Politecnico di Torino

Corso di laurea magistrale in Ingegneria Informatica (Computer Engineering)
A.a. 2021/2022
Sessione di Laurea Marzo/Aprile 2022

Applicazione mobile a supporto della Manutenzione Predittiva Industriale

Relatore:
Prof. Massimo Poncino

Candidato:
Andrea FRANCO

Sommario

Indice delle Figure

1 Introduzione

Negli ultimi anni gli sviluppi ottenuti nell'ambito delle innovazioni tecnologiche e nel progressivo e sempre più significativo abbassamento dei costi di queste ultime ha portato il mondo industriale a rinnovarsi e a mutare in quella che oggi è definita Industria 4.0. Tale termine è utilizzato per indicare la tendenza dell'attuale automazione industriale a sviluppare e implementare nuove tecniche per perfezionare e affinare l'intero processo industriale, sia in termini di miglioramento della produttività e della validità del prodotto, così come nell'incremento delle condizioni lavorative e nella possibilità di nuove frontiere di business. In questo senso, non è esagerato affermare che l'Industria 4.0 rappresenti una nuova pagina delle rivoluzioni industriali che hanno caratterizzato l'evoluzione del settore, o quantomeno una futura (ed in numerose realtà già attuale) tendenza.

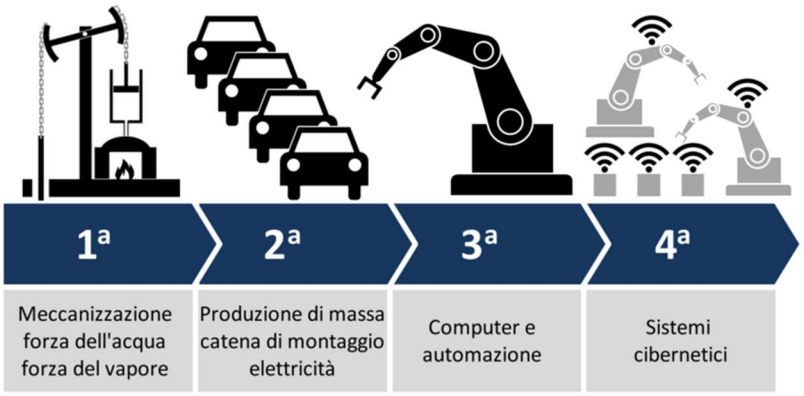

Figura 1 - Quattro "rivoluzioni industriali" con descrizione [1]

È in questo contesto che la società ZIRAK s.r.l. si è proposta di operare negli ultimi anni. In particolare, il lavoro svolto per la stesura di questo elaborato è dovuto alla collaborazione con la suddetta nella sua partecipazione al progetto PREMA. Nello specifico, la trattazione seguente verterà sulla porzione di tale progetto riservata allo sviluppo di un'interfaccia per dispositivi mobili.

[1] Christoph Roser, Illustration of Industry 4.0, showing the four "industrial revolutions" with a brief Italian description, AllAboutLean.com, 2016

I restanti paragrafi di questo capitolo presenteranno brevemente l'azienda ed il progetto di interesse, infine si concluderà con una descrizione di come sarà strutturato il resto del documento.

1.1 ZIRAK s.r.l. – L'Azienda

Figura 2 - Logo ZIRAK s.r.l.[2]

ZIRAK s.r.l. nasce nei primi mesi del 2000 come fusione di diverse realtà complementari operanti nel settore dell'Information Technology.

Essa è composta da numerosi dipartimenti autonomamente gestiti e in grado di cooperare su sistemi integrati. Tali dipartimenti sono:

ZIRAK Automotive

ZIRAK IoT & Machine Learning

ZIRAK Mobile

ZIRAK Software

ZIRAK WebSolutions

La spiccata competenza e l'attitudine a gestire progetti di svariate dimensioni ha portato ZIRAK a beneficiare, negli ultimi anni, di un incremento sia in termini di fatturato che di risorse umane, motivate ed ampiamente qualificate.

[2] ZIRAK s.r.l., Logo Zirak, www.zirak.it

La sede principale dell'azienda si trova a Mondovì, città raggiungibile in meno di 1 ora da Torino. Oltre a ciò, si avvale di un laboratorio di Ricerca e Sviluppo situato in Portogallo, collabora con istituti superiori ed università internazionali e prende parte a numerosi progetti di ricerca Europei.

Nel corso del suo operato la società si è saputa contraddistinguere per l'ottenimento di due obiettivi cardine: l'innovazione (intesa come lo sviluppo e la proposta di soluzioni innovative) e la soddisfazione del cliente, ottenuta mediante lo studio congiunto di soluzioni ottimali e altamente personalizzate.

ZIRAK si propone quindi come singolo interlocutore da ricercare per questioni connesse all'ambito dell'Information and Communication Technology, capace di fornire per esse soluzioni personalizzate a partire dall'analisi delle specifiche esigenze, proseguendo con la loro progettazione e concludendo con la loro realizzazione.

1.2 PREMA – Il progetto

Come già evidenziato in precedenza, una delle attività in cui ZIRAK è attualmente impegnata è il progetto PREMA. Tale progetto è classificato come "Progetto di Ricerca nell'ambito del Polo di Innovazione ICT"[3] afferente al Bando PRISM-E. PREMA è un acronimo per piattaforma di PREdictive MAintenance vibrazionale, per l'appunto il progetto si è proposto come elemento abilitante la Predictive Maintenance (PDM, Manutenzione Predittiva) nell'ambito del modello di Industria 4.0, sfruttando componenti come la Big Data Analysis e l'IoT al fine di ricavare informazioni ed ottenere conoscenze in tempo reale riguardo il comportamento di entità sottoposte a tensioni vibrazionali o macchinari che siano composti da unità intensamente esposte a vibrazioni. Il progetto si pone come fine quello di minimizzare i guasti e le anomalie degli stessi per mezzo di un'analisi predittiva che permetta operazioni mirate e intelligenti di manutenzione ed eviti quindi delle

[3] Polo di Innovazione ICT, Format del progetto, Format-di-Progetto_PRISM-E_PREMA V4, 2020

situazioni critiche di emergenza o rottura dell'apparato in esame, grazie ad una combinazione efficace di:

1) Sensoristica IoT specializzata che non necessita di alcun tipo di cablaggio.
2) Un avanzato apparato di Machine Learning che si fa forza dell'efficacia del Cloud Computing.
3) Unità Edge che si avvalgono della complementarità offerta dal Fog Computing.

Il progetto è considerato come "disruptive", ossia capace di creare valore in un modo così nuovo da sconvolgere un mercato esistente[4], da realtà Europee di punta del settore, inserendosi in un contesto storico favorevole per lo sviluppo di IoT e avvalendosi di tre società piemontesi complementari e due enti di ricerca (rispettivamente in Italia e Portogallo) profondamente specializzati e sfruttando infine un copioso elenco di novità tecnologiche "intelligenti".

Nell'ambito del progetto è stata stabilita un'architettura che potesse definire una piattaforma completa, composta da 4 macro componenti:

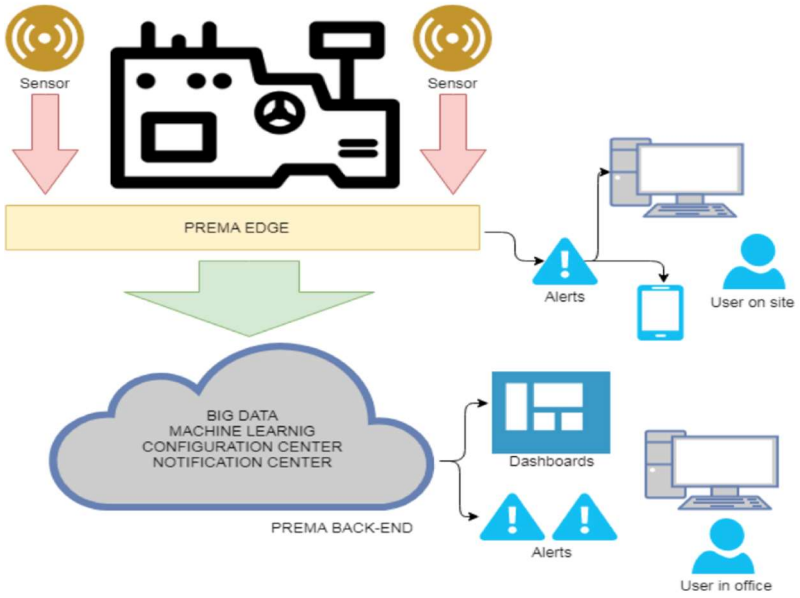

Figura 3 - Architettura ad alto livello del progetto PREMA[5]

[4] Clayton Christensen, What is Disruptive Innovation?, Harvard Business Review, 2015
[5] ZIRAK s.r.l., PREMA Architecture, www.zirak.com/iot/prema

1) Sensori a bordo macchina o sullo specifico componente: dispositivo specializzato nella rilevazione di vibrazioni, in grado di ricevere parametri di configurazione e memorizzare tali parametri, così come di rivelare i dati vibrazionali di interesse e comunicare tali dati durante le svariate operazioni della macchina o direttamente durante il ciclo di vita completo di una certa struttura.
2) PREMA edge, un componente in grado di sopperire ai problemi di scalabilità del sistema considerato nelle situazioni in cui l'elevato numero di macchinari, l'alta frequenza di campionamento o l'esistenza di sensori ausiliari facciano presagire una mole di dati da comunicare al backend eccessiva. L'edge si inframmezza tra il backend ed i sensori con il fine di non svolgere unicamente la raccolta dei dati, ma altresì di compiere un pre-processamento degli stessi in modo di ridurne la mole, individuando quelli più significativi da trasferire al backend. Inoltre, uno dei suoi compiti è altresì quello di fornire predizioni "localizzate" al sito produttivo, in contrapposizione all'analisi più globale del backend.
3) PREMA backend, costituito da un'unità che consenta la gestione centralizzata dei dati sfruttando le tecnologie Big Data, il reperimento di informazioni utili fornendone allo stesso tempo una consultazione, il controllo dello storico e l'analisi a partire dalle statistiche relative. In aggiunta, è presente un sottosistema per la comunicazione di allarmi e notifiche e uno per la configurazione dei sensori. I dati raccolti, raggruppati in cluster relativamente alle condizioni applicative o al tipo di lavorazione specifico, daranno la possibilità di individuare immediatamente possibili problematiche e di effettuare una manutenzione predittiva, sfruttando avanzate tecniche di "Early PDM" (quali ad esempio algoritmi di Outliers Detection).
4) Frontend, composto a sua volta da:
    a) Presentation Dashboard su pagine web per l'interazione remota, che consentano una immediata gestione e consultazione dei dati con la capacità di creazione di grafici ad hoc, consultazioni dedicate e grafiche personalizzate in base al cliente. È altresì

previsto l'interfacciamento dell'intero sistema PREMA con i framework e server specifici delle varie aziende, per consentire la presentazione delle conoscenze sviluppate dal sistema PDM internamente alla rappresentazione aziendale relativa al processo produttivo.

b) Pannelli di configurazione, intesi come punti di interfaccia per il back office che permettano di configurare in primis il modo in cui le informazioni generate dal sistema PDM vengano visualizzate, ma anche gli specifici sensori o il relativo centro di notifiche e allarmi.

c) Mobile App dedicate, per la configurazione dei sensori, la consultazione del loro stato in tempo reale, la lettura di dati ricavati dal PDM in modo intuitivo e diretto e la segnalazione di notifiche che consentano, in una situazione di emergenza, di intervenire proprio quando alcuni minuti potrebbero fare la differenza. Ciò permette la trasformazione del processo produttivo in un cosiddetto Cyber Physical System (CPS) integrato, in pieno accordo con il modello di Industria 4.0.

Vale la pena evidenziare come l'analisi svolta in primis dall'edge e poi dal backend sia di fatto da considerarsi complementare. Infatti, se l'edge nella sua analisi impatta localmente gli apparati connessi ad esse grazie al dettaglio e alla specificità dei dati utilizzati, l'analisi del backend è effettuata contemplando indicazioni più eterogenee e di conseguenza è in grado di correlare informazioni di macro livello che hanno un impatto sulla conoscenza ottenuta dal PDM.

Relativamente a PREMA, ZIRAK si configura non soltanto come partner di progetto, ma come suo capofila, in quanto ad essa si devono la sua ideazione e lo sviluppo della tecnologia relativa al backend fino a TRL 4 (Technology Readiness Level 4 – tecnologia validata in laboratorio). Congiuntamente con gli altri partner e grazie al supporto di più dipartimenti di due Organismi di Ricerca, intende portare il progetto ad un grado di sviluppo TRL 7 (dimostratore in campo). In particolare, nel corso del progetto, oltre alle funzioni di amministrazione ZIRAK prende parte nella determinazione dell'architettura software, nello sviluppare ogni componente

software collegata al mobile/frontend e al backend, così come nel testare e validare le prime, nel documentare e finalizzare il dimostratore sul campo e in conclusione nella divulgazione dei risultati. Come precedentemente indicato, l'attenzione di questo documento si focalizzerà sullo sviluppo dell'applicazione mobile relativa a PREMA.

1.3 Obiettivi ed Organizzazione della Tesi

L'obiettivo principe della tesi è la realizzazione di un app mobile che vada ad integrarsi come uno dei componenti del frontend individuati nella architettura del progetto PREMA. In particolare, ad inizio lavori è stato concordato con il team aziendale impiegato nella realizzazione del progetto che l'applicazione permettesse una immediata ed intuitiva consultazione degli allarmi generati dai sensori o dal sistema di Predictive Maintenance, così come la possibilità di interfacciarsi con una panoramica delle macchine e dei sensori dell'impianto considerato ed avere la capacità di inserire nuovi modelli di predizione e visualizzarne la lista ed il relativo skeleton e statistiche acquisite. Una spiegazione più ampia ed accurata verrà fornita nel relativo capitolo di questo elaborato.

Il secondo obiettivo è la divulgazione di quanto considerato e ottenuto durante la realizzazione di tale app, preceduta da un'opportuna e doverosa panoramica sugli aspetti che si pongono al contorno di un'applicazione di questo genere.

Oltre a questo primo capitolo introduttivo, la tesi è divisa in quattro ulteriori capitoli. Il secondo capitolo ha la finalità di fornire una visuale più approfondita su temi come la manutenzione predittiva e l'Industria 4.0, per cogliere il contesto e le motivazioni che sono posti alla base in primis del progetto PREMA ed in subordine alla app mobile associata. Il terzo capitolo mostrerà una panoramica del mondo delle app mobili, con particolare riferimento al sistema operativo Android e allo stato dell'arte in questo campo. Il quarto capitolo è dedicato alla esposizione di quanto svolto per la concretizzazione dell'applicazione, con una descrizione approfondita delle sue viste in termini di layout grafico e logiche di funzionamento del software. Al termine dei capitoli sopraccitati sarà presente una conclusione che conterrà una discussione degli obiettivi raggiunti ed eventuali possibili futuri sviluppi o miglioramenti che potrebbero essere integrati nell'applicazione.

2 Scenario e background tecnologico

Nel primo capitolo di questa trattazione è stato indicato come il progetto PREMA si vada ad inserire nell'ambito dell'Industria 4.0. È stato altresì segnalato che la principale materia di interesse dell'intero proponimento sia la Manutenzione Predittiva, ottenuta in questo caso tramite un'analisi vibrazionale conseguita grazie all'utilizzo di tecnologie profondamente attuali come l'Internet of Things e la Big Data Analytics. Nonostante alcuni di questi termini possano essere, almeno in minima parte, conosciuti ai più, è opportuno fornire per essi una spiegazione più approfondita di quanto già esposto precedentemente.

2.1 Industria 4.0

In aggiunta alla definizione e alla spiegazione fornita per introdurre l'Industria 4.0, se si vuole comprendere appieno l'essenza di questa nuova rivoluzione industriale è impossibile tralasciare concetti quali quello di smart factory e di tecnologie abilitanti.

La concezione di smart factory è talmente legata a quella di Industria 4.0 che si può intendere la prima come una concretizzazione del modello individuato dalla seconda.

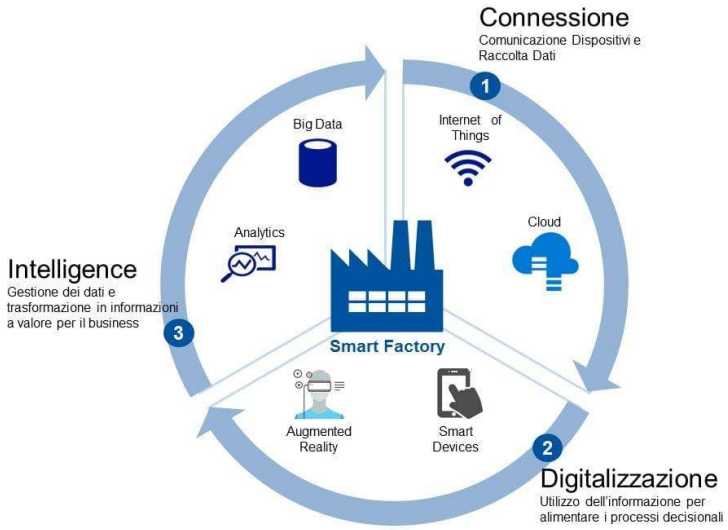

Figura 4 - Smart Factory[6]

[6] Minifaber, smart-factory-schema, minifaber.it/blog/la-smart-factory-o-industry-4-0

Perché uno stabilimento industriale sia considerabile come realizzazione di una smart factory, esso deve prima di tutto implementare una smart production, ossia tutte quelle tecnologie produttive innovative in grado di creare una collaborazione tra l'operatore, gli strumenti che ha a disposizione e le macchine con cui è a stretto contatto. Inoltre, il secondo aspetto che deve essere presente è indentificato dagli smart services, definibili come la serie di strutture, tecniche e installazioni informatiche che consentono l'integrazione tra i sistemi aziendali e tra lo stabilimento e le strutture esterne. Il complesso deve avvenire in un'ottica di smart energy, prestando quindi attenzione ai dispendi energetici attraverso l'impiego di energie sostenibile e l'analisi e riduzione degli sprechi.

In un articolo[7] relativo ad uno studio del Boston Consulting Group viene affermato che il fulcro della nuova rivoluzione industriale sia da ricercarsi nella decisione di adottare i cosiddetti "Pillars of Technological Advancement", traducibili come tecnologie abilitanti.

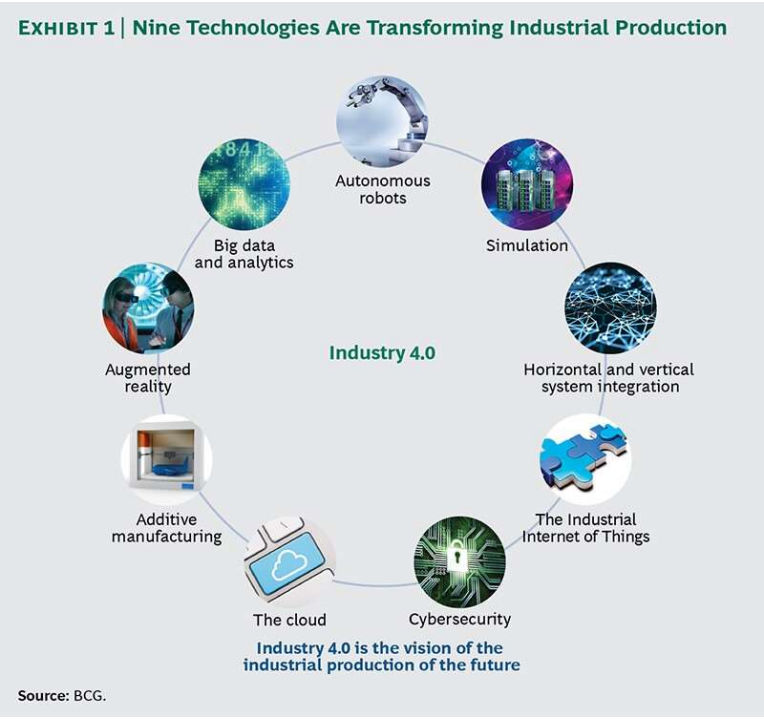

Figura 5 - Le nove tecnologie abilitanti[8]

[7] Michael Rüßmann, Lorenz, Gerbert, Waldner, Engel, Harnisch, and Justus, Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries, Boston Consulting Group, 2015

[8] Boston Consulting Group BCG, Exhibit 1, www.bcg.com, 2015

Di queste non tutte sono necessariamente concetti innovativi, ma se in passato erano state implementante solo in ambiti di ricerca o in via sperimentale, nel contesto economico-sociale attuale esse possono imporsi ed essere adottati in veri e propri sistemi di produzione. Vengono di seguito elencate e descritte le nove tecnologie abilitanti precisate dal Boston Consulting Group:

a) Autonomous Robots, o Robot Autonomi: le industrie produttrici hanno tratto profitto dall'utilizzo di robot per effettuare compiti complessi e precisi, ma negli ultimi anni i robot si stanno evolvendo per utilizzi ancora più grandiosi. Stanno diventando più flessibili, autonomi e soprattutto cooperativi. In alcuni casi infatti tali robot sono in grado di interagire tra di loro e addirittura operare fianco a fianco con le persone, da cui riescono ad imparare. Rispetto a quelli utilizzati in passato, questi robot promettono di essere più economici e risultare all'altezza di una maggiore varietà di compiti.

b) Horizontal and Vertical System Integration, o integrazione verticale e orizzontale: la maggior parte degli attori negli odierni processi produttivi non è completamente integrata. Ma con l'Industria 4.0 è possibile avere dipartimenti, mansioni e capacità fortemente coesi, così da ottenere delle reti di integrazione di dati universali.

c) Cybersecurity, o sicurezza informatica: numerose aziende fanno ancora affidamento su sistemi di produzione e gestionali sconnessi o sbarrati. Tuttavia, con l'incremento della connettività e l'utilizzo di protocolli standard di comunicazione propri della realizzazione del modello di Industria 4.0, la necessità di proteggere tali sistemi e comunicazioni aumenta considerevolmente. Per questo motivo canali di comunicazione affidabili e sicuri, così come i meccanismi di autenticazione ed accesso di macchine e utenti risultano essenziali. Nell'ideazione del progetto PREMA uno degli aspetti tenuti in considerazione è per l'appunto relativo all'utilizzo di canali e protocolli di comunicazione sicuri tra le varie entità architetturali, impiegando ad esempio meccanismi di autenticazione tramite certificati.

d) Additive Manufacturing, o manifattura additiva: le aziende hanno cominciato da poco ad utilizzare la manifattura additiva, ossia la fabbricazione di oggetti a partire da modelli 3D progettati al computer ottenuti aggiungendo vari strati in fila (al contrario delle tradizionali metodologie di fabbricazione proprie della produzione sottrattiva, come nel caso di torni e fresatrici). Con l'Industria 4.0 questa produzione additiva può essere portata all'estremo con la fabbricazione di tanti piccoli lotti di prodotti su misura che offrono numerosi vantaggi costruttivi.

e) Big Data Analytics: l'analisi basata su vaste collezioni di dati è emersa solo di recente nell'ambiente manifatturiero, al cui interno è già stata in grado di ottimizzare i risparmi energetici e la qualità della produzione. In un contesto di Industria 4.0, il reperimento e la valutazione integrale di dati eterogenei nei tipi e nelle sorgenti diventerà lo standard per il supporto agli eventi decisionali in tempo reale, nonché uno strumento per effettuare previsioni o predizioni. Una porzione importante del progetto PREMA, su cui si basa il concetto di manutenzione predittiva, aspirazione principale dell'intero progetto, ha proprio come concretizzazione l'impiego di tecniche di Big Data Analytics. Il concetto verrà sviscerato più approfonditamente nel seguito.

f) Simulation, o simulazione: nel moderno mondo produttivo le simulazioni 3D di processi produttivi, materiali e prodotti sono già una realtà, ma in un futuro prossimo verranno utilizzate in maniera più estesa anche in altri ambiti di operazioni impiantistiche. Queste simulazioni sfrutteranno dati raccolti in tempo reale per mimare il mondo fisico in un modello virtuale, che potrà includere macchine e operatori. Questi ultimi potranno quindi testare ed eventualmente ottimizzare le impostazioni dei macchinari per l'ordine di produzione successivo nella linea, prima che questo sia in lavorazione nell'ambiente fisico, in modo da ridurre le tempistiche di settaggio e migliorare la qualità del processo.

g) Industrial Internet of Things, o IoT industriale: attualmente, nella grande maggioranza dei casi soltanto alcune delle macchine e dei sensori di uno stabilimento sono collegati in rete e fanno uso di computazione embedded. Solitamente, sono inoltre tipicamente organizzati in uno schema di automazione piramidale verticale, in cui quindi i sensori con limitati controllori automatici comunicano e delegano ad un sistema di controllo più generale. Con l'IoT industriale molti più dispositivi verranno arricchiti ed integrati di controllori embedded e connessi l'un l'altro utilizzando protocolli e tecnologie standard di comunicazione. In questo modo potranno continuare, se necessario, a dialogare con un controllore più centralizzato, ma saranno anche in grado di interagire uno con l'altro, decentralizzando così quando opportuno l'analisi ed il processo decisionale in maniera da reagire in tempo reale. Componenti fondamentali dell'architettura del progetto PREMA sono proprio dei sensori IoT, in grado di collegarsi all'Edge o al Backend per comunicare ed interagire con essi. Una ulteriore analisi verrà esposta nel proseguo della trattazione.

h) Cloud: le imprese stanno già utilizzando il cloud, soprattutto per l'archiviazione dei dati e per l'esecuzione di programmi aziendali e di analisi. Con l'Industria 4.0, tuttavia, si renderà necessario uno scambio e condivisione di dati tra più stabilimenti e oltre i confini fisici delle aziende. Inoltre, le performance dei cloud in termini di risposta aumenteranno considerevolmente, arrivando a tempi di reazione nell'ordine di pochi millisecondi. Come risultato, sempre più dati verranno archiviati e sempre più processi distribuiti sui cloud. Le logiche e gli algoritimi necessari alle tecniche di analisi predittiva impiegate nel progetto PREMA si appoggiano infatti su sistemi in Cloud e distribuiti.

i) Augmented Reality, o realtà aumentata: i sistemi basati sulla realtà aumentata sono attualmente in una fase primordiale, ma le loro potenzialità li rendono di vivo interesse per le industrie manifatturiere. Grazie ad essi saranno possibili numerosi servizi quali la formazione virtuale degli

operatori sul campo, la selezione di componentistiche di magazzino e la visualizzazione di operazioni di riparazione.

Sebbene un'implementazione complementare di tutte queste tecnologie abilitanti avrebbe un impatto cumulativo sui benefici che queste possono portare, l'adozione anche scaglionata nel tempo o di solo alcune di esse può comunque garantire benefici tangibili nel contesto aziendale.

2.2 Predictive Maintenance

Qualunque macchinario industriale, con tempistiche e modalità altamente variabili, va inevitabilmente incontro nel corso del suo utilizzo a fenomeni di usura, disallineamenti o danneggiamenti che se non correttamente monitorati e gestiti possono portare a malfunzionamenti, a volte anche molto gravi e repentini. Per tale motivo, è pratica comune e fondamentale inserire nelle attività industriali standard dei piani di manutenzione attentamente studiati e massi in atto.
Una prima definizione di manutenzione è stata enunciata nel corso dei primi anni Sessanta, la quale dichiara: "S'intende per manutenzione quella funzione aziendale alla quale sono demandati il controllo costante degli impianti e l'insieme dei lavori di riparazione e revisione necessari ad assicurare il funzionamento regolare e il buono stato di conservazione degli impianti produttivi, dei servizi e delle attrezzature di stabilimento"[9]. Da questa esposizione è importante porre l'attenzione su tre principali aspetti. Il primo punto che viene sollevato è che il controllo deve essere costante, non può quindi venire effettuato ad intervalli discreti, ma in maniera continuativa nel corso del tempo. Successivamente, viene evidenziato come tale controllo deve essere compiuto non valutando unicamente il solo funzionamento regolare degli impianti di produzione, ma tenendo conto anche del corretto stato di conservazione degli stessi. Non considerando quest'ultimo aspetto, nonostante le attività possano sembrare svolgersi correttamente in un dato istante, non c'è garanzia ed è anzi probabile che in un momento futuro il deterioramento non scrupolosamente vigilato porti ad un malfunzionamento. Infine,

[9] Organizzazione per la Cooperazione e lo Sviluppo Economico, Definizione di Manutenzione, Delibera OCSE, 1963

la questione conclusiva su cui viene posto l'accento è l'assunto che a risultare soggetti a manutenzione non debbano essere solamente i macchinari e gli impianti di produzione, ma anche tutti i servizi ad essi collegati. La manutenzione deve quindi essere intesa come un'operazione che copra nel suo insieme l'intero processo produttivo.
Una volta compreso il concetto generale di manutenzione è essenziale osservare che questa non sia univoca, ma si suddivide in diverse accezioni. Ne esistono infatti diverse tipologie e configurazioni[10], che verranno ora di seguito elencate.

a) Manutenzione autonoma: anche chiamata automanutenzione o AM, è quella che viene effettuata direttamente dall'operatore che ha a che fare con il sistema in esame. Questa si basa sull'assunto che, rimanendo spesso a contatto e maturando quindi una rilevante familiarità con il macchinario considerato, il lavoratore che lo utilizza sia in grado di percepire con i propri sensi indizi di un iniziale o futuro malfunzionamento e che possa quindi intervenire autonomamente o segnalare l'anomalia al personale di competenza per sopperire all'alterazione.

b) Manutenzione preventiva: indicata anche come PM, si tratta di quella categoria di accorgimenti ed interventi volti alla sostituzione o riparazione dei componenti che ne necessitano prima che il guasto o malfunzionamento abbia effettivamente avuto luogo. Essa si divide solitamente in altre concezioni, tra cui quelle di manutenzione statistica (la riparazione o sostituzione viene effettuata tenendo conto di considerazioni di tipo statistico sul tempo di vita medio di un certo componente) e manutenzione secondo condizione (la valutazione viene svolta in base alla misurazione di alcune grandezze fisiche che possano suggerire l'attuale stato di salute del componente in esame).

[10] Riferimento 191.07.16 della Norma UNI 9910 e Riferimenti 3.9, 3.9.1, 3.9.2, 3.9.3, 3.7, 3.10, 3.11 e 3.12, della Norma UNI 10147, rispettivamente del 1991 e 1993

c) Manutenzione incidentale: conosciuta anche come manutenzione “a guasto”, si identifica con la particolare modalità in cui l’intervento sostitutivo o di riparazione viene eseguito soltanto a guasto o malfunzionamento già avvenuto. Non vi è in questo caso quindi alcun monitoraggio particolare se non quello evidente di rilevazione del guasto.

d) Manutenzione migliorativa: è una peculiare classe di manutenzione in cui l’intervento sull’impianto o macchinario è effettuato al fine di incrementare le prestazioni o l’usabilità dello stesso e non come negli altri casi per fare fronte ad un guasto o malfunzionamento. L’esigenza di tale miglioramento può essere portata alla luce dal manutentore così come direttamente dall’operatore.

e) Manutenzione opportunistica: è una fascia caratterizzata dal compimento dell’intervento manutentivo in un periodo di tempo in cui nell’azienda è stato stabilito che la disponibilità del sistema considerato non sia necessaria. Nonostante questo consenta evidenti risparmi in termini di tempo sottratto alla produzione in sé, non è sempre possibile né certo che esista una porzione temporale in cui il macchinario o impianto sia non necessario.

f) Manutenzione produttiva: dal termine inglese Total Productive Maintenance (TPM), mira all’ottenimento di un elevato grado di efficienza relativamente ad ogni entità aziendale. In questo caso quindi, le prestazioni dell’impianto vogliono essere garantite spostando l’attenzione dall’impianto in sé al personale tecnico, operativo e di manutenzione.

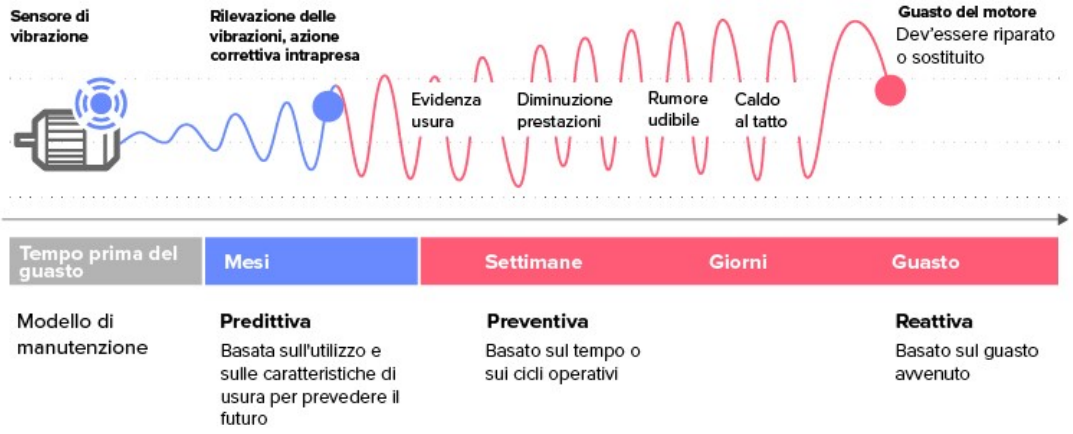

Figura 6 - Manutenzione Predittiva[11]

Una particolare tipologia di manutenzione preventiva non citata nel precedente listato è la manutenzione predittiva (o predictive maintenance, PDM). Nello specifico la prima fase di questo tipo di manutenzione si concretizza nella ricerca e identificazione di determinati parametri di interesse. Successivamente tali parametri devono essere misurati e registrati attraverso opportune tecnologie di monitoraggio, quali sensori o test chimico-fisici. In ultima battuta, i dati raccolti sono sviluppati alimentando opportuni modelli matematici, il cui scopo ultimo è quello di restituire un'indicazione dello stato di salute dell'entità considerata al fine di generare una stima riguardo il tempo di vita rimanente prima del verificarsi del guasto. Esistono numerosi esempi di applicazione di questo metodo, quali ad esempio l'esame delle correnti assorbite in un determinato circuito, la rilevazione di attriti anomali evidenziati da produzioni di calore identificate mediante termografia, la valutazione di un olio lubrificante attraverso analisi tribologiche ed accurate indagini vibrazionali idonee per vagliare eventuali squilibri o disallineamenti. Rispetto alle attuali metodologie di manutenzione preventiva, le cui operazioni sono limitate a degli interventi pianificati in funzione del monte ore complessivo di funzionamento e sulla statistica cadenza di malfunzionamento dei sistemi in esame, i sistemi di predictive maintenance specializzati consentono di ottenere risultati di gran lunga preferibili.

[11] Gian Luca Panizzari, Manutenzione-base-tempo-e-condizione, Management Problem Solving, consultingmps.com/manutenzione-predittiva-quando-sceglierla/

Relativamente all'impatto[12] che dei sistemi di predictive maintenance possono comportare, questo può essere quantificato evidenziando i seguenti aspetti:

- diminuzione fino al 50% della durata di fermo-macchina causati dalla rottura di un macchinario, ossia il lasso di tempo che si rende necessario per sostituire o riparare il componente danneggiato, durante il quale la macchina non può continuare il suo processo produttivo;

- estensione tra il 3% ed il 5% del tempo di vita dell'apparecchio, se messo a confronto con altre tecniche di manutenzione come la semplice manutenzione statistica;

- riduzione delle spese dovute alla manutenzione da sostenere che oscilla tra il 10% e il 40%;

- decremento della frequenza di incidenti e potenziali infortuni sul posto di lavoro che va dal 10% al 25%;

- abbassamento netto degli sprechi, con evidenti conseguenze sia in termini di impronta ambientale che di costi evitabili, che può variare dal 10% fino al 20%.

A questi effetti calcolabili si aggiungono tutta una serie di conseguenze difficilmente quantificabili, ma ugualmente considerevoli e di notevole vantaggio. Nel dettaglio:

- un ridotto impatto ambientale, dovuto all'assottigliamento dei consumi energetici e al calo dei materiali altrimenti utilizzati per sostituire componenti con ancora un tempo di vita residuo;

[12] blog.appliedai.com/predictive-maintenance/

- la generazione massiva di dati analitici evoluti, vantaggiosi al fine di contraddistinguere delle potenziali occasioni di perfezionamento del processo produttivo considerato;

- un accrescimento dell'affidabilità delle tempistiche di consegna attese e della bontà del prodotto in sé, con un conseguente incremento della soddisfazione finale del cliente;

- un sostanziale miglioramento relativo al morale degli operatori di macchina, grazie alla riduzione del turbamento causato da attrezzature guaste o che non operano efficacemente come dovrebbero che si porrebbero quindi come sorgente di problematiche inattese;

- l'accresciuta conoscenza e consapevolezza in relazione al processo produttivo passato proprio dell'azienda, ottenuta in virtù di un immagazzinamento dei dati e della loro storicizzazione;

- la cancellazione del rischio di dare origine a superflui effetti collaterali dei macchinari dovuti all'errato giudizio sulla necessità di un intervento di manutenzione, la cui applicazione può portare in alcuni casi alla generazione di danni inattesi.

Recenti studi[13] pubblicati dal Wall Street Journal advertising department hanno considerato con una stima che un'inefficace programma strategico di manutenzione possa ridurre di un fattore compreso tra il 5% ed il 20%, in media, l'effettiva capacità produttiva di un complesso industriale. In tale contesto le spese complessive globali dovute ai fermo-macchina non previsti impattano ogni anno sull'intera industria manufatturiera per un valore stimato di 50 miliardi di dollari statunitensi. Di questo tempo in cui le macchine non operano per fermo-macchina

[13] IndustryWeek in collaborazione con Emerson, Unlocking Performance, partners.wsj.com/emerson/unlocking-performance/how-manufacturers-can-achieve-top-quartile-performance/

imprevisti, ben il 42% è dovuto ad una rottura o malfunzionamento delle apparecchiature.

La sola manutenzione reattiva (RM, precedentemente indicata come manutenzione incidentale o "a guasto"), effettuata ovvero solamente in seguito ad aver appurato un guasto o una rottura già avvenuta, non ha un peso esclusivo relativamente all'intervento manutentivo, ma può dare origine a flussi e ripercussioni che impattano negativamente lungo l'intera catena del processo produttivo. Lo sfruttamento combinato di entrambi i benefici scaturiti dall'utilizzo di tecniche di manutenzione preventiva (PM) e manutenzione predittiva (PDM) porta al raggiungimento del miglior programma di manutenzione ottenibile[14]. A tale scopo si rende opportuno utilizzare il metodo adatto nelle situazioni in cui è applicabile e stabilire quale strategia mettere in atto in un'ottica di facoltà di adattamento alle particolari necessità e flessibilità operativa. Proprio con l'obiettivo di ottenere il miglior risultato, nel progetto PREMA è stato deciso di includere un sistema ibrido di manutenzione preventiva e manutenzione predittiva, che sappia sfruttare i benefici che ognuna delle due classi di tecniche, se opportunamente implementata, è in grado di mettere a disposizione.

Uno degli strumenti più utilizzati nel campo della manutenzione industriale è quello identificato dai cosiddetti Computerized Maintenance Management System, o CMMS. Anche chiamato sistema computerizzato di gestione della manutenzione, è un programma la cui funzione è quella di permettere la centralizzazione delle informazioni riguardanti la manutenzione ed una conseguente semplificazione delle procedure relative alle operazioni ed interventi di manutenzione[15]. In particolare, un CMMS favorisce la gestione degli asset (intesi come beni di proprietà di un'azienda), consente una pianificazione ordinata e studiata delle manutenzioni da effettuare e aiuta a tracciare e monitorare gli ordini di lavoro aziendali.

[14] Accelix, Predictive versus preventive maintenance, www.accelix.com/predictive-versus-preventive-maintenance/, 2018

[15] International Business Machines Corporation IBM, Cosa è un CMMS?, www.ibm.com/it-it/topics/what-is-a-cmms

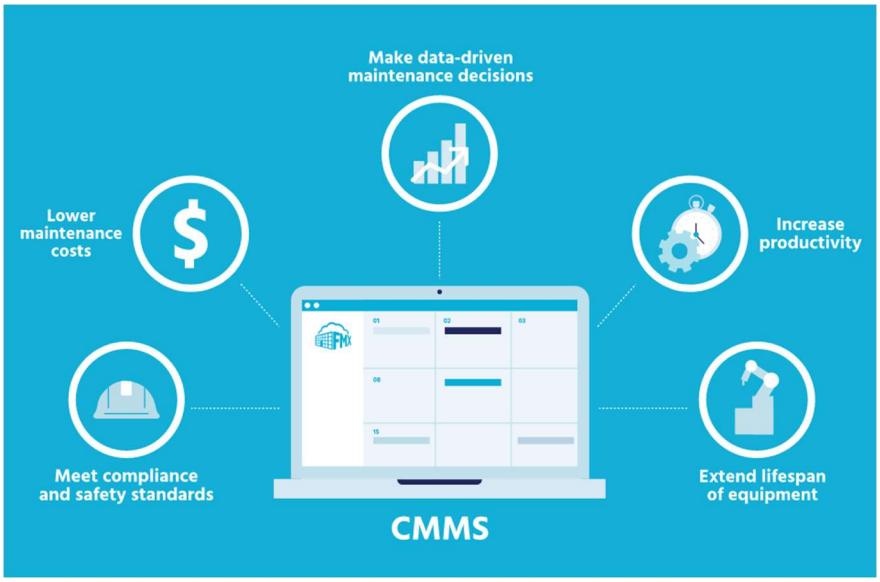

Figura 7 - Diagramma CMMS[16]

In un'indagine riguardante gli stessi è stato evidenziato come nel 2018 quasi il 60% degli impianti aziendali di produzione individui nella presenza di apparati di predictive maintenance una delle proprietà peculiari più desiderate di un CMMS. A tale interesse sarebbe associato uno sviluppo globale nel settore dei sistemi PDM specializzati ed una previsione di aumento di capitale investito in questi ultimi pari a più di 3 miliardi di dollari statunitensi. Mettendo in conto un incremento annuo stimabile intorno al 90%, tale ammontare potrebbe crescere fino a sfiorare nel 2022 la quota degli 11 miliardi di dollari americani. Questo aumento di attrazione nei confronti della manutenzione predittiva è confermato anche secondo le analisi delle tendenze di ricerca offerte da Google Trends, come è possibile apprezzare dal grafico mostrato nella figura successiva. Tale grafico mostra l'andamento nel tempo del quantitativo di interesse relativo al concetto di Predictive Maintenance, basato sulle ricerche effettuate con il motore di ricerca di Google su scala globale nella categoria identificata come "Commercio e industria", nel periodo compreso tra l'inizio del 2011 e la fine del 2018.

[16] FMX, CMMS Software Diagram, www.gofmx.com/wp-content/uploads/2020/04/cmms-software-diagram.svg

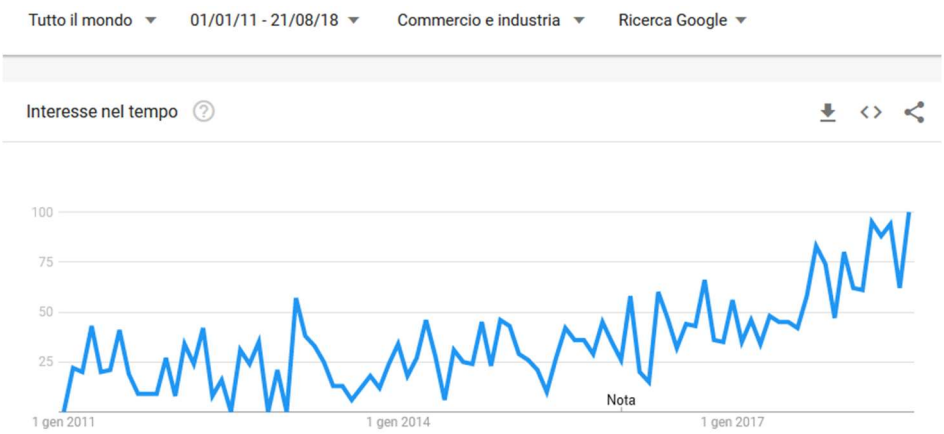

Figura 8 - Trend di ricerca "Predictive maintenance" nel settore Industriale[17]

Tra gli aspetti e le motivazioni che hanno portato all'ideazione del progetto PREMA, la sostenibilità ambientale riveste un ruolo di primaria importanza. Infatti, uno dei principali aspetti da considerare quando si parla di sostenibilità ambientale è costituito dalla limitazione degli sprechi[18]. Come già indicato precedentemente l'impiego di sistemi di predictive maintenance può giocare un ruolo fondamentale in questo campo. Grazie all'utilizzo di strumenti PDM infatti uno stabilimento industriale è in grado di ridurre i costi dovuti al consumo di energia elettrica fino al 40%. Gli stessi strumenti inoltre forniscono la possibilità di riconoscere e mettere in luce frangenti anomali di funzionamento che portano a consumi subottimali, con una relativa crescita delle spese per i consumi energetici che oscilla tra il 30% ed il 60%[19]. Quanto descritto non tiene inoltre conto dell'impronta ambientale generata dalle materie prime sprecate nel corso del processo produttivo e delle immissioni nell'ambiente di prodotti di scarto e gas di scarico, figli di una non corretta e insufficientemente studiata operazione di manutenzione.

[17] Google Trends, *Trend di ricerca "Predictive maintenance" nel settore Industriale, 2011-2018*
[18] Rapporto Tecnico dell'Autorità Ambientale 12/2013 relativo al POR FESR 2007-2013
[19] Jonathan Cooper, Stop wasting energy: Use predictive maintenance, embeddedcomputing.com, 2017

2.3 Internet of Things e Big Data Analytics

Storicamente, il termine Internet è stato utilizzato a partire dal 1982, con la definizione dei protocolli TCP (Transmission Control Protocol) e IP (Internet Protocol), per indicare la moltitudine di reti collegate tra loro attraverso tali protocolli. Da quel momento, la sua diffusione è aumentata esponenzialmente sia in termini geografici che di terminali connessi. È tuttavia nel nuovo millennio che si è assistito alla più vertiginosa crescita della cifra dei suddetti, basti pensare che nel 1996 il numero globale di computer connessi ad Internet fosse di appena 10 milioni, mentre alle porte del 2021 circa 4,66 miliardi di persone in tutto il mondo accedono ad Internet[20]. Tale impressionante ammontare tuttavia misura l'utenza che fa uso di Internet solamente in termini di singoli individui e non il bacino smisurato di dispositivi effettivamente connessi. Una valutazione circa il numero di questi device suggerisce che esso possa attestarsi al giorno d'oggi intorno alla soglia dei 25 miliardi[21].

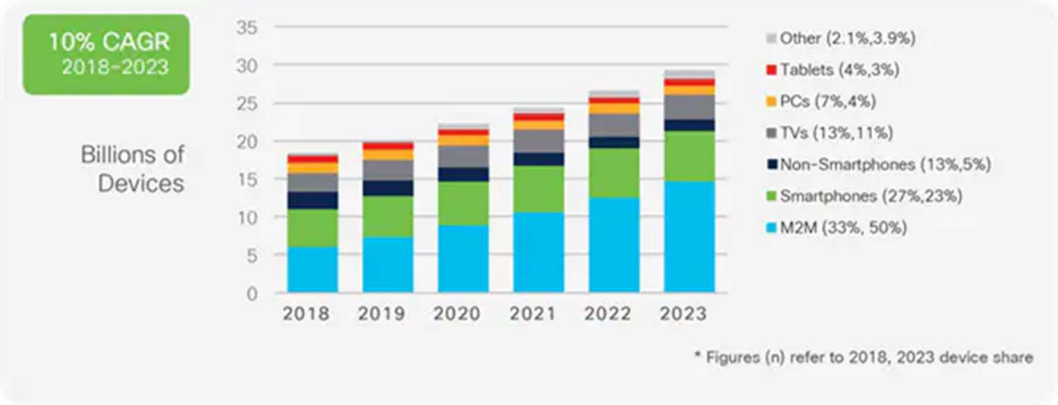

Figura 9 - Stime riguardo al numero di dispositivi connessi[22]

Una delle cause di questo disaccoppiamento tra l'ammontare di utenti fisici che utilizzano Internet e dispositivi effettivamente connessi è individuata dal fenomeno dell'Internet of Things (IoT). Anche chiamato in italiano Internet delle Cose, questo

[20] Matteo Starri di We Are Social, Digital 2021: i dati globali, wearesocial.com, 2021
[21] Cisco, Annual Internet Report White Paper c11-741490, cisco.com, 2021
[22] Cisco, 2018-2023 Device Share, cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

neologismo è stato utilizzato per la prima volta da Kevin Ashton[23], il quale ne fornì inoltre una iniziale definizione. Il concetto può essere visto inizialmente come un'espansione dell'utilizzo di Internet nell'ambito di luoghi ed oggetti concreti. Se in precedenza Internet era concepito come la via per mettere in comunicazione e far interagire gli esseri umani tra di loro, o per fare interfacciare gli individui con delle macchine su cui erano installati dei programmi al fine di erogare determinati servizi, in questa moderna concezione di Internet of Things la componente "dialogante" umana viene a mancare. Le cose, infatti, intese come tutti quei dispositivi il cui compito nelle loro funzionalità di base esulava fino a quel punto dall'obbiettivo di una connessione in rete, incominciano a dotarsi di componenti che permettono loro di effettuare una comunicazione autonoma con gli altri oggetti dotati di tale capacità, senza un diretto ed esplicito comando da parte di una persona. In questo modo i device acquisiscono la capacità di rendersi identificabili e maturano una "intelligenza" dovuta alla possibilità di effettuare comunicazioni tra di loro fornendo le proprie informazioni e ricevendo dati semplici o aggregati da parte delle altre entità[24].

Figura 10 - Rete di dispositivi IoT[25]

[23] Kevin Ashton, RFIDjournal-That Internet of Things Thing, 2009
[24] Paolo Magrassi, Dizionario dell'economia digitale, Il Sole 24Ore, 2002
[25] jeferrb, rete-iot-internet-delle-cose, pixabay.com

Non si deve tuttavia fraintendere e considerare questa possibilità di dialogo tra dispositivi come un qualcosa di estraneo agli esseri umani e ai servizi loro offerti, ma una modalità nuova con cui questa gamma di compiti può essere ampliata. A titolo esemplificativo, si può pensare ad una lampada da comodino: il suo utilizzo principale rimane sempre quello di illuminare facendo luce grazie alla lampadina o il led con cui è equipaggiata, ma in una versione in cui questa integri un dispositivo IoT può svolgere funzioni più diversificate rispetto alle sole di accensione e spegnimento con un interruttore, quale ad esempio regolare il suo stato di funzionamento rispetto all'ora del giorno comunicatole da un altro dispositivo di rete. I campi di applicazione dell'Internet of Things sono stati molteplici fin dall'inizio del nuovo millennio e di anno in anno la lista che li elenca si popola di nuove voci. Di seguito verranno riportati alcuni degli ambiti applicativi più diffusi ed in voga nei diversi contesti.

- Domotica e Smart home: l'Internet delle Cose si inserisce anche nel più ampio quadro dell'automazione domestica. Spesso, in questi casi, i dispositivi IoT sono collegati e orchestrati attraverso l'impiego dei cosiddetti hub, ossia degli oggetti con una componente computazionale più ricca e potente in grado di gestire logiche e moli di dati più intense rispetto a quelle per certi versi più "banali" o elementari dei device IoT che si collegano ad esso. Inoltre, prerogativa di molti di questi oggetti intelligenti è la possibilità di interagire con essi attraverso un controllo vocale o mediante la programmazione, con l'ausilio di opportune interfacce utente, di alcune routine di funzionamento in base ad orari o eventi di contesto. Questi tipi di dispositivi IoT, come ad esempio lampadine, televisori e stufette elettriche, vanno a porre le basi per il concetto di smart home (casa intelligente).

Figura 11 - Foto di due hub della famiglia Amazon Echo[26]

Il beneficio che l'utilizzo di questi dispositivi comporta, oltre a quello più immediato individuato dalle maggiori e più avanzate meccaniche di funzionamento e servizio che possono offrire rispetto alle controparti non IoT, si manifesta anche nel più lungo periodo in termini di risparmi energetici dovuti ad esempio allo spegnimento automatico degli stessi qualora la situazione suggerisse la cessazione della necessità di funzionamento.

- Medicina e Sanità: anche definita come Internet of Medical Things (IoMT), è una sfera delle applicazioni IoT per scopi relativi alle cure e alla salute, per il monitoraggio delle funzioni patologiche e fisiologiche dei pazienti e per la raccolta e le analisi nell'ambito di ricerche mediche. Alcuni ospedali e cliniche all'avanguardia, infatti, sono diventati implementazioni del concetto di Smart Healthcare[27], in cui i macchinari per le risorse sanitarie ed i servizi clinici sono stati resi intelligenti e connessi in modo da creare un sistema ospedaliero digitalizzato e collegato. In questo modo l'intero personale della struttura, con le opportune autorizzazioni di ruolo e privacy, è in grado di consultare più rapidamente ed efficacemente le informazioni

[26] Pocket-lint, image1 in "Cos'è Alexa e cosa può fare Amazon Echo?", pocket-lint.com
[27] Nilanjan Dey, Bhatt, Satapathy, Hassanien, Ashour, Internet of things and big data analytics toward next-generation intelligence, Springer, 2018

relative a pazienti come parametri fisici e referti, ed a tutte le attrezzature smart dell'ospedale come i macchinari per gli esami.

- Industria Manufatturiera: in questo campo numerosi elementi della catena produttiva possono implementare dispositivi IoT, a partire dai sensori a bordo dei macchinari fino alle attrezzature utilizzate dagli operatori di macchina. Questo permette di avere più controllo sull'inventario dei prodotti e delle materie prime così come della loro logistica, una maggiore capacità di gestione di situazioni e processi produttivi e una possibilità più concreta di individuazione di criticità e una rapidità di intervento associata. Possono inoltre venire effettuate ed immagazzinate misure di ogni genere, implementati vari meccanismi di monitoraggio e sicurezza sul posto di lavoro così come numerose strategie di predictive maintenance e ottimizzazioni energetiche. È proprio questa la tipologia di dispositivi Internet of Things utilizzati nel progetto PREMA, con l'installazione a bordo macchina di sensori IoT intelligenti.

Solitamente, accade che per qualsiasi tecnologia legata all'utilizzo di Internet nascano, in coppia con le opportunità e le innovazioni che questa si dimostra in grado di fornire, alcune preoccupazioni e difficoltà inedite dovute anch'esse alle novità introdotte dalla stessa tecnologia. L'Internet of Things in questo senso non costituisce un'eccezione, ma ha infatti portato a sua volta alla luce problematiche collegate al suo utilizzo.

Un primo esempio di queste tematiche che destano preoccupazione è individuato dalla questione riguardante la privacy. Tutti i prodotti e servizi legati alle soluzioni IoT implementate in maniera diffusa nel corso degli ultimi anni fanno tipicamente impiego di raccolte massive di dati e informazioni relative all'ambiente in cui operano e agli utilizzatori che interagiscono con esse[28]. Uno dei propositi dei dispositivi IoT, infatti, al fine di essere in grado di fornire i servizi da loro offerti in modo tale da aumentarne l'efficacia e il grado di adeguatezza alle situazioni

[28] Charith Perera, Ranjan, Wang, Khan, Zomaya, Big Data Privacy in the Internet of Things Era, IEEE IT Professional Magazine: Special Issue Internet of Anything, 2015

specifiche, si concretizza nel tentativo di imparare da ciò che li circonda. Questi dati e informazioni che vengono assimilati devo ovviamente venire immagazzinati per fare in modo che la conoscenza da essi ricavabile possa essere utilizzata non soltanto nell'istante in cui gli stessi sono registrati, ma sia possibile sfruttarla anche in futuro ed eventualmente con tecniche che rendano questo apprendimento additivo ed incrementale. Per sopperire a questa necessità di immagazzinamento sono disponibili due strategie principali. La prima consiste nel dotare il dispositivo IoT di componentistiche che implementino una memoria di massa in modo da consentire la realizzazione di un database locale interno al sistema, interrogabile quindi senza la necessità di sfruttare una connessione esterna. In alternativa, si può ricorrere all'utilizzo di infrastrutture di archiviazione Cloud raggiungibili in rete. In quest'ultima soluzione i dati acquisiti invece di essere conservati direttamente sul dispositivo vengono caricati online e accumulati all'interno di periferiche di memorizzazione remote.

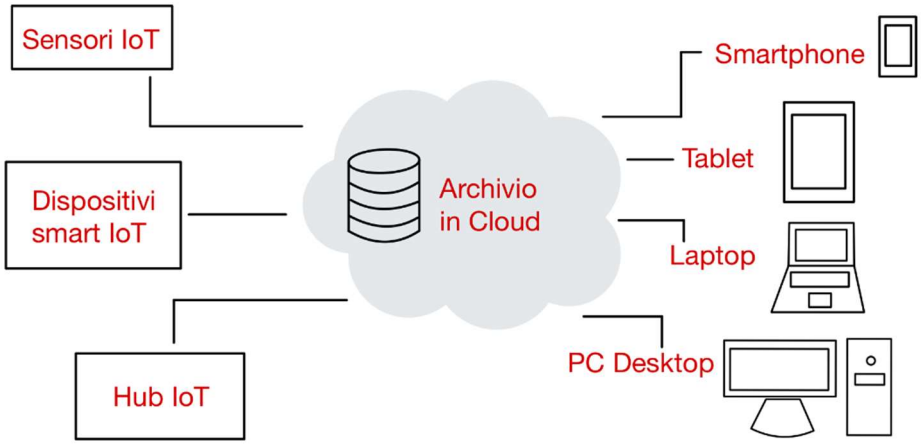

Figura 12 - Schema di archivio in Cloud

Nell'accezione relativa al settore IT (Information Technology) per Cloud[29], o in lingua italiana nuvola informatica, si intende un modello di riferimento in cui le funzionalità offerte ed i servizi messi a disposizione da un fornitore vengano devoluti, sfruttando una connessione in rete, dietro esplicita ed eventualmente

[29] Peter Mell, Grance, The NIST Definition of Cloud Computing, National Institute of Standards and Technology, 2011

incrementale richiesta mossa da un consumatore, sfruttando una collezione di mezzi e infrastrutture fabbricati in precedenza e organizzati secondo un sistema distribuito che possono essere in larga misura configurati in base alle necessità del caso e messi a disposizione per lo sfruttamento degli stessi da remoto. Questa seconda soluzione è spesso preferita proprio perché rende più facile ottenere il reale valore della raccolta dei dati, che si configura attraverso l'aggregazione su vasta scala ed il processamento ad ampio spettro delle informazioni acquisite al fine di estrarre una nuova e più complessa conoscenza. Ed è proprio l'impiego di certe procedure che può comportare alcune problematiche e questioni in termini di privacy. In una ricerca condotta a proposito del suddetto argomento, sono state evidenziate tre principali tematiche che possono destare preoccupazione e alle quali si rende necessario prestare particolare attenzione:

- anonimato: il timore e, in alcuni casi, la certezza che si prova nell'osservare come le informazioni assimilate vengono trasmesse è dovuto all'insufficiente cautela nel disaccoppiare il dato dall'utente a cui si riferisce. È auspicabile che in un prossimo avvenire sia possibile utilizzare tecnologie che non permettano il tracciamento di profili troppo particolari propri degli utilizzatori;
- libertà di scelta: le norme e regolamentazioni erogate in materia di tutela della privacy devono essere studiate in modo da favorire le preferenze libere ed individuali dei singoli;
- consenso: deve essere implementato un meccanismo attraverso il quale l'utilizzatore sia messo in condizione di manifestare un consenso informato sulla raccolta e sugli utilizzi relativi ai propri dati personali tracciati.

Un altro ambito connesso all'Internet of Things che sta sollevando qualche controversia ed è motivo di apprensione è costituito dalle vulnerabilità e relativi meccanismi di difesa in termini di sicurezza che l'utilizzo di dispositivi IoT potrebbe ed in certi casi ha già fatto nascere. Nel contesto individuato dalla connessione alla rete da parte di cose e persone, se la privacy può essere definita come il livello di protezione delle proprie informazioni di cui un individuo è dotato

quando utilizza Internet[30], il concetto di sicurezza spazia in maniera più ampia e articolata. In particolare, la privacy copre solamente l'ammontare di sicurezza online disponibile per quanto concerne i propri dati finanziari e relativi all'identità, le comunicazioni effettuate e le singole abitudini e preferenze e le minacce ad essa associate si concretizzano in numerose classi, che spaziano dal rischio di incorrere in truffe organizzate e messe in atto al fine di estorcere beni e denaro fino ai furti di identità veri e propri. Per quanto riguarda la sicurezza sulla rete, anche chiamata Internet Security, essa consiste in tutto il panorama di tattiche e strategie messe in atto per ricavare protezione durante il compimento di qualsiasi attività e operazioni che hanno luogo durante una connessione[31]. In quest'ottica negli ultimi anni è stata sollevata la questione anche per quanto riguarda l'adozione sempre più diffusa di implementazioni IoT nei suoi molteplici campi di applicazione.

Uno dei timori mossi nello scorso decennio, per l'appunto, è dato dalla considerazione che se il campo dell'Internet of Things si stia ampliando in maniera piuttosto rapida, altrettanto non si può affermare per quanto concerne le considerazioni da mettere in atto riguardanti le specifiche tematiche di sicurezza associate e le nuove regolamentazioni da varare che potrebbero risultare essenziali[32]. Quando si parla di sicurezza, infatti, è bene non soltanto conoscere quali siano le vulnerabilità e gli attacchi che possono essere portati in atto sfruttandole, ma anche stabilire quali siano le responsabilità che tutte le entità coinvolte nell'intero ciclo di implementazione di meccaniche IoT devono avere, a partire dalla produzione conforme a determinati standard del dispositivo da parte dei produttori fino all'utilizzo consapevole e corretto da parte degli utilizzatori. A titolo esemplificativo, si può considerare quanto accaduto nel 2013 nei confronti di un'azienda manufatturiera di dispositivi connessi nell'ambito IoT. La Federal Trade Commission, un'agenzia federale del governo degli Stati Uniti che si occupa della promozione della tutela dei consumatori, ha portato avanti un'azione legale[33] nei

[30] Winston & Strawn LLP, What is the Definition of Online Privacy?, winston.com/en/legal-glossary
[31] McAfee Enterprise, What Is Internet Security?, mcafee.com
[32] Chris Clearfield, Why The FTC Can't Regulate The Internet Of Things, Forbes, 2013
[33] Federal Trade Commission, Marketer of Internet-Connected Home Security Video Cameras Settles FTC Charges It Failed to Protect Consumers' Privacy, ftc.gov, 2013

confronti della TRENDnet, un'impresa produttrice, tra le altre cose, di videocamere connesse alla rete.

Figura 13 - Fotografia di una TRENDnet Wireless Cloud Camera[34]

Secondo l'accusa, l'azienda avrebbe fallito nell'ottemperare alle proprie affermazioni rivolte agli utenti relativamente all'efficacia delle misure di sicurezza adottate sui prodotti in questione. Infatti, nel 2012 un gruppo di hacker ha trovato e sfruttato una falla che ha permesso loro di individuare di contenuti video privati degli utenti, che erano rimasti esposti senza che questi ultimi ne fossero a conoscenza. La vicenda si è conclusa con l'imposizione all'azienda di un programma ventennale relativo alle conformità in termini di sicurezza.

Oltre a queste situazioni, in cui l'attacco all'apparecchiatura IoT è compiuto al fine di sfruttare le sue vulnerabilità per ottenere informazioni e materiali multimediali personali, ce ne sono altre in cui il dispositivo IoT stesso è uno dei vettori attraverso cui l'attacco è portato a termine. È il caso dei Distributed Denial-of-Service (DDoS, in italiano negazione del servizio distribuito), un tipo particolare di attacco in cui il server o terminale designato come bersaglio viene inondato da un numero spropositato di richieste da parte di una cosiddetta botnet, un insieme di molteplici dispositivi spesso non effettivamente coinvolti in modo intenzionale nell'intervento ostile con un comando impartito dal legittimo utilizzatore, volte a saturare le risorse della vittima che quindi non è più in grado di funzionare efficacemente e risulta

[34] TRENDnet, TRENDnet TV-IP751WC Wireless Cloud Camera photo, trendnet.com

quindi impossibilitato a servire le richieste genuine che possono venirle consegnate. In un contesto come quello dell'Internet of Things, costituito infatti da un numero molto nutrito di dispositivi spesso venduti a basso costo e in alcuni casi trascurando alcuni importanti aspetti in termini di meccanismi di sicurezza, è possibile sfruttare gli apparecchi connessi e infettarli con un programma malevolo al fine di organizzare la botnet che verrà utilizzata in fase di messa in atto dell'attacco. Un esempio di tale dinamica è costituto dalla botnet IoT soprannominata dark_nexus, che utilizza una tattica di DDoS che traveste il traffico malevolo in un apparentemente innocuo traffico generato da un browser[35]. La prima implementazione individuata di questa temibile minaccia risale alla fine del 2019 e nei mesi successivi è stata ampliata ed aggiornata a nuove versioni.
In ultima analisi un ulteriore elemento da tenere attentamente in considerazione è costituito dalla presenza sul mercato di veicoli in cui gli svariati dispositivi elettronici di cui sono equipaggiati, come ad esempio quelli coinvolti nell'impianto frenante e nello sblocco delle serrature, sono in qualche caso connessi tra di loro e ad Internet[36]. È lampante come un'eventuale introduzione non autorizzata in questi dispositivi IoT e delle successive operazioni di manomissione e controllo svolte sugli stessi possano portare, in situazioni delicate come quella costituite da veicoli in movimento, ad effetti drammatici e potenzialmente in grado di mettere a rischio l'incolumità delle persone coinvolte.

Un concetto che viene spesso accompagnato nelle discussioni relative alle tematiche trattate in precedenza, quali Cloud, Predictive Maintenance e Internet of Things, è quello individuato dalla Big Data Analytics. Il termine, ultimamente sempre più in voga e utilizzato fin dalla fine dello scorso millennio, è usato per definire il vasto insiemi di tecniche sofisticate e spesso complicati processi di esaminazione di moli immense di dati al fine di svelare nuove conoscenze, quali correlazioni tra eventi ed entità, pattern comportamentali nascosti, preferenze dei consumatori ed attuali tendenze del mercato, che possono aiutare le aziende a

[35] Bitdefender Investigations and Forensics Unit, New dark_nexus IoT Botnet Puts Others to Shame, bitdefender.com, 2020
[36] Andy Greenberg, Hackers Remotely Kill a Jeep on the Highway—With Me in It, wired.com, 2015

compiere scelte consapevoli in fase di decisione delle strategie di business da mettere in atto[37]. A sua volta, infatti, il termine Big Data è utilizzato per indicare una collezione di dati, intesi come caratteri, quantità, simboli o segnali che possono essere immagazzinati e trasmessi da un calcolatore, enorme nei volumi e comunque ancora in continua crescita esponenziale col passare del tempo[38].

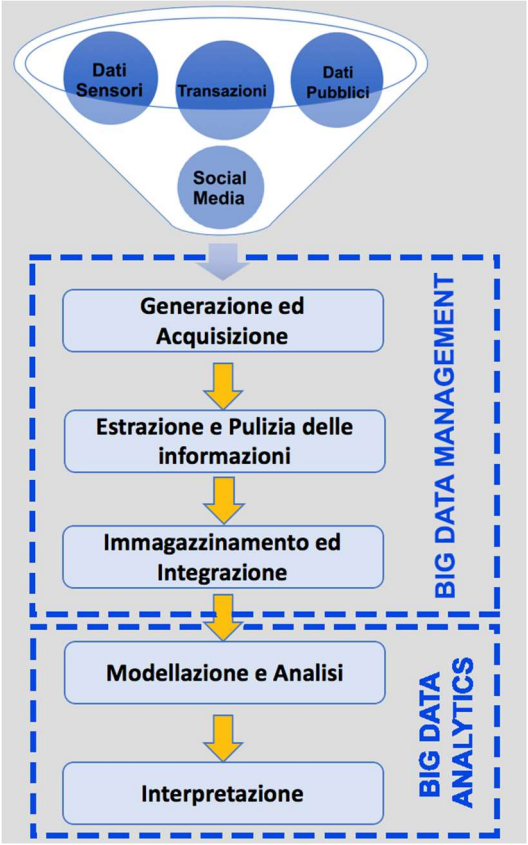

Figura 14 – Ciclo di vita dei Big Data[39]

È una particolare categoria di dati la cui dimensione e complessità la rendono impossibile da amministrare e archiviare efficacemente attraverso gli strumenti tradizionali di gestione dei dati. Una volta compresa la loro definizione, è possibile notare come i Big Data possano essere categorizzati in tre classi a seconda del loro tipo.

[37] Wesley Chai, Labbe, Stedman, The ultimate guide to big data for businesses – big data analytics definition, searchbusinessanalytics.techtarget.com, 2021
[38] David Taylor, What is Big Data? Introduction, Types, Characteristics, Examples, guru99.com, 2021
[39] Giuseppe.sepe, The main processes that make up the life cycle of the Big Data, 2017

- Structured: anche definiti strutturati, sono quei dati che possono essere in qualche modo immagazzinati, acceduti e processati sotto forma di un formato fisso. Ad esempio, le entità che con i propri campi costituiscono le tabelle e le relative colonne all'interno di un database costituiscono un tipo di Big Data strutturato.

- Unstructured: in italiano non strutturati, è quella tipologia in cui non è possibile definire un formato o una struttura standard. Il fatto che la mole di dati sia considerevole, in aggiunta a questa condizione, comporta numerose sfide in termini del processamento atto al fine di estrarne un qualche genere di valore. Un esempio tipico di dato non struttutrato è quello costituito da una fonte eterogenea contentente un misto di contenuti multimediali come video, righe di testo e immagini, come nel caso della pagina restituita in seguito ad una ricerca effettuata su Google.

- Semi-structured: chiamati anche semi-strutturati, sono i dati che contengono in qualche misura entrambe le tipologie, structured e unstructured. Possono essere visti come dati per cui è possibile intravedere una struttura di base che però non è definibile, ad esempio, come tabella di un database relazionale. Esempi di dati semi-strutturati sono rappresentati dai file di tipo XML (eXtensible Markup Language).

A corredo di quanto detto, per aiutare a meglio comprendere e definire i Big Data è stato comune fin dai primi anni di utilizzo del termine associare ad esso il cosiddetto modello delle "3 V", una trattazione in cui sono listate le tre parole chiave (che iniziano tutte con la lettera V, da cui il nome) da avere in mente quando si sta cercando di intendere fino in fondo il concetto in questione[40]. Nello specifico, queste sono elencate nel seguito, insieme ad una breve spiegazione del significato di ognuna di esse.[41]

[40] Gartner, Gartner Information Technology Glossary: Big Data, gartner.com
[41] The unbelievable Machine Company, What is Big Data? – A definition with five Vs, blog.unbelievable-machine.com

- Volume: indica la principale caratteristica dei Big Data, ossia il fatto di essere composti da un vasto ammontare di dati generati spesso su base giornaliera, talmente grande da non poter essere immagazzinato e analizzato utilizzando i metodi di processamento convenzionali.

- Variety: o Varietà, si riferisce all'elevato grado di eterogeneità sia dei tipi di dato che delle sorgenti degli stessi. Nonostante il fatto che l'80% dei dati in tutto il mondo sia appartenente alla tipologia unstructured, attraverso l'utilizzo di algoritmi Big Data è possibile ordinare e organizzare tali dati in modo da renderli idonei ad un processamento.

- Velocity: o Velocità, si riferisce alla rapidità con cui i dati vengono generati, acquisiti, processati ed analizzati. Al giorno d'oggi è spesso possibile che tutto ciò avvenga non più in intere giornate, ma in frazioni di secondo, al punto che queste operazioni vengono classificate come effettuate in tempo reale.

Sebbene questo modello delle 3 V sia ancora oggi assolutamente valido e corretto, si è rivelato nel tempo doveroso aggiungere due ulteriori termini cruciali, che consentano una trattazione più completa.

- Validity/Veracity: in italiano Validità/Veridicità, indicano il livello di garanzia della qualità del dato, così come la sua autenticità e conseguente credibilità. Visto che spesso il fattore Volume si accompagna ad un limitato grado di genuinità, lavorare coi Big Data significa dover avere a che fare con tutti i gradi ranghi di bontà.

- Value: ossia valore, sta ad indicare il valore aggiunto di cui le aziende beneficiano interessandosi a certi dati. Molte compagnie hanno ad oggi investito risorse e denaro per erigere le proprie piattaforme di dati, riempire gli specifici bacini di informazioni e costruire le infrastrutture adeguate e relative. È quindi necessario che tale investimento si traduca in un bene inedito è utilizzabile per generare guadagni.

La Big Data Analytics si inserisce all'interno delle tattiche di Business Intelligence, intesa come lo sfruttamento di programmi e servizi in grado di trasformare le informazioni in possesso in intenzioni perseguibili dall'azienda nella sua organizzazione strategica di decisioni imprenditoriali[42], con la particolarità di utilizzare i Big Data come base. Grazie all'impiego della Big Data Analytics, infatti, i manager e gli amministratori aziendali hanno la possibilità di dotarsi di nuove conoscenze che li supportino nel compiere le decisioni più corrette, in modo da ridurre al minimo le dispersioni di risorse e raggiungere i massimi guadagni ottenibili. Tuttavia, per conseguire tali risultati, un'azienda deve dotarsi degli strumenti adatti non sempre già presenti nella propria organizzazione produttiva, visto che come è già stato assodato l'attitudine dei Big Data di interesse è solitamente quella di appartenere alla classe di dati non strutturati, per cui i tradizionali approcci non si rivelerebbero efficaci.

In quest'ottica, si sono rivelate di particolare interesse negli ultimi anni alcune innovative tecnologie, da subito in rapida ascesa. Tra queste, è impossibile non citare MapReduce, un'architettura logica di supporto alla creazione di software registrata e messa disponibile da Google, al fine di rendere fruibile un calcolo computazionale ripartito su moli massicce di informazioni, effettuato in datacenter composti da svariati server[43]. Le funzioni di libreria relative sono disponibili nei numerosi linguaggi di programmazione più diffusi, tra cui ad esempio Java, C++ e Python. Sull'onda di questa architettura si è preso un posto all'interno del panorama globale relativo alla Big Data Analytics un prodotto della Apache Software Foundation: Hadoop. Con la sua prima versione che risale al 2006, Apache Hadoop è un progetto che sviluppa software open-source per una computazione distribuita affidabile e scalabile, ideato e fondato da Doug Cutting in collaborazione con Mike Cafarella[44]. La libreria di programmazione costituita da Hadoop è un framework che consente il processamento distribuito di insiemi di dati estesi attraverso cluster informatici di computer, utilizzando semplici ma profondamente efficaci modelli di

[42] Josh Fruhlinger, Pratt, What is business intelligence? Transforming data into business insights, cio.com, 2019

[43] Colby Ranger, Raghuraman, Penmetsa, Bradski, Kozyrakis, Evaluating MapReduce for Multi-core and Multiprocessor Systems, *IEEE 13th International Symposium on High Performance Computer Architecture*, ieeexplore.ieee.org, 2007

[44] Redazione LineaEDP, 10 anni di Apache Hadoop, lineaedp.it, 2016

programmazione[45]. È progettato per scalare in maniera incrementale partendo da singoli server fino a migliaia di macchine, ognuna in grado di offrire una computazione e un'archiviazione locali. Invece di affidarsi alla bontà dell'hardware, ossia alle prestazioni in termini di qualità delle componentistiche dei vari calcolatori, per fornire un alto grado di disponibilità, la libreria stessa è progettata per intercettare e gestire difetti o fallimenti sullo strato applicativo, mettendo quindi a disposizione un servizio ad alta disponibilità posizionato in cima ad una base costituita da cluster di computer, ognuno dei quali potrebbe essere prono a errori e malfunzionamenti. Il suo sviluppo è in continua progressione e al momento della stesura di questo elaborato la sua ultima versione stabile è costituita dalla 3.3.1, la cui documentazione è stata aggiornata per l'ultima volta sul sito di Apache a Giugno 2021.

Nel 2012 la ditta di ricerca in ambiti di Information Technology canadese IDC Canada (International Data Corporation) ha rilasciato il suo report annuale riguardante le sue predizioni per l'anno a venire e, come in quel momento ci si poteva aspettare, a prendere posto tra tematiche come la sicurezza e il Cloud figurarono proprio i Big Data e la relativa Analytics[46]. Con questo fatto in mente, è stata stilata una lista che, al fine di supportare le imprese nell'adottare strategie di Big Data Analytics che potessero portare valore aggiunto nelle proprie aziende, elencasse sette tecniche già largamente usate e che avrebbero costituito argomento di interesse nel corso dei dodici mesi successivi.[47] Queste verranno ora illustrate con, per ognuna, una loro spiegazione e qualche caso tipico di utilizzo.

- Association rule learning: in italiano apprendimento delle regole di associazione, è un metodo per cercare di tirare fuori interessanti correlazioni tra le variabili contenute in estese basi di dati. È stato usato in primo luogo da parte delle maggiori catene di supermercati, con l'intento di trovare relazioni tra i prodotti acquistati o meno, sfruttando i dati ottenuti dai loro

[45] The Apache Software Foundation, Apache Hadoop, hadoop.apache.org
[46] Ryan B. Patrick, The three things IDC Canada predicts will matter in 2013, expertIP, blog.allstream.com, 2012
[47] Debbie Stephenson, 7 Big Data Techniques That Create Business Value, TouchPoint by Firmex, firmex.com

sistemi POS (Point Of Sale). Questo metodo viene usato per fornire supporto in differenti situazioni decisionali, come ad esempio nella scelta relativa a quali prodotti mettere l'uno vicino all'altro per incrementarne le vendite o nel monitoraggio delle linee dei log di sistema per individuare accessi non autorizzati o attività malevole.

- Classification tree analysis: l'analisi degli alberi di classificazione, consiste, sfruttando un insieme di addestramento di oggetti correttamente classificati in precedenza, nell'identificare le categorie di appartenenza proprie della nuova entità osservata. Un uso tipico di tale tecnica coincide con l'assegnazione automatica di un documento ad una certa categoria, così come con la generazione di profili tipici di studenti che seguono dei corsi online.

- Genetic algorithms: anche chiamati algoritmi genetici, si ispirano al metodo di funzionamento dell'evoluzione, ossia attraverso meccanismi quali l'ereditarietà, le mutazioni e la selezione naturale. Questi procedimenti sono utilizzati per maturare soluzioni utili a problemi che necessitano una ottimizzazione, per i quali non esistono o non si conoscono algoritmi con complessità polinomiale. Sono usati per esempio nella generazione delle combinazioni ottimali di materiali e pratiche ingegneristiche richiesti per lo sviluppo di automobili dagli efficienti consumi di carburante.

- Machine Learning: uno dei termini più conosciuti e utilizzati, talvolta anche a sproposito, include genericamente del software che "apprende" da dei dati. Conferisce ad un calcolatore l'abilità di apprendere senza che la conoscenza sia esplicitamente programmata e si focalizza sulla generazione di previsioni basate su delle proprietà note imparate da degli insiemi di dati di addestramento, spesso definiti training set. È stato utilizzato per distinguere i messaggi di spam da quelli genuini nella messaggistica via e-mail, così come viene spesso sfruttato per imparare i gusti e le preferenze di un utente per fornirgli contenuti consigliati sulla base degli stessi, ad esempio dalle piattaforme streaming di contenuti video on demand. Tecniche ed algoritmi

di machine learning sono implementati all'interno del progetto PREMA sia all'interno dell'Edge che del Backend.

- Regression Analysis: in italiano analisi della regressione, ad un livello di base, include la manipolazione di una qualche variabile indipendente per osservare come questo influenzi una certa variabile dipendente. Descrive in che modo il valore della variabile dipendente cambi quando la variabile indipendente è fatta variare. Funziona al meglio con quantità continue come il peso o la velocità. È utilizzata per determinare come il livello di soddisfazione del cliente influisca sulla sua fedeltà, o quanto il vicinato circostante influenzi il prezzo di vendita degli immobili.

- Sentiment Analysis: aiuta i ricercatori a determinare le sensazioni e gli stati d'animo di oratori o scrittori rispetto ad un certo argomento. Viene sfruttata per personalizzare servizi ed incentivi per andare incontro a ciò che la clientela sta effettivamente chiedendo, così come a determinare ciò che i clienti pensano veramente basandosi sulle opinioni e recensioni sui social media.

- Social Network Analysis: definita in italiano analisi delle reti sociali, è una tecnica usata in prima battuta dall'industria delle telecomunicazioni, per essere poi in seguito rapidamente adottata anche dai sociologi nello studio delle relazioni interpersonali. Viene adesso applicata in svariati campi per analizzare le relazioni tra le persone e tra le attività commerciali. I nodi della rete rappresentano gli individui, mentre gli archi simboleggiano le relazioni tra questi individui. È utilizzata per trovare l'importanza o influenza che un individuo particolare ha all'interno di un gruppo, così come per comprendere la struttura sociale di una base di clientela.

3 Applicazioni per dispositivi mobili

Nei capitoli precedenti è stato indicato come uno dei componenti relativi al frontend del progetto PREMA, ossia la parte costitutiva dall'architettura progettuale adibita alla interazione con l'utente in termini di ricezione di comandi e presentazione dei risultati, fosse rappresentato da applicazioni mobili dedicate che permettessero, rispetto agli altri elementi costitutivi del frontend, un'interazione coi dati più agile ed in tempo reale. Considerato inoltre che il punto focale della tesi si concentra proprio su questo ambito progettuale, si rende necessario fornire in questo elaborato una trattazione relativa alle applicazioni mobili, che sia in grado di illustrarne una panoramica comprensiva sia dell'attuale stato dell'arte di tale campo, che delle pratiche comuni e linee guida da osservare nella loro realizzazione. Nel seguito verranno quindi in un secondo momento descritti i principali attori operanti nel campo delle applicazioni mobili e gli strumenti utilizzati per il loro sviluppo, con particolare riferimento all'ecosistema Android e agli aspetti ad esso collegati.

3.1 Introduzione alle mobile app

Un'applicazione mobile, anche chiamata mobile application o più semplicemente e comunemente app o app mobile, è una particolare tipologia di programma, il cui codice è progettato per essere eseguito su un dispositivo mobile, tipicamente identificato da uno smartphone o un tablet[48]. Le funzioni che le applicazioni di tipo mobile forniscono sono in molti casi simili a quelle offerte dai programmi eseguiti sui personal computer, tuttavia i dispositivi su cui le prime sono installate sono tipicamente costituiti da componentistiche elettroniche più modeste in termini di performance, in quanto montate su apparecchi di dimensioni più contenute e perché sfruttano, invece di una costante alimentazione elettrica esterna, delle batterie integrate, a loro volta in un formato più ridotto. Proprio per sopperire ai limiti imposti da queste prestazioni hardware più moderate e per ottimizzare il più possibile i consumi energetici, rispetto ai programmi dell'ambito desktop le applicazioni mobili sono caratterizzate da una progettazione e implementazione più snella e da un impiego più sobrio delle risorse a disposizione.

[48] Techopedia, Mobile Application (Mobile App), techopedia.com

Figura 15 - Dispositivi per le applicazioni mobili[49]

In quest'ottica di attenzione ai consumi di risorse, nelle app mobili è tipicamente rilevante escludere tutte le funzioni e componenti non necessarie al corretto funzionamento del programma. Questo non significa necessariamente che l'applicazione mobile, rispetto alla controparte desktop, sia scarna o limitata dal punto di vista del numero o della completezza dei servizi che essa mette a disposizione, ma soltanto che viene posto particolare interesse all'eliminazione del superfluo. Al contrario, è possibile vedere nelle app mobili delle opportunità originale ed uniche rispetto ai programmi utilizzabili su dei computer, come ad esempio la possibilità di utilizzare l'applicazione in qualsiasi momento o luogo e non necessariamente a casa o in ufficio, o quella di fornire una presentazione del servizio preso in considerazione inedita e in grado di far ottenere una esperienza utente più elastica e funzionale, in modo da renderla più completa[50]. Dal punto di vista tecnico e realizzativo, è possibile suddividere le applicazioni mobili nelle seguenti categorie[51]:

- Native: sono applicazioni progettate e costruite in maniera esclusiva e dedicata per essere eseguite su un particolare dispositivo o famiglia di dispositivi. Il loro sviluppo è quindi subordinato al sistema operativo

[49] geralt, mobile-smartphone-tablet-bianca, pixabay.com
[50] dwb Web Agency, Vantaggi e svantaggi delle app mobile, dwb.it, 2018
[51] Education-wiki, Tipi di applicazioni mobili, it.education-wiki.com

installato su tali dispositivi. Se ciò impone alcuni vincoli architetturali e di progetto, come l'impossibilità per l'applicazione di funzionare su dispositivi diversi rispetto a quelli considerati in fase di progettazione, fornisce al programma l'opportunità di sfruttare direttamente le periferiche hardware di cui sono formati, come ad esempio schermi, microfoni o fotocamere, con impatti evidenti sulle funzionalità e le prestazioni offerte. Di contro, uno svantaggio di questa categoria di applicazioni è che il codice di cui sono costituite deve essere duplicato per ognuna delle piattaforme a cui sono destinate, con i linguaggi e le librerie dedicati. Inoltre, va posta particolare attenzione alla manutenzione delle stesse, in quanto eventuali aggiornamenti del sistema operativo potrebbero comportare perdite di funzionalità o problematiche inattese, da risolvere con corrispondenti aggiornamenti delle applicazioni stesse. Tutti i file contenenti le risorse e le logiche di funzionamento dell'applicazione infine devono essere memorizzati localmente sul dispositivo, fattore di cui va tenuto conto in relazione allo spazio di archiviazione disponibile sullo stesso.

- Web: è una classe di applicazioni mobili in cui l'applicazione non è effettivamente installata sul dispositivo mobile, ma viene visualizzata attraverso dei collegamenti a delle pagine web raggiunte tramite i browser di cui è dotato il device. Essendo in linea di principio un sito web visualizzato sotto forma di applicazione, il rendering dei contenuti viene costruito in base alle dimensioni e proporzioni dello schermo in questione. In questo modo, rispetto ad una applicazione mobile nativa, una app web funziona su qualsiasi dispositivo mobile, a patto che sia dotato di un browser, indipendentemente dal sistema operativo utilizzato o dalla famiglia di apparecchi a cui il dispositivo appartiene. Questo rende le applicazioni mobili di tipo web semplici da manutenere e in fase di sviluppo il codice da scrivere è unico. Per contro, ponendole sempre a confronto con le mobile app native, queste hanno l'importante limitazione di non poter operare in assenza di una connessione ad Internet, in quanto il codice e la logica di funzionamento invece di risiedere localmente sul dispositivo sono

memorizzati in un server remoto. Contestualmente, le prestazioni dell'applicazione sono fortemente determinate dalla qualità e dalla velocità della connessione, invece che dalla potenza del dispositivo in sé. A seconda delle situazioni, questo fattore può rivelarsi un punto di forza considerevole o un collo di bottiglia non trascurabile.

- Ibride: le applicazioni che fanno parte di questa categoria sono sviluppate in maniera particolare. Esse, infatti, sono costituite da un misto di elementi propri delle applicazioni native in integrazione con componenti basate su tecnologie web. In questo modo è possibile tentare di sfruttare al meglio i vantaggi propri di entrambe le classi di appartenenza, cercando di limitarne e mitigarne invece le criticità. Ad esempio, è possibile lo sviluppo di applicazioni in grado allo stesso tempo di sfruttare appieno le periferiche hardware di interazione con l'utente e delegare delle elaborazioni complesse a dei server web remoti da cui ricavare i risultati perché siano fruibili. Inoltre, sebbene una parte di codice sia ancora personalizzata per i differenti sistemi operativi, rimane una porzione comune a tutti i dispositivi su cui si vuole installare l'applicazione, esente quindi dalla necessità di venire duplicata. È tuttavia fondamentale notare come esista, anche in questo caso, la dipendenza da una connessione ad Internet sufficientemente affidabile e prestazionale.

- Universali: a differenza delle altre categorie, gli elementi appartenenti a questa classe si collocano in una posizione intermedia tra i tradizionali programmi per personal computer e le applicazioni mobili. Queste applicazioni, infatti, possono essere installate indifferentemente sia su dispositivi fissi come i computer che su device mobile come smartphone e tablet. È il caso di alcune app di Windows, che sono disponibili ed installabili a partire dagli store di entrambe le categorie di dispositivi. Recentemente questa soluzione è stata utilizzata da parte di Microsoft per

ampliare questo concetto ad un'implementazione destinata a console di gioco Xbox o a visori di realtà virtuale[52].

A seconda delle funzioni che si vogliono implementare, della quantità di risorse in termini di strumenti e sviluppatori e più in generale della situazione corrente al momento di ideazione e progettazione dell'applicazione, può risultare conveniente optare per una delle categorie precedenti rispetto ad un'altra. In linea di massima, tuttavia, qualora le suddette condizioni lo permettano, optare per una app nativa è tipicamente la soluzione ottimale in termini di performance e funzionalità dell'applicazione stessa, in quanto le librerie e le API (Application Programming Interface) dedicate all'implementazione sui dispositivi e sistemi operativi relativi permettono operazioni più specifiche e adatte che si riflettono con un impatto evidente sulla qualità finale del prodotto.

Questa classificazione tiene conto della forma in cui un'applicazione mobile può essere costruita e distribuita, ma è possibile categorizzare le stesse in base all'ambito sociale e commerciale all'interno del quale le funzionalità proposte vanno a collocarsi. Proprio per questo motivo, i cosiddetti market o store, ossia le piattaforme spesso dedicate ad un singolo ambiente o sistema operativo su cui è possibile cercare, sfogliare ed eventualmente portare a termine l'acquisto e la successiva installazione delle varie applicazioni, sono sempre provvisti di una suddivisione in categorie, così che sia più immediato comprendere l'ambito di competenza in cui le app mobili si propongono di operare.

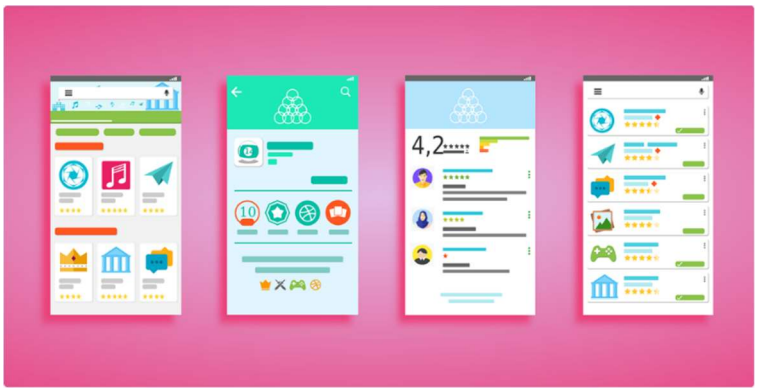

Figura 16 - Tipiche schermate di uno store di applicazioni mobili[53]

[52] Microsoft Windows, Che cos'è un'app UWP (Universal Windows Platform)?, docs.microsoft.com, 2021

[53] 200degrees, interfaccia-utente-androide, pixabay.com

Da un punto di vista consumer, ossia ad uso privato e non strettamente professionale, nonostante il numero delle suddette categorie sia spesso elevato e differente per i diversi store, è possibile identificarne sei tra le più rilevanti[54] e per ognuna di esse gli sviluppatori devono tenere conto di fattori e considerazioni differenti.

- Lifestyle: in italiano "stile di vita", sono attualmente tra le più in voga ed utilizzate. È quella tipologia di applicazioni in cui lo scopo delle stesse è quello di supportare e migliorare le sfaccettature individuali che definiscono le proprie abitudini o routine giornaliere. Fanno parte di questa classe tutte quelle mobile app operanti nell'ambito degli esercizi per la forma fisica, degli incontri e appuntamenti, delle ricette di piatti e tracciamento alimentare, della musica da ascoltare o suonare e comporre e di quelle collegate ai viaggi e al turismo. Sono applicazioni che gli utilizzatori avviano su base giornaliera per tracciare i propri progressi o trovare nuove ispirazioni, guide, contenuti e destinazioni relative ai propri interessi. Proprio perché il mercato relativo è così pieno di elementi e alternative, nello sviluppo di un'applicazione di questo genere è importante porsi come obiettivo la creazione di una piattaforma accogliente, che si adatti in qualche modo ai singoli individui e che sappia sorprenderli con delle funzionalità inedite o più efficaci, di modo che il prodotto ottenuto spicchi e si differenzi dagli altri già presenti. Esponenti di questa categoria sono ad esempio Spotify, una diffusa applicazione di streaming musicale che mette a disposizione una libreria contenente migliaia di brani e podcast, o Tripadvisor, un portale statunitense contenente innumerevoli recensioni di attività alberghiere e della ristorazione.

- Social Media: conosciuti anche come social network, sono tra le app che registrano il maggior numero di utenti attivi ogni giorno, arrivando a far registrare miliardi di accessi nelle ventiquattro ore. Questi servizi, seppur

[54] DuckMa Blog, What Are the Different Types of Mobile Apps? Breaking Down Industries and Functionalities, blog.duckma.com

utilizzati anche nella loro versione non mobile attraverso siti Internet o applicazioni desktop, sfruttano appieno le loro potenzialità quando installate come app di dispositivi mobili. In questa variante, è importante che l'esperienza risulti fluida ed intuitiva, ma anche divertente ed in continua evoluzione in termini di contenuti e funzionalità. Gli individui della società attuale condividono su questi portali una mole di contenuti riguardante le proprie attività quotidiane o i propri interessi di portata smisurata. Proprio per questo motivo, anche applicazioni mobili le quali non ricadono strettamente nella categoria considerata sono coinvolte nell'includere al proprio interno delle funzionalità di condivisione sui social. Numerose app mobili infatti forniscono la possibilità di caricare e condividere contenuti quali prodotti, notizie od obiettivi personali direttamente sulla stessa, senza dover costringere l'utente a doverla chiudere per aprire l'applicazione social in questione.

Figura 17 – Classifica Social Media Mobile Applications[55]

L'interesse dello sviluppatore nei riguardi di questa categoria di applicazioni non dev'essere quindi rivolto alla creazione di nuove app

[55] Mehmet Can Cavas, Top 10 Social Media Apps February 2021, mobileaction.co/blog, 2021

social, ma al prendere in considerazione, in fase di progetto e sviluppo della app considerata, di introdurre all'interno di quest'ultima degli strumenti di condivisione social, in modo da fornire all'utilizzatore un'ulteriore motivazione per continuare ad utilizzare l'applicazione mobile. Tra le app appartenenti a questa classe spiccano realtà come quella di Facebook, la piattaforma social che si pone come capostipite nella classifica di numero di utenti attivi arrivando ad una quota giornaliera media di 1.84 miliardi[56], così come Pinterest, il servizio che integra la creazione di bacheche digitali riempite con immagini, fotografie e video condivisi dagli utenti.

- Utility: la traduzione letterale sarebbe "utilità", ma è da intendersi più come strumenti e servizi di utilità e rappresenta quella categoria di applicazioni mobili che vengono spesso utilizzate senza nemmeno avere in testa in quel momento di stare interagendo con un app. Spesso sono preinstallate sui dispositivi mobili, almeno in una versione basilare ed il loro insieme di servizi e spesso ridotto ad una singola funzionalità. Sono quelle applicazioni mobile che vengono solitamente utilizzate con un'elevata frequenza ma per intervalli di tempo ridotti. Ad esempio, di norma, un individuo interagisce con l'applicazione della calcolatrice per i pochi secondi necessari ad effettuare il calcolo che necessita in un certo momento, per poi in seguito chiudere istantaneamente l'applicazione. Proprio per questo motivo, nella costruzione dell'app mobile lo sviluppatore deve cercare di implementarla in modo che sia rapida e funzionale e quindi renderla più piacevole da adoperare, così da poter sopperire alla ridotta durata di impiego con un numero cospicuo di utilizzi. Tra le più scaricate figurano le app che svolgono la funzione di calcolatrice, di torcia (sfruttando il flash della fotocamera del dispositivo) e le applicazioni per la consultazione delle previsioni meteo ed il monitoraggio dei temporali.

[56] Meta Facebook, Facebook Reports Fourth Quarter and Full Year 2020 Results, investor.fb.com, 2021

- Games/Entertainment: in italiano giochi/intrattenimento, è una categoria particolarmente popolosa e, per questo motivo, tra le differenti case produttrici è presente una forte competizione e una spiccata concorrenza. La porzione relativa ai giochi è tra le classi più intuitive e non ha bisogno di spiegazione, per quanto riguarda l'intrattenimento, invece, si tratta dell'insieme di applicazioni mobili per la fruizione di contenuti multimediali come lo streaming di film e serie televisive o la condivisione e propagazione dei cosiddetti "meme"[57], termine inglese utilizzato per definire tutti quegli stili, idee o azioni che si diffondono nella cultura di massa. Queste tipologie di applicazioni suscitano un vigoroso interesse da parte degli sviluppatori in quanto portano tendenzialmente l'utilizzatore a compiere numerosi accessi durante la settimana o addirittura molteplici in una sola giornata. Per questa motivazione, nel loro sviluppo è di primaria importanza tentare di rendere per l'applicazione mobile il più alto grado di dipendenza da parte degli utenti, implementando meccanismi che offrano a questi ultimi degli incentivi per incrementarne l'utilizzo. Esponenti di rilievo di questa categoria sono applicazioni come Netflix e Amazon Prime Video, colossi di distribuzione streaming e on demand di prodotti dell'industria cinematografica.

- Productivity: nonostante possa non essere la prima categoria a venire presa in considerazione in una riflessione sulle applicazioni mobili, questa risulta, tuttavia, una classe sorprendentemente popolare e diffusa. Questa tipologia di applicazioni supporta gli utilizzatori nell'assolvere i propri compiti in maniera rapida ed efficiente, rendendo quelli che rappresentano attività in alcuni casi quotidiane più semplici e gestibili. Spesso queste applicazioni mobili si sono rivelate utili e già dopo i primi utilizzi possono arrivare a risultare indispensabili, se correttamente realizzate. L'obiettivo che deve porsi uno sviluppatore nella progettazione e realizzazione di un app mobile di questo genere consiste nel trovare dei meccanismi per

[57] Richard Dawkins, The Selfish Gene o Il gene egoista – la parte immortale di ogni essere vivente, 1976

agevolare gli utilizzatori nella finalizzazione di una mansione in maniera più semplice, rapida ed efficiente rispetto alle soluzioni portate alla luce dai lavori della concorrenza. Ciò si traduce nel dover compiere uno sforzo creativo che porti all'ideazione o scoperta di modalità e funzionalità innovative che guidino gli utenti da un punto di partenza fino al completamento di un certo compito, in maniera differente rispetto a quelle già presenti in altri prodotti del settore. Esponenti di rilievo della categoria sono applicazioni come Google Docs, in italiano Documenti Google, il famoso word processor online della compagnia multinazionale californiana, o Evernote, progettata al fine di fornire strumenti efficaci per la stesura di appunti e la gestione di note e attività.

- News/Information: l'obiettivo della categoria è piuttosto lineare, queste applicazioni mobili forniscono ai propri utilizzatori le notizie e gli articoli informativi a cui sono interessati. Ciò è portato a compimento con la realizzazione di impaginazioni grafiche intuitive nella navigazione e nei contenuti, che permettano di raggiungere agevolmente gli elementi più significativi per i suddetti. La considerazione su cui riflettere quando si parla di informazione consiste nella sostanziale immutabilità delle singole notizie, ossia i fatti che esse riportano non possono essere cambiati. Per questo motivo, le due strade percorribili dagli sviluppatori nell'ideazione della loro applicazione, per distinguere il proprio prodotto da quelli al momento già esistenti, si concretizzano in due possibilità: restringere il bacino di informazioni e notizie messe a disposizione ad un ambito specifico e non saturo di alternative, o trovare un modo unico per rendere particolare e caratteristica l'esperienza utente dell'applicazione mobile, con funzionalità o meccanismi singolari. Tra le applicazioni più utilizzate di questa categoria è possibile citare Feedly, un aggregatore di notizie che permette di fruire e organizzare gli articoli e le novità delle proprie fonti fidate e preferite in un'unica schermata, o BBC News, che fornisce i contenuti prodotti dalla società britannica incaricata del relativo servizio pubblico radiotelevisivo e dalla propria rete globale di giornalisti.

Education: rappresenta quella categoria di applicazioni mobili volte a rendere fruibile o integrare l'educazione e la formazione scolastica degli studenti di qualsiasi grado e età. La pandemia globale dovuta alla diffusione del COVID-19 ha fortemente incrementato l'interesse e l'utilizzo relativo a questa tipologia di applicazioni mobili, tuttavia, già in precedenza si era assistito negli ultimi anni una diffusione sempre più intensa delle stesse.

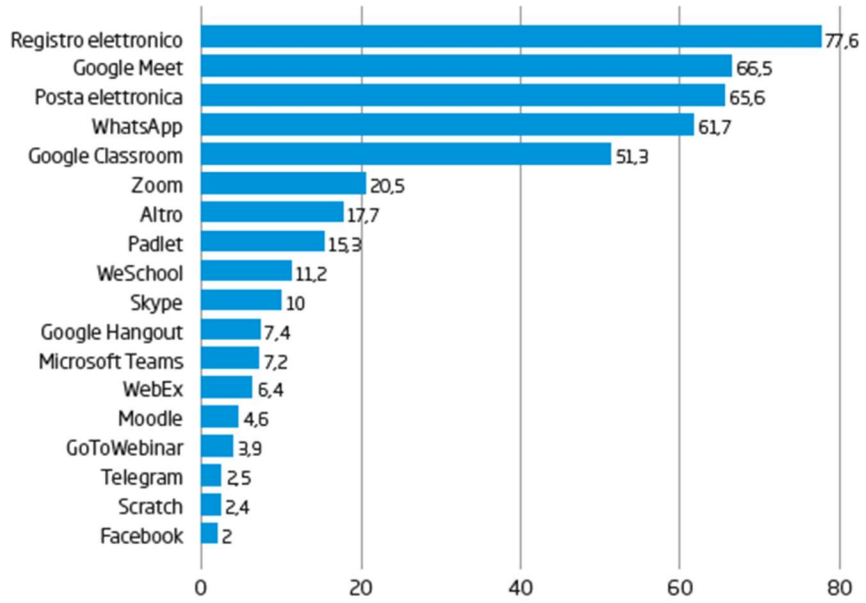

Figura 18 - Grafico raffigurante la distribuzione percentuale di utilizzo di applicazioni per la didattica a distanza nel periodo di lockdown dei primi mesi del 2020[58]

La possibilità di accesso alle informazioni legate all'educazione da qualsiasi luogo e in qualsiasi momento rende infatti il processo dell'apprendimento più semplice e conveniente[59]. Attraverso un tablet o uno smartphone permette la semplificazione di numerosi compiti e un notevole risparmio di tempo impiegato. I vantaggi che comporta l'utilizzo di queste applicazioni mobili sono molteplici, tra cui l'impiego di moderne tecniche di apprendimento, un'efficace comunicazione tra studenti, docenti e genitori, la possibilità di un approccio più completo e sistematico, così come una maggiore efficienza in termini di costi da sostenere. Una delle mobile app

[58] INDIRE, Indagine Tra I Docenti Italiani Pratiche Didattiche Durante Il Lockdown Report Preliminare, 2020

[59] Vidyalankar, Benefits of Mobile Apps in Education, vidyalankar.org, 2021

più utilizzate in questo ambito è Google Classroom, un servizio che consente, tra le altre cose, di limitare l'utilizzo della carta sfruttando documenti digitali, di assegnare compiti e valutare gli elaborati consegnati e di creare sondaggi e avvisi così da ottenere una comunicazione più chiara e sempre a portata di mano.

- Industry and Manufacture: le mobile app di questa tipologia sono spesso commissionate su richiesta alle aziende che si occupano dello sviluppo di applicazioni mobili personalizzate ed è quindi più inusuale trovarle direttamente nei cataloghi degli store. Le funzionalità di cui queste possono essere equipaggiate sono molteplici, ma gli obiettivi principali ricadono nelle intenzioni di fornire meccanismi per monitorare e ricercare dati di interesse, collaborare con i colleghi, ricevere notifiche e reagire alle stesse, così come effettuare operazioni di configurazione e controllo[60]. Nonostante l'utilizzo di personal computer portatili ed il semplice e diffuso scambio di e-mail sia in grado di sopperire ad alcune delle necessità descritte in precedenza, la proliferazione di dispositivi mobili e la conseguente possibilità per operatori, tecnici ed ingegneri di potersi avvalere delle potenzialità di un computer direttamente nel palmo della mano, ha cambiato il modo in cui molti lavoratori portano a termine i loro incarichi quotidiani. Nonostante ad un primo impatto l'introduzione di queste applicazioni mobili sul posto di lavoro possa comportare un'iniziale insofferenza da parte degli utilizzatori, dovuta al doversi confrontare con nuovi meccanismi per i quali non possiedono esperienza, spesso dopo alcune settimane gli stessi si rendono conto dell'assistenza e dei benefici che queste app possono offrire fino ad arrivare a considerarle, in alcuni casi, degli strumenti indispensabili nella loro routine lavorativa. Se queste possono infatti essere utilizzate per la semplificazione di compiti ordinari e consueti, il loro utilizzo può altresì rivelarsi fondamentale in situazioni critiche e vitali legate alla sicurezza. Uno degli aspetti principali da

[60] Renee Bassett, Industrial Mobile Apps: Who's Using Them and Why, automationworld.com, 2014

considerare nella progettazione del layout e delle funzionalità dell'applicazione consiste nel mirare ad ottenere una grafica semplice ed intuitiva, priva di controlli sofisticati per l'utilizzo delle sue funzionalità e con delle logiche di funzionamento esenti da inutili complicazioni, di modo che l'app mobile sia utilizzabile comodamente nel maggior numero di situazioni possibili. In altre parole, a titolo esemplificativo, si potrebbe rendere consigliabile evitare di fare affidamento su movimenti tattili particolari, le cosiddette "gesture", ma preferire a queste dei semplici bottoni o icone.

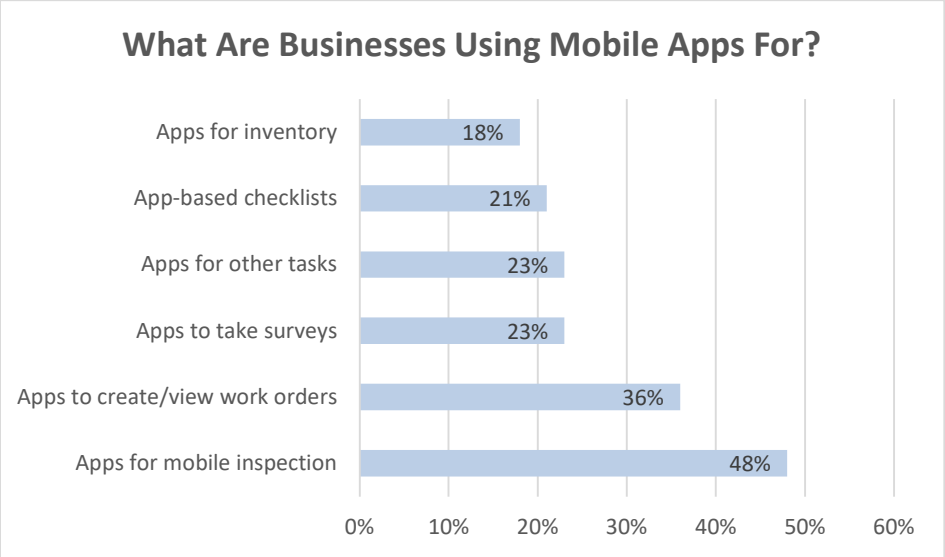

Figura 19 - Grafico che mostra i risultati di un sondaggio sugli ambiti di utilizzo di app mobili in campo industriale[61]

Nonostante come è stato già indicato queste applicazioni siano spesso costruite ad hoc, alcune tra le più utilizzate e presenti sugli store dei dispositivi mobili sono ad esempio Pipe Tools, una soluzione che si prefigge l'obiettivo di implementare un manuale digitale delle informazioni concernenti le tubature in acciaio, così come Plant Manager, un programma che assiste nella pianificazione, ottimizzazione ed esecuzione degli ordini di lavoro.

[61] Jason Peck, What Are Businesses Using Mobile Apps For? Infographic + Survey Results, gocanvas.com, 2014

3.2 Principali attori: Android e iOS

Nonostante nel corso dei primi anni del nuovo millennio siano stati presenti sul panorama dei sistemi operativi per dispositivi mobili numerosi esponenti di rilievo, in grado di ottenere un elevato grado di affermazione e consenso, la quasi totalità dei suddetti è stata nel tempo rimossa dal mercato e sostituita. È il caso di attori rinomati quali Symbian OS, sviluppato e integrato per un lungo periodo sui dispositivi Nokia; BlackBerry OS, sistema operativo proprietario utilizzato sui famosi smartphone "palmari" dell'omonima azienda canadese; Windows Phone, una famiglia con numerose release che sostituì Symbian OS sui dispositivi Nokia poi diventati Lumia; e infine Bada OS, la piattaforma sviluppata e utilizzata per i dispositivi della Samsung Electronics, poi sostituita da Tizen. Se questi ed altri esponenti hanno rappresentato in tempi e modalità diverse lo scenario dei sistemi operativi mobili, ad oggi questo scenario è globalmente dominato da due singole entità: Android e iOS. Congiuntamente, infatti, questi due sistemi operativi per dispositivi mobili hanno costituito, nel corso del 2021, una quota del mercato globale superiore al 99%, con singole porzioni che oscillano rispettivamente intorno al 70% per Android e 29% per iOS[62].

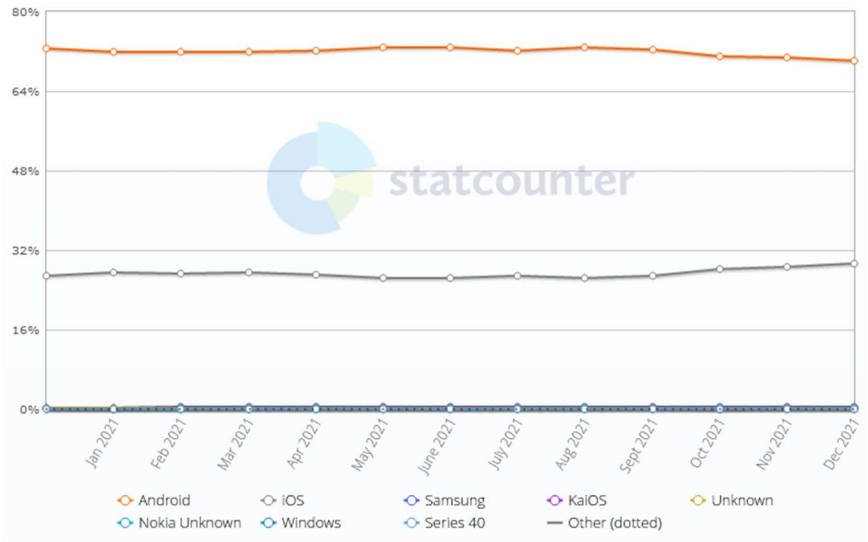

Figura 20 - Distribuzione di mercato dei principali sistemi operativi mobili nel 2021

[62] StatCounter GlobalStats, Mobile Operating System Market Share Worldwide - December 2021, gs.statcounter.com

Il primo sistema operativo per dispositivi mobili di cui verrà esposta una trattazione in questo elaborato è iOS, per due motivazioni principali. La prima è strettamente legata ad un ordine cronologico, in quanto, per l'appunto, la versione iniziale del suddetto è stata rilasciata nel 2007 sotto il nome poi modificato di iPhone OS, in concomitanza con la presentazione del precursore di una lunga serie di device mobili a marchio Apple, l'iPhone 2G[63]. La seconda è dovuta all'innegabile svolta e contributo che l'introduzione di questo sistema operativo ha comportato nella moderna industria dei dispositivi mobili. Uno dei fattori principali a cui questa rivoluzione è dovuta, congiuntamente con le prestazioni e funzionalità del primo iPhone, quali ad esempio l'innovativo touch screen di cui era dotato, si concretizza nella presenza dell'App Store, l'iconico catalogo di applicazioni mobili da cui poter acquistare ed installare le medesime. Se infatti, in un primo momento, le app mobili sviluppate da entità esterne all'azienda californiana non godettero dell'autorizzazione della stessa, per considerazioni legate a possibili problematiche in termini di sicurezza e al timore di effetti nocivi sulle performance del sistema in generale, dopo pochi mesi Steve Jobs, l'indimenticabile cofondatore e amministratore delegato dell'azienda scomparso nel 2011, intuì le potenzialità che queste potessero offrire e stabilì un cambio di rotta, rivelatosi nel seguito vincente, mettendo a punto e distribuendo un software development kit (SDK, in italiano interpretabile come pacchetto di strumenti per sviluppo di programmi e applicazioni) per applicazioni native, prima implementabili solamente come personalizzazioni web ottenute attraverso il browser Safari[64]. Nel corso del 2008 venne quindi presentato al pubblico il già citato App Store, al cui interno erano rese disponibili per l'acquisto 500 applicazioni mobili[65]. Questo numero negli anni successivi è cresciuto vertiginosamente, arrivando nel corso dell'anno seguente a superare quota centomila, sfondando nel 2014 il muro del milione di unità e arrivando, secondo i dati disponibili al momento della stesura di questo elaborato, ad oltrepassare la cifra delle 4.5 milioni di applicazioni presenti[66]. Ufficialmente,

[63] Lisa Eadicicco, TIME, Watch Steve Jobs Unveil the First iPhone 10 Years Ago Today, time.com, 2017
[64] Philip Elmer-DeWitt, Steve Jobs: Apple Will Open iPhone to 3rd Party Apps in February, fortune.com, 2007
[65] Apple Newsroom, The App Store turns 10, apple.com/newsroom, 2018
[66] David Curry, App Store Data (2022), businessofapps.com, 2022

infatti, le applicazioni native per iOS posso essere scaricate unicamente dal catalogo presente sull'App Store, che attualmente è suddiviso in 21 categorie. Ciononostante, è possibile in alcuni casi che l'azienda di sviluppo di una certa applicazione mobile fornisca e distribuisca la stessa sotto forma di archivio compresso con estensione .ipa (iOS App Store Package), attraverso canali differenti dall'App Store e considerati quindi non ufficiali. Sebbene per un lungo periodo iOS è stato il sistema di riferimento comune sia per dispositivi appartenenti alla famiglia degli smartphone che quella dei tablet dei prodotti a marchio Apple, rispettivamente gli iPhone e gli iPad, per questi ultimi è stata introdotta durante la Worldwide Developers Conference tenutasi nel 2019 una variante dedicata del sistema operativo chiamata iPadOS, con l'obiettivo di ottimizzare e ampliare le funzioni che questi mettono a disposizione[67]. Molte delle caratteristiche presenti già in iOS, tuttavia, permangono in egual misura all'interno di questo più recente sistema operativo. Dal punto di vista degli sviluppatori, Apple fornisce loro un SDK che li mette in condizione di effettuare la creazione e lo sviluppo di app mobili native per dispositivi basati su iOS[68]. La versione più recente utilizzabile al momento è la iOS 15 SDK, che fornisce tutta una serie di innovative tecnologie chiave e nuove caratteristiche. Tra le principali novità rilasciate, è possibile estrapolarne alcune delle più significative:

- GroupSessionMessenger and Group Activities, delle nuove API che consentono la creazione di applicazioni che permettano la condivisione tra più utenti di contenuti video con un grado di fedeltà totale e una sincronizzazione gestita dal sistema, così come, per i contenuti multimediali diversi dallo streaming video, dei canali di scambio dati sicuri che siano in grado di sincronizzare le informazioni tra le differenti istanze dell'applicazione fra molteplici utilizzatori;

- Focus, un nuovo meccanismo di gestione delle notifiche che permetta agli utenti di ricevere le notifiche in istanti per loro funzionali e agli sviluppatori di avvalersi delle nuove Interruption Levels API, che consentono una

[67] Stefano Bontempi, Apple, tutte le novità della WWDC 2019, hdblog.it, 2019
[68] Apple iOS, What's New in the iOS SDK, developer.apple.com, 2022

generazione delle notifiche da implementare più sfumata, con quattro livelli di interruzione;

- Swift 5.5, che introduce il supporto alla concorrenza, costruita a partire dal linguaggio attraverso l'utilizzo di meccanismi di async e wait;

- ARKit and RealityKit, strumenti che forniscono potenti capacità di personalizzazione che possano supportare la creazione di applicazioni per esperienze di realtà aumentata che risultino ancora più convincenti rispetto a quelle sviluppabili in precedenza;

- Create ML, ora disponibile come framework Swift anche su iOS e iPadOS, consente la creazione di funzionalità dinamiche per le applicazioni che sfruttano delle nuove API per effettuare il training dei modelli a partire dagli input e comportamenti degli utenti direttamente a bordo del dispositivo, per preservarne la privacy.

L'Integrated Development Environment (IDE, in italiano ambiente di sviluppo integrato) ufficiale fornito e mantenuto da parte di Apple per lo sviluppo di applicazioni mobili è Xcode, il quale supporta la scrittura di codice in numerosi linguaggi di programmazione, tra cui Java, C++, Ruby, Swift e Python.

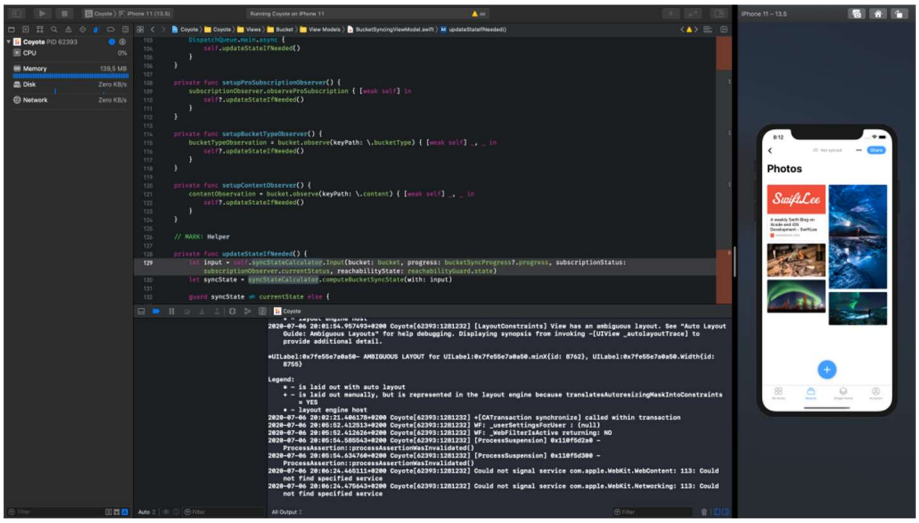

Figura 21 - Schermata dell'IDE Xcode accostata all'iPhone Simulator[69]

[69] Antoine van der Lee, Full-screen development with Xcode and the Simulator, avanderlee.com

In aggiunta a quanto descritto in precedenza, l'SDK in questione consente altresì l'utilizzo di un simulatore grafico rappresentante un iPhone, di modo che sia possibile testare il layout ed il comportamento dell'applicazione direttamente dalla schermata su cui viene scritto il codice. Se questo Software Development Kit è distribuito da Apple attraverso una licenza ottenibile, per i possessori di computer Mac, senza alcun costo, lo stesso non si può dire per la pubblicazione e messa in commercio delle applicazioni mobili sull'App Store. Per questo procedimento è infatti necessario effettuare una sottoscrizione a pagamento, individuale o per organizzazione, al cosiddetto Apple Development Program, un programma il cui abbonamento annuale è offerto a 99 dollari statunitensi[70]. In aggiunta alla possibilità di distribuzione delle proprie app mobili sullo store di riferimento, in questo pacchetto sono inclusi strumenti e risorse volti a consentire l'accesso a servizi quali gli ultimi programmi nelle loro versioni beta, fornire meccanismi per testare e analizzare i software sviluppati e mettere a disposizione un sostegno tecnico da parte di ingegneri di supporto.

Temporalmente successivo ad iOS, ma con un'adozione da parte dei dispositivi sul mercato che si quantifica in una percentuale grande più del doppio rispetto a quest'ultimo, il secondo sistema operativo per dispositivi mobili di cui verrà fornita una trattazione nel presente elaborato è Android. Sebbene il suo ambito di utilizzo principale si concretizzi appunto nei dispositivi quali smartphone e tablet, l'ambiente Android prevede l'esistenza di ambienti dedicati anche per ulteriori e differenti categorie di apparecchiature elettroniche. In questi casi la GUI, acronimo per indicare la Graphical User Interface, è rilasciata sotto forma di personalizzazione specifica di Android, volta a sfruttare ed ottimizzare i rapporti di aspetto e le dimensioni delle differenti tipologie di device. Esempi di questo concetto sono realizzati nei sistemi quali Android TV, dedicato ai televisori di ultima generazione comunemente indicati come Smart TV per la loro possibilità di essere connessi ad Internet, o Android Auto, destinato ad una implementazione del sistema sulle automobili messe sul mercato a partire dallo scorso decennio, così come l'ex Android Wear, ormai noto come Wear OS, rivolto ai dispositivi indossabili dotati di schermo tattile solitamente noti come Smartwatch. Questa

[70] Apple Inc., Apple Developer Program - What You Need To Enroll, developer.apple.com, 2022

disponibilità di differenti caratterizzazioni proprie per ogni specifico ambito rende Android un vero e proprio ecosistema digitale che può essere sfruttato per integrare e personalizzare l'esperienza utente complessiva.

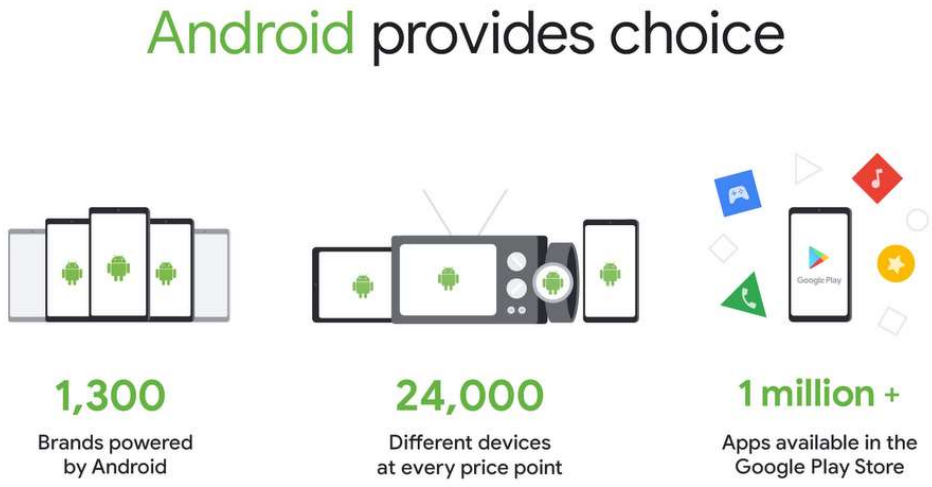

Figura 22 - Statistiche del 2018 in riferimento all'ecosistema Android[71]

Per vedere la comparsa di Android è necessario attendere il 2003, anno in cui il progetto nacque con lo scopo, da parte della compagnia tecnologica statunitense Android Inc., di sviluppare un sistema operativo che potesse essere utilizzato a bordo di macchine fotografiche digitali[72]. L'anno successivo il progetto subì una svolta che lo portò a cambiarne l'obiettivo in termini di ambito di interesse, spostando quindi l'attenzione sulla sfera dei dispositivi mobili cellulari. Il momento chiave nella crescita del progetto, che mise le basi per il sistema operativo mobile come oggi lo conosciamo, si identifica con l'acquisizione da parte di Google, nel 2005, del progetto Android[73]. Nel corso di quell'anno, infatti, degli esponenti del colosso californiano incontrarono i fondatori della Android Inc. e si fecero convincere della bontà del progetto in seguito all'esposizione di una versione

[71] Sundar Pichai, Android provides choice, blog.google, 2018
[72] The Editors of Encyclopaedia Britannica and Erik Gregersen, Android operating system, britannica.com
[73] John Callaham, Google made its best acquisition nearly 16 years ago: Can you guess what it was?, androidauthority.com, 2021

prototipale del sistema operativo mobile. Durante i tre anni successivi gli sviluppatori di Google lavorarono per ampliare e portare a compimento la prima versione pubblica di Android. Nel 2008, per l'appunto, venne presentato il primo dispositivo mobile con a bordo il suddetto sistema operativo, con la comparsa sul mercato del T-Mobile G1, anche conosciuto come HTC Dream.

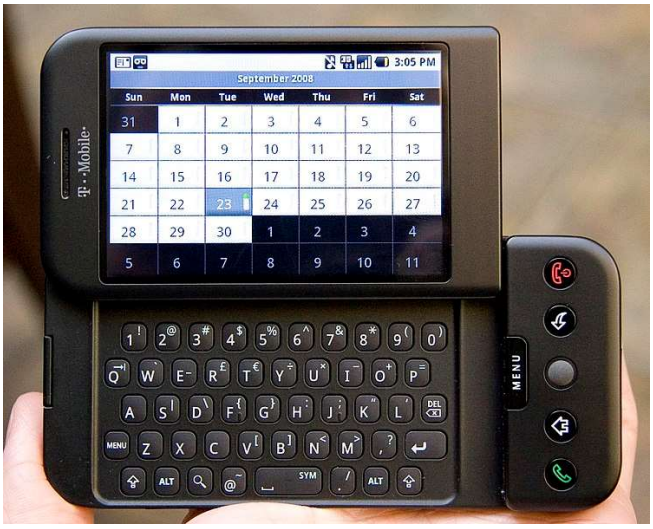

Figura 23 - Fotografia del primo dispositivo mobile con sistema operativo Android: il T-Mobile G1[74]

L'uscita del dispositivo in questione costituisce il primo esempio di una lunga serie, ad oggi in continua espansione, di aziende elettroniche che utilizzano Android sui propri apparecchi. Il numero di questi ultimi è cresciuto enormemente arrivando a superare, nel corso del 2021, la cifra di 3 miliardi di dispositivi attivi[75]. Uno degli elementi che costituì motivo di interesse nei confronti del T-Mobile G1, congiuntamente con l'implementazione degli assodati servizi e strumenti di ricerca e navigazione certificati da Google, fu la presenza di uno store di applicazioni da cui poterle sfogliare, acquistare ed installare, il cosiddetto Android Market[76]. Nonostante il confronto con uno store di applicazioni mobili già affermato e maturato nei mesi precedenti, ossia quello identificato dall'App Store della controparte Apple, in termini di quantità e qualità delle app in sé, potesse non soddisfare pienamente, l'esistenza di questo store mostrò solide promesse. Attualmente le applicazioni mobili native sviluppate per dispositivi Android

[74] Michael Oryl, T-Mobile G1 launch event, flickr.com
[75] Alex Cranz, There are over 3 billion active Android devices, theverge.com, 2021
[76] Bonnie Cha, N. Lee, T-Mobile G1 review:, cnet.com, 2018

possono essere scaricate da numerosi store presenti sulla rete e, in alternativa, queste possono essere distribuite ed installate sui device direttamente dalle aziende che distribuiscono il software attraverso file in formato .apk (da Android PacKage). In questo scenario di store per applicazioni Android, oltre ad affermati esponenti come Aptoide e l'Amazon App Store, ad oggi il punto di riferimento consiste nel catalogo messo a disposizione dalla piattaforma ufficiale Google Play, anche conosciuta come Play Store. Nonostante il numero preciso di applicazioni mobili presenti su quest'ultimo risulti complicato da misurare, dato il continuo processo di creazione e inserimento di nuove app da parte degli sviluppatori e di monitoraggio e rimozione di altre effettuato dai tecnici di Google, negli ultimi anni le stime relative al suddetto oscillano intorno alle 3 milioni di unità[77]. Al momento della stesura di questo elaborato questo store ufficiale presenta all'interno del suo catalogo una quantità elevata di categorie, che si quantifica con più di 30 singole componenti, di modo che sia più semplice ed intuitivo per gli utilizzatori sfogliare ed individuare le applicazioni che possano incontrare le proprie necessità o interessi. Dal punto di vista degli sviluppatori, per poter pubblicare i propri contenuti su Google Play, oltre a dover creare l'account e accettare i contratti e regolamenti del caso, questi devono effettuare il pagamento di una quota di registrazione, al momento offerta al prezzo di 25 dollari statunitensi da versare una tantum[78]. In questo modo, allo sviluppatore sarà garantito l'accesso alla cosiddetta Play Console, una piattaforma messa a disposizione da Google che offre al suddetto una serie di strumenti e servizi volti a pubblicare, impreziosire e monitorare le applicazioni che lo stesso ha generato in precedenza. Più nello specifico, la piattaforma consente una gestione chiara e trasparente dell'account dello sviluppatore, un controllo sul ciclo di vita dell'applicazione in termini di pubblicazioni, revisioni e aggiornamenti della stessa, una personalizzazione delle impostazioni relative alla sua distribuzione con particolare attenzione alle zone geografiche e alle tecniche di promozione e partecipazione a differenti programmi, una modalità con cui appurare la bontà e l'andamento dell'applicazione attraverso

[77] L. Ceci, Number of available applications in the Google Play Store from December 2009 to September 2021, statista.com, 2022
[78] Google, Come usare Play Console, support.google.com

l'impiego di test e report statistici, un'amministrazione relativa ai prezzi e alle modalità con cui si vuole distribuire l'app con differenti opzioni per i pagamenti e abbonamenti della stessa o degli elementi presenti al suo interno, così come un supporto per la creazione di contenuti che siano conformi a tutte le norme di utilizzo e quelle relative alla privacy e la sicurezza. L'ambiente di sviluppo integrato principale per la creazione di applicazioni mobili native Android è costituito dal programma Android Studio. Tale software è stato concepito in maniera specifica per la generazione applicazioni della suddetta tipologia ed è stato implementato personalizzando IntelliJ IDEA, l'Integrated Development Environment dell'azienda di software ceca JetBrains[79].

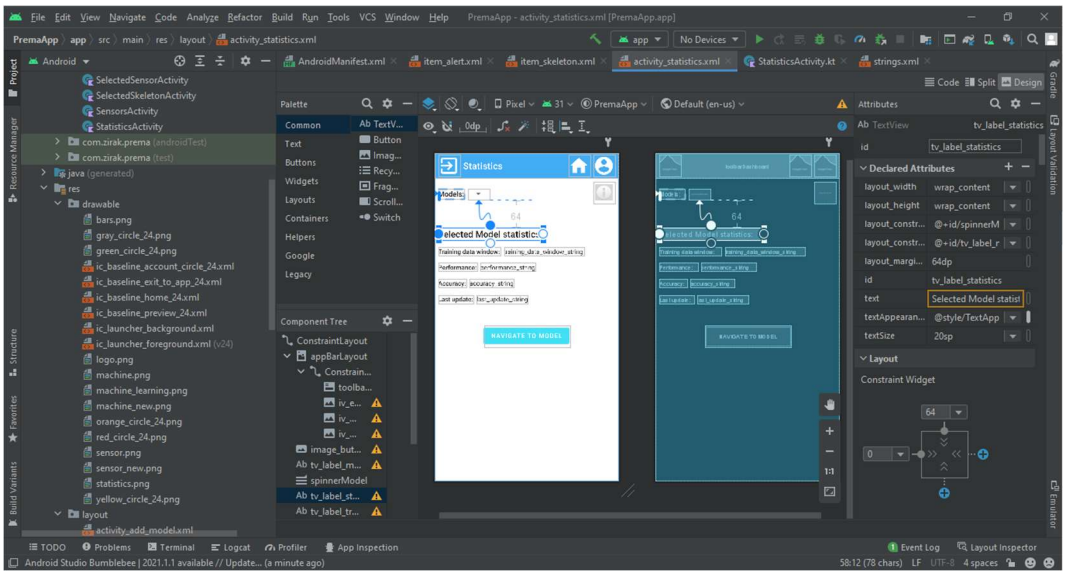

Figura 24 - Schermata dell'ambiente di sviluppo integrato Android Studio

Tra le funzionalità che questo IDE mette a disposizione e che lo rendono così valido ed utilizzato, verranno elencate nel seguito alcune tra le più caratteristiche e rilevanti[80].

- Editor di layout visuale: è possibile creare complessi layout grafici per le schermate dell'applicazione utilizzando numerose categorie di arrangiamento dei contenuti, modificando ad esempio i vincoli del

[79] Eugene Toporov, IntelliJ IDEA is the base for Android Studio, the new IDE for Android developers, 2013

[80] Google Developers, Android Studio, developer.android.com

ConstraintLayout. Inoltre, è possibile visualizzare un'anteprima del layout generato su una finestra di visualizzazione che può essere ridimensionata a piacimento o selezionata tra una di quelle relative alle numerose configurazioni di dispositivi messe a disposizione.

- Analizzatore APK: fornisce l'opportunità di ridurre le dimensioni della propria applicazione effettuando un'ispezione dei contenuti del relativo file APK, sia esso stato creato su Android Studio o meno. È infatti possibile esaminare il manifest, ossia il file in formato .xml che che descrive le informazioni essenziali relative all'app, così come i vari file per le risorse del progetto, come immagini o valori di diversa natura. Successivamente, al termine del completamento di tali operazioni, è possibile confrontare il nuovo APK con quello creato in precedenza, per verificare le differenze in termini di spazi di archiviazione necessari.

- Emulatore rapido: sebbene sia possibile provare le proprie applicazioni mobili, sia durante il loro sviluppo che in seguito al completamento delle stesse, installandole su dei dispositivi fisici collegati su cui siano state abilitate le cosiddette "Opzioni svilupatore", è possibile eseguire le stesse in maniera più rapida e semplificata direttamente sull'emulatore presente nell'ambiente di sviluppo integrato. Questo emulatore può assumere le configurazioni di numerosi dispositivi e simularne la totalità delle funzioni di interesse, così da avere un riscontro immediato sul comportamento e la qualità del lavoro svolto.

- Editor di codice intelligente: l'opportunità di scrivere codice più efficace e pulito, incrementando al contempo la produttività dello sviluppatore e la velocità con cui esso può portare a compimento i propri compiti, è ottenuta sfruttando nella scrittura del codice stesso un editor che fornisca meccanismi di completamento e assistenza per i linguaggi Kotlin, Java, C e C++.

- Sistema di compilazione flessibile: alimentato da Gradle, la piattaforma per l'automazione nello sviluppo software, lo strumento di compilazione presente in Android Studio permette agli sviluppatori di personalizzare e configurare le proprie build in modo da generarne numerose varianti, per i singoli dispositivi in esame, a partire da un unico progetto.

- Profilatori in tempo reale: all'interno dell'IDE sono presenti degli strumenti integrati per la profilazione dei consumi di risorse, che sono in grado di fornire statistiche in tempo reale relative all'attività dell'applicazione in esame negli ambiti della CPU, della memoria e della rete. In questo modo è possibile identificare eventuali colli di bottiglia, ispezionando ad esempio le allocazioni o i pacchetti di rete in entrata e in uscita.

Sebbene Android e iOS costituiscano di fatto, come già evidenziato in precedenza, la pressappoco totalità delle alternative nell'ambito dei sistemi operativi per dispositivi mobili, è doveroso citare in questa panoramica un terzo attore, non tanto per la sua rilevanza attuale ma per l'impatto che potrebbe portare nel prossimo futuro. Si tratta di HarmonyOS, il sistema operativo messo a punto dal colosso cinese Huawei per l'implementazione dello stesso sui propri dispositivi mobili[81]. Inizialmente svelato dall'azienda con il nome di HongMeng OS, rappresenta la risposta della stessa al bando imposto nei suoi confronti da parte dell'amministrazione Trump nel corso del 2019, che la additava come elemento di rischio per la sicurezza nazionale[82]. Tale bando, infatti, tra le sue altre implicazioni, impose a Google di interrompere il supporto dei suoi servizi alla casa di produzione cinese. In un primo momento Huawei continuò ad utilizzare il sistema operativo Android sui suoi dispositivi mobili, smettendo però di includere negli stessi lo store Google Play e sostituendolo con il proprio, il recente AppGallery. Anche questo sistema operativo è stato progettato per venire implementato non soltanto su

[81] Deng Li, HarmonyOS (HongMeng OS): Everything you need to know, huaweicentral.com, 2021
[82] David Shepardson, Freifeld, Trump extends U.S. telecom supply chain order aimed at Huawei, ZTE, reuters.com, 2020

smartphone e tablet, ma altresì su dispositivi indossabili come gli Smartwatch e su apparecchi con schermi più grandi come le SmartTV. Se come accennato l'adozione odierna di tale sistema operativo non sia eccessivamente significativa, grazie a questo atteggiamento volto alla creazione di un ecosistema tra dispositivi e all'affermata presenza dell'azienda sul mercato globale, i suoi sviluppi futuri rappresentano uno dei motivi di interesse nel campo di riferimento.

Per quanto concerne iOS ed Android, è possibile confrontare i due sistemi operativi in base a numerose considerazioni. Essendo stati negli anni i principali concorrenti l'uno dell'altro, nel tempo sono stati redatti numerosi articoli e contenuti che li paragonassero e mettessero in comparazione i punti di forza e di debolezza di entrambe le alternative, in modo da far risaltare quali siano le motivazioni per preferirne una rispetto all'altra. Infatti, è evidente, guardando le vendite in entrambi i casi massicce dei prodotti che implementano questi sistemi operativi, che ci siano delle platee di utilizzatori che prediligono per lungo tempo uno dei due e altri che in differenti occasioni decidono di optare per dispositivi di una e dell'altra tipologia. Questo fenomeno può essere spiegato realizzando che non esista in assoluto una piattaforma che sia superiore rispetto all'altra, ma differenti situazioni e necessità che rendono una delle due soluzioni più adatta a soddisfare l'esigenza in questione. Nell'intento di stilare una lista delle tematiche e degli aspetti su cui i due sistemi operativi per dispositivi mobili siano confrontabili, è possibile identificarne dieci principali, come mostrato nella figura successiva.

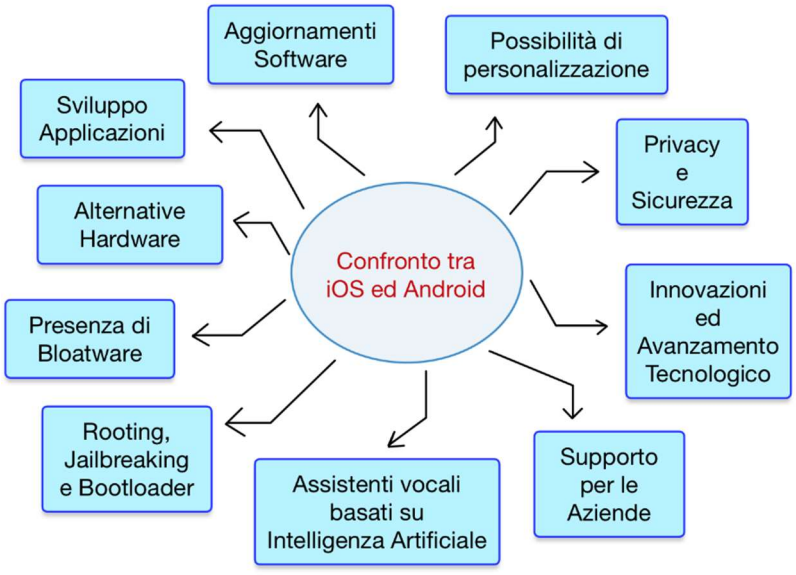

Figura 25 - Diagramma che illustra 10 ambiti di confronto tra iOS ed Android

L'interesse relativo a queste categorie di confronto, sia dal punto di vista degli utilizzatori così come da quello degli sviluppatori, le copre nella loro totalità, in quanto, a titolo esemplificativo, se un acquirente possa non essere coinvolto direttamente nella creazione di applicazioni ma nel solo utilizzo delle stesse, sapere in che modalità queste possono venire alla luce può tuttavia costituire un ulteriore spunto decisionale nel momento in cui questo si trovi a dover stabilire per quale prodotto propendere[83]. Ciononostante, da un punto di vista più strettamente relativo alla creazione e all'impiego delle applicazioni native di questi due sistemi operativi in ambito professionale, è possibile isolare tre ambiti di paragone più fondamentali.

- Sviluppo: è uno degli aspetti più tecnici che differenzia le due entità, sia in termini di complessità procedurali che dei costi da sostenere, sia relativi a queste ultime che da esse slegati. Il primo aspetto da considerare consiste nel modello implementativo del kernel del sistema: Android costituisce una piattaforma di tipo open-source, con un kernel, un'interfaccia grafica e parte delle applicazioni completamente appartenenti a questo paradigma. Al contrario, il kernel iOS non è di tipo open-source. Questo fattore è di particolare rilevanza in quanto consente alle case produttrici di dispositivi mobili di generare la propria specifica personalizzazione di Android. Collegato a questo punto di interesse, è di primaria rilevanza una considerazione sulla complessità relativa allo sviluppo delle applicazioni. Infatti, se Apple è l'unica casa produttrice di dispositivi con a bordo iOS, Android è implementato da numerose aziende del settore. Questo rende lo sviluppo delle app mobili Android più complicato, in quanto devono essere funzionali in una più grande varietà di dispositivi e con una quantità di tipologie di hardware differenti. Ciò si traduce in tempi di sviluppo più elevati, rispetto a quelli necessari per portare a termine la creazione di una app nativa iOS. Questa considerazione ha impatti considerevoli anche sui costi veri e propri da sostenere nelle due casistiche. Infatti, tempi di creazione più prolungati si traspongono spesso in aumenti delle spese necessarie. Ciononostante, questo non è l'unico fattore da considerare dal

[83] Harikrishna Kundariya, Android VS iOS: A Comparison of Distinctive Attributes, esparkinfo.com

punto di vista economico. Per l'appunto, a tal proposito è necessario valutare come l'ambiente di sviluppo Android Studio sia disponibile gratuitamente su tutti i computer con a bordo i principali sistemi operativi, mentre Xcode è distribuito unicamente su macchine Mac, che, come è noto, hanno tipicamente prezzi più elevati. Questa disparità è individuabile allo stesso modo nella distribuzione e pubblicazione delle app sugli store ufficiali di riferimento, a partire dal fatto che, come già indicato, la possibilità di sfruttare l'AppStore per queste necessità sia subordinata ad una tassa di utilizzo annuale piuttosto onerosa, quando un quarto di tale cifra sia sufficiente per ottenere una registrazione a vita per l'utilizzo di Google Play. Infine, un ulteriore motivo di divisione è dovuto ai linguaggi di programmazione utilizzabili, diversi per le due alternative. Non è quindi semplice selezionare l'una o l'altra soluzione come la più indicata dal punto di vista dello sviluppo, vanno infatti valutati i punti di forza e gli svantaggi più considerevoli nelle differenti situazioni e in base alle necessità e risorse del caso.

- Privacy e Sicurezza: la prima considerazione su cui porre l'attenzione consiste nell'essenza e filosofia stessa del sistema operativo Android. Questo, infatti, se da un lato può trarre beneficio dall'essere più "aperto" rispetto alla controparte, a partire dalla stessa condizione può invece risultare più suscettibile all'attacco e alla successiva azione di programmi malevoli. In questo senso, quindi, è possibile affermare che sia, in una certa misura, più vulnerabile. Dal punto di vista dell'installazione sui dispositivi di applicazioni mobili con procedure diverse dal reperimento sugli store ufficiali dei due sistemi, anche in questo caso Android risulta più "aperto" e di conseguenza più esposto. Inoltre, la cosiddetta procedura di jailbreaking, ossia la ricerca e lo sfruttamento delle falle presenti su un dispositivo elettronico al fine di ottenerne i permessi di root del sistema operativo e sbloccarne la completa lista di funzionalità[84], non è mai completamente realizzabile sui dispositivi che implementano iOS. Come

[84] Kaspersky Lab, What is Jailbreaking – Definition and Explanation, kaspersky.com

conseguenza, tutta una serie di funzionalità che sono bloccate per motivi di sicurezza continuano ad essere protette. Infine, rispetto alla controparte, Apple rilascia per iOS aggiornamenti e patch di sicurezza in maniera più frequente, rendendo più complicato tentare di sfruttare eventuali vulnerabilità. Per le motivazioni descritte in precedenza, il suddetto sistema operativo risulta, paragonato ad Android, più efficace e completo nell'ambito della sicurezza.

- Supporto per le Aziende: entrambe le piattaforme supportano i prodotti del pacchetto Microsoft Office, come Word ed Excel e i servizi offerti da parte di Google, come Google Docs. Dal punto di vista della compatibilità con il sistema operativo Windows, tuttavia, con Android questa risulta più ampia ed efficace. Ciononostante, i dispositivi iOS risultano più interfacciabili con i computer Mac, prodotti dalla stessa azienda Apple e quindi progettati per collaborare in un unico ecosistema. In aggiunta a ciò, il modello open-source implementato da Android permette una maggiore personalizzazione ed infine, in relazione alle applicazioni di terze parti, è spesso più semplice lavorare con le stesse utilizzando il medesimo sistema operativo. Anche in questo caso sarebbe opportuno valutare in base ai differenti scenari presenti all'interno delle diverse aziende, tuttavia se ci si deve sbilanciare, tenendo conto di quanto evidenziato in precedenza, Android risulta la scelta in un certo senso più flessibile.

3.3 Sviluppo applicazioni Android: aspetti principali

È già stato evidenziato in precedenza all'interno di questo elaborato quanto il settore delle applicazioni mobili native per il sistema operativo Android sia stato, negli ultimi anni e in maniera specifica nel periodo attuale, particolarmente attivo ed in continua crescita, come dimostra l'esorbitante quantità delle suddette nei cataloghi dei principali store relativi. Per questo motivo, è stato deciso di includere nella trattazione effettuata in questo elaborato, una panoramica sulle principali tecniche e componenti ad oggi presenti sul panorama dello sviluppo di tali applicazioni mobili, con particolare riferimento ai principali elementi grafici

inseribili all'interno delle schermate delle stesse e alle componenti architetturali più consigliate, diffuse ed utilizzate nella loro progettazione e creazione. La piattaforma di riferimento nel proseguo del paragrafo sarà l'ambiente di sviluppo integrato Android Studio. Un primo aspetto che è ragionevole approfondire è quello relativo alle tipologie di file che è possibile mettere a punto utilizzando questo IDE. La lista di questi ultimi è particolarmente fitta ed estesa, tuttavia ne esistono due che risultano imprescindibili per la creazione di qualsiasi applicazione mobile, ossia quelli relativi al layout delle schermate e alle logiche di funzionamento ed interazione associate. Per quanto riguarda i primi, l'interfaccia grafica è realizzata mediante la scrittura di stringhe in file di tipo XML, acronimo per il termine inglese eXtensible Markup Language, traducibile come linguaggio di marcatura estendibile. Questi sono infatti costituiti da componenti racchiuse entro dei cosiddetti tag personalizzabili in funzione delle necessità del caso, motivo per cui il linguaggio è considerato estendibile. Per quanto concerne invece le logiche con cui gli elementi grafici sono eseguiti e le interazioni che l'utilizzatore può avere con essi, è necessario generare delle cosiddette Activity o Fragment, che possono essere considerate come gli elementi fondamentali per agire sulle varie schermate. I linguaggi di programmazione maggiormente diffusi per la scrittura di tali logiche, e le relative tipologie di file utilizzabili, sono costituiti da Java, il famoso ed affermato linguaggio rivolto agli oggetti e da Kotlin, un linguaggio basato a sua volta sulla Java Virtual Machine e indicato da Google, a partire dal 2019, come favorito per la creazione di app mobili Android[85]. Il fatto che quest'ultimo sia in buona misura affine a Java li rende interoperabili, al punto che, integrata in Android Studio, esiste una funzione per convertire una classe scritta in Java in una classe Kotlin che possieda la stessa semantica. Rispetto a Java, con il quale condivide numerosi aspetti, in svariate occasioni il codice Kotlin necessario per descrivere un certo comportamento è considerevolmente più snello e costituito da un minor numero di righe, il che rende, solitamente, lo sviluppo in tale linguaggio più agile e scorrevole. Nella generazione del file .xml adibito al contenimento del layout grafico di una data schermata, l'ambiente di sviluppo integrato fornisce tre diverse

[85] Chet Haase, Google I/O 2019: Empowering developers to build the best experiences on Android + Play, android-developers.googleblog.com, 2019

alternative operazionali: Code, Split e Design. Nella modalità Code, l'ambiente assume le sembianze di un editor di codice intelligente, con meccanismi di supporto e completamento, in cui è possibile andare ad inserire e modificare i vari elementi costitutivi del layout agendo direttamente sul codice del file XML. Questo si traduce in un controllo e manipolazione a basso livello, utile ad esempio nelle situazioni in cui un certo elemento grafico sia stato copiato da un altro file per inserirlo, con le dovute modifiche, all'interno di una nuova vista. Al contrario, sfruttando l'opzione Design lo sviluppatore può beneficiare di un ambiente di lavoro ad interfaccia grafica. Quest'ultima è composta da numerose finestre di supporto, che circondano la vista principale costituita da un'anteprima grafica della schermata che è in fase di creazione. Tale vista non è da confondersi con l'emulatore su cui simulare l'utilizzo dell'applicazione, in quanto nel caso della prima lo scopo è fornire un rendering rudimentale di una singola schermata, per renderne la progettazione più agevole, mentre non è possibile, ad esempio, interagire con essa premendo sui pulsanti al suo interno. Ciononostante, anche in questo caso è possibile personalizzare la vista in modo che rappresenti lo schermo di svariati device, differenti in termini di dimensioni e risoluzione, al fine di arrangiare i vari elementi in uno spazio ben specifico a seconda della necessità, invece di essere limitati all'utilizzo di un unico schermo per tutte le situazioni. A contorno di tale vista, in primis è presente una porzione denominata "Palette", all'interno del quale è possibile sfogliare e selezionare, trascinandoli e posizionandoli al di sopra della schermata, gli elementi grafici e funzionali di interesse. Inferiormente a questa è posizionata la sezione etichettata come "Component Tree", che implementa per l'appunto un albero delle componenti e sottocomponenti principali che sono attualmente inserite all'interno della grafica, identificati dal loro id assegnato. Infine, è presente un ritaglio definito "Attributes" che, trasformandosi nella forma e nei contenuti a seconda del singolo elemento selezionato, mostra i valori correnti dei suoi attributi e fornisce per gli stessi una manipolazione puntale ed intuitiva. In ultima battuta, la modalità Split costituisce una via di mezzo tra le due alternative descritte in precedenza, in quanto fornisce, affiancandoli, sia l'editor testuale che l'interfaccia grafica, in modo che possano venire sfruttate le potenzialità ed opportunità di entrambi gli strumenti senza dover

cambiare videata. Sebbene possa risultare, almeno ad un primo impatto, la soluzione più completa e funzionale, questa è realmente utilizzabile soltanto lavorando su schermi con una diagonale di visione sufficiente, in quanto, nel caso in cui questa condizione venisse a mancare, le varie porzioni non avrebbero sufficiente spazio e collasserebbero in dei menu a tendina, vanificandone l'utilità. Gli elementi principali che è possibile trascinare ed inserire all'interno della schermata in esame sono descritti, all'interno della Palette, in sette categorie principali che verranno a questo punto elencate.

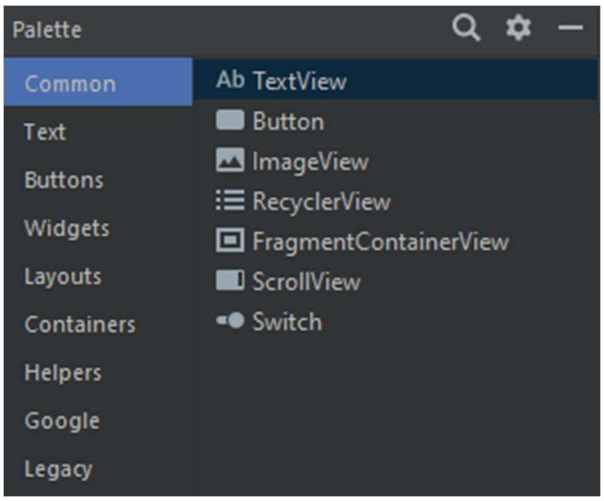

Figura 26 - Schermata della "Palette" di Android Studio

- Text: è la sezione relativa alle cosiddette TextView ed EditText, elementi rispettivamente delegati alla presentazione all'utente di contenuti testuali e all'inserimento da parte di questo ultimo dei suoi input da tastiera. Per quanto riguarda le EditText, ne esistono di svariate tipologie che differiscono, tra le altre cose, in base alla tastiera che viene presentata all'utilizzatore nel momento in cui le tocca con il dito, che cambia in funzione del contenuto che dovrà ricevere: sarà ad esempio numerica se destinata a contenere un recapito telefonico, mentre provvista di lettere e simboli se verrà utilizzata per l'inserimento di un indirizzo e-mail.

- Buttons: costituisce la categoria in cui il funzionamento degli elementi che ne fanno parte è solitamente subordinato ad un singolo tocco del dito sullo

schermo da parte dell'utilizzatore. Esempi di tali componenti sono i semplici bottoni, in inglese Button, così come le conosciute CheckBox per la scelta multipla e i RadioButton per la selezione individuale di un'opzione all'interno di un gruppo.

- Widgets: in questa sezione sono presenti gli elementi relativi alle componenti multimediali, come ImageView e VideoView, così come quelli relativi a un qualche tipo di servizio ausiliario come la possibilità di fornire una valutazione, con le RatingBar, oppure visualizzare e manipolare una certa progressione dei contenuti all'interno della schermata, rispettivamente con le ProgressBar e le SeekBar.

- Layouts: in questa accezione, i layout, ossia gli elementi costituitivi di questa categoria, costituiscono le modalità che è possibile adottare per la strategia di disposizione delle altre entità all'interno della schermata o di una sua porzione. Ad esempio, lo sviluppatore può optare per un LinearLayout, nelle sue varianti verticale ed orizzontale, per disporre gli elementi in colonna o in riga, oppure preferire l'arrangiamento ad ancoraggi fornito dal ConstraintLayout, in cui ad ogni elemento vengono assegnati dei vincoli di posizionamento relativo rispetto ai bordi della schermata o ad altre entità presenti nella stessa. L'utilizzo di quest'ultima soluzione è caldeggiata dagli ingegneri informatici di Google[86], in quanto fornisce numerosi vantaggi, tra cui la possibilità di rendere, in combinazione con altre tecniche e accorgimenti, la schermata responsiva in funzione delle dimensioni e proporzioni del singolo dispositivo su cui è in esecuzione l'applicazione mobile.

- Containers: come suggerisce il nome, gli elementi di questa categoria sono accomunati dalla proprietà di contenere al loro interno altre componenti. Tra i rappresentati più rilevanti, spiccano entità come gli Spinner, in grado di implementare i cosiddetti menu a tendina, e le diffusamente utilizzate

[86] Takeshi Hagikura, Understanding the performance benefits of ConstraintLayout, android-developers.googleblog.com, 2017

RecyclerView, un moderno strumento utilizzato per visualizzare a schermo liste di elementi seguendo un approccio flessibile e performante.

- Helpers: le entità costitutive di questa sezione si differenziano in maniera particolare da quelle appartenenti alle altre categorie fin qui elencate. Queste, infatti, anche se inserite nel layout non appaiono sulla schermata durante l'esecuzione dell'applicazione, in quanto il loro scopo consiste nel fornire allo sviluppatore degli elementi grafici di supporto per la progettazione e la disposizione degli elementi all'interno della schermata. È il caso delle Guideline, a cui è possibile ancorare elementi all'interno di un ConstraintLayout, o dei Layer, che consentono ad esempio di muovere e ruotare coerentemente le entità al loro interno.

- Google: gli elementi di questa categoria sono rappresentati, per definizione, dall'implementazione di servizi proprietari dell'azienda detentrice del sistema operativo Android. Tali elementi si riducono a due soli esponenti, composti dalle AdView, i componenti adibiti alla delimitazione di strisce di contenuti pubblicitari, eventualmente utilizzati come formula di introito per gli sviluppatori e le MapView, elementi atti a contenere finestre di mappe e navigazione fornite dal popolare servizio Google Maps.

- Legacy: all'interno di questa sezione sono presenti gli elementi che nel corso del tempo sono stati deprecati o sostituiti da alternative più funzionali o performanti. È il caso delle ListView, utilizzate per la visualizzazione di elenchi di elementi e sostituite dalle già citate RecyclerView, e del RelativeLayout, che permette l'arrangiamento relativo dei componenti al suo interno, comportamento oggi implementato attraverso il più rigoroso ConstraintLayout.

In aggiunta alle categorie appena elencate, sempre internamente alla stessa porzione costituita dalla Palette, ne è presente una ulteriore, denominata Common. Al suo interno è possibile avvalersi della presenza dei componenti tipicamente più utilizzati, come gli elementi grafici per eccellenza costituiti dalle TextView e dai

Button, così come quelli più supportati ed in voga nel recente passato: è il caso delle RecyclerView.

È stato evidenziato nel corso del paragrafo precedente come, in subordine al fatto che esistono numerose case produttrici di apparecchiature elettroniche con a bordo il sistema operativo Android, siano presenti sul mercato una pletora di dispositivi mobili, ognuno con schermi di dimensioni e rapporti d'aspetto differenti. Per questo motivo, il team di ingegneri di Android ha reso disponibile una serie di consigli e pratiche da seguire al fine di creare layout di schermate che possano adattarsi ai differenti device[87]. All'interno di questa guida, oltre all'indicazione relativa all'utilizzo del già accennato ConstraintLayout, sono presenti altri accorgimenti da mettere in atto, come la creazione di differenti layout alternativi suddivisi per classi di dispositivi in base alla dimensione dello schermo (Compact, Medium, Expanded) e l'evitare il più possibile l'utilizzo di valori fissi nelle dimensioni dei componenti, preferendo a questi etichette per valori dinamici come "wrap_content" e "match constraint", valori che indicano rispettivamente una dimensione sufficiente al contenimento dei componenti interni e una che si estenda fino a occupare tutto lo spazio disponibile, ma senza superare i vincoli imposti nel layout.

Nel corso del Google I/O 2018, ossia durante l'evento annuale di conferenze relative alla presentazione delle ultime novità sul panorama dell'azienda, è stato annunciato il progetto Android Jetpack[88]. È stato considerato come la generazione successiva di componenti Android e l'obiettivo principale di tale soluzione consiste nella riduzione dei tempi di sviluppo delle applicazioni. Jetpack è una raccolta di librerie create per fornire agli sviluppatori: un supporto nell'adozione di alcune migliori pratiche; una riduzione del cosiddetto "boilerplate code", ossia del codice ripetuto più volte senza sostanziali variazioni; una possibilità di scrivere codice che funzioni in maniera consistente attraverso differenti dispositivi e versioni del sistema operativo. In questo modo, gli sviluppatori possono quindi concentrarsi sull'effettivo codice specifico della singola applicazione, senza doversi preoccupare dei suddetti aspetti[89]. In cooperazione con Android KTX, ossia una

[87] Google Developers, Support different screen sizes, developer.android.com
[88] Stephanie Cuthbertson, Google I/O 2018: What's new in Android, android-developers.googleblog.com, 2018
[89] Google Developers, Android Jetpack, developer.android.com

serie di estensioni del linguaggio Kotlin incluse nelle librerie Jetpack, queste ultime gestiscono attività quali l'esecuzione di compiti in background, la navigazione tra schermate dell'applicazione e la coordinazione dei cicli di vita delle varie componenti.

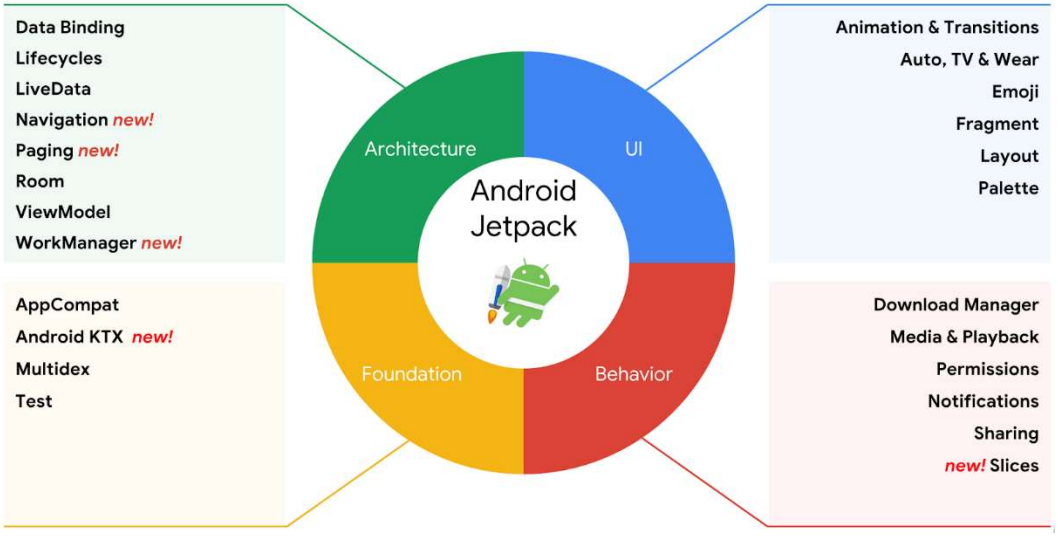

Figura 27 - Componenti delle librerie Android Jetpack[90]

Negli anni l'insieme di queste librerie si è arricchito di nuove entità, a tal punto che le componenti di Android Jetpack sono state raggruppate in quattro categorie principali, delle quali verrà fornita una panoramica qui di seguito[91].

- Foundation: sono le componenti il cui scopo va ad inserirsi nell'ottica di fornire una compatibilità all'indietro nelle versioni più datate di Android, delle funzionalità di test delle componenti dell'interfaccia grafica e un arricchimento del supporto al linguaggio Kotlin. Fanno parte di questo gruppo le già citate estensioni Android KTX, che includono il supporto alle funzioni lambda e alle coroutine, utilizzate per l'esecuzione asincrona del codice.

[90] Chris Sells, Poiesz, Ng, Android Jetpack per accelerare lo sviluppo delle app, developers-it.googleblog.com, 2018
[91] Anand Gaurav, What is Android Jetpack and why should we use it?, blog.mindorks.com, 2019

- Architecture: la funzione delle componenti di questa categoria è da ricercarsi nell'intento di fornire strumenti per la costruzione di applicazioni mobili più robuste, testabili e mantenibili. La lista di queste entità è particolarmente numerosa e contiene al suo interno componenti fondamentali nelle moderne applicazioni, quali: i LiveData, un modo per implementare un observer pattern, anche conosciuto come publish-subscribe, di modo che le viste dell'interfaccia grafica vengano notificate di eventuali cambiamenti nei dati al loro interno e possano quindi reagire opportunamente; Room, una libreria per la persistenza dei dati che fornisce un'astrazione per un accesso robusto ai database implementati con SQLite; ViewModel, una soluzione che permette di gestire, in maniera conscia del ciclo di vita dell'applicazione e delle singole schermate, i dati relativi all'interfaccia grafica, così che nel codice che descrive il funzionamento di quest'ultima siano presenti unicamente logiche relative al comportamento dei suoi componenti.

- Behaviour: i componenti di questo raggruppamento supportano l'integrazione di servizi standard di Android, come il controllo delle notifiche, la gestione dei permessi e la condivisione di file. Esponenti di rilievo della categoria sono costituiti da Download Manager, per la programmazione e la gestione dei download di grandi dimensioni in background, Permissions, API per controllare e richiedere permessi all'interno dell'app e Slices, per la creazione di flessibili elementi dell'interfaccia utente che possano mostrare contenuti dell'applicazione al di fuori della stessa.

- UI: le entità costitutive di questa categoria forniscono elementi e supporto per lo sviluppo di applicazioni al contempo semplici e gradevoli da utilizzare. Tra le appartenenti a questa sezione, è possibile citare componenti quali Animations and Transitions, per i movimenti e gli effetti grafici relativi all'utilizzo degli elementi di una schermata o nella navigazione tra la stessa ed un'altra, ed Emoji2, per un utilizzo retrocompatibile dei popolari simboli pittografici.

4 Applicazione Mobile Android – Prema (provvisorio)

Nei capitoli precedenti sono state presentate delle panoramiche riguardanti i temi che si pongono a contorno del progetto PREMA e successivamente una trattazione relativa alle applicazioni mobili e ad alcune tecniche ed accorgimenti da mettere in pratica, sia nella fase della loro progettazione che in quella della loro creazione. Nel seguito del capitolo corrente, al contrario, verrà invece riportato quanto svolto in merito alla realizzazione dell'applicazione mobile associata al progetto. Al momento della stesura del presente elaborato, l'applicazione ha raggiunto uno stadio di maturità che si colloca, nel ciclo di sviluppo software, a cavallo tra la fase test e integrazione.

Figura 28 - Fasi del ciclo di sviluppo software[92]

In particolare, verranno in primo luogo descritte le motivazioni che hanno portato alla necessità della suddetta applicazione mobile congiuntamente con le scelte progettuali che sono state selezionate. In subordine, verrà presentata l'implementazione pratica della stessa, con una trattazione ed analisi relative alle singole schermate che la compongono, ponendo l'attenzione al contempo sia sull'aspetto grafico dei layout che costituiscono la user interface, sia sulla struttura

[92] Vito Lavecchia, Fasi del ciclo di vita del software in informatica, vitolavecchia.altervista.org

del codice che va ad implementare le logiche necessarie al funzionamento e alla fruizione del software.

4.1 Motivazioni e analisi dei requisiti, funzionalità e progettazione architetturale

Come già indicato in precedenza, nel paragrafo relativo alla presentazione e descrizione del progetto PREMA, oltre ai pannelli di configurazione presenti nel back office e alle dashboard di presentazione erogate come pagine web, uno degli elementi costitutivi del frontend dello stesso è rappresentato da un'applicazione per dispositivi mobili dedicata. In fase di definizione della struttura architetturale del progetto, infatti, è stato messo in evidenza come la presenza di quest'ultima risulti fondamentale al fine di fornire una modalità attraverso la quale l'utilizzatore sia in grado di effettuare un'analisi e maturare una comprensione dei risultati ottenuti in seguito all'elaborazione dei dati raccolti. Il raggiungimento di questo obbiettivo risulta di fatto una questione critica in quanto, in sua assenza, verrebbero vanificate tutte le caratteristiche ed i punti di forza che la porzione rimanente del progetto PREMA vanta. In quest'ottica, per l'appunto, un'applicazione mobile dedicata si configura in maniera esaustiva nel fornire un ulteriore vantaggio per l'utente, in quanto questa è in grado di promuovere e mettere a sua disposizione un accesso alle informazioni in esame in qualsiasi luogo ed in ogni istante, facilitando inoltre le operazioni messe in atto quotidianamente dal personale aziendale, potenziando le loro attività e consentendo una, seppur parziale e circoscritta, digitalizzazione delle stesse. Una volta chiarita la motivazione alla base della necessità di una applicazione mobile dedicata, in una fase preliminare del lavoro svolto per la realizzazione della stessa, sono stati stabiliti i requisiti e le funzionalità che questa dovesse possedere. Queste caratteristiche sono state discusse e selezionate in seguito ad incontri svolti con la partecipazione di tutti gli elementi della squadra impiegata dall'azienda nei molteplici ambiti realizzativi assegnati a Zirak, preposti all'implementazione delle diverse esigenze proprie del progetto. Questa discussione effettuata coinvolgendo non soltanto le persone il cui ruolo imponesse un certo grado di coinvolgimento nella realizzazione dell'applicazione, si è rivelata fondamentale al fine di generare una consapevolezza d'insieme dei vari ambiti

progettuali, fornendo come risultato l'adozione di alcune strategie implementative che non fossero intraprese in un'ottica di separazione netta dei confini di interesse dei singoli settori di lavoro, bensì perseguendo l'obiettivo di una intensa combinazione in grado di rendere armonicamente collegate le varie componenti architetturali e produrre un'interazione naturale ed efficace tra le stesse. A titolo esemplificativo, le API di cui fa uso l'applicazione per il reperimento dei dati relativi alle diverse entità di interesse, che verranno riprese e presentate nel seguito del capitolo corrente del presente elaborato, non sono state messe a punto tenendo in considerazione le sole esigenze specifiche della stessa, ma generate in maniera da consentirne un utilizzo esteso ad altre entità integrate nel progetto, come le già citate dashboard di presentazione su pagine web. Per quanto concerne la trattazione dei requisiti stabiliti durante questi incontri, è stato deciso in primo luogo che l'applicazione dovesse essere connessa ad Internet per il suo funzionamento. Nonostante, infatti, la presenza di logiche e funzionalità che vengono implementate dall'applicazione strettamente in locale, senza cioè la necessità di una comunicazione esterna al dispositivo su cui essa è in esecuzione, per la realizzazione di alcuni ulteriori comportamenti, quali un generico reperimento di dati immagazzinati su una base di dati contenuta all'interno di un server esterno, non possono, proprio per la propria natura, prescindere dalla possibilità di una connessione, rendendo necessario questo vincolo funzionale. Questa condizione ha reso necessario, durante una successiva fase implementativa dell'applicazione, la presenza all'interno di un file specifico dei sorgenti dell'applicazione la seguente porzione di codice:

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

In ogni progetto relativo ad un'applicazione mobile Android, infatti, deve sempre essere incluso, nella cartella radice dello stesso, un file denominato "AndroidManifest.xml"[93]. Questo file, scritto come per l'appunto indica la sua estensione nel metalinguaggio XML, descrive al suo interno una serie di informazioni inerenti all'applicazione, ritenute essenziali per la fruizione delle stesse da parte degli strumenti di compilazione Android, dall'omonimo sistema

[93] Google Developers, App Manifest Overview, developer.android.com

operativo e dallo store di riferimento Google Play. Tra le altre cose, il file manifest deve dichiarare indicazioni riguardanti le seguenti categorie:

- I componenti costitutivi dell'applicazione, come ad esempio le Activity[94], ossia le classi elementari di una app mobile che si occupano della creazione delle schermate e della gestione delle interazioni che l'utente può avere con esse. Per ogni componente devono essere descritte alcune sue proprietà di base come il relativo nome della classe Kotlin.

- Le caratteristiche ed i requisiti che l'applicazione richiede in termini di risorse hardware, come la presenza di determinati apparati elettronici, e software, come la versione minima di SDK e sistema operativo necessaria, che il dispositivo mobile deve possedere al fine di risultare idoneo per l'installazione al suo interno della suddetta. Se questi vincoli non sono soddisfatti, per l'appunto, in seguito ad un tentativo di acquisto di una certa app sullo store Google Play, quest'ultimo dopo aver verificato ed individuato la non compatibilità del dispositivo in questione non ne permette l'installazione, motivando la causa scatenante.

- I permessi dell'applicazione, intesi sia come le autorizzazioni di cui la stessa necessita per fare accesso a porzioni protette del sistema in termini di dati o azioni sottoposti a restrizioni, sia come quelle di cui ha bisogno una ulteriore app per utilizzare i contenuti presenti nella suddetta.

La stringa precedentemente citata, relativa all'utilizzo di Internet all'interno dell'applicazione, fa appunto parte delle autorizzazioni che è necessario dichiarare nell'ambito dell'ultima categoria indicata, ossia quella dei permessi. Se questa fosse stata omessa, in fase di connessione e inizializzazione di una richiesta HTTP quest'ultima non sarebbe andata a buon fine, pregiudicando il comportamento atteso dall'applicazione. Il permesso per l'utilizzo di Internet, etichettato come autorizzazione normale al pari di numerosi altri permessi, è un tipo particolare di

[94] Google Developers, Activity, developer.android.com

autorizzazione che è incluso nelle cosiddette "Install-time permissions"[95]. Questa tipologia di autorizzazioni, come viene suggerito dal nome stesso, vengono automaticamente garantite dal sistema operativo in fase di installazione dell'applicazione, in quanto sono ritenute in grado di accedere a dati o compiere azioni protette in maniera limitata, con un grado potenziale di influenza su altre applicazioni o sul sistema operativo in sé considerato minimo. Tipicamente, sugli store di applicazioni queste autorizzazioni vengono elencate in una lista contenuta nei dettagli delle informazioni relative all'applicazioni, così che l'utente intenzionato all'acquisto ne sia informato. In contrapposizione a questa categoria di autorizzazioni, ne è presente una che racchiude permessi più delicati, ossia le "Runtime permissions". In questo caso, il grado di influenza che queste possono scatenare è largamente più elevato, motivo per cui le stesse vengono considerate autorizzazioni pericolose. La maggioranza di queste autorizzazioni, infatti, è relativa all'accesso a dati privati dell'utilizzatore o del suo dispositivo, che possono includere informazioni anche sensibili. Per questo motivo, proprio perché l'utente che sta utilizzando l'applicazione sia estremamente consapevole della delicatezza insita nella concessione di tali permessi, un suo consenso esplicito è richiesto in fase di esecuzione di ogni app mobile che richieda l'autorizzazione ad uno di questi ultimi, attraverso un prompt generato dal sistema operativo grazie al quale l'utilizzatore sia in grado di concedere o negare il suo benestare. Nonostante la portata e la delicatezza di questa tipologia di autorizzazioni, non soltanto non è raro che le applicazioni le richiedano, ma è anzi pratica comune in numerosissime casistiche. Relativamente al tema delle autorizzazioni richieste dalle applicazioni mobili, dal punto di vista degli sviluppatori queste rappresentano una questione a cui essi devono porre particolare attenzione, in quanto spesso, come precedentemente indicato, alcune funzionalità possono portare all'utilizzo di informazioni private e sensibili e, se non correttamente trattati, dare luogo a comportamenti inattesi che si traducono in funzionamenti anomali o pericolosi delle applicazioni. In molti casi, inoltre, è possibile che la funzionalità considerata, se correttamente implementata, possa essere fornita senza la necessità di dichiarare e conseguentemente richiedere permessi particolari. La fase di studio delle

[95] Google Developers, Permissions on Android, developer.android.com

autorizzazioni si configura come un compito da svolgere attentamente anche per questa motivazione.

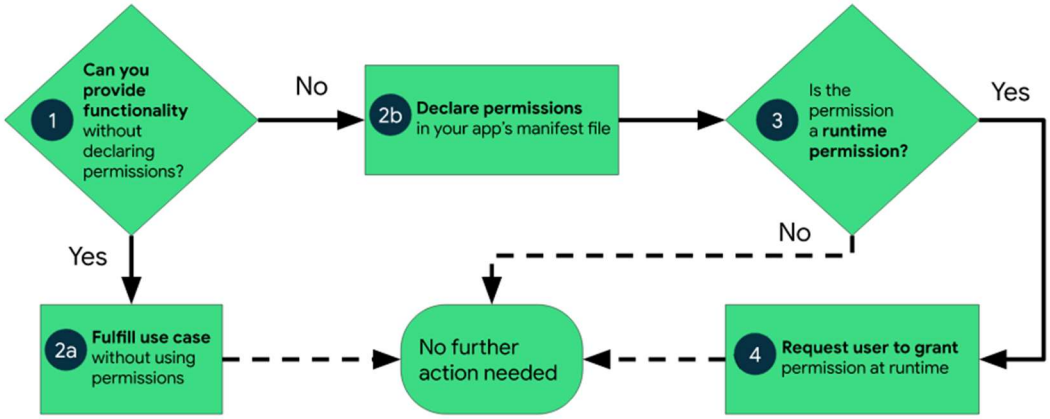

Figura 29 - Diagramma di flusso relativo alla dichiarazione di autorizzazioni per applicazioni mobili Android[96]

Come ulteriore prerogativa definita in fase di analisi dei requisiti dell'applicazione considerata, è stato stabilito che la stessa dovesse risultare in grado di poter essere installata su dispositivi mobili con schermi di diverse dimensioni e rapporti di forma. In particolare, come piattaforma obiettivo sono stati considerati gli smartphone Android di qualsiasi casa produttrice. Per far fronte a questo ulteriore vincolo progettuale, nella fase implementativa è stato necessario tenerne conto nella realizzazione dei layout delle varie schermate. All'interno di questi ultimi infatti, fatto salvo di alcune inevitabili eccezioni, è stato fatto ampio impiego del già descritto Constraint Layout, per la disposizione degli elementi grafici costitutivi del layout attraverso la definizione di vincoli posizionali relativi ai confini della schermata o delle altre entità presenti sulla schermata, in accoppiata con l'utilizzo, per la definizione delle dimensioni degli elementi grafici e dei loro contenuti, di valori dinamici come wrap_content e match constraint, a loro volta descritti in precedenza, così come di valori espressi in dp ed sp[97]. Per dp si intende dei Density-independent Pixels, ossia un valore dimensionale che, invece di tener conto di dimensioni fisse fortemente dipendenti dallo schermo e dalla sua risoluzione come

[96] Google Developers, permissions workflow-overview, developer.android.com
[97] Google Developers, Dimension, developer.android.com

centimetri o numero di pixel, fa riferimento alla densità dello schermo, in modo che ci sia una consistenza relativa alla dimensione assunta nel mondo fisico dai singoli elementi grafici sui vari dispositivi su cui l'applicazione può essere installata. Le unità dimensionali sp, ossia Scale-independent Pixels, funzionano allo stesso modo dei dp, ma in aggiunta scalano la propria dimensione in funzione delle preferenze del dispositivo impostate dall'utilizzatore in termini di font grafico. Per questo motivo, in caso di dimensioni da assegnare a caratteri alfanumerici di campi testuali presenti sulle varie schermate, è stata fornita una dimensione in sp. In aggiunta a quanto indicato finora, relativamente alla discussione dei requisiti propri dell'applicazione considerata, è stato stabilito che questa dovesse essere, almeno in una sua prima versione, implementata ed erogata in lingua inglese, coerentemente con le informazioni immagazzinate sul database relativo. In ultima analisi, sono state effettuate delle considerazioni in merito funzionamento dell'applicazione in funzione dell'orientamento dello schermo. La porzione più significativa dei dispositivi mobili presenti sul mercato, infatti, presenta nella sua dotazione hardware accelerometri e giroscopi in grado di stabilire l'orientamento del dispositivo. Il sistema operativo Android è in grado di intercettare eventuali rotazioni dello stesso e notificare gli eventi relativi all'applicazione che è quindi in grado di reagire in base al comportamento descritto al suo interno in fase di programmazione. Le applicazioni possono infatti essere visualizzate in modalità portrait, nel caso in cui il dispositivo sia orientato in modo tale da rendere la larghezza visiva dello schermo più piccola dell'altezza dello stesso, o in alternitiva in configurazione landscape, ossia nella circostanza inversa in cui la larghezza visiva dello schermo risulta maggiore della sua altezza[98]. Lavorando sull'IDE Android Studio è possibile, in fase di creazione di un nuovo layout, configurare quest'ultimo in modo che da rendere esplicita la sua implementazione per un utilizzo con il dispositivo orientato in una delle due suddette modalità. Questo costituisce un vantaggio per lo sviluppatore, in quanto fornisce la possibilità di creare per ogni schermata dell'applicazione mobile due layout, corrispondenti ad entrambi gli orientamenti assumibili dai dispositivi mobili, in cui gli elementi grafici contenuti al loro interno possano essere arrangiati e dimensionati in maniera

[98] Google Developers, ScreenOrientation, developer.android.com

differente per le due configurazioni, di modo che la disposizione risultante si dimostri conveniente ed efficace, a seconda dello spazio disponibile sui due assi orizzontale e verticale. In seguito, durante l'esecuzione dell'applicazione, il layout utilizzato per la presentazione della schermata corrente viene selezionato e caricato in funzione della configurazione di orientamento in tempo reale del dispositivo. Considerando le tempistiche a disposizione per lo sviluppo dell'applicazione, a tal proposito è stato stabilito di implementare la stessa in modo che venisse erogata nella sola modalità portrait, al fine di destinare le risorse per implementazioni considerate, sul momento, prioritarie nei confronti di questo aspetto. Per questo motivo, all'interno del già citato file manifest, è stata inserita nei dettagli relativi alle proprietà di ogni Activity dell'applicazione, la seguente stringa:

```
android:screenOrientation="portrait"
```

Quest'ultima indicazione segnala l'intenzione che l'Activity in questione sia da considerarsi erogabile nella sola modalità portrait, in modo da evitare la rotazione del layout relativo nel caso di orientamento del dispositivo nella configurazione landscape. In assenza di questa segnalazione, infatti, non essendo stato implementato per la schermata corrente un layout dedicato alla modalità landscape, sarebbe stato comunque caricato il layout della modalità portrait, ma ruotato di 180°, con il risultato della generazione di una interfaccia utente non ottimale e, nei casi più estremi, utilizzabile nella non totalità delle sue funzioni.

Terminata la fase di analisi dei requisiti, l'attenzione degli incontri e discussioni relativi all'applicazione per il progetto PREMA è stata spostata e focalizzata su un differente argomento principale, quello ossia inerente alle funzionalità da implementare per la stessa. Queste, come spesso accade in progetti di tale tipologia, sono state definite in questa fase per poi essere successivamente, durante la seguente illustrazione dei modelli dei layout delle schermate, affinate e rettificate di conseguenza. L'applicazione è rivolta a due tipologie principali di utilizzatori: gli operatori di macchina ed il personale tecnico adibito all'analisi dei processi produttivi. Per questo motivo, l'applicazione è strutturata per rispondere alle esigenze delle due differenti entità di destinazione, assumendo una diversa

configurazione di schermate e flusso di navigazione tra le stesse. È stato stabilito, a tal proposito, che fosse presente come schermata iniziale dell'applicazione una vista destinata a fornire una procedura di autenticazione e login. In seguito a quest'ultima, l'applicazione avrebbe dovuto riconoscere la tipologia di utente autenticatosi in base alle autorizzazioni corrispondenti alle credenziali del suo profilo e procedere con la navigazione alla corrispondete schermata principale. In particolare, le funzionalità previste per un utente assegnato alla prima tipologia indicata, quella relativa ad un operatore di macchina e nel seguito indicato come "utente view" o "view account", sono le seguenti:

- Possibilità di consultazione rapida ed immediata della lista degli allarmi presenti sul database in tempo reale, di modo che possa reagire di conseguenza
- Possibilità di consultazione dei dettagli del singolo allarme considerato
- Possibilità di consultazione delle macchine e dei sensori presenti sul database
- Possibilità di consultazione dei dettagli della singola macchina o del singolo sensore considerato
- Possibilità di poter navigare in qualunque momento alla schermata principale
- Possibilità di consultazione e modifica delle credenziali dell'utente che ha effettuato il login, così come la consultazione dei permessi garantiti all'account in questione
- Possibilità di logout con conseguente navigazione alla schermata di login

In aggiunta alle suddette, sono state stabilite le funzionalità per la fruizione da parte di un utente che faccia parte del personale tecnico adibito all'analisi dei processi produttivi, nel seguito indicato come "utente edit" o "edit account". Queste comprendono altresì tutte le funzionalità descritte per gli utenti view, con la differenza che invece di visualizzare una lista di allarmi attualmente in atto, vengono messe a disposizione dell'utente edit delle etichette che indichino il numero di allarmi in corso nelle diverse categorie. In aggiunta a tali funzionalità

precedentemente elencate, per un edit account sono state previste al contempo le ulteriori seguenti:

- Possibilità di consultazione dei modelli di machine learning presenti sul database
- Possibilità di consultazione dei dettagli del singolo modello, così come delle statistiche relative
- Possibilità di creazione ed aggiunta di un nuovo modello, a partire da una lista di skeleton disponibili
- Possibilità di effettuare il retrain di un determinato modello

Nei proseguo dell'elaborato verrà presentata una spiegazione più approfondita delle singole entità prese in considerazione, come allarmi, macchine e modelli, così come del loro significato, dei loro campi caratteristici e dei loro utilizzi all'interno delle logiche del progetto in maniera contestualizzata sulla applicazione. Una volta conclusa la definizione delle funzionalità desiderate, ha avuto luogo un periodo di lavoro dedicato alla produzione dei modelli di riferimento per la successiva implementazione effettiva dei layout grafici delle schermate.

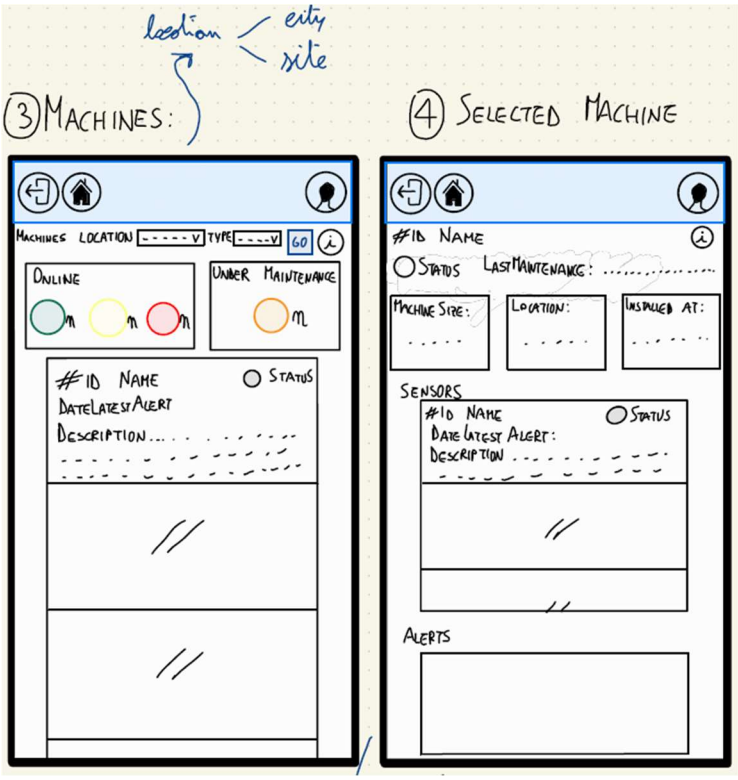

Figura 30 - Esempio di modelli di layout grafici prodotti

Questa attività è stata svolta in diverse occasioni, partendo da un'idea di base modificata ed ampliata di volta in volta. Tra la produzione di ogni fascicolo di template e quello successivo, sono stati effettuati degli incontri sistematici con tutti i membri aziendali del progetto, al fine di analizzare e discutere i risultati ottenuti fino a quel momento per stabilire eventuali modifiche, aggiunte o nuovi accorgimenti volti a migliorarne la qualità acquisita. Al termine di queste rielaborazioni è stata svolta una validazione conclusiva che ha portato ad un definitivo fascicolo di modelli di layout grafici. All'interno di quest'ultimo, è stato stabilito che fossero inserite 14 schermate, di cui alcune dedicate ad una singola tipologia di account ed altre condivise da entrambe.

In questa fase preliminare di analisi e progettazione, un'ultima ulteriore porzione significativa del lavoro svolto ha avuto a che fare con la definizione delle componentistiche strutturali volte a costituire l'architettura dell'applicazione mobile. Come per le attività presentate in precedenza, anche questa mansione è stata in parte svolta attraverso uno sforzo condiviso da tutti gli elementi della squadra aziendale del progetto. In questo caso particolare, tuttavia, data la sua spiccata specificità in termini di componentistiche proprie del sistema operativo di riferimento, per il completamento dell'attività in questione si è rivelato più efficace un approccio più circoscritto al personale tecnico specializzato. Per la realizzazione dell'applicazione sono stati seguiti dei criteri che si sposassero con la filosofia dell'intero progetto. A tal proposito, è stato preso in considerazione come, al fine di fornire una piattaforma per la manutenzione predittiva che risultasse modulare, scalabile e decentralizzata, il progetto PREMA consista in un'aggregazione di differenti servizi, ognuno con i relativi sottosistemi ed operazioni eseguite in maniera isolata rispetto agli altri. Per gli scambi di dati ed informazioni necessari tra questi servizi, sono stati definiti molteplici protocolli e parametri di comunicazione. Con questo concetto in mente, è stato stabilito che l'applicazione Android dovesse essere realizzata in maniera conforme a questa idea di modularità e decentralizzazione e, in quest'ottica, fare tra le altre cose uso di alcuni dei suddetti protocolli di comunicazione. Tra tutti quelli messi a disposizione, quelli elencati nel seguito sono stati considerati come maggiormente pertinenti e adatti per lo sviluppo dell'applicazione.

- Per le comunicazioni con il database MongoDB, era stato inizialmente proposto l'utilizzo di un driver apposito denominato Mongo Java Driver. Quest'ultimo, proprio perché i principali linguaggi di programmazione per lo sviluppo di applicazioni Android sono Java e Kotlin, entrambi basati su Java Virtual Machine, era stato considerato come l'opzione più naturale per il compito preposto. Tuttavia, in seguito ad un'ulteriore analisi, è stato considerato come l'accesso diretto al database potesse non essere la soluzione ottimale[99]. Come alternativa, è stata proposta ed in seguito confermata ed applicata la creazione di un insieme di API che avessero il compito di interagire con il database. Infatti, esistono due principali motivazioni per cui uno sviluppatore potrebbe voler preferire l'utilizzo di API in contrapposizione al diretto accesso e consultazione del database, che hanno a che fare con la semplicità di utilizzo e la sicurezza. In quest'ottica, infatti, l'utilizzo di API permette l'introduzione di uno strato di astrazione aggiuntivo che consente allo sviluppatore di non doversi preoccupare della particolare implementazione del database e degli specifici meccanismi di interrogazione delle sue entità, senza quindi dover avere che fare con tematiche legate al versionamento e alla sintassi propria delle suddette interrogazioni. In aggiunta a questa considerazione, l'utilizzo di API consente di fare fronte alle criticità in termini di sicurezza che l'accesso diretto al database potrebbe comportare. Infatti, è possibile costruire le suddette in modo da essere sicuri che i soli utenti autorizzati siano in grado di accedere ai contenuti del database e che soltanto le richieste valide siano in grado di interagire con le informazioni corrette, prevenendo la possibilità di comportamenti malevoli da parte di utenti malintenzionati. Per le suddette motivazioni, è stato messo a punto un insieme di API per il reperimento delle informazioni necessarie alla corretta erogazione delle funzionalità messe a disposizione dall'applicazione. Tali informazioni sono relative alle svariate entità di interesse da parte della suddetta, come, ad esempio, i dettagli relativi a macchine, sensori, allarmi e modelli di machine learning.

[99] DaNeil C, API =/= Database - Why Would You Want to use an API to Talk to a Database Rather Than Just Query The Database?, dev.to, 2020

Come spesso accade, tali API sono state rese fruibili attraverso il ricorso a richieste http.

- Per le comunicazioni con il backend, relative a tematiche inerenti ai suoi processi di machine learning, sono state utilizzate delle API dedicate. Proprio per la natura strettamente legata a fattori temporali caratteristica dei modelli di predizione relativi, queste forniscono gli strumenti per effettuare le predizioni, il training, il retraining e la cancellazione dei modelli esistenti, così come la creazione di nuovi. Anche in questo caso, le suddette API sono state disposte per essere usufruite tramite richieste http.

In aggiunta a queste API, ne sono state predisposte, seppur in maniera soltanto prototipale al momento della stesura del presente elaborato, alcune che consentano una comunicazione con il backend in merito alla configurazione dei sensori considerati nel progetto. Nonostante questa funzionalità non sia stata inclusa nell'elenco relativo dell'applicazione mobile in esame, non è infatti escluso che in futuro questa possa essere discussa ed eventualmente approvata ed integrata.

Dopo aver stabilito quali fossero le connessioni esterne necessarie e i protocolli di comunicazione attraverso i quali queste sarebbero state instaurate, sono state fissate le componenti strutturali volte a generare nel loro complesso l'architettura vera e propria dell'applicazione. Il primo fattore considerato è stata l'implementazione del cosiddetto pattern MVVM, sigla che sta ad indicare il termine Model, View, ViewModel. La decisione di seguire un pattern di questo genere è stata presa in seguito alla considerazione secondo cui inserire tutto il codice relativo alla gestione dell'interfaccia utente e alla presentazione dei dati al suo interno, così come quello relativo al reperimento e all'immagazzinamento di tali dati, avrebbe portato come risultato la generazione di un codice verboso, difficilmente mantenibile e più prono alla generazione di errori o comportamenti inaspettati[100]. Implementando tali attività secondo il pattern considerato è possibile al contrario ottenere una separazione strutturale delle stesse, con tutti i benefici che questo comporta anche

[100] Anupam Chugh, Android MVVM Design Pattern, journaldev.com

in ottica di un eventuale futuro ampliamento o cambiamento delle funzionalità dell'applicazione in sé. Per quanto riguarda i termini dell'acronimo MVVM, Model sta per la componente che contiene i dati in sé, View sta per la componente adibita alla creazione dell'interfaccia utente e della gestione dei suoi comportamenti in seguito all'interazione con l'utente, mentre ViewModel rappresenta lo stadio intermedio tra le prime due, è quindi una componente responsabile di fornire alla componente View i dati contenuti nella componente Model in maniera che siano fruibili dalla prima. Al netto di questa considerazione sull'implementazione di un pattern di tipo MVVM, l'architettura stabilità per l'applicazione si concretizza in quella illustrata nella figura seguente:

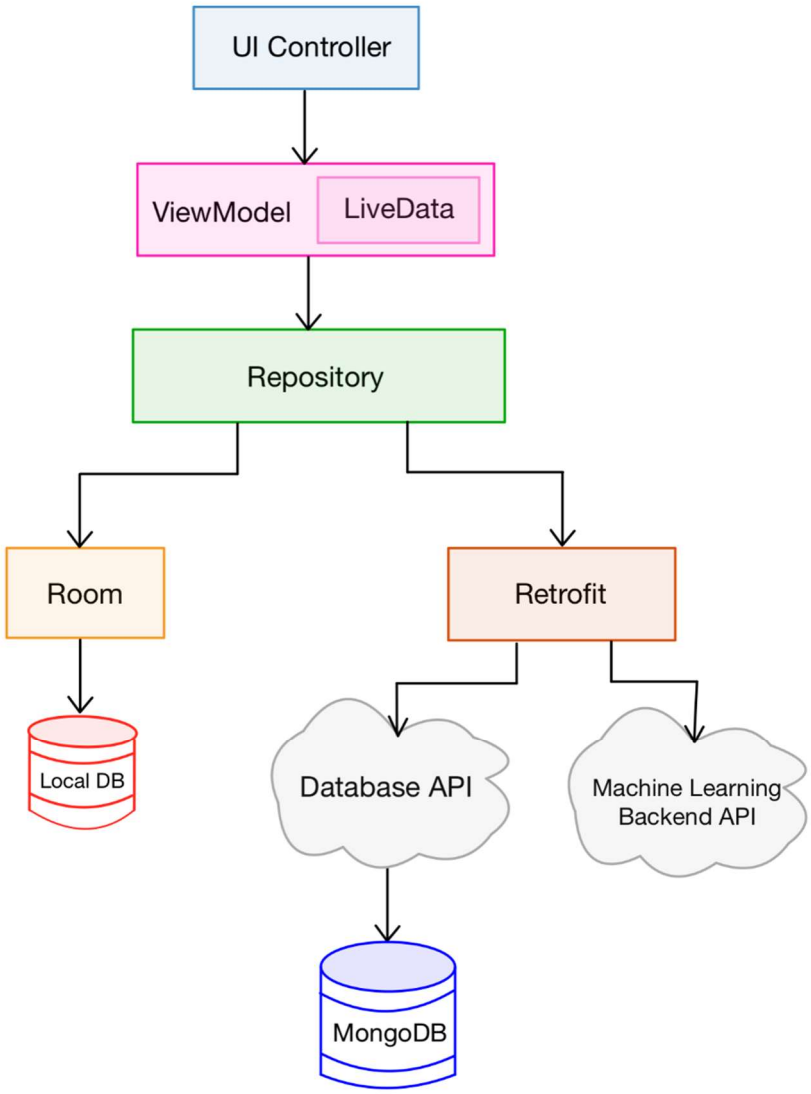

Figura 31 - Schema dell'architettura dell'applicazione mobile per il progetto PREMA

Nel seguito verranno elencate le principali entità che compongono tale architettura, indicando per ognuna di esse il proprio ruolo e significato.

- UI Controller: è la componente adibita alla presentazione della schermata contenente l'interfaccia con l'utente, alla gestione dei comportamenti degli elementi grafici contenuti al suo interno in seguito ad interazioni con l'utente o ai segnali provenienti dall'esterno, così come alla navigazione verso le ulteriori schermate dell'applicazione. L'UI controller può essere costituito da una semplice Activity o da una Activity contenente al suo interno uno o più Fragment: nel caso dell'applicazione in esame è stato scelto di impiegare solamente Activity della prima tipologia. In relazione al precedentemente descritto pattern MVVM, l'UI controller è la comopnente che assume il ruolo di View. L'UI Controller dipende dal ViewModel corrispondente, in quanto il suo ciclo di vita è da quest'ultimo gestito e i dati con cui è popolata l'interfaccia utente sono altresì forniti dallo stesso.

- ViewModel: è l'entità che si interpone tra l'UI Controller e i dati di cui esso necessità per la correttezza delle sue funzionalità. Di norma, per ogni UI Controller è implementato un ViewModel. Fanno eccezione i casi in cui due UI Controller posseggano le stesse necessità, per cui la creazione di un secondo ViewModel risulterebbe ridondante. La presenza di questa componente è importante in quanto i dati contenuti al suo interno sopravvivono ad eventuali cambi di configurazione della schermata. Un esempio di questi ultimi è costituito dal cambio dell'orientamento del dispositivo. In una situazione del genere, infatti, il sistema reagisce al cambio di configurazione distruggendo la vista corrente e generata in precedenza, con la chiamata del metodo onDestroy() dell'UI Controller, creandone al suo posto una nuova, con la successiva chiamata del metodo onCreate(). In questa sequenza vengono distrutti tutti gli elementi grafici della schermata considerata e di conseguenza anche tutti i dati al loro interno, sia che essi costituiscano informazioni inserite dall'utente sia che risultino da eventuali elaborazioni od operazioni di lettura interne. In un'implementazione in cui, al contrario, sia presente un ViewModel, i

suddetti dati non vengono persi in quanto contenuti non direttamente all'interno dell'UI Controller, ma nel ViewModel stesso, il quale al contrario del primo non viene distrutto e ricreato ad ogni cambio di configurazione. Nella figura seguente è possibile a tal proposito constatare questo diverso comportamento tra una Activity ed il relativo ViewModel, in seguito alla rotazione dello schermo del dispositivo mobile su cui è in esecuzione l'applicazione in esame.

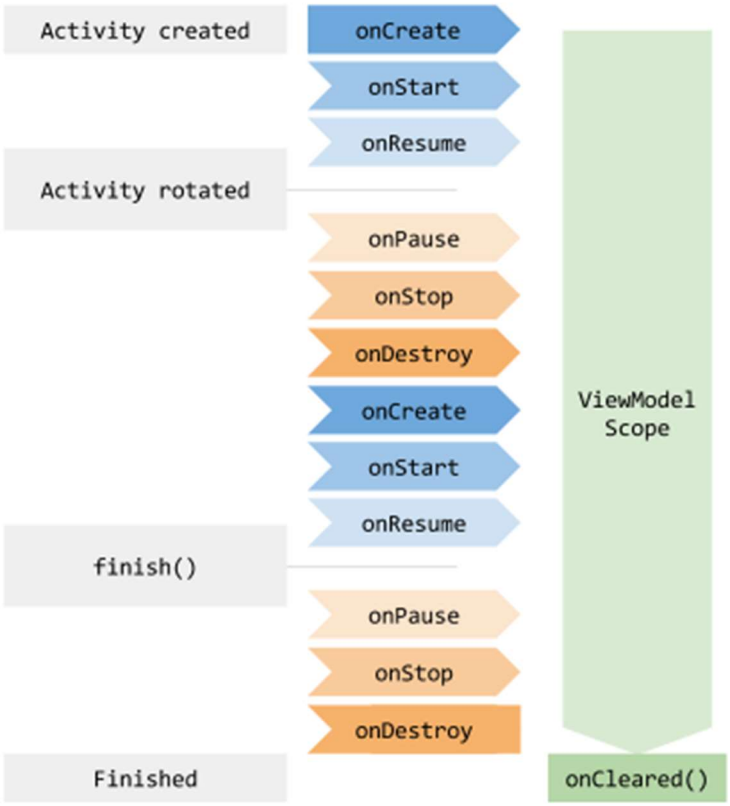

Figura 32 - Confronto tra il ciclo di vita di una Activity e del relativo ViewModel in seguito alla rotazione del dispositivo mobile[101]

I dati non generati direttamente dalla logica della applicazione, di cui fanno parte, ad esempio, contatori o altre variabili di appoggio contenute nel ViewModel, ma proveniente da una sorgente esterna, non vengono recuperati direttamente dal suddetto, ma attraverso la chiamata di funzioni che vadano ad interpellare l'entità architetturale definita come Repository. Nella maggior parte dei casi, questi dati non vengono forniti sotto forma di

[101] Google Developers, ViewModel Overview – viewmodel-lifecycle, developer.android.com

semplici istanze della tipologia di classe a cui fanno riferimento, ma vengono racchiusi in un oggetto di tipo LiveData. Come già indicato in precedenza, l'importanza degli oggetti di questo tipo è dovuta al fatto che, grazie alla loro proprietà di essere osservabili, le entità che ne fanno uso vengono automaticamente informate dei cambiamenti al loro interno, così che eventuali modifiche apportate da entità terze vengano propagate in tutti gli elementi attivi dell'applicazione che li utilizzano.

- Repository: è la componente che si occupa della comunicazione dei dati necessari per il corretto funzionamento dell'applicazione. La particolarità della suddetta è che si configura come l'unica entità che dipende da molteplici componenti, invece che da una singola come per le altre. È responsabile di fornire ai ViewModel che ne posseggono un'istanza le funzioni che permettano l'utilizzo delle API per il reperimento dei dati esterni al dispositivo, così come di quelle che consentano di effettuare operazioni di inserimento, lettura, modifica o cancellazione dei dati archiviati localmente sul dispositivo. Per sopperire a tali necessità, il Repository fa affidamento sui servizi messi a disposizione da due ulteriori componenti architetturali: Room e Retrofit.

- Room: è una libreria messa a punto dalla squadra di sviluppatori Android di Google, la quale permette l'utilizzo di database locali interni all'applicazione, isolati dalle altre porzioni della stessa. Il suo funzionamento è dovuto a classi con le seguenti annotazioni:

  @Entity, per segnalare alle librerie di Room che una certa classe rappresenti una tabella all'interno del database, i cui membri corrispondano alle colonne della tabella;

  @Dao, sigla per Data Access Object, rappresentano le classi al cui interno sono contenuti i metodi di accesso al database, in cui sono descritte le query relative. In particolare, per oggi entità contenuta nel database è tipicamente codificato un DAO specifico;

@Database, corrisponde alla classe che funge da punto di accesso principale per la connessione ai dati sottostanti. All'interno della stessa è presente una lista delle classi che rappresentano le entità incluse nel database, le funzioni astratte per il reperimento dei relativi DAO e la versione corrente del database. Solitamente la classe etichettata con questa annotazione, che estende la classe RoomDatabase, è implementata seguendo il singleton pattern, in modo da possedere una singola istanza condivisa dal resto dell'applicazione.

- Retrofit: è l'entità che costituisce il più stabile è comune client http in ambiente Android. È costituito da una libreria che permette al Repository di effettuare richieste http, quali ad esempio POST e GET, alle API sviluppate e messe a disposizione per il reperimento di dati esterni, come quelle precedentemente descritte per la comunicazione con il database del progetto ed il backend. Le operazioni effettuate dalle API sono invisibile ed isolate rispetto all'applicazione, in modo che in seguito ad un malfunzionamento delle suddette questa possa non soffrire di malfunzionamenti critici.

4.2 Codice sviluppato per il funzionamento dell'Applicazione

In seguito alla presentazione sul lavoro svolto nelle fasi di analisi, pianificazione e progettazione architetturale e dei layout grafici delle schermate dell'applicazione mobile per il progetto PREMA, verrà in questo paragrafo descritto quanto eseguito nella successiva fase implementativa della stessa. Il periodo di tempo impiegato per l'attività in questione è risultato, per la mole di lavoro necessaria al suo completamento, di gran lunga più significativo rispetto a quanto realizzato nelle fasi precedenti. L'obiettivo di questa sezione dell'elaborato consiste nella presentazione delle schermate contenute nell'applicazione e delle logiche a loro contorno. Prima di procedere a tale trattazione, risulta tuttavia necessaria una presentazione delle entità di interesse nell'ambito della suddetta, al fine di fornire le informazioni utili a meglio comprendere come queste siano collegate tra di loro attraverso il flusso logico del codice relativo. Per questo motivo, tali entità verranno qui di seguito elencate, attraverso una lista che mostri per ognuna di esse il nome del modello

relativo, ossia quello della corrispondente classe Kotlin, i principali campi contenuti al suo interno ed il significato che le suddette possiedono.

- Alert: rappresenta gli allarmi che possono venire generati e visualizzati all'interno dell'applicazione. Per ogni alert, oltre a campi volti ad illustrare delle caratteristiche proprie dello stesso, come una data di creazione ed una descrizione, riveste una particolare importanza la presenza dei campi "origin" ed "origin_ID". Ogni allarme infatti, può essere generato a partire da tre sorgenti differenti: un sensore, nel caso in cui un valore misurato da parte di un tale dispositivo montato su una particolare macchina ecceda una certa soglia predefinita; un output di un modello, quando cioè i meccanismi di machine learning implementati per una certa macchina scatenino il sistema di manutenzione predittiva per segnalare una previsione di gusto della stessa; una macchina, nei casi in cui il macchinario stesso sia dotato di sistemi per il riconoscimento e la relativa comunicazione di malfunzionamenti. A tal proposito, i campi menzionati in precedenza risultano specialmente significativi in quanto descrivono, rispettivamente, la tipologia di entità che ha scatenato l'alert e il suo codice identificativo univoco, di modo che sia possibile recuperarne i dettagli in una schermata dedicata.

- Machine: rappresenta i macchinari industriali per cui è stata manifestata l'intenzione di renderli dotati di un sistema di manutenzione predittiva. Al momento dell'inserimento nel database dei dettagli di una specifica macchina, infatti, questa può non essere soggetta all'analisi di nessun modello di predizione, ma è possibile che tale eventualità possa essere implementata in futuro. Oltre ai campi relativi ai dettagli della macchina considerata, come il nome della stessa ed il luogo in cui essa è installata, ne sono presenti di spiccata rilevanza, come la lista degli allarmi generati, dei modelli attivamente presenti sulla macchina e dei sensori di cui essa è equipaggiata, così come la data dell'ultima manutenzione effettuata sulla stessa.

- Model: questa classe è utilizzata per rappresentare i modelli di machine learning che possono essere applicati ad un macchinario sul quale si intenda implementare un sistema di manutenzione predittiva. Ogni modello si basa su uno skeleton, che può essere inteso come una struttura comune a tutti i modelli che ne fanno uso. In altre parole, un modello può essere visto come un'istanza di un certo skeleton, in cui sono stati impostati diversi parametri e configurazioni. Questi ultimi valori costituiscono una parte degli attributi di ogni modello, in aggiunta a numerosi altri campi. Tra i più significativi è possibile evidenziare la macchina su cui il modello è stato implementato, lo stato del modello, le sue performance ed altre statistiche proprie dello stesso.

- Sensor: rappresenta i sensori che possono essere installati a bordo di un macchinario, al fine di registrate e monitorare alcuni valori e grandezze, come temperature e vibrazioni, che possano essere ritenuti significativi al fine di fornire indicazioni sul suo stato di operatività o per evidenziare eventuali disallineamenti dai parametri standard di funzionamento. Oltre a contenere indicazioni riguardo il codice identificativo della macchina su cui sono installati, tra i campi presenti all'interno della classe che modellizza tali dispositivi sono presenti una lista degli allarmi generati e un valore di soglia, il cui superamento porta per l'appunto alla formazione di un nuovo ed ulteriore allarme che si vada ad aggiungere alla suddetta lista.

- Skeleton: è la classe preposta alla rappresentazione degli skeleton dei modelli di machine learning. Oltre ai campi descrittivi e identificativi, sono presenti due attributi principali che caratterizzano uno skeleton: il datatype e gli hyperparameters. Per quanto riguarda il primo parametro, come viene suggerito dal nome, esso contiene un'indicazione sulla tipologia di dati utilizzata dallo skeleton, che può assumere uno tra i valori di raw data, ossia nelle situazioni in cui i dati che sfrutta non abbiano subito preelaborazioni, e statistical features, quando cioè le informazioni in essi contenute siano utilizzate per fornire indicazioni di natura statistica.

La lista delle classi utilizzate all'interno dell'applicazione per modellare le entità di interesse conta, al momento della stesura del presente elaborato, un numero di elementi pari a 20. Tuttavia, la grande maggioranza delle restanti è soltanto utilizzata per modellare dei campi specifici di quelle elencate in precedenza. A queste è possibile aggiungerne una ulteriore, seppure questa non sia direttamente collegata alla descrizione di elementi fisici o astratti coinvolti nella realizzazione di un sistema per la manutenzione predittiva. Ciononostante, infatti, per il corretto funzionamento dell'applicazione mobile risulta imprescindibile la presenza di una classe che funga da modello per l'entità che più di tutte è fonte di interazione con la suddetta, ossia l'utilizzatore in sé. A tal proposito è stata creata la classe User, che contenga al suo interno, in aggiunta ai campi per lo username e l'indirizzo di posta elettronica dell'utilizzatore, i permessi legati alla tipologia di account di cui quest'ultimo è stato provvisto. A partire da questi ultimi infatti, come precedentemente indicato, in seguito all'esito positivo della procedura di login effettuata nella schermata iniziale mostrata all'avvio dell'applicazione, viene stabilito il comportamento assunto dalla suddetta nel proseguimento del flusso di schermate.

A titolo esplicativo, verrà ora presentato un esempio che mostri nel dettaglio le modalità con cui, all'interno dei sorgenti della app mobile, sono state realizzate le classi utilizzate per la modellizzazione delle entità precedentemente descritte. In particolare, verranno qui di seguito illustrate le righe di codice volte a modellare la classe Kotlin Alert, relativa agli allarmi del sistema di predictive maintenance considerato, con una spiegazione dei suoi principali aspetti implementativi. Al termine della suddetta descrizione è inserita una figura che mostri per intero la classe considerata, al fine di poter fruire di una vista da consultare in parallelo alla descrizione della stessa. Il primo termine di interesse nella realizzazione della classe è costituito dalla riga in cui è presente la precedentemente citata annotazione @Entity. Questa è utilizzata in primis per segnalare al sottostante database locale Room che tale classe è intesa per essere trasformata in una delle sue tabelle e, in aggiunta, per assegnare il nome con qui questa verrà identificata: in questo caso, "alerts". Successivamente, è importante evidenziare come tale classe sia preceduta dalla parola chiave "data". Quest'ultima, è utilizzata per comunicare al compilatore

che la classe in questione appartenga alle cosiddette "data classes", una caratteristica tipologia di classi fornite dal linguaggio Kotlin per definire classi il cui scopo principale è quello di contenere dati[102]. Utilizzando tale attributo del linguaggio è possibile sfruttare una serie di funzionalità automaticamente derivabili dal compilatore a partire dai membri della classe, come le funzioni membro equals() e toString(). Proseguendo nel codice, si nota come uno dei membri elencati tra le parentesi sia preceduto dall'annotazione @PrimaryKey. Questa serve ad indicare alle librerie di Room che il campo in questione, ossia "ID", debba essere utilizzato come chiave primaria per le entry della tabella del database presa in considerazione. Inoltre, specificando che il campo autoGenerate debba assumere valore booleano false, si impedisce alle funzioni di Room di generare automaticamente, in fase di inserimento di una nuova entry nella tabella, un valore numerico incrementale, in quanto gli allarmi recuperati dalle API per il reperimento delle informazioni contenute nel database esterno posseggono per questo campo un codice identificativo impostato in precedenza. Nel codice relativo alla classe in questione, si nota come tutti i campi presenti risultino di tipo String. Room, costituendo uno strato di astrazione posto sopra un'infrastruttura SQLite, condivide con quest'ultimo la lista di tipologie di dato supportate, ossia: NULL, INTEGER, REAL, TEXT e BLOB[103]. Tuttavia, nelle situazioni e nei casi d'uso tipici, alcune classi possono possedere tra i propri attributi svariati campi la cui tipologia non sia compresa nelle cinque elencate in precedenza. A tal proposito, per informare le librerie di Room sul modo in cui un campo di una qualsiasi tipologia di dato debba essere convertito per essere inserito in una colonna di una tabella nel database, è messa a disposizione degli sviluppatori l'annotazione @TypeConverter. Questa viene utilizzata ponendo in successione alla stessa una funzione che indichi il modo con cui un parametro in ingresso possa venire convertito in un valore che appartenga ad una tipologia di dato interpretabile dalle librerie di Room e, viceversa, il modo in cui sia ottenibile il processo contrario. Di seguito è riportato un esempio di una coppia di funzioni per la serializzazione e deserializzazione di un parametro costituito da una lista di valori Double, utilizzando i metodi forniti

[102] Kotlin Docs, Data classes, kotlinlang.org
[103] SQLite, Datatypes In SQLite, sqlite.org

dalla libreria di Google Gson per la conversione di oggetti Java nella loro rappresentazione in formato JSON, acronimo per JavaScript Object Notation[104].

```
@TypeConverter
fun doubleListToJson(value: List<Double>?) = Gson().toJson(value)

@TypeConverter
fun jsonToDoubleList(value: String) =
Gson().fromJson(value, Array<Double>::class.java).toList()
```

Grazie a questo esempio è possibile notare come sia possibile inserire nelle colonne del database locale non soltanto valori singoli, ma altresì collezioni di dati che vengano codificate in delle stringhe JSON. Affinché queste funzioni di conversione sia correttamente utilizzate, è necessario comunicare alle librerie Room, all'interno della classe annotata con @Database, la classe in cui le suddette siano inserite, utilizzando l'annotazione @TypeConverters. Proseguendo nella visione del codice, è possibile notare come un campo sia stato etichettato con l'annotazione @Transient. Lo scopo di tale operazione è escludere l'attributo in questione dalla forma serializzata dell'oggetto. In altre parole, ciò sta a significare che il campo in questione non verrà utilizzato per la creazione di una colonna della tabella nel database locale. In questo caso, infatti, tale campo non è presente nella lista dei campi contenuta all'interno del database remoto. Esso è presente al solo scopo di permettere l'implementazione di una delle logiche presenti nell'applicazione, che verrà discussa in un secondo momento. Il suo valore è generato all'invocazione della funzione initNumericalSeverity(), in base all'informazione contenuta all'interno del campo severity, normalmente serializzato. La procedura descritta in precedenza per la realizzazione della classe che modelli l'entità considerata, ossia quella relativa agli allarmi, è stata utilizzata allo stesso modo, seppur con sottili differenze del caso in base alle diverse situazioni, per la generazione delle classi che sono state utilizzate per la modellizzazione delle restanti entità.

[104] Google, google/gson, github.com

```
@Entity(tableName = "alerts")
data class Alert(
    @PrimaryKey(autoGenerate = false)
    val ID: String,
    val date: String,
    val description: String,
    val origin: String,
    val origin_ID: String,
    val severity: String,
    val type: String
){
    @Transient var numericalSeverity: Int = 0

    fun initNumericalSeverity() {
        numericalSeverity = when (severity){
            "low" -> 3
            "medium" -> 2
            "high" -> 1
            else -> 0
        }
    }
}
```

Figura 33 - Schermata della classe utilizzata per modellare l'entità Alert

A seguito di questa presentazione riguardante le principali entità di interesse all'interno dell'applicazione mobile per il progetto PREMA, è possibile procedere alla descrizione dei dettagli delle singole schermate dell'applicazione. Tale trattazione verrà effettuata seguendo un possibile ordine di navigazione tra le schermate, in prima battuta seguendo il flusso percorribile da un utente provvisto delle autorizzazioni di un view account ed in un secondo momento percorrendo quello disponibile per un utente edit, evidenziando eventuali funzionalità comuni ad entrambe le liste di opportunità fornite per le due differenti tipologie di account. Per ogni schermata, oltre ad una cattura che ne mostri il layout grafico risultante, verranno descritti gli elementi grafici ed interattivi che la compongono, così come la lista delle sue funzionalità ed eventuali spezzoni di codice ritenuti di particolare importanza. Nonostante l'esistenza di ovvie differenze rese necessarie per adattarsi alle diverse situazioni, in molti casi all'interno delle schermate sono state ripetute alcune logiche e procedure standard, che, al fine di non rendere verbosa e ripetitiva la trattazione corrente, verranno descritte nei particolari solamente nella prima

occasione in cui risultano presenti. Nei casi successivi, al contrario, verranno solamente citate e corredate di eventuali specificità del caso.

La schermata iniziale dell'applicazione mobile consiste nella schermata di autenticazione, a cui è associata la classe Kotlin denominata LoginActivity.

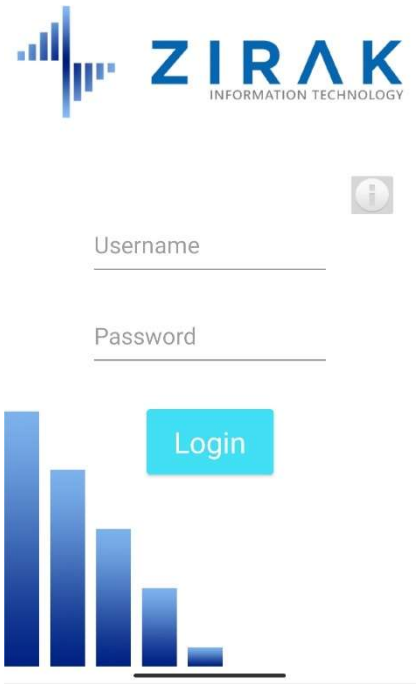

Figura 34 - Cattura della schermata relativa alla LoginActivity

Come è naturale intuire, la schermata in considerazione è comune ad entrambe le tipologie di account. Se, infatti, lo scopo principale della suddetta consiste nel fornire un meccanismo di autenticazione per l'utilizzo dell'applicazione, di modo che i soli utenti in possesso di credenziali corrette siano autorizzati ad accedere alle sue funzionalità, la seconda funzionalità prevista risulta per l'appunto essere la verifica della tipologia di account a cui l'utilizzatore appartiene. Al momento della stesura del presente elaborato, la procedura di login non è ancora stata correttamente implementata, per questioni relative alla sicurezza e alle particolari necessità in termini di confidenzialità ed integrità rese necessarie dall'inserimento e successiva trasmissione di credenziali personali. Tuttavia, è stato ipotizzato che questa procedura possa avvenire, come per le altre comunicazioni esterne all'applicazione locale, attraverso lo sfruttamento di una API appositamente studiata ed

implementata. Tale API, in seguito alla ricezione di una corretta coppia di credenziali username-password, invierà nella risposta i dettagli relativi all'utente, tra cui i permessi di cui esso gode in base alla tipologia di account assegnatagli. Tali permessi verranno quindi salvati in una classe appositamente creata, che disponga delle funzionalità utili per la consultazione degli stessi nelle situazioni in cui queste verranno richieste per il corretto funzionamento dell'applicazione e la navigazione verso le schermate associate. In primo piano all'interno della schermata, oltre al logo della azienda ZIRAK realizzato mediante il widget ImageView, sono presenti due caselle di testo modificabili, implementate attraverso l'utilizzo del componente Android EditText. Dopo un tocco all'interno dell'area di schermo in cui queste sono contenute, verrà generata dal sistema una tastiera virtuale attraverso cui l'utente potrà digitare le proprie credenziali. In aggiunta a queste due, è presente un ulteriore componente costituito da un bottone alla cui pressione, intercettata nel codice della Activity attraverso l'utilizzo del metodo setOnClickListener, viene scatenata la procedura di navigazione verso la schermata successiva, costituita dalla pagina iniziale relativa alla tipologia di account considerata. Tale navigazione è realizzata, in questo contesto come nel resto dell'applicazione, attraverso la preparazione di un cosiddetto Intent, all'interno del quale è possibile specificare la classe della Activity della schermata desiderata, il quale verrà successivamente utilizzato come parametro per il metodo startActivity. Infine, all'interno di questa schermata, così come in tutte quelle successive, è presente un bottone alla cui pressione viene generata una finestra di dialogo contenente delle informazioni sulle funzionalità della suddetta schermata, di modo che l'utilizzatore sia in grado di consultarle in maniera contestuale all'esecuzione dell'app in sé. Tale finestra è resa disponibile attraverso la creazione nel codice di un cosiddetto AlertDialog, di cui è stato impostato un titolo, contenente il nome della schermata, ed un messaggio, con la spiegazione delle funzionalità di quest'ultima.

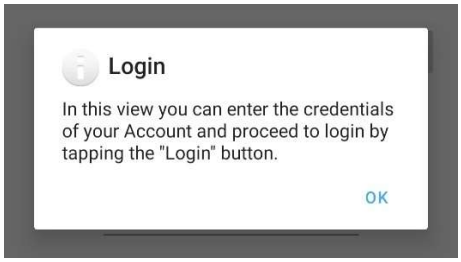

Figura 35 - Cattura della finestra di dialogo contenente le informazioni sulle funzionalità della schermata

Le stringhe utilizzate per il completamento dei suddetti campi, sono state rese disponibili attraverso la creazione, all'interno del file strings.xml contenuto nella cartella delle risorse del progetto, di coppie chiave-valore in cui il secondo attributo corrisponde alla stringa da predisporre. Questa procedura è stata utilizzata per la grande maggioranza delle situazioni analoghe di definizione di stringhe, di modo che il codice risulti più pulito e mantenibile, in quanto, nel caso in cui si rendesse necessario modificare il contenuto di una stringa utilizzato in molteplici sezioni dello stesso, si sufficiente per tale modifica un intervento in un'unica riga di un singolo file.

Nel caso in cui un utente provvisto di autorizzazioni relative ad un view account abbia effettuato il login, esso viene indirizzato alla relativa schermata principale, corrispondente alla classe Kotlin DashboardUserViewActivity.

Figura 36 - Cattura della schermata relativa alla DashboardUserViewActivity

In cima a questa schermata, è stata inserita una barra dell'applicazione personalizzata, contenente il nome della schermata e due icone per la navigazione, rispettivamente, verso la schermata iniziale di autenticazione (eseguita

contestualmente ad una procedura di logout dell'utente correntemente autenticato) e quella di consultazione del profilo dell'utilizzatore. Questa strategia verrà ripresa altrettanto nelle ulteriori schermate dell'applicazione, con le differenze del caso in termini di icone presenti. Nella porzione superiore della schermata, oltre ad una casella di testo che informa sulla tipologia di account corrispondente e al già citato bottone per il reperimento delle informazioni sulla schermata, sono presenti due immagini con relativa didascalia, implementate attraverso il widget di ImageView, utilizzate per la navigazione, in seguito alla pressione di una delle due, verso la relativa schermata di consultazione delle macchine o dei sensori. In aggiunta a tali componenti, è presente in primo piano una lista degli allarmi rilevati e presenti sul database remoto. Per l'implementazione di questa lista è stato utilizzata la componente denominata RecyclerView, la quale, come suggerito dal nome, in seguito allo scorrimento delle viste che vanno ad implementare gli elementi della lista rappresentata, invece di cancellare le viste create in precedenza le ricicla sostituendone i contenuti, con evidenti impatti in termini di consumo di risorse e prestazioni[105]. Il primo passo per l'utilizzo di questa componente consiste nella creazione del layout volto a contenere i dettagli di un singolo elemento della lista. In relazione al fatto che la lista in esame è adibita al contenimento di dettagli relativi ad allarmi e, quindi, alla corrispondente entità Alert, tale layout è stato definito all'interno di un file denominato item_alert.xml. In seguito, è necessario definire per il RecyclerView il suo Adapter ed il relativo ViewHolder, ossia i componenti che si occupano di gestire la maniera con cui i dati sono trattati e visualizzati. Per l'Adapter è stata creata la classe kotlin AlertAdapter all'interno di una cartella appositamente creata per contenere tutti gli Adapter necessari nel progetto. All'interno della suddetta è stata inoltre definita una inner class per la definizione del ViewHolder. Rispettivamente, il ruolo di quest'ultimo è costituito dal fornire tutte le funzionalità per un singolo elemento della lista contenuta nel RecyclerView, mentre l'Adapter si occupa di creare le istanze dei ViewHolder e di impostare i dati da inserire nelle viste relative. In aggiunta a queste tipiche componenti di un RecyclerView, è stata sfruttata una funzione DiffUtil al fine di permettere al suddetto, in fase di aggiunta di una nuova collezione di elementi, di ricalcolare i

[105] Google Developers, Create dynamic lists with RecyclerView, developer.android.com

soli elementi della lista che sono cambiati, invece di cancellare e ricostruire la lista per intero. Come è immediatamente intuibile, ciò concorre all'incremento delle prestazioni dell'applicazione. Per il reperimento degli allarmi da visualizzare nella lista della schermata, è stato considerato di creare una classe kotlin relativa al ViewModel corrispondente alla Activity corrente, coerentemente denominata DashboardViewModel, all'interno di una ulteriore cartella appositamente creata. Al solo scopo di poter passare come parametro a tale ViewModel un'istanza del Repository, è stata contestualmente creata un'ulteriore classe, denominata DashboardViewModelProviderFactory. All'interno del ViewModel, oltre alle interfacce per le interazioni con il database locale, sono presenti porzioni di codice volte alla chiamata del metodo fornito dal Repository per l'utilizzo della API relativa al reperimento degli allarmi immagazzinati sul database remoto e alla gestione della risposta ottenuta in seguito a tale chiamata. Internamente al ViewModel è per questo motivo inserito un membro la cui tipologia di dato è MutableLiveData, in grado di contenere una risorsa specifica per la tipologia di risposta ricevuta dalla API in questione. In aggiunta a tale contenuto, la risorsa in questione può anche assumere dei valori che riflettano lo stato di caricamento o di errore relativi alla suddetta chiamata di API. Se la risposta della API termina con esito positivo, da questa vengono derivati gli allarmi presenti sul database remoto e trattati attraverso i corrispondenti modelli Alert al fine di essere inseriti sul database locale. All'interno della Activity considerata, per il popolamento del RecyclerView, viene osservato il MutableLiveData presente nel ViewModel corrispondente, per due principali ragioni. La prima consiste, nel caso in cui la risorsa contenuta nell'oggetto osservato sia corrispondente all'esito positivo della risposta della API, nell'alimentare l'Adapter con la lista degli allarmi recuperata, di modo che questi siano visualizzati per l'interazione con l'utente. Tale lista viene ordinata in base al grado di criticità dell'allarme e, in subordine, alla data di generazione degli allarmi. La seconda motivazione è dovuta alla presenza, all'interno del layout della schermata, di una componente che indichi il progresso del processo di reperimento degli allarmi, al fine di far intendere all'utilizzatore che l'applicazione stia effettuando un'elaborazione. In assenza di tale componente, denominata ProgressBar, all'utente potrebbe sorgere il dubbio erroneo che l'app sia bloccata o

in attesa di un comando. In base al contenuto della risorsa osservata, questa componente viene mostrata attraverso un'animazione circolare o nascosta rendendola invisibile. Immediatamente al di sopra della lista degli allarmi, nella schermata è presente un menu a tendina a scomparsa che, se premuto dall'utilizzatore, fornisce un elenco di opzioni tra cui quest'ultimo può scegliere per filtrare tale lista, in base alla sorgente che ha generato l'allarme.

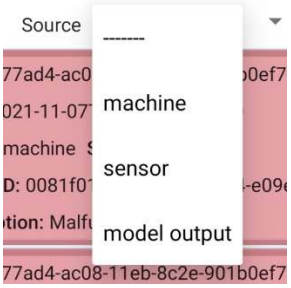

Figura 37 - Cattura del dettaglio dello Spinner per il filtraggio della lista degli allarmi

Tale menu a tendina è implementato mediante l'utilizzo del componente Android denominato Spinner, a cui può essere fornita una lista di stringhe che vada a costituire l'elenco delle entry selezionabili. È possibile, inoltre, reagire al tocco dell'utente sullo stesso andando a sovrascrivere i metodi onNothingSelected e onItemSelected del relativo onItemSelectedListener. In ultima analisi, è presente all'interno della schermata un bottone con un'icona di refresh, volto ad eseguire, in seguito alla sua pressione e attraverso un apposito metodo del ViewModel, un'esplicita chiamata alla API per un nuovo reperimento degli allarmi presenti sul database remoto. A tal proposito, infatti, il metodo per la raccolta di questi ultimi attraverso la suddetta API viene invocato, senza considerare tale esplicita richiesta in seguito alla pressione del bottone di refresh, solamente durante la fase di inizializzazione del DashboardViewModel attraverso l'inserimento di questa chiamata nel metodo init di quest'ultimo. Numerose logiche e strategie utilizzate per le funzionalità della schermata appena descritta sono state applicate allo stesso modo in buona parte delle restanti schermate dell'applicazione. Riassumendo, la lista di tali tecniche consiste nell'utilizzo di: RecyclerView per la visualizzazione di elementi, con la relativa creazione di layout, Adapter e ViewHolder dedicati; ViewModel e ViewModelProviderFactory dedicati per il reperimento dei dati di interesse attraverso l'utilizzo di metodi per la chiamata di API e per le interazioni

col database locale; MutableLiveData alimentati nel ViewModel e osservati nella Activity, per l'aggiornamento della lista di elementi di interesse e le animazioni della ProgressBar; Button di refresh per l'invocazione esplicita della API in questione, altrimenti chiamata nella sola fase di inizializzazione del ViewModel associato.

Dalla schermata principale descritta in precedenza, tra le differenti ulteriori schermate raggiungibili, è possibile navigare verso la schermata per la consultazione dei dettagli di un singolo allarme, attraverso la pressione, all'interno della lista di questi ultimi, di un suo singolo componente. La classe Kotlin adibita alla gestione delle funzionalità della suddetta è identificata dalla cosiddetta SelectedAlertActivity.

Figura 38 - Cattura della schermata relativa alla SelectedAlertActivity

Nella parte superiore della schermata, ossia nella porzione di quest'ultima relativa alla barra dell'applicazione personalizzata, è stata posizionata un'ulteriore icona, in aggiunta alle due citate in precedenza, la cui funzione consiste nella navigazione,

in seguito alla pressione della stessa, verso la schermata principale appena descritta. Oltre al pulsante per il reperimento delle informazioni sulle funzioni della schermata, sono elencati all'interno di quest'ultima i dati di tutti i campi contenuti nell'entità relativa all'allarme selezionato in precedenza. L'informazione relativa all'allarme da considerare in questo frangente è ottenuta attraverso lo sfruttamento dei cosiddetti extras. Questi ultimi, sono delle informazioni aggiuntive di cui l'Intent per la navigazione verso una schermata può essere corredato nella sua preparazione. Nel caso in questione, questi sono sfruttati per l'inserimento di una coppia chiave-valore in cui il contenuto associato al valore è il codice identificativo dell'allarme. Tale inserimento è effettuato attraverso il metodo putExtra, nella maniera seguente:

```
intent.putExtra("alertId", alertId)
```

Il parametro così messo a disposizione è nella Activity di destinazione considerata recuperato attraverso la ricerca del valore associato alla chiave, in questo modo:

```
val alertId = intent.extras?.getString("alertId")
```

A questo punto, il codice identificativo recuperato è utilizzato come parametro per l'invocazione di una funzione messa a disposizione dal ViewModel associato, la quale a sua volta effettua la chiamata di un metodo del Repository per il reperimento dell'entità relativa alla entry presente nel database locale, la cui chiave primaria corrisponda al codice identificativo in questione. Il valore di ritorno relativo a quest'ultimo metodo corrisponde ad un LiveData, motivo per cui per accedere ai suoi contenuti è necessario osservarlo all'interno del codice della Activity. Le funzioni di libreria messe a disposizione da Room per il reperimento dei dati contenuti nelle tabelle del database forniscono dei dati di tipo LiveData, invece di utilizzare direttamente tipologie di dato più standard adatti al contenimento delle informazioni contenute, come i modelli delle entità relative. Ciò è dovuto al fatto che il reperimento delle suddette informazioni viene completato in un lasso di tempo non noto a priori e che potrebbe potenzialmente risultare anche relativamente lungo, motivo per cui, in assenza di una strategia publish-subscribe come quella messa a disposizione dai LiveData, non sarebbe possibile stabilire quando il dato in questione sia effettivamente disponibile. Una volta recuperata l'entità di interesse,

i campi di quest'ultima sono utilizzati per aggiornare le informazioni contenute nella schermata, relative ai dettagli dell'allarme precedentemente selezionato. L'ultima componente presente all'interno della suddetta schermata è costituita da un pulsante per la navigazione verso la sorgente che ha originato l'allarme. In base al valore contenuto all'interno del campo origin di quest'ultimo, utilizzando l'espressione when del linguaggio Kotlin (simile al costrutto switch dei linguaggi C-like) viene preparato e fatto partire l'Intent per la suddetta navigazione. In particolare: quando l'origine corrisponde alla stringa "machine" viene raggiunta la schermata relativa alla SelectedMachineActivity, quando corrisponde invece alla stringa "sensor" viene raggiunta la schermata relativa alla SelectedSensorActivity e quando, infine, corrisponde alla stringa "model output" viene raggiunta la schermata relativa alla StatisticsActivity. Tali schermate verranno presentate nel proseguo della trattazione. Ponendo l'attenzione sulla schermata in questione, è possibile osservare come questa possa risultare piuttosto spoglia, con molto spazio libero a disposizione. A tal proposito, in seguito alle considerazioni effettuate nelle discussioni relative all'applicazione con i membri assegnati al progetto PREMA, non è da escludere che le entità poste alla modellazione degli allarmi si arricchiscano di nuovi campi o di informazioni più estese, andando quindi a colmare questo "vuoto" percepibile all'interno della schermata considerata. In aggiunta a questo aspetto, è altresì possibile che nel corso del tempo l'applicazione venga dotata di nuove funzionalità, per cui lo spazio non occupato nella sua versione attuale potrebbe essere sfruttato in un secondo momento.

Nella descrizione della schermata principale precedentemente descritta, è stato posto in evidenza come, in seguito al tocco da parte dell'utente nella porzione di schermo relativa all'immagine la cui didascalia riporta come descrizione la parola "Machines", sia possibile navigare verso un'ulteriore schermata. Le logiche di funzionamento di quest'ultima sono state realizzate all'interno della classe Kotlin MachinesActivity. Lo scopo di tale schermata è di fornire una panoramica sui macchinari attualmente presenti nel database remoto, su cui sia in atto o in via di attuazione un sistema di predictive maintenance. In particolare, attraverso questa schermata si fornisce all'utente la possibilità di esplorare la lista di tali macchinari e di questi viene presentata una panoramica sul loro stato di salute, attraverso una

grafica che sfrutti una logica di colori comunemente riconosciuta come quella presente su un semaforo stradale.

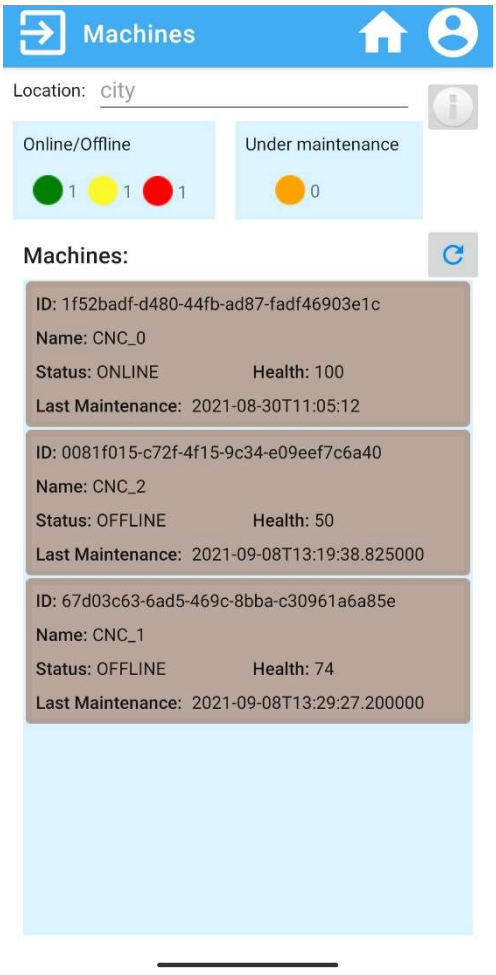

Figura 39 - Cattura della schermata relativa alla MachinesActivity

Nella porzione di schermo posta immediatamente al di sopra della lista di macchinari, implementata anche in questo caso attraverso un RecyclerView, sono infatti presenti due sezioni che forniscano la funzionalità appena descritta. In particolare, quella di destra intitolata "Under maintenance" illustra il numero di macchinari che si trovano, nell'istante corrente, in stato di manutenzione, eventualmente a seguito di una segnalazione portata alla luce dai modelli di machine learning contenuti nel processo di manutenzione predittiva. Tale numero è riportato in una casella di testo posta a fianco di una spia arancione. Nella sezione di sinistra denominata "Online/Offline", al contrario, sono presenti delle indicazioni riguardanti il numero di macchinari non in manutenzione, suddivisi in base al proprio grado di conservazione. Questa suddivisione è comunicata attraverso tre

spie colorate (nell'ordine verde, gialla e rossa) interpretabili come tre livelli di salute (rispettivamente sano, leggermente insalubre e fortemente insalubre), al cui fianco destro è disposta una specifica casella di testo in cui è mostrata la cifra associata. Le quantità utilizzate in queste viste sono ottenute mediante l'utilizzo di dei contatori dedicati, aggiornati in maniera solidale ad eventuali cambiamenti nella lista dei macchinari presenti. Il contatore per i macchinari in stato di manutenzione è incrementato ogni qualvolta venga identificata un elemento della lista in cui il campo status assume il valore "MAINTENANCE". Al contrario, i contatori dei macchinari non in manutenzione vengono incrementati in base al valore numerico assunto dal campo health, confrontandolo con delle soglie prestabilite. In particolare: per valori superiori a 80 viene incrementato il contatore corrispondente alla spia verde, per valori compresi tra 60 escluso e 80 incluso viene incrementato quello corrispondente alla spia gialla, mentre per valori pari o inferiori a 60 è incrementato il contatore della spia rossa. Oltre alle componenti già presenti nelle schermate precedenti e per questo già esposte nella loro trattazione, è presente nella parte superiore dello schermo, al di sotto della barra dell'applicazione, una casella di testo utilizzata per filtrare la lista dei macchinari in base alla città in cui è collocato lo stabilimento che le ospita. Tale componente rappresenta una particolare modalità di inserimento del testo, in cui in seguito alla digitazione di alcuni caratteri vengono forniti, inferiormente alla casella, dei suggerimenti di opzioni in cui la parola che le rappresenta inizia con le lettere inserite fino a quel momento.

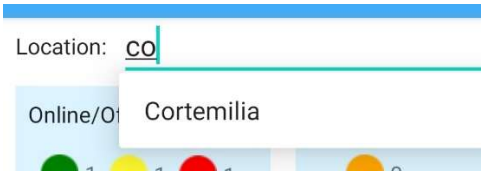

Figura 40 - Cattura del dettaglio della schermata relativo alla casella di testo auto-completante

Sono state quindi implementate, attraverso il metodo setOnItemClickListener di tale componente (denominata per l'appunto AutoCompleteTextView), le logiche per il filtraggio e l'aggiornamento degli elementi della lista di macchinari mostrata a video. In seguito ad un corretto esito di queste operazioni, al fine di informare l'utente dei cambiamenti effettuati, viene generato un breve messaggio di conferma

all'interno di una nuvoletta a scomparsa, attraverso un cosiddetto Toast, implementato attraverso il codice seguente:

```
Toast.makeText(this, "Machines list filtered on city:
${cityAutoCompleteTextView.text.toString()}",
Toast.LENGTH_SHORT).show()
```

Il risultato ottenuto in seguito alla chiamata del metodo show e visualizzato nella figura successiva.

Figura 41 - Cattura del dettaglio del Toast per informare l'utente dell'avvenuta operazione di filtraggio

La funzionalità conclusiva messa a disposizione da questa schermata consiste nella possibilità di navigare, in seguito alla pressione di uno degli elementi presenti nella lista dei macchinari, verso una vista che mostri i dettagli della macchina selezionata. Quest'ultimo compito è assolto dalla classe Kotlin denominata SelectedMachineActivity.

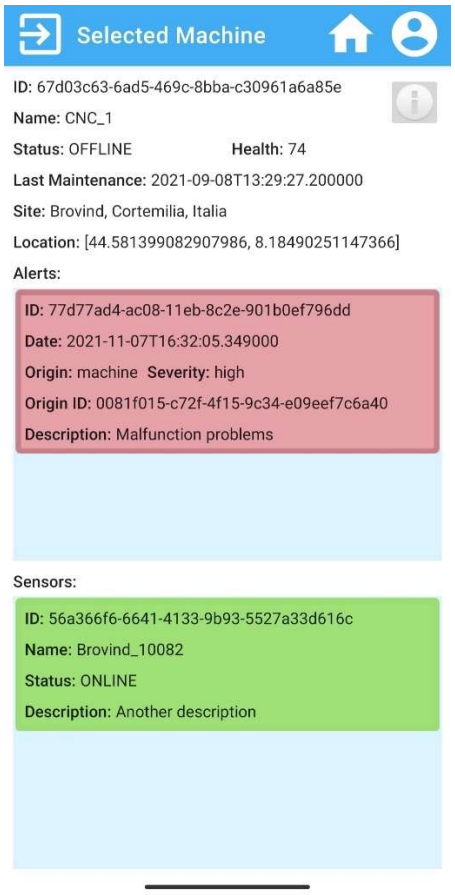

Figura 42 - Cattura della schermata relativa alla SelectedMachineActivity

L’entità corrispondente al macchinario di cui è necessario mostrare i dettagli è recuperata a partire dal codice identificativo passato come parametro negli extras dell’Intent per la navigazione verso la schermata corrente. Questo parametro viene utilizzato all’interno della Activity per chiamare la funzione, fornita dal ViewModel corrispondente, il cui compito è costituito dall’invocazione del metodo per il reperimento di una singola entità di tipo Machine all’interno del database locale. Il LiveData contente la suddetta entità viene quindi osservato all’interno della Activity e, quando il suo contenuto risulta disponibile, quest’ultimo viene utilizzato per popolare le componenti testuali della schermata ed inizializzare due MutableSet atti al contenimento delle stringhe relative alla lista dei codici identificativi corrispondenti agli allarmi e ai sensori associati al macchinario considerato. Queste collezioni di dati vengono in seguito utilizzate per filtrare le liste di allarmi e sensori contenute nelle tabelle del database locale, in maniera analoga rispetto all’operazione descritta in precedenza. Le suddette liste filtrate vengono infine utilizzate per alimentare i due RecyclerView presenti nella schermata, adibiti per l’appunto al mostrare a video, rispettivamente, la lista degli allarmi associati al macchinario selezionato (preceduta dall’etichetta “Alerts:”) e la lista dei sensori montati sullo stesso (preceduta dall’etichetta “Sensors:”). Oltre ai campi precedentemente descritti sono inoltre presenti le già citate componenti per la barra dell’applicazione e per il bottone adibito al reperimento delle informazioni sulle funzionalità dell’applicazione, opportunamente modificate in maniera coerente con la schermata considerata.

Oltre alla navigazione verso le schermate relative ai dettagli di un singolo allarme e alla lista dei macchinari attualmente presenti nel database remoto, la schermata principale descritta in precedenza mette a disposizione dell’utente con autorizzazioni associate ad un account view un’ulteriore funzionalità. Quest’ultima è costituita dalla possibilità di navigare, in seguito alla pressione dell’immagine identificata dalla didascalia “Sensors”, verso una schermata adibita alla visualizzazione e consultazione dei sensori impiegati, attualmente o in potenza, all’interno del sistema di manutenzione predittiva. Le logiche relative alla schermata corrispondente sono contenute all’interno del codice della classe Kotlin SensorsActivity.

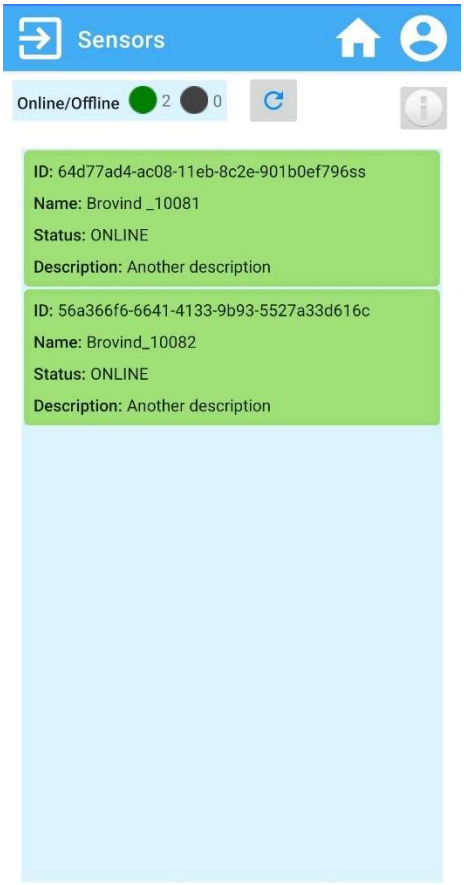

Figura 43 - Cattura della schermata relativa alla SensorsActivity

Una porzione considerevole dello schermo è riservata alla visualizzazione della lista dei sensori presenti sul database remoto, implementata attraverso il consueto utilizzo di un RecyclerView, Come nelle casistiche descritte in precedenza, anche in questo caso tale lista viene reperita attraverso l'osservazione del MutableLiveData contenuto all'interno del ViewModel corrispondente, il cui contenuto viene aggiornato in funzione dello stato di caricamento o esito positivo/negativo relativi alla chiamata della API necessaria. In caso di esito positivo, nella porzione di codice adibita all'alimentazione dell'Adapter per la visualizzazione a video degli elementi all'interno del RecyclerView, la lista di questi ultimi viene scansionata al fine di incrementare il valore di due contatori, adibiti rispettivamente al conteggio dei sensori che, in funzione di quanto contenuto nel campo status dell'entità corrispondente, risultano nel dato istante online oppure offline. I suddetti contatori sono quindi utilizzati per aggiornare le caselle di testo presenti nella porzione superiore della schermata. Nella fattispecie, il valore

contenuto nel contatore relativo ai sensori online viene visualizzato nella casella posta di fianco all'icona della spia verde, mentre quello relativo allo stato contrario in quella che si trova alla destra dell'icona della spia grigia. In aggiunta a queste funzionalità e a quelle relative ai componenti già descritti nella trattazione delle schermate viste in precedenza, un ulteriore comportamento implementato e la navigazione, in seguito alla pressione di un singolo elemento all'interno della lista, verso una schermata contente i dettagli del sensore selezionato.

Le logiche della schermata relativa alla fruizione delle informazioni proprie di un singolo sensore sono contenute all'interno della classe Kotlin SelectedSensorActivity.

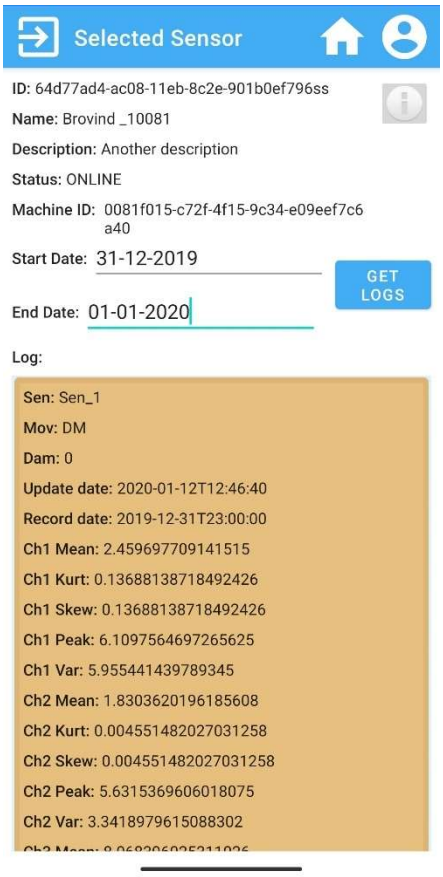

Figura 44 - Cattura della schermata relativa alla SelectedSensorActivity

Il codice identificativo del sensore selezionato, estratto anche in questo caso dall'Intent precedentemente preparato, è utilizzato per recuperare l'entità relativa al record corrispondente nella tabella del database locale, attraverso la consueta metodologia descritta in precedenza. All'interno della schermata corrente è altresì possibile consultare, oltre ai valori dei campi propri del suddetto sensore, i log delle

elaborazioni effettuate a partire dalle misurazioni compiute e dai dati raccolti da quest'ultimo. Tuttavia, come spesso accade nelle situazioni relative a dei log, l'elenco dei record di quest'ultimo è particolarmente nutrito. È stata quindi implementata una strategia con un duplice fine: il principale consiste nel fornire all'utilizzatore una consultazione più chiara e fruibile; il secondo è costituito dalla necessità di limitare il tempo necessario a reperire le informazioni dal database remoto. Grazie a tale strategia, l'utente beneficia di una limitazione delle attese dovuta al minor tempo di risposta del server e all'individuazione più immediata delle informazioni di suo interesse in un dato istante. La suddetta consiste nel fornire all'utente una modalità con cui filtrare i log del sensore all'interno di una finestra temporale delimitata da una data iniziale ed una finale. Quest'ultime sono inseribili dall'operatore in delle caselle di testo editabili in cui è messo in evidenza, attraverso l'uso di un cosiddetto hint testuale, il formato con cui tali date sono riconoscibili dalle logiche dell'applicazione mobile.

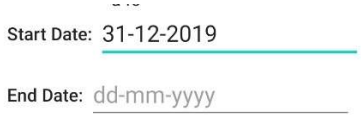

Figura 45 - Cattura del dettaglio delle caselle di testo con suggerimento sul formato di data riconosciuto

In seguito all'inserimento delle date nella sezione appena descritta, l'effettivo reperimento dei log così filtrati, è effettuato in seguito alla pressione del pulsante identificato dall'etichetta "GET LOGS". È importante sottolineare come le informazioni presenti nel RecyclerView dei log, reperite attraverso una apposita API a cui viene comunicato l'intervallo temporale di interesse, non siano costituite da tutte le misurazioni "grezze" effettuate e registrate dal sensore. Al contrario, tali misure sono aggregate in dei valori statisticamente significativi, ritenuti di maggiore interesse rispetto a puri e semplici valori di grandezze misurate.

L'ultima schermata utilizzabile da un utente provvisto di autorizzazioni relative ad una tipologia di account view è quella relativa ai dettagli del proprio profilo, ossia quello con cui si è autenticato nel contesto dell'applicazione mobile. Questa schermata è raggiungibile a partire da ogni altra schermata, con l'eccezione, per ovvie motivazioni, di quella adibita alla procedura di login. Per accedervi, all'utilizzatore basta premere sull'icona posta all'interno del margine superiore della schermata, all'estrema destra della barra dell'applicazione. Coerentemente

con quanto esposto durante la presentazione della schermata di login, al momento della stesura del presente elaborato il codice necessario alla realizzazione delle funzionalità richieste da quest'ultima non è ancora stato terminato. Tuttavia, è allo stesso modo possibile fornire una panoramica riguardante i componenti del layout della schermata e le modalità attraverso cui è ipotizzabile che tali funzionalità vengano, in futuro, erogate. La classe Kotlin adibita al contenimento delle logiche relative è costituita dalla ProfileActivity.

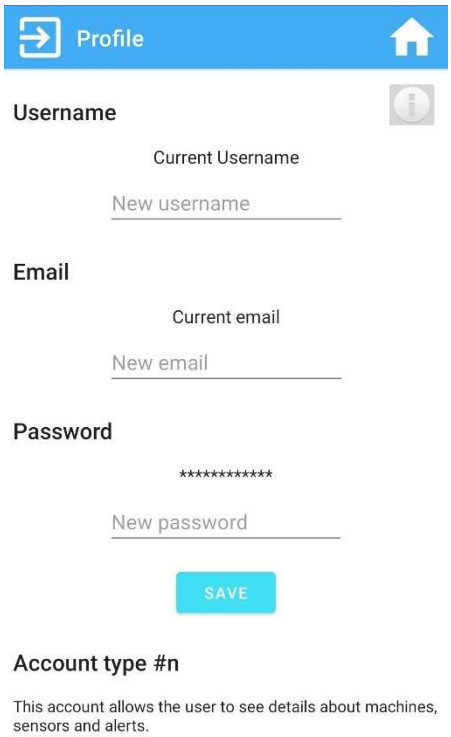

Figura 46 - Cattura della schermata relativa alla ProfileActivity

All'interno della schermata, sono presenti tre sezioni: una relativa allo username dell'utente all'interno dell'applicazione, una relativa al suo indirizzo di posta elettronica ed una relativa alla password per accedere all'account. Nella porzione superiore delle prime due sezioni, è presente una TextView il cui scopo consiste nel contenere il valore corrente del campo relativo. È ipotizzabile che tali valori saranno contenuti alternativamente: in una classe apposita, nel database locale o reperiti con una API a partire dal database remoto. Per tutte e tre le sezioni è inoltre presente, nella loro porzione inferiore, una casella di testo editabile atta a contenere i nuovi

valori che l'utente intende utilizzare per sostituire quelli attualmente in uso. È quindi presente un pulsante, contente l'etichetta "SAVE", in seguito alla pressione del quale venga scatenata la metodologia con cui salvare le modifiche effettuate. Tale procedura verrà con ogni probabilità erogata attraverso l'utilizzo di una API appositamente studiata e messa a disposizione. In ultima analisi, nella parte inferiore della schermata è presente un campo testuale volto a descrivere la tipologia di account correntemente autenticato, con una breve descrizione dei suoi permessi all'interno dell'applicazione.

Successivamente a questa trattazione relativa alle schermate utilizzabili da un utente i cui permessi siano propri della tipologia di account view, verranno ora presentate quelle messe a disposizione di un utente in possesso di un account edit. In tale nuova casistica, è possibile notare come tutte le schermate descritte in precedenza siano altrettanto raggiungibili anche utilizzando un account di questa tipologia, con l'unica eccezione della schermata principale, la quale è stata modificata per permettere la navigazione verso ulteriori schermate. La classe Kotlin che gestisce le logiche per il funzionamento di quest'ultima è denominata DashboardUserEditActivity.

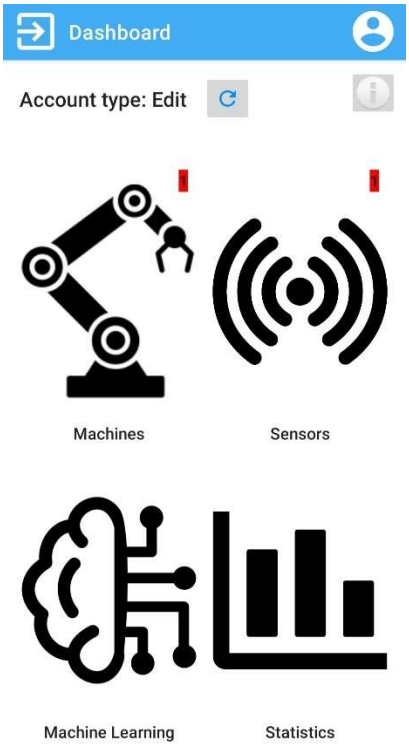

Figura 47 - Cattura della schermata relativa alla DashboardUserEditActivity

A differenza della schermata principale per la tipologia di account view, in questo caso la lista degli allarmi è stata sostituita da due nuove sezioni identificate da due ulteriori ImageView. Ciononostante, viene anche in questo caso richiesta un'indicazione relativa ai suddetti, motivo per cui il ViewModel associato a questa classe e lo stesso già citato per l'altra schermata principale, ossia il DashboardViewModel. In questo caso, il MutableLiveData atto a contenere la risposta della API utilizzata per il reperimento degli allarmi sul database remoto è osservato al fine di incrementare tre diversi contatori, uno per ogni tipologia di sorgente dei suddetti: "machine", "sensor", o "model output". Questi contatori vengono utilizzati per aggiornare il contenuto delle caselle di testo poste nell'angolo superiore destro delle tre rispettive sezioni: Machines, Sensors e Statistics. In caso di assenza di allarmi per una di queste, la casella relativa viene resa invisibile.
In seguito ad una pressione nella porzione di schermo relativa all'immagine etichettata con la didascalia "Machine Learning", viene effettuata una navigazione verso la schermata che fornisce la lista dei modelli di machine learning utilizzati nel sistema di manutenzione predittiva. Le logiche volte ad implementare le funzionalità della suddetta sono contenute nella classe Kotlin ModelsActivity.

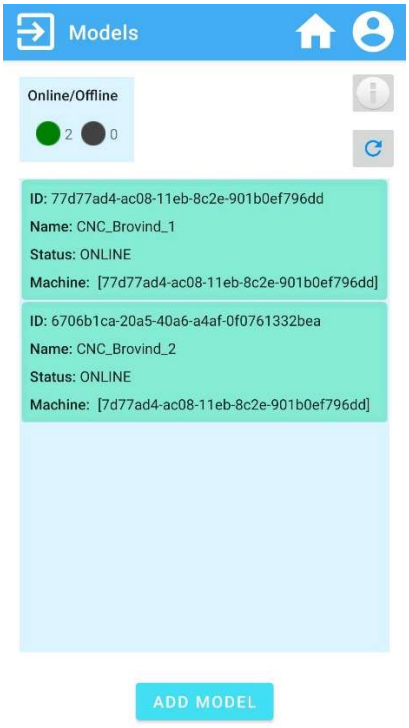

Figura 48 - Cattura della schermata relativa alla ModelsActivity

Il suo contenuto e le logiche utilizzate per implementare le sue funzionalità sono simili a quanto è stato descritto per la presentazione della SensorsActivity. In aggiunta agli elementi presenti nella schermata relativa a quest'ultima, tuttavia, nella schermata considerata è presente un ulteriore pulsante, identificato dall'etichetta "ADD MODEL", alla cui pressione è effettuata la navigazione verso la schermata per l'aggiunta di un nuovo modello di predizione. In seguito alla pressione di un singolo elemento all'interno della lista contenuta nel RecyclerView, viene scatenata la navigazione verso una schermata contenente i dettagli del modello selezionato.

La classe Kotlin contenente le logiche per la schermata relativa alle informazioni proprie di un singolo modello di predizione è la SelectedModelActivity.

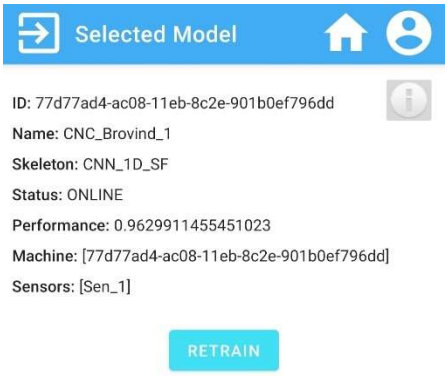

Figura 49 - Cattura della schermata relativa alla SelectedModelActivity

Come è possibile notare nella figura, il layout di questa schermata, così come la lista delle sue funzionalità, è molto simile a quanto è stato descritto per la SelectedAlertActivity, comprese le considerazioni in merito allo spazio vuoto ancora disponibile all'interno della stessa. Oltre ad una serie di campi testuali

contenenti i dettagli del modello di predizione considerato, come di consueto reperiti attraverso un codice identificativo passato in precedenza alla Activity ed utilizzato come parametro per il metodo del ViewModel corrispondente all'osservazione di una singola entry all'interno del database locale, alla comune barra dell'applicazione ed al pulsante per il reperimento delle informazioni sulla schermata, è infatti presente un singolo bottone identificato dall'etichetta "RETRAIN". Quest'ultimo, tuttavia, non è utilizzato per implementare una navigazione verso un'ulteriore schermata, ma bensì per effettuare il retrain del modello, ottenuto tramite la chiamata ad una apposita API.

A partire dalla schermata contenente la lista dei modelli implementati nel sistema di machine learning è possibile raggiungere una ulteriore schermata, oltre a quella appena descritta. Quest'ultima contiene le funzionalità per fornire all'utente autenticato con un account edit la possibilità di configurare ed aggiungere un nuovo modello di predizione. La classe Kotlin contenente il codice per l'implementazione delle logiche relative è costituita dalla AddModelActivity.

Figura 50 - Cattura della schermata relativa alla AddModelActivity

Proprio a partire dalla sua natura di elemento volto alla configurazione, questa schermata risulta particolarmente densa in termini di componenti e, di conseguenza, con uno spiccato grado di interazione con l'utente. Nella parte superiore dello schermo sono presenti tre caselle di testo editabili in cui l'utente ha la possibilità di inserire il nome che il modello di predizione avrà all'interno del sistema e la finestra temporale di riferimento, identificata da una data di inizio e una di fine, che indichi quali dati utilizzare per il suo funzionamento. Poste inferiormente a questa sezione, ne sono presenti altre tre, allo scopo di fornire all'utente una modalità con cui consultare, selezionare e configurare i restanti parametri necessari alla corretta creazione del modello. La prima selezione che il suddetto è tenuto ad effettuare consiste nella scelta dello skeleton su cui il modello dovrà basarsi. La lista dei possibili skeleton è fornita con un RecyclerView i cui elementi sono recuperati attraverso il ViewModel associato alla classe. Nel layout grafico relativo ad ognuno dei suddetti è presente, accanto al margine destro, un'icona, in seguito alla cui pressione viene effettuata la navigazione verso una schermata contenente le informazioni relative allo skeleton corrispondente. In aggiunta, una volta selezionato uno skeleton, attraverso una pressione sull'elemento corrispondente all'interno della lista, l'entità relativa a quest'ultimo viene utilizzata per il popolamento del secondo RecyclerView presente nella schermata. Quest'ultimo, per l'appunto, è utilizzato per la definizione degli hyperparameter propri dello skeleton selezionato. Ognuno degli elementi contiene a tal proposito, oltre ad un'etichetta che ne identifichi il significato, una casella di testo editabile configurata in modo che, in seguito alla pressione all'interno della stessa, la tastiera virtuale generata per la compilazione presenti soli valori numerici. L'ultima sezione presente è relativa alla selezione del macchinario a cui applicare il modello di predizione considerato e dei relativi sensori da utilizzare per le sue analisi. La prima scelta è effettuata mediante l'utilizzo di uno Spinner contenente i macchinari presenti nel sistema. In seguito alla selezione di uno dei suddetti, il RecyclerView sottostante viene popolato con i sensori su di esso montati, che l'utente potrà selezionare attraverso una pressione sui singoli elementi. Una volta inseriti e selezionati tutti i campi necessari precedentemente descritti, in seguito alla pressione del pulsante nuovamente identificato dall'etichetta "ADD MODEL",

viene invocato il metodo messo a disposizione dal ViewModel per la chiamata della API appositamente realizzata.

La schermata per la consultazione dei dettagli relativi ad un singolo skeleton, citata all'interno della descrizione appena conclusa, è implementata attraverso le logiche contenute nella classe Kotlin SelectedSkeleton.

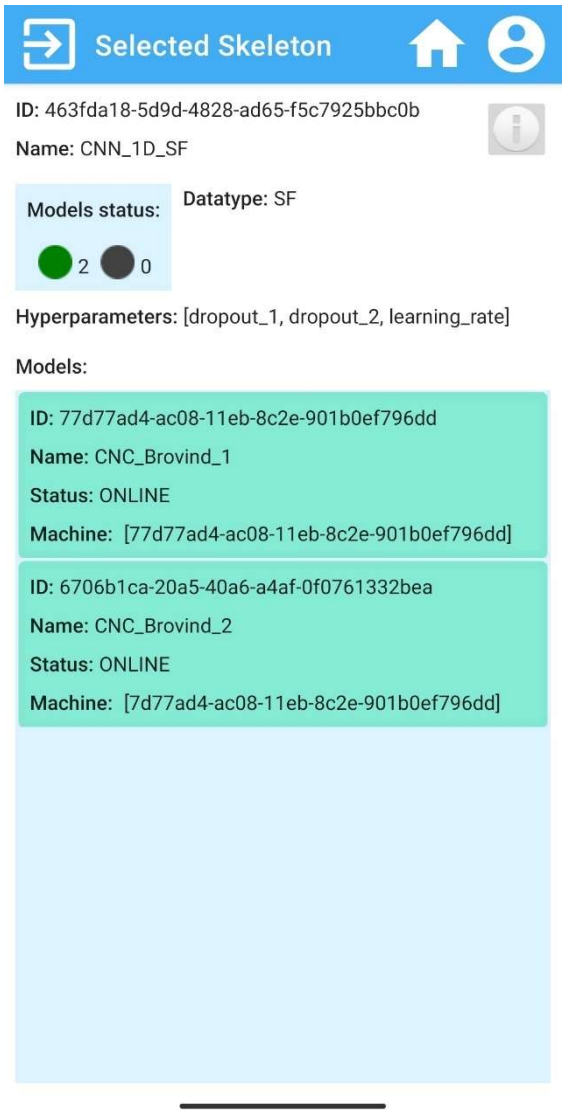

Figura 51 - Cattura della schermata relativa alla SelectedSkeletonActivity

Nella porzione superiore di tale schermata sono presenti le informazioni relative alla singola entità corrispondente al codice identificativo passato come parametro nell'Intent per la navigazione verso la suddetta. In aggiunta ai campi testuali relativi a tali dettagli è presente una lista dei modelli di predizione basati sullo skeleton corrispondente, inserita, come di consueto, in un RecyclerView. A proposito degli elementi contenuti in tale lista è inoltre presente una sezione volta a segnalare il

quantitativo di modelli il cui stato corrisponde al valore “ONLINE” oppure “OFFLINE”, mediante la già osservata tecnica a spie colorate.

L’ultima schermata presente nell’applicazione mobile per il progetto PREMA è raggiungibile a partire dalla schermata principale per la tipologia di account edit, in seguito alla pressione da parte dell’utente sulla porzione di schermo contenente l’immagine etichettata con la didascalia “Statistics”. Per l’appunto, la classe Kotlin contenente le logiche per le sue funzionalità è costituita dalla StatisticsActivity.

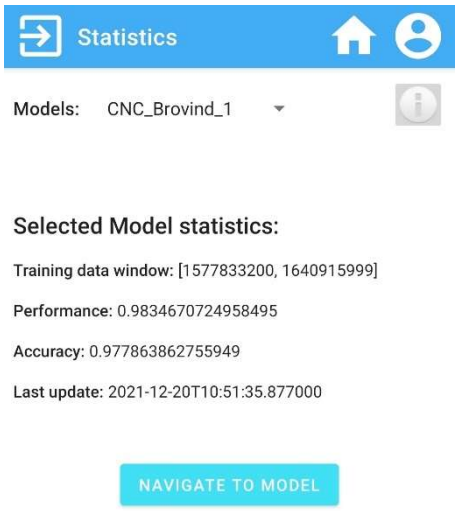

Figura 52 - Cattura della schermata relativa alla StatisticsActivity

Lo scopo della schermata considerata è fornire all’utente la possibilità di consultare le statistiche relative ai modelli di machine learning implementati nel sistema di manutenzione predittiva. A tale scopo è presente, nella porzione superiore della suddetta, uno Spinner attraverso il quale l’utente può selezionare un singolo modello su cui effettuare la suddetta consultazione. In seguito a tale selezione i campi testuali presenti nella schermata sono popolati con le statistiche relative. In fase di creazione della schermata questi ultimi possono già venire definiti se l’Intent per la navigazione verso di essa presenta, all’interno dei suoi extras, un codice

identificativo relativo ad un certo modello. Questa situazione si concretizza nel caso in cui la navigazione sia effettuata a partire dalla SelectedAlertActivity, in seguito alla pressione del bottone identificato dall'etichetta "NAVIGATE TO ORIGIN", nel caso in cui la sorgente dell'allarme sia, per l'appunto, l'output di un modello. In ultima analisi, è presente all'interno della schermata un'ulteriore componente costituita da un pulsante per la navigazione verso la schermata contenente i dettagli più generali del modello selezionato, ossia quella concretizzata dalla SelectedModelActivity, presentata in precedenza.

Con quest'ultima schermata si conclude la trattazione relativa al lavoro svolto per la progettazione e la realizzazione dell'applicazione mobile. In termini di ciclo di vita del software, le restanti fasi da portare a compimento sono costituite dal successivo stadio di testing e dal prolungato periodo di mantenimento della suddetta.

5 Conclusioni

Al termine delle tematiche trattate in precedenza, viene posto all'interno dell'elaborato questo ulteriore capitolo conclusivo, il cui scopo è illustrare gli obiettivi raggiunti attraverso il lavoro svolto nel periodo della tesi ed esporre alcune considerazioni in merito ad alcuni spunti per la realizzazione di modifiche o nuove funzionalità da integrare nell'applicazione su cui l'intera trattazione è stata basata.

Relativamente agli obiettivi che sono stati presentati nell'introduzione dell'elaborato ed in particolare a quello principale di realizzazione di un'applicazione mobile per la manutenzione predittiva industriale, è possibile stabilire che questo sia stato raggiunto. Come è già stato evidenziato nel capitolo precedente, infatti, l'unica funzionalità prevista per tale applicazione non ancora implementata al momento della stesura del presente elaborato consiste nell'effettivo meccanismo di autenticazione da utilizzare. Tuttavia, le logiche che erano state stabilite come conseguenza di questo procedimento, ossia relative alle diverse implicazioni di una certa tipologia di account, sono state stabilite ed in buona parte già implementate, motivo per cui il risultato ottenuto può considerarsi quasi del tutto completo, nonostante questa mancanza. Le comunicazioni ed interazioni con i componenti aziendali che hanno collaborato al progetto si sono rivelate efficaci e,

in accoppiata col lavoro svolto per la creazione dell'applicazione in sé, hanno portato ad una app mobile risultante fluida, intuitiva e funzionale.
Per quanto riguarda invece le modifiche la cui integrazione potrebbe essere discussa in futuro, esse sono raccolte nel seguente elenco.

- Supporto specifico per dispositivi mobili di tipologia tablet: allo stato attuale, l'applicazione è altresì in grado di funzionare su tablet Android nella stessa maniera; tuttavia, data la significativa grandezza degli schermi di questi ultimi sarebbe preferibile la generazione di layout appositi che possano sfruttare più efficacemente lo spazio a disposizione.

- Supporto multilingua: come evidenziato in precedenza, infatti, al momento l'applicazione è erogata nella sola lingua inglese. Sarebbe quindi ipotizzabile una sua estensione al fine di renderla utilizzabile anche da parte di utilizzatori con una diversa preferenza linguistica.

- Supporto per l'utilizzo in modalità landscape: in maniera coerente a quanto considerato in merito all'utilizzo di dispositivi tablet, è possibile ipotizzare la creazione di appositi layout per l'utilizzo dell'applicazione con il dispositivo ruotato in modalità landscape, al fine di gestire in maniera più efficace la disposizione degli elementi nella schermata.

- Supporto per la gestione degli allarmi già gestiti: a tal proposito è ipotizzabile un meccanismo di eliminazione degli stessi o di assegnazione a questi ultimi di una proprietà che li descriva come tali, in modo che possano essere trattati diversamente dagli allarmi non ancora gestiti.

- Supporto per la configurazione dei sensori: si potrebbe immaginare l'implementazione di una schermata, raggiungibile da quella relativa ai dettagli di un singolo sensore, per qualche tipo di configurazione di questi ultimi effettuabile direttamente dall'applicazione, con i vantaggi che ciò potrebbe generare.

Bibliografia:

[1] Christoph Roser, Illustration of Industry 4.0, showing the four "industrial revolutions" with a brief Italian description, AllAboutLean.com, 2016

[2] ZIRAK s.r.l., Logo Zirak, www.zirak.it

[3] Polo di Innovazione ICT, Format del progetto, Format-di-Progetto_PRISM-E_PREMA V4, 2020

[4] Clayton Christensen, What is Disruptive Innovation?, Harvard Business Review, 2015

[5] ZIRAK s.r.l., PREMA Architecture, www.zirak.com/iot/prema

[6] Minifaber, smart-factory-schema, minifaber.it/blog/la-smart-factory-o-industry-4-0

[7] Michael Rüßmann, Lorenz, Gerbert, Waldner, Engel, Harnisch, and Justus, Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries, Boston Consulting Group, 2015

[8] Boston Consulting Group BCG, Exhibit 1, www.bcg.com, 2015

[9] Organizzazione per la Cooperazione e lo Sviluppo Economico, Definizione di Manutenzione, Delibera OCSE, 1963

[10] Riferimento 191.07.16 della Norma UNI 9910 e Riferimenti 3.9, 3.9.1, 3.9.2, 3.9.3, 3.7, 3.10, 3.11 e 3.12, della Norma UNI 10147, rispettivamente del 1991 e 1993

[11] Gian Luca Panizzari, Manutenzione-base-tempo-e-condizione, Management Problem Solving, consultingmps.com/manutenzione-predittiva-quando-sceglierla/

[12] blog.appliedai.com/predictive-maintenance/

[13] IndustryWeek in collaborazione con Emerson, Unlocking Performance, partners.wsj.com/emerson/unlocking-performance/how-manufacturers-can-achieve-top-quartile-performance/

[14] Accelix, Predictive versus preventive maintenance, www.accelix.com/predictive-versus-preventive-maintenance/, 2018

[15] International Business Machines Corporation IBM, Cosa è un CMMS?, www.ibm.com/it-it/topics/what-is-a-cmms

[16] FMX, CMMS Software Diagram, www.gofmx.com/wp-content/uploads/2020/04/cmms-software-diagram.svg
[17] Google Trends, *Trend di ricerca "Predictive maintenance" nel settore Industriale, 2011-2018*
[18] Rapporto Tecnico dell'Autorità Ambientale 12/2013 relativo al POR FESR 2007-2013
[19] Jonathan Cooper, Stop wasting energy: Use predictive maintenance, embeddedcomputing.com, 2017
[20] Matteo Starri di We Are Social, Digital 2021: i dati globali, wearesocial.com, 2021
[21] Cisco, Annual Internet Report White Paper c11-741490, cisco.com, 2021
[22] Cisco, 2018-2023 Device Share, cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html
[23] Kevin Ashton, RFIDjournal-That Internet of Things Thing, 2009
[24] Paolo Magrassi, Dizionario dell'economia digitale, Il Sole 24Ore, 2002
[25] jeferrb, rete-iot-internet-delle-cose, pixabay.com
[26] Pocket-lint, image1 in "Cos'è Alexa e cosa può fare Amazon Echo?", pocket-lint.com
[27] Nilanjan Dey, Bhatt, Satapathy, Hassanien, Ashour, Internet of things and big data analytics toward next-generation intelligence, Springer, 2018
[28] Charith Perera, Ranjan, Wang, Khan, Zomaya, Big Data Privacy in the Internet of Things Era, IEEE IT Professional Magazine: Special Issue Internet of Anything, 2015
[29] Peter Mell, Grance, The NIST Definition of Cloud Computing, National Institute of Standards and Technology, 2011
[30] Winston & Strawn LLP, What is the Definition of Online Privacy?, winston.com/en/legal-glossary
[31] McAfee Enterprise, What Is Internet Security?, mcafee.com
[32] Chris Clearfield, Why The FTC Can't Regulate The Internet Of Things, Forbes, 2013

[33] Federal Trade Commission, Marketer of Internet-Connected Home Security Video Cameras Settles FTC Charges It Failed to Protect Consumers' Privacy, ftc.gov, 2013

[34] TRENDnet, TRENDnet TV-IP751WC Wireless Cloud Camera photo, trendnet.com

[35] Bitdefender Investigations and Forensics Unit, New dark_nexus IoT Botnet Puts
Others to Shame, bitdefender.com, 2020

[36] Andy Greenberg, Hackers Remotely Kill a Jeep on the Highway—With Me in It, wired.com, 2015

[37] Wesley Chai, Labbe, Stedman, The ultimate guide to big data for businesses – big data analytics definition, searchbusinessanalytics.techtarget.com, 2021

[38] David Taylor, What is Big Data? Introduction, Types, Characteristics, Examples, guru99.com, 2021

[39] Giuseppe.sepe, The main processes that make up the life cycle of the Big Data, 2017

[40] Gartner, Gartner Information Technology Glossary: Big Data, gartner.com

[41] The unbelievable Machine Company, What is Big Data? – A definition with five Vs, blog.unbelievable-machine.com

[42] Josh Fruhlinger, Pratt, What is business intelligence? Transforming data into business insights, cio.com, 2019

[43] Colby Ranger, Raghuraman, Penmetsa, Bradski, Kozyrakis, Evaluating MapReduce for Multi-core and Multiprocessor Systems, *IEEE 13th International Symposium on High Performance Computer Architecture*, ieeexplore.ieee.org, 2007

[44] Redazione LineaEDP, 10 anni di Apache Hadoop, lineaedp.it, 2016

[45] The Apache Software Foundation, Apache Hadoop, hadoop.apache.org

[46] Ryan B. Patrick, The three things IDC Canada predicts will matter in 2013, expertIP, blog.allstream.com, 2012

[47] Debbie Stephenson, 7 Big Data Techniques That Create Business Value, TouchPoint by Firmex, firmex.com

[48] Techopedia, Mobile Application (Mobile App), techopedia.com

[49] geralt, mobile-smartphone-tablet-bianca, pixabay.com
[50] dwb Web Agency, Vantaggi e svantaggi delle app mobile, dwb.it, 2018
[51] Education-wiki, Tipi di applicazioni mobili, it.education-wiki.com
[52] Microsoft Windows, Che cos'è un'app UWP (Universal Windows Platform)?, docs.microsoft.com, 2021
[53] 200degrees, interfaccia-utente-androide, pixabay.com
[54] DuckMa Blog, What Are the Different Types of Mobile Apps? Breaking Down Industries and Functionalities, blog.duckma.com
[55] Mehmet Can Cavas, Top 10 Social Media Apps February 2021, mobileaction.co/blog, 2021
[56] Meta Facebook, Facebook Reports Fourth Quarter and Full Year 2020 Results, investor.fb.com, 2021
[57] Richard Dawkins, The Selfish Gene o Il gene egoista – la parte immortale di ogni essere vivente, 1976
[58] INDIRE, Indagine Tra I Docenti Italiani Pratiche Didattiche Durante Il Lockdown Report Preliminare, 2020
[59] Vidyalankar, Benefits of Mobile Apps in Education, vidyalankar.org, 2021
[60] Renee Bassett, Industrial Mobile Apps: Who's Using Them and Why, automationworld.com, 2014
[61] Jason Peck, What Are Businesses Using Mobile Apps For? Infographic + Survey Results, gocanvas.com, 2014
[62] StatCounter GlobalStats, Mobile Operating System Market Share Worldwide - December 2021, gs.statcounter.com
[63] Lisa Eadicicco, TIME, Watch Steve Jobs Unveil the First iPhone 10 Years Ago Today, time.com, 2017
[64] Philip Elmer-DeWitt, Steve Jobs: Apple Will Open iPhone to 3rd Party Apps in February, fortune.com, 2007
[65] Apple Newsroom, The App Store turns 10, apple.com/newsroom, 2018
[66] David Curry, App Store Data (2022), businessofapps.com, 2022
[67] Stefano Bontempi, Apple, tutte le novità della WWDC 2019, hdblog.it, 2019
[68] Apple iOS, What’s New in the iOS SDK, developer.apple.com, 2022

[69] Antoine van der Lee, Full-screen development with Xcode and the Simulator, avanderlee.com
[70] Apple Inc., Apple Developer Program - What You Need To Enroll, developer.apple.com, 2022
[71] Sundar Pichai, Android provides choice, blog.google, 2018
[72] The Editors of Encyclopaedia Britannica and Erik Gregersen, Android operating system, britannica.com
[73] John Callaham, Google made its best acquisition nearly 16 years ago: Can you guess what it was?, androidauthority.com, 2021
[74] Michael Oryl, T-Mobile G1 launch event, flickr.com
[75] Alex Cranz, There are over 3 billion active Android devices, theverge.com, 2021
[76] Bonnie Cha, N. Lee, T-Mobile G1 review:, cnet.com, 2018
[77] L. Ceci, Number of available applications in the Google Play Store from December 2009 to September 2021, statista.com, 2022
[78] Google, Come usare Play Console, support.google.com
[79] Eugene Toporov, IntelliJ IDEA is the base for Android Studio, the new IDE for Android developers, 2013
[80] Google Developers, Android Studio, developer.android.com
[81] Deng Li, HarmonyOS (HongMeng OS): Everything you need to know, huaweicentral.com, 2021
[82] David Shepardson, Freifeld, Trump extends U.S. telecom supply chain order aimed at Huawei, ZTE, reuters.com, 2020
[83] Harikrishna Kundariya, Android VS iOS: A Comparison of Distinctive Attributes, esparkinfo.com
[84] Kaspersky Lab, What is Jailbreaking – Definition and Explanation, kaspersky.com
[85] Chet Haase, Google I/O 2019: Empowering developers to build the best experiences on Android + Play, android-developers.googleblog.com, 2019
[86] Takeshi Hagikura, Understanding the performance benefits of ConstraintLayout, android-developers.googleblog.com, 2017
[87] Google Developers, Support different screen sizes, developer.android.com

[88] Stephanie Cuthbertson, Google I/O 2018: What's new in Android, android-developers.googleblog.com, 2018
[89] Google Developers, Android Jetpack, developer.android.com
[90] *Chris Sells, Poiesz, Ng, Android Jetpack per accelerare lo sviluppo delle app, developers-it.googleblog.com, 2018*
[91] Anand Gaurav, What is Android Jetpack and why should we use it?, blog.mindorks.com, 2019
[92] Vito Lavecchia, Fasi del ciclo di vita del software in informatica, vitolavecchia.altervista.org
[93] Google Developers, App Manifest Overview, developer.android.com
[94] Google Developers, Activity, developer.android.com
[95] Google Developers, Permissions on Android, developer.android.com
[96] Google Developers, permissions workflow-overview, developer.android.com
[97] Google Developers, Dimension, developer.android.com
[98] Google Developers, ScreenOrientation, developer.android.com
[99] DaNeil C, API =/= Database - Why Would You Want to use an API to Talk to a Database Rather Than Just Query The Database?, dev.to, 2020
[100] Anupam Chugh, Android MVVM Design Pattern, journaldev.com
[101] Google Developers, ViewModel Overview – viewmodel-lifecycle, developer.android.com
[102] Kotlin Docs, Data classes, kotlinlang.org
[103] SQLite, Datatypes In SQLite, sqlite.org
[104] Google, google/gson, github.com
[105] Google Developers, Create dynamic lists with RecyclerView, developer.android.com