POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Well being app - Progetto della user interface

Relatore

Prof. Maurizio MORISIO

Candidato

Matteo PELLEGRINO

Aprile 2023

Sommario

Ad oggi più dell'80 percento della popolazione mondiale possiede uno smartphone e più del 30 percento di essa utilizza una mobile app o un fitness tracker per monitorare la propria salute. Con il termine fitness tracker si intende un dispositivo indossabile, simile a un orologio o braccialetto, in grado di registrare l'attività fisica di un individuo attraverso l'utilizzo di dati come la frequenza cardiaca, i passi percorsi e la durata del sonno. Questo viene associato ad una mobile app, nella maggior parte dei casi sviluppata dalla stessa casa produttrice del tracker, che mostra i dati rilevati da quest'ultimo e permette all'utente di controllare i propri eventuali progressi. L'obiettivo della presente tesi di laurea è la creazione della user interface di una mobile app che permetta ai medici dell'Azienda Ospedaliera Universitaria Integrata di Verona, di monitorare le abitudini giornaliere dei loro pazienti. L'app è in grado di analizzare e processare i dati raccolti, per fornire settimanalmente delle "recommendation" specifiche che mirano al miglioramento dello stile di vita. La raccolta dati avviene tramite il fitness tracker, che il paziente dovrà indossare lungo il corso della giornata e per tutta la durata dell'esperimento, che sarà di circa sei mesi. La tesi ripercorre prima di tutto l'attività di ricerca svolta al fine di trovare la soluzione di sviluppo più adatta per il cliente. Tale ricerca viene fatta osservando le diverse proposte già in commercio, cercando di distinguersi da esse per funzionalità e facilità di utilizzo. Successivamente si presenta come i dati vengono trasmessi dal fitness tracker alla mobile app. Inizialmente si è percorsa una strada che prevedeva di creare una mobile app che si interfacciasse direttamente con il fitness tracker e utilizzando solamente la tecnologia BLE come canale di comunicazione per prelevare i dati di cui si aveva bisogno, successivamente però questa idea è stata abbandonata per far spazio alla creazione di una mobile app che si servisse dell'applicazione proprietaria del fitness tracker come via di comunicazione con esso e che come funzione primaria avesse quella di gestire questi dati e mostrarli all'utente. Infine si dà spazio allo sviluppo e implementazione dell'interfaccia dell'app, descrivendo passo dopo passo il processo di creazione e presentando le pagine implementate. La conclusione del lavoro di tesi fornisce uno sguardo verso le future versioni dell'applicazione, esplicando le funzionalità aggiuntive che essa potrà includere.

Tabella dei contenuti

Elenco delle tabelle

Elenco delle figure

Acronyms

API
Application programming interface

AES
Advanced Encryption Standard

BLE
Bluetooth Low Energy

CSS
Cascading Style Sheets

ECB
Electronic code book

HTML
HyperText Markup Language

GATT
Generic Attribute Profile

MET
Metabolic equivalent of task

UUID
Universally Unique Identifier

Capitolo 1

Approccio ad uno stile di vita salutare

Per trattare questo argomento si è fatto uso delle informazioni provenienti dal testo "Path to Longevity" di Luigi Fontana, scienziato e professore di medicina e nutrizione alla Università di Sidney, che studia la longevità e il concetto di restrizione calorica. Nell'ultimo secolo infatti, l'aspettativa di vita mondiale ha avuto un incremento del 55% arrivando nel 2020 a 73.2 anni [1]. Nonostante questo dato però, ad oggi, meno dell'1% della popolazione mondiale raggiunge i 100 anni di età e di questi solo il 19% ci arriva in salute, il resto è affetto da malattie croniche [2], tra queste ci sono il diabete, malattie cardiovascolari, artrite, asma o cancro. Negli Stati Uniti il 78% degli adulti con età superiore a 55 anni è affetto da almeno una malattia cronica [3]. Tra queste, le malattie cardiovascolari, sono la causa primaria di morte nei paesi occidentali, si sviluppano nell'individuo, lungo decenni della sua vita. Un infarto o un ictus avvengono dopo che, per molti anni, l'individuo, ha sviluppato livelli sempre più alti di colesterolo e pressione sanguigna. Prescrivere quindi un medicinale che riduce il colesterolo a 50 o 60 anni, ridurrà, ma non azzererà il rischio di una morte improvvisa. Il tempismo quindi è molto importante poiché agire dopo alcuni anni potrebbe risultare inefficace. Per rallentare il processo di invecchiamento e mantenere o migliorare la salute, è necessario modificare la quantità e la qualità di cibo che vengono mangiate ogni giorno, in combinazione con una serie di attività fisiche e cognitive. I fattori principali da considerare sono quindi: l'alimentazione, l'attività fisica e la cura del sonno.

1.1 Alimentazione

Determinare il miglior fabbisogno calorico e nel dettaglio l'apporto di proteine, carboidrati e grassi è molto importante per migliorare lo stile di vita salutare poiché ognuno di questi macronutrienti ha un effetto diverso nel nostro corpo.

1.1.1 Proteine

Secondo l'Harvard Medical School [1], l'apporto corretto di proteine che una persona dovrebbe assumere è di 0.8g per ogni kg del proprio peso. L'assunzione di troppe proteine porta all'accelerazione dell'invecchiamento e all'aumento del rischio di sviluppare molte malattie croniche. Tra i tre macro-nutrienti, si è dimostrato che la proteina è quella che riduce l'aspettativa di vita molto più velocemente delle altre. Una soluzione potrebbe essere quella di sostituire in parte o totalmente le proteine animali con quelle vegetali, questo perché quest'ultime contengono gli stessi amminoacidi essenziali ma riducono il rischio di malattie cardiovascolari. [4]

1.1.2 Carboidrati

Cibi ricchi in carboidrati sono cereali, verdure amidacee (come patate ed altri tuberi), frutta, fagioli e zuccheri. Quest'ultimi sono semplici mentre gli amidacei sono carboidrati complessi. Cibi a basso contenuto glicemico e grano integrale offrono una protezione dal rischio di sviluppare un diabete di tipo 2. Per esempio un consumo regolare di grano integrale influisce su un 20-40% di riduzione di rischio nelle malattie cardiovascolari e un 20-30% in meno di sviluppare diabeti. Per esempio, pane bianco e carote hanno indici glicemici simili, ma le carote hanno molti meno carboidrati. In contrasto, cibi come patatine fritte o panini per hamburger dovrebbero essere eliminati o drasticamente ridotti dalla nostra dieta. Anche il modo in cui si cucinano i carboidrati è molto importante. Per esempio, il cibo non dovrebbe essere stracotto poiché cambia la struttura degli amidi e la velocità di digestione. Per esempio gli spaghetti cotti al dente, vengono digeriti molto lentamente e ciò induce a un livello glicemico più basso. [5] Un altro modo per rallentare l'assorbimento dei carboidrati nell'intestino è quello di aggiungere un cucchiaino di olio extra-vergine di oliva al pasto.

1.1.3 Fibre

Tra i cibi che possiedono un alto contenuto di fibre si possono annoverare la frutta secca, i legumi, i carciofi e cereali come il mais, grano, riso ed orzo. Un consumo

[1] `https://www.health.harvard.edu/blog/how-much-protein-do-you-need-every-day-201506188096`

sostanzioso giornaliero di fibre al può ridurre il rischio di mortalità, infatti uno studio [2]ha riportato che le persone che consumano 14 g o meno di fibre al giorno hanno un rischio di mortalità del 30% più alto rispetto a chi ne consuma 30g o più. Altri studi epidemiologici affermano che chi consuma tra i 25 e i 29g di fibre al giorno, ha un rischio ridotto di morire di diabete, infarto e cancro al seno. La fibra più salutare sembra essere quella del grano integrale, cereali e verdure non amidacee. Tuttavia, è sconsigliato prendere integratori di fibre. I dati di uno studio clinico randomizzato in tre anni ha mostrato che gli individui che assumevano 3,5g di fibre come integratori avevano un incremento significativo nel rischio di tumori. Un modo per incrementare l'assimilazione di ferro e zinco è aggiungere vitamina C. Un consiglio è quello di aggiungere succo di limone per cucinare e condire le verdure, legumi e cereali.

1.1.4 Frutta secca e legumi

È fondamentale avere sempre una varietà di grano integrale e legumi all'interno della dieta. Questi cibi possono essere utilizzati come contorno per i pasti o anche come pasto principale, una porzione di legumi, approssimativamente metà tazza, fornisce una media di 120 calorie, 20g di carboidrati, 6-10g di fibre e 8g di proteine. Studi epidemiologici indicano che, persone che consumano legumi, dalle due alle quattro volte a settimana, hanno un rischio ridotto di morte, in particolare per le malattie cardiovascolari. Analisi condotte dal Nurses' Health Study hanno scoperto che le donne che consumano due o più porzioni di legumi a settimana hanno il 24% in meno di possibilità di sviluppare un cancro al seno. Non solo i legumi ma anche semi e frutta secca sono una grande fonte di vitamine essenziali e minerali come potassio, magnesio, calcio e ferro. Uno studio rivela che chi consuma almeno cinque porzioni di frutta secca alla settimana riduce del 40-60% il rischio di sviluppare malattie al cuore. La frutta secca contiene bassi valori di grassi saturi e ricca nei grassi mono saturi e polinsaturi ed è ricca in fibre solubili che riducono l'assorbimento del colesterolo nel nostro intestino. Tutta la frutta secca, in particolare mandorle e nocciole, sono la fonte principale di vitamina E antiossidante e dei minerali antiossidanti. Tra gli altri vantaggi la frutta secca possiede valori bassi di sodio e alti in potassio e magnesio, fattori che hanno un grosso effetto nell'abbassamento della pressione. Le noci sono fonte di omega-3, il quale abbassa i livelli trigliceridi, infiammazione e protegge contro irregolari frequenze cardiache. Molta frutta secca in commercio però, contiene sale, olio vegetale e zucchero, i quali hanno effetti dannosi sulla pressione sanguigna e sul peso e incrementano il rischio di diabeti, quindi si consiglia di acquistare frutta secca ancora nel proprio guscio così da evitare di assumere cibi già trattati.

1.2 Attività fisica

L'inattività fisica rientra tra i nove principali fattori responsabili dello sviluppo di malattie cardiache in tutto il mondo.[6] È importante tenere in mente che anche piccole dosi di attività fisica sono benefiche per la salute. Ogni settimana bisognerebbe incorporare più sessioni di attività fisiche che possiamo. Idealmente bisognerebbe trovare 30 minuti liberi per fare attività fisica o altrimenti anche 10 minuti distribuiti lungo il giorno sono sufficienti a fornire benefici per la salute. Secondo un Report stilato dal Physical Activity Guidelines Advisory Committee Scientific nel 2018, per raggiungere dei benefici sostanziali alla salute bisognerebbe svolgere almeno 150-300 minuti di attività moderata o 75-150 minuti di attività ad alta intensità alla settimana.
Il VO2max è la metrica che rappresenta il massimo volume di ossigeno consumato per minuto (in millilitri) per chilogrammo di peso e definisce il livello cardiorespiratorio e aerobico personale. È un valore che descrive quindi la capacità dell'organismo di portare l'ossigeno raccolto dai polmoni in tutte le parti del corpo, attraverso il sistema circolatorio, per poterlo usare come parte del processo di produzione di energia. Un atleta allenato che ha un VO2max di 5 litri per minuto brucia 1100 calorie durante un esercizio ad alta intensità, mentre una persona non allenata con un VO2max di 1.6 litri per minuto brucerà solo 360 calorie. Tuttavia, per quest'ultima persona, tre mesi di regolare attività fisica moderata, porteranno a un incremento del 15-25% del VO2max. Anche negli individui più anziani, un programma di attività aerobica moderata indurrà un incremento del 20% del VO2max. Il beneficio dell'allenamento aerobico, tuttavia, non è ristretto solamente alle calorie bruciate e al controllo del peso corporeo, ma ha anche effetti positivi sulla prevenzione della glicemia alta e diabete di tipo 2.
Dati da uno studio clinico ci dicono che camminare per circa 18km a settimana, solo 2.5 km al giorno, ha un'alta efficienza nel miglioramento della tolleranza al glucosio orale. Dati provenienti da studi randomizzati hanno mostrato che un anno di esercizio fisico può anche incrementare il volume dell'ippocampo, un'area del cervello che è responsabile delle attività mnemoniche. In altre parole, l'esercizio aerobico previene in maniera importante l'annebbiamento del cervello che peggiora con l'invecchiamento. Praticare con regolarità attività fisiche riduce lo stress mentale, l'ansia e migliora i sintomi della depressione. Praticare attività fisica con regolarità è molto importante poiché quando le persone smettono di allenarsi perdono i benefici positivi guadagnati attraverso l'esercizio fisico precedente.
Questo fenomeno è chiamato detraining, sono sufficienti dai 7 ai 14 giorni per perdere i benefici metabolici e di salute. Bastano quindi soltanto dieci giorni di detraining negli atleti allenati affinché questi verifichino un peggioramento della tolleranza del glucosio e un aumento dei livelli insulinici. D'altro canto, però, un esercizio eccessivo è associato a problemi cardiaci. Aumenta anche l'incidenza di

pericolosi problemi del ritmo cardiaco irregolare (aritmie), inclusa la fibrillazione atriale. In particolare, la fibrosi miocardica, che può aumentare il rischio di aritmia fatale, è stata rilevata nel 12-50% degli atleti che svolgono attività di resistenza (es. lunghe maratone). Per capire come sia correlato il dispendio di energie con lo sviluppo di patologie, viene utilizzato un parametro che si chiama MET. Un MET è un rapporto tra il tasso metabolico di lavoro rispetto al tasso metabolico a riposo. Il tasso metabolico è il tasso di energia spesa per unità di tempo. È un modo per descrivere l'intensità di un esercizio o di un'attività.
Un MET è l'energia che spendi seduto a riposo - il tuo metabolismo a riposo o basale. Quindi, un'attività con un valore MET uguale a 4, significa che stai esercitando quattro volte l'energia che avresti se fossi seduto fermo. Per metterlo in prospettiva, una camminata veloce a 5 chilometri all'ora ha un valore di 4 MET. La corda da salto, che è un'attività più vigorosa, ha un valore MET di 12,3. Studi hanno dimostrato che piccole dosi di attività fisica al giorno (dai 5 ai 10 minuti) sono sufficienti per avere benefici, mentre estendere l'allenamento aggiungendo esercizi ad alta intensità non provocherebbe benefici maggiori e in alcune persone potrebbe causare danni irreversibili. Dati derivanti da due studi clinici randomizzati hanno mostrato che sei mesi di attività fisica hanno portato a miglioramenti nella cognizione nelle persone anziane a rischio demenza, specialmente quando combinata con una dieta più sana. Si è anche dimostrato che l'esercizio aerobico applicato a persone che vanno dai 60 ai 79 anni ha effetti sull'incremento di materia grigia e bianca nella corteccia prefrontale del cervello. Altri studi hanno dimostrato che camminare a un ritmo intenso per 50 minuti, tre volte alla settimana per sei mesi, può migliorare la plasticità neuronale.

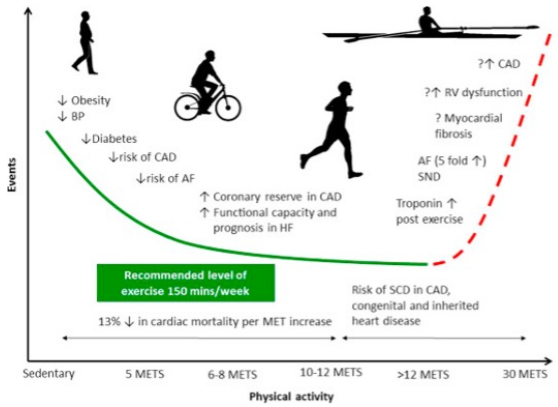

Figura 1.1: Relazione a forma di U tra salute ed esercizio fisico [7]

1.3 Sonno

Un altro fattore che influisce sulla longevità di una persona è il sonno, in particolare, la qualità di esso, lungo tutta la durata della vita.

1.3.1 Fasi del sonno

Il sonno può essere diviso in due fasi: REM(Rapid Eye Movement) e NREM. La prima viene chiamata anche sonno paradosso ed è la fase nella quale la persona sogna. La seconda invece, N-REM, viene divisa in tre stadi, NREM1, NREM2 e NREM3.

- NREM1: Il primo stadio, NREM1, dura all'incirca pochi minuti (da uno a sette), i muscoli sono ancora attivi, il movimento oculare diviene lento e il battito cardiaco inizia ad abbassare il proprio ritmo.
- NREM2: Questo stadio viene anche chiamato sonno leggero, occupa all'incirca il 25% del sonno, in questa fase i muscoli si rilassano ancora di più dello stadio precedente e la consapevolezza del mondo esterno sparisce.
- NREM3: Detto anche sonno profondo, durante questo stadio non c'è nessun movimento oculare nè muscolare, il respiro e il battito cardiaco sono rallentati ulteriormente e una persona difficilmente può venire svegliata. Durante questo stadio, vengono rilasciato gli ormoni della crescita il corpo si ristora al massimo delle proprie possibilità.

Dopo il terzo stadio, NREM3, si entra nella fase REM, il periodo del sonno, nel quale l'individuo sogna e comprende il 20-25% del sonno. In una notte di sonno che va dalle sette alle dieci ore, un adulto medio avrà all'incirca dai quattro ai sei cicli di sonno REM, intervallati ogni 80-100 minuti. Questa fase, si modifica durante gli anni, infatti se nei primi anni di età questa occupa il 50% del sonno totale, diminuisce costantemente, fino ad arrivare al 25% nella vecchiaia.
Con il passare degli anni infatti, il sonno si deteriora e i cicli di questo, che in giovane età, si manifestano regolarmente, dopo, diventano più instabili e interrotti. Studi riportano che le persone che dormono con costanza, meno di sei ore a notte, incrementano del 30% il rischio di sviluppare demenza. [8] Altri studi hanno fatto emergere che anche dormire tante ore, più di 9 ore al giorno, può causare un aumento del rischio di mortalità. [9] Per concludere, non si ha con certezza un numero preciso di ore di sonno consigliato per migliorare la longevità però è stato studiato che un interruzione del ciclo notturno e un'irregolarità dell'ora in cui si va a letto sono cause associate a un invecchiamento non sano. [10]

Capitolo 2

Applicazioni per il monitoraggio della salute

2.1 Diffusione e utilizzo

Lo sviluppo di mobile app dedicate al controllo della salute, dette anche health app, ha rivoluzionato il rapporto tra utente e le proprie abitudini. Nel solo 2020, sono state aggiunge, ai vari servizi di distribuzione di applicazioni per dispositivi mobili, come Google Play Store e Apple Store, più di 90mila health app, con una media di 250 app al giorno. Quando si parla di health app spesso ci si può confondere poichè il settore della salute è molto ampio, quindi bisogna fare una distinzione tra le migliaia che possiamo trovare. Fanno parte delle health app, due categorie principali: quelle che si concentrano sul benessere della persona, le quali permettono all'individuo di monitorare ed eventualmente cambiare le proprie abitudini fisiche ed alimentari, tra le app più scaricate facenti parte di questa categoria ci sono Google Fit, e quelle invece che riguardano la gestione delle condizioni di salute, le quali mostrano dettagli sulle patologie dell'individuo e consentono di accedere a dati clinici o somministrazione di farmaci. Tra quest'ultime, la categoria che rappresenta la maggioranza è quella delle patologie croniche come salute mentale e disturbi del comportamento, seguite da diabete e app che controllano il sistema circolatorio[11]. Tra le app più scaricate riguardanti questa categoria si trovano mySugr, OneTouch Reveal, che aiuta i pazienti a controllare il proprio diabete tracciando i livelli di glucosio nel sangue, l'uso di medicine e dati riguardanti l'attività fisica, MediSafe Medication Management app, che permette ai pazienti di inserire medicinali e le loro dosi all'interno dell'app e fornisce dei reminder e avvisi all'utente riguardanti l'assunzione di tali medicinali. Tra le app che invece si concentrano sul controllo del benessere, si possono trovare diverse tipologie con scopi e servizi diversi tra di loro. In base alle preferenze che l'utente possiede,

vengono proposte diverse aree di benessere tra le quali: il sonno, l'attività fisica, la nutrizione e la meditazione. Tra le app più scaricate che gestiscono la salute mentale si possono citare Mindshift, che aiuta alle persone a rilassarsi e a gestire situazioni di panico ed ansia e Headspace, che guida l'utente attraverso dei percorsi meditativi a ridurre lo stress e a migliorare la qualità del sonno.

2.2 Esempi di health app in commercio

Di tutte le health app che sono in commercio, solo 110 sono state scaricate per più di 10 milioni di volte [1]. Tra queste possiamo trovare applicazioni sviluppate da compagnie leader nel settore come Xiaomi, Samsung, FitBit, Google che accompagnano all'uso dell'app il loro relativo fitness tracker. Di queste applicazioni è stata fatta un' analisi e una comparazione per capire quali dati vengono mostrati all'utente e in che modo questo si interfaccia ad essi.

2.2.1 Samsung Health

Questa applicazione, prodotta dal colosso coreano Samsung supera il miliardo di download nel Google Play Store e si piazza al primo posto tra le fitness app con utenti più attivi [2]. L'applicazione permette di tracciare i dati, quali passi, battito cardiaco e sonno sia attraverso i sensori del proprio smartphone (accelerometro e fotocamera) oppure attraverso il collegamento di un fitness tracker. L'utilizzo di un fitness tracker è fortemente consigliato in quanto uno studio condotto per 2 mesi [12] ha analizzato che i risultati riguardo la misurazione dei passi possono differire fino al 30 % poiché l'utente non porta sempre con se lo smartphone durante il trascorrere della giornata.

[1] `https://www2.deloitte.com/us/en/blog/health-care-blog/2021/how-digital-health-apps-are-empowering-patients.html`

[2] `https://www.similarweb.com/it/apps/top/google/app-index/us/health-fitness/top-free/`

Figura 2.1: Sezione home-page

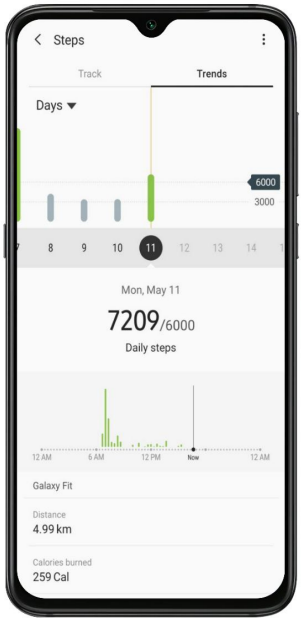

Figura 2.2: Sezione passi

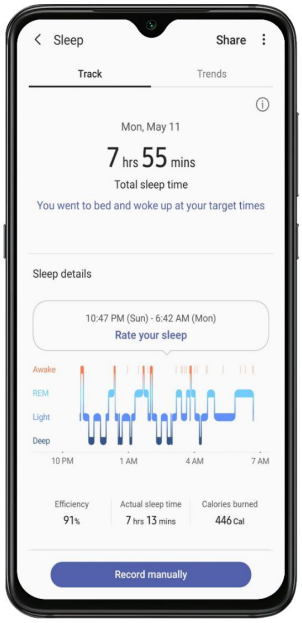

Figura 2.3: Sezione sonno

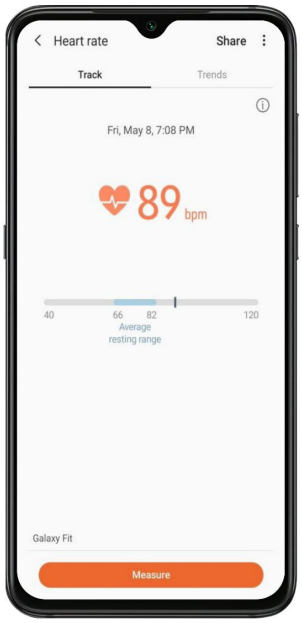

Figura 2.4: Sezione battito cardiaco

Figura 2.5: Sezione peso

2.2.2 Google Fit

L'applicazione del fitness di Google supera i 100 milioni di download nel Google Play Store e si piazza al quinto posto tra le fitness app più utilizzate al momento [3]. L'app come Samsung Health, permette sia di ricevere i dati direttamente dallo smartphone e sia da un fitness tracker collegato. Tra le sezioni che si possono trovare al suo interno, spicca quella dell'attività fisica, con il conteggio dei passi, che riveste un ruolo centrale nella homepage. Attraverso la navigazione si trovano le sezioni del sonno, parametri vitali e misurazioni corporee.

Figura 2.6: Sezione homepage

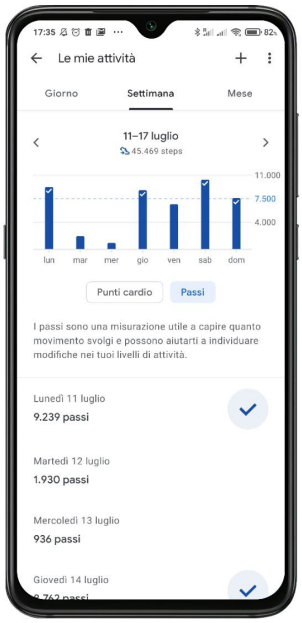

Figura 2.7: Sezione passi

[3] `https://www.similarweb.com/it/apps/top/google/app-index/us/health-fitness/top-free/`

Figura 2.8: Sezione sonno

Figura 2.9: Sezione battito cardiaco

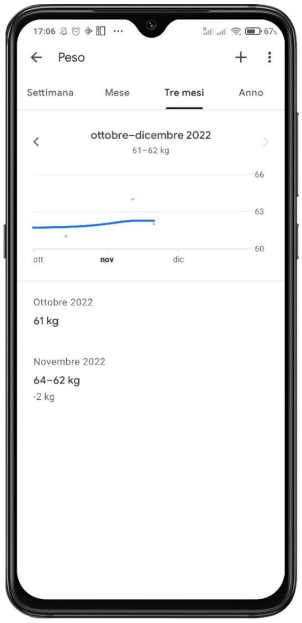

Figura 2.10: Sezione peso

2.2.3 Zepp Life

Questa applicazione, prodotta dall'azienda cinese Xiaomi, supera anch'essa i 100 milioni di download nel Google Play Store ma annovera meno utenti attivi piazzandosi al quarantacinquesimo posto tra le fitness app più utilizzate al momento [4]. L'app, che si connette ad uno dei molteplici fitness tracker di marca Xiaomi, presenta diverse funzionalità per monitorare l'attività fisica e la salute del paziente che vanno dal conteggio dei passi percorsi fino ad arrivare al monitoraggio del peso dell'utente lungo il tempo.

Figura 2.11: Sezione home-page

Figura 2.12: Sezione passi

[4] `https://www.similarweb.com/it/apps/top/google/app-index/us/health-fitness/top-free/`

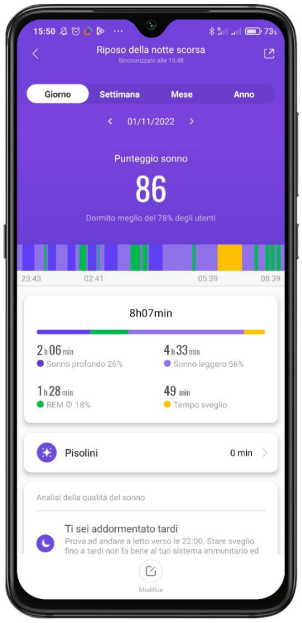

Figura 2.13: Sezione sonno

Figura 2.14: Sezione battito cardiaco

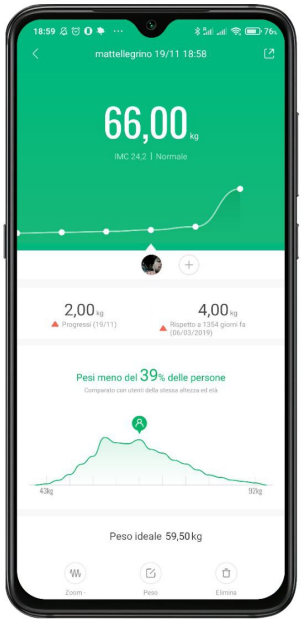

Figura 2.15: Sezione peso

2.3 Considerazioni

L'utilizzo di una applicazione mobile connessa con un fitness tracker diventa imprescindibile se si vuole avere una stima più accurata delle proprie abitudini giornaliere. In tutte le applicazioni analizzate il dato viene mostrato attraverso istogrammi, grafici ad area o grafici a ciambella che servono a fornire un'idea delle abitudini e dei progressi dell'utente. La maggior parte di queste app si presta però a mostrare solamente i dati mentre HealthApp, oltre a fare questo, grazie anche all'accurato studio dei medici del centro Ospedaliero di Verona, è in grado anche di fornire un riscontro costante al paziente attraverso degli indici di feedback e delle recommendation specifiche, prodotte automaticamente dopo l'analisi delle abitudini del paziente. Il design dell'applicazione prende spunto da tutte le applicazioni che sono state analizzate, cercando però di avere una sua identità e personalizzazione.

Capitolo 3

Comunicazione con il fitness tracker

3.1 Primo approccio

Al fine di analizzare i comportamenti e le abitudini giornalieri di una persona, è necessario che questa possieda un fitness tracker e che questo si connetta ad un'applicazione mobile. Grazie alla connessione tra questi due dispositivi, l'applicazione riceve i dati provenienti dal braccialetto, li processa e li mostra all'utente. Questa connessione è garantita da una tecnologia wireless chiamata Bluetooth Low Energy. In un primo momento si è pensato di acquisire i dati, quali i passi effettuati e la misurazione del battito cardiaco dal fitness tracker, però senza ricorrere ad API ed applicazioni proprietarie. La prima idea è stata quella di sviluppare una applicazione mobile che gestisse la comunicazione con il fitness tracker del paziente. Attraverso una ricerca che è stata effettuata da un altro testista si è deciso di scegliere come fitness tracker, il Mi Band 3, di marca Xiaomi. La scelta è ricaduta su questo dispositivo poiché permette una connessione e autenticazione senza dover ricorrere alla propria applicazione proprietaria, Zepp Life. Prima di eseguire dei tentativi per collegare il braccialetto alla nostra applicazione mobile, si è studiato lo standard Bluetooth Low Energy per capire come funziona e permettere di accedere ai dati che si volevano acquisire dall'utente.

3.2 Bluetooth Low Energy

Bluetooth® Low Energy è uno standard a basso consumo usato per connettere dispositivi vicini tra loro. Come dice il nome stesso, questa tecnologia garantisce un consumo energetico ridotto rispetto al Bluetooth "classico". Il BLE è una tecnologia ideale per casi d'uso come fitness tracker e altri dispositivi portatili ed è anche utilizzato nei dispositivi domestici intelligenti e sensori industriali. BLE è stato introdotto nel 2010 insieme all'uscita della versione Bluetooth 4.0 e si è fatto strada nelle nuove applicazioni che fanno parte del mondo dell'Internet of Things dove piccole quantità di dati vengono trasferite a basse velocità.

3.2.1 Concetti generali

Lo stack del protocollo BLE è diviso in tre parti:

- Applicazione: è responsabile della logica dell'applicazione e dell'user interface.
- Host: l'host è composto da diversi layer:
  - Generic Access Profile (GAP)
  - Generic Attribute Profile (GATT)
  - Logical Link Control and Adaptation Protocol (L2CAP)
  - Attribute Protocol (ATT)
  - Security Manager (SM)
  - Host Controller Interface (HCI)
- Controller
  - Link layer (LL)
  - Physical Layer (PHY)

La comunicazione tra i dispositivi è garantita dal GAP (Generic Access Profile), uno dei layer dello stack del Bluetooth Low Energy che definisce diversi aspetti specifici, tra i quali i ruoli dei dispositivi, i modi con cui comunicano tra di loro e i parametri di comunicazione e connessione che utilizzano. Due dispositivi possono comunicare tra di loro attraverso la tecnologia BLE in due modi che si differenziano in broadcasting e connessione. Nella prima i dispositivi sono chiamati broadcaster e observer, dove il broadcaster è il dispositivo che manda dei pacchetti a qualsiasi dispositivo che si trovi nel proprio range di comunicazione, mentre l'observer legge i pacchetti. La tipologia a rete è a stella e la comunicazione è unidirezionale, uno a molti, partendo dal broadcaster e arrivando a uno o più observer.

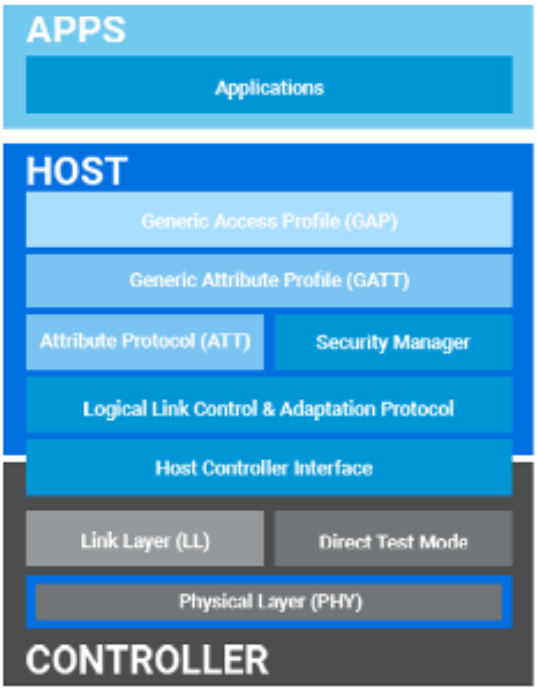

Figura 3.1: Stack layer Bluetooth Low Energy

Per stabilire un link sicuro ed efficiente tra i due dispositivi si avrà bisogno di una connessione poichè questa abiliterà uno scambio bidirezionale di dati tra i dispositivi e limiterà il consumo di energia di entrambi. Il Central device, che è il dispositivo più complesso (di solito può essere uno smartphone o un tablet o un pc), effettua una scansione di altri dispositivi e avvia il processo di connessione al peripheral, che è invece il dispositivo che ha funzioni limitate (di solito lo smartband), il quale aspetta la connessione e manda segnali al central device.

Il BLE è una tecnologia asimmetrica in quanto, la maggior parte delle operazioni è eseguita dal central device, il quale permette così al peripheral device, di lavorare solo quando necessario, risparmiare batteria e durare a lungo. La comunicazione è solamente possibile tra un central e un peripheral device, due central o due peripheral non possono comunicare tra di loro. Il dispositivo che agisce da Peripheral manda in broadcast i pacchetti di modo che il Central riesca a trovarlo e ad inviargli un pacchetto di richiesta di connessione. A questo punto la connessione è stata creata e il Peripheral smette di inviare advertisements. Dopo uno scambio di altri due pacchetti di dati la connessione è stabilita e i dispositivi sono pronti a scambiarsi i dati di cui hanno bisogno. Dopo aver stabilito una connessione il Central diventa Master e il Peripheral diventa Slave. Il primo ha il compito di iniziare la connessione e di gestirla durante la sua durata e una volta stabilita la connessione entrambi i

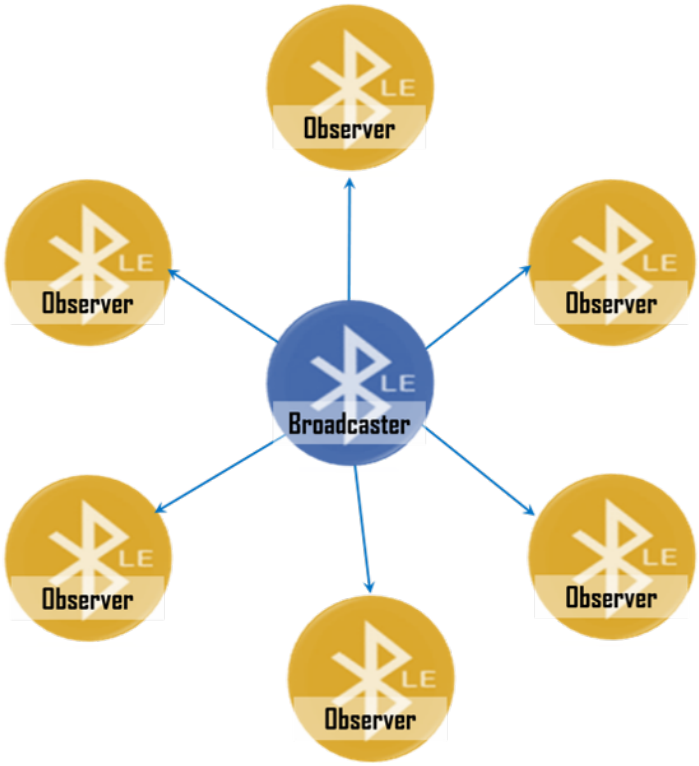

Figura 3.2: Topologia di rete

dispositivi potranno scambiarsi i dati in qualsiasi direzione. I modi e le procedure con cui questi dati vengono scambiati attraverso una connessione BLE, sono definiti dal Generic Attribute Profile, detto GATT, il quale usa l'Attribute Protocol (ATT), un protocollo che definisce come sono disposti i dati di un server ad un client e come sono strutturati all'interno del server. Ci sono due ruoli all'interno dell'Attribute Protocol:

- GATT Server: che contiene le risorse esposte al Client, organizzate come un database di attributi. Un GATT server contiene uno o più servizi GATT. Un servizio può contenere zero o più funzionalità che vengono chiamate caratteristiche.
- GATT Client: che controlla il comportamento del server e legge i dati che sono esposti, è il dispositivo che invia comandi e richieste al server.

In alcuni casi un dispositivo può anche svolgere questi due ruoli in contemporanea.
Il **GATT server** stabilisce una gerarchia per gestire i suoi dati, i quali vengono denominati attributi e utilizzano una struttura dati composta da 4 campi:

- Handle: è un identificatore lungo 16 bit usato dal GATT server per identificare ogni attributo
- UUID (Universally Unique Identifier): è un numero di 128 bit (16bytes) che per efficienza viene ridotto a 16 o 32 bit. Questi ultimi due formati possono essere usati solo se sono definiti dal Bluetooth SIG come standard Bluetooth UUID. (https://novelbits.io/uuid-for-custom-services-and-characteristics/)
- Valore: contiene il dato che viene esposto dal server, ha una lunghezza variabile e il formato dipende dal tipo di dato che contiene. Il valore può essere qualsiasi cosa, dal valore del battito cardiaco misurato in battiti per minuto o i passi giornalieri misurati da un fitness tracker
- Permessi: descrivono le proprietà dell'attributo (lettura, scrittura, notifica) e i livelli di sicurezza che proteggono il suo valore

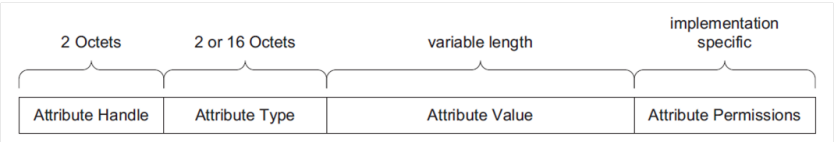

Figura 3.3: Struttura dati attributi

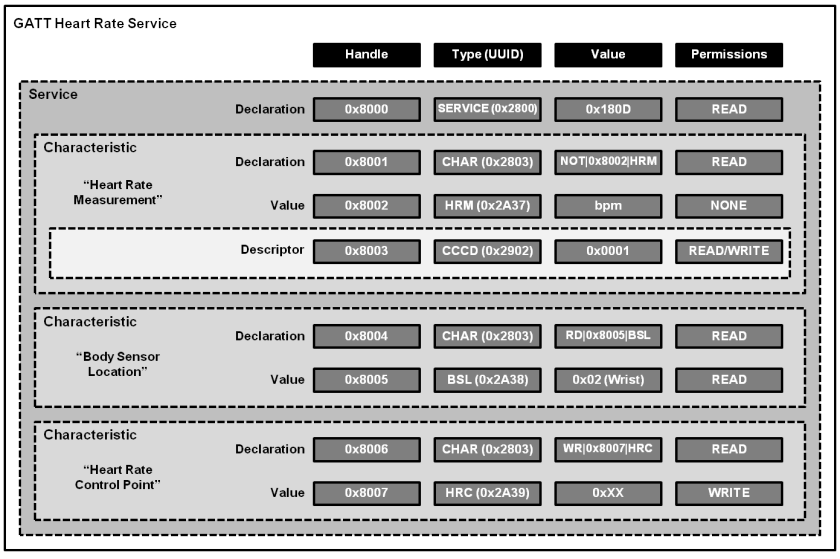

Figura 3.4: Gerarchia all'interno del Heart Rate Service [13]

Ci sono tre livelli in questo modello:

- Profilo: è un oggetto non visibile nel dispositivo BLE, è semplicemente un contenitore di servizi che vengono utilizzati per un uso specifico. Un esempio di profilo è quello dell'Heart Rate Profile, il quale viene utilizzato in campo medico e sportivo con sensori che permettono di misurare il battito cardiaco.

- Servizio: un servizio è un contenitore di una o più caratteristiche. Tra i servizi più comuni si possono trovare l'Heart Rate Service, che contiene al suo interno tre caratteristiche, delle quali una, Heart Rate Measurement, fornisce il valore istantaneo in battiti per minuto (bpm) del battito cardiaco. Uno dei servizi più utilizzati è il Battery Service, il quale presenta una caratteristica che contiene il dato relativo alla percentuale di batteria residua del dispositivo BLE.

- Caratteristica: è un contenitore di dati che hanno almeno un minimo di due attributi:

  – Attributo di dichiarazione (Characteristic Declaration Attribute): che come dice dal nome è il metadato che descrive la caratteristica. Al suo interno è contenuto l'UUID della caratteristica, che potrebbe essere sia un UUID conforme allo standard Bluetooth-SIG e quindi essere lungo 16 bits o un UUID specifico del proprietaria e in questo caso essere 128 bits.

  – Attributo del valore (Characteristic Value Attribute): contiene il valore del dato.

    Le caratteristiche hanno un ruolo fondamentale nel BLE poichè contengono l'informazione che si vuole leggere o scrivere. Esse sono usate anche per scambiarsi dati quando si sta cercando di stabilire una connessione tra due dispositivi.

Il **GATT Client**: solitamente questo ruolo viene implementato da uno smartphone, tablet o un computer. Il client è interessato ad accedere ai dati che sono esposti dal GATT Server. Prima di ottenere i dati il cliente deve seguire delle operazioni o transazioni:

1. Scoperta dei servizi:il client non essendo a conoscenza dei servizi disponibili, deve performare una scansione di questi per poter avere una struttura completa di servizi e caratteristiche che gli vengono esposte.

2. Leggere caratteristiche: con questa operazione il client è in grado di leggere il valore di una caratteristica a seconda della propria preferenza. In base al livello di sicurezza della connessione, alcune caratteristiche potrebbero non essere leggibili dal client poichè il server declina l'accesso ai propri dati se il livello di sicurezza non è quello requisito.

3. Scrivere caratteristiche: con questa operazione il client è in grado di scrivere il valore di una caratteristica a seconda della propria preferenza. Come succede per la lettura, alcune di queste caratteristiche potrebbero non essere scrivibili dal client per questioni di sicurezza.

3.2.2 Vantaggi e svantaggi BLE

Vantaggi

I vantaggi di questa tecnologia sono molti, a partire dal basso consumo di energia che offre ai dispositivi che la sfruttano. Per ridurre il consumo energetico, un dispositivo BLE è tenuto in sleep mode per la maggior parte del tempo. Quando un evento è scatenato, il dispositivo di sveglia e invia un messaggio contenente i dati di cui ha bisogno il client e poi torna in sleep mode. Una connessione BLE accesa per 24 ore può consumare tra 1-3% della batteria di uno smartphone , mentre una connessione Bluetooth può arrivare anche fino al 5-8% nello stesso periodo [1]. Un altro vantaggio è dato dalla robustezza e della affidabilità della connessione che può arrivare ad una portata massima di oltre 100 metri[2] ed essere in grado di gestire improvvise perdite di segnale o interferenze. Un altro vantaggio è dato dal fatto che chip e moduli BLE hanno un costo ridotto rispetto ad altre tecnologie e ad oggi sono presenti in quasi tutti gli smartphone.

Svantaggi

Uno degli svantaggi di questa tecnologia è il basso data throughput che possiede. BLE ha un bitrate designato di 1 Megabit per secondo (1Mbps), ma può scendere in modo significativo a causa del sovraccarico del protocollo o delle prestazioni radio. Un altro svantaggio risiede nel range di comunicazione: dato il suo scopo di risparmio energetico, BLE si concentra principalmente su una comunicazione con un raggio ridotto. Due dispositivi BLE possono comunicare fino a 30 metri di distanza se non ci sono interferenze tra di loro ma di norma una tipica comunicazione avviene dai 2 ai 5 metri [3]

[1] `https://www.pointr.tech/blog/bluetooth-low-energy-ble-use-phone-data-battery`

[2] `https://www.sciencedirect.com/science/article/abs/pii/S1570870515001663`

[3] `https://www.rfid-wiot-search.com/iot-knowledge/ble,https://pcng.medium.com/the-basic-concepts-of-bluetooth-low-energy-ble-for-beginner-c0fe062190c5`

3.2.3 Connessione e autenticazione con il fitness tracker

Dopo una prima fase di studio del protocollo BLE, si è deciso quindi di sviluppare un'applicazione che fosse in grado di connettersi al fitness tracker scelto e di ottenere dati come i passi completati durante la giornata e il battito cardiaco sia istantaneo che a riposo. Per l'applicazione si è deciso di utilizzare un'implementazione nativa come quella di Android, poiché risultava più ampia in documentazione online per l'implementazione di un servizio come quello di BLE. Come punto di partenza si è presa come spunto l'applicazione mobile nRF Connect for Mobile [4], la quale è in grado di scannerizzare, esplorare i dispositivi Bluetooth Low Energy e comunicare con loro attraverso l'utilizzo in lettura e scrittura delle caratteristiche e delle notifiche. Inoltre, si è fatto uso di guide online per capire il procedimento di acquisizione dati e di connessione con il braccialetto. [5].
Nello sviluppo dell'applicazione si è dichiarato un valore chiamato SCAN_PERIOD, il quale definisce la durata, in millisecondi, della scansione dei dispositivi. Durante la scansione vengono trovati tutti i dispositivi BLE che sono nel range del Central device e se ne seleziona uno per stabilire una connessione con esso. Per garantire un'esperienza migliore all'utente si è deciso di filtrare i dispositivi per nome, mostrandogli solamente i fitness tracker con casa produttrice Xiaomi, quindi i Mi Band.

[4] `https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp&hl=fr&gl=US`

[5] `https://medium.com/@yogeshojha/i-hacked-xiaomi-miband-3-and-here-is-how-i-did-it-43d68c27`

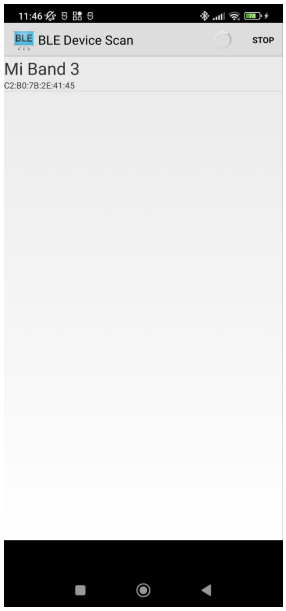

Figura 3.5: Scanning dispositivi con filtro per Mi Band

Una volta selezionato il Peripheral a cui l'utente vuole connettersi, il protocollo GATT, una volta stabilita la connessione prova a scoprire i servizi che sono esposti dal fitness tracker. Dopo aver scoperto i servizi che il fitness tracker espone,

```
// Attempts to discover services after successful connection.
Log.i(TAG, msg: "Attempting to start service discovery:" +
        mBluetoothGatt.discoverServices());
```

Figura 3.6: Ricerca dei servizi esposti dal fitness tracker

inizia la ricerca della caratteristica desiderata; per lo sviluppo dell'applicazione è necessario ottenere i dati provenienti dalle caratteristiche n° 00000007-0000-3512-2118-0009af100700, la quale espone i passi giornalieri compiuti dall'utente e dalla caratteristica n° 00002a37-0000-1000-8000-00805f9b34fb, che espone il valore del battito cardiaco istantaneo misurato in bpm (battiti per minuto). Una volta connessi al dispositivo e dopo aver richiesto i dati necessari attraverso le caratteristiche, si è verificato che il braccialetto si sconnetteva dal GATT Server dopo soli 30 secondi. Questo avveniva perché il Mi Band non aveva ancora effettuato la fase di autenticazione con il Central device, lo smartphone.
Per effettuare una corretta autenticazione, il Mi Band e lo smartphone si parlano attraverso una delle caratteristiche che espone il peripheral device, in questo caso la caratteristica di autenticazione(che viene chiamata auth carachteristic). Ogni volta che questa viene scritta e quindi conseguentemente cambiata, il central device procederà con gli step per portare a termine l'autenticazione. Durante il

primo step, il central device scrive la auth carachteristic esposta dal Peripheral inviando una chiave segreta di 16 byte concatenata con il byte 0x01 e richiede una chiave random scrivendo sempre la auth characteristic con il byte 0x02. Una volta che la auth characteristic è correttamente modificata vengono presi gli ultimi 16 byte del valore della caratteristica e vengono cifrati utilizzando l'algoritmo di cifratura AES/ECB/No Padding insieme alla chiave segreta. Questo valore cifrato viene poi concatenato al byte 0x03 e inviato al peripheral sempre scrivendo l'auth characteristic. Se, dopo quest'ultima operazione, la auth characteristic del peripheral espone i tre byte 0x10, 0x03 e 0x01 allora l'autenticazione è andata a buon fine altrimenti, se fallirà l'ultimo byte sarà 0x04.

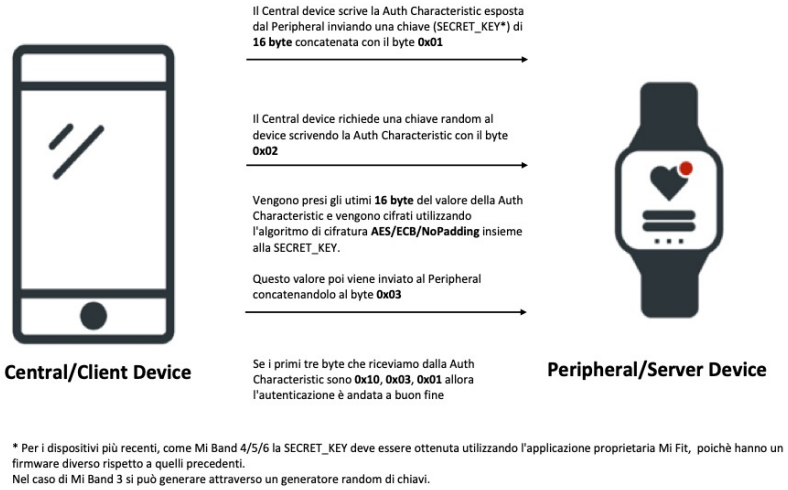

Figura 3.7: Autenticazione tra braccialetto e applicazione

Dopo aver effettuato l'autenticazione, il braccialetto e lo smartphone rimangono collegati fino a che non si decide manualmente di scollegarli. Attraverso il GATT Server quindi, i due dispositivi possono comunicare e scambiarsi dati. Una prima bozza di interfaccia è stata creata attraverso il programma Android Studio per mostrare i dati dei passi giornalieri provenienti dal fitness tracker dopo aver effettuato l'autenticazione.

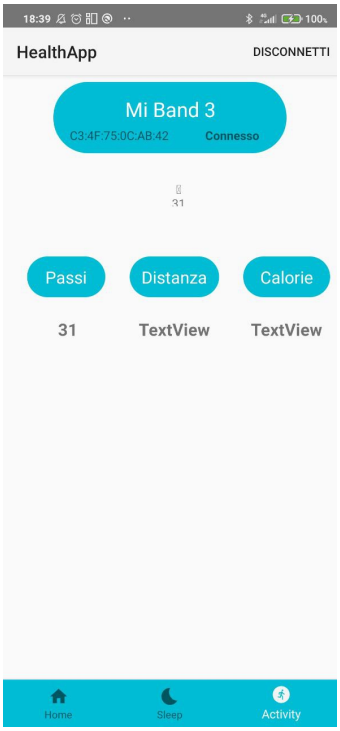

Figura 3.8: Prima bozza homepage HealthApp

3.2.4 Considerazioni e cambio approccio

Dopo aver connesso e autenticato il fitness tracker allo smartphone attraverso un'applicazione mobile nativa si è deciso però di cambiare approccio poiché questa soluzione mostrava dei limiti, come la velocità di sviluppo attraverso un'applicazione nativa, che è risultata non sufficiente per raggiungere l'obiettivo prefissato, cioè quello di creare un primo prototipo di applicazione che fosse disponibile per entrambi i sistemi operativi, Android e iOS. Un altro motivo per cui si è deciso di cambiare strada e optare per l'utilizzo di un'app proprietaria, per autenticare il fitness tracker e gestire il profilo del paziente, è stato quello che fitness tracker più recenti del Mi Band 3, affinché possano trasmettere dati ad una applicazione terza, richiedono inizialmente una registrazione e autenticazione iniziale da eseguire attraverso l'app proprietaria. La seconda strada quindi è stata quella di sviluppare un'applicazione mobile in modo cross-platform così che l'applicazione fosse disponibile immediatamente per dispositivi Android e iOS. Oltre a questo si è deciso anche di cambiare tecnologia di fitness tracker e preferire un braccialetto Fitbit al Mi Band 3 poiché Fitbit espone pubblicamente le API che sono necessarie per fornire i dati di cui HealthApp aveva bisogno, al contrario di Xiaomi, casa produttrice di Mi Band, che non permette di utilizzare le proprie API.

Capitolo 4

HealthApp: Sviluppo e Implementazione

4.1 Sviluppo cross-platform app

Per lo sviluppo dell'applicazione mobile si è deciso di scegliere un approccio cross-platform che permettesse così di avere nell'immediato l'applicazione disponibile sia per dispositivi mobili con sistema operativo Android che iOS. Grazie ad un approccio multipiattaforma lo sviluppatore può creare il codice una sola volta per più piattaforme, questo non succede se si sceglie di programmare un'app nativa, in questo caso bisognerebbe sviluppare un codice con due linguaggi diversi e molto probabilmente con framework differenti. Uno dei benefici delle app multipiattaforma è quello della riutilizzabilità del codice che garantisce una migliore produttività ed efficienza. La scelta dell'implementazione dipende dalle esigenze che il progetto vuole soddisfare in termini di budget, requisiti, vastità del pubblico a cui destinarlo e altri parametri. Ogni mese, vengono rilasciate nel solo Apple Store mediamente più di 30mila app [1], quindi la competizione nel mercato delle mobile app è ai massimi livelli, dato questo aspetto, scegliere un approccio cross-platform è nettamente la soluzione migliore rispetto ad un app nativa, in termini di velocità di sviluppo. Prendendo in considerazione il budget impiegato nella creazione di un app, anche qui la scelta di un sviluppo multipiattaforma viene preferito rispetto a un approccio nativo. Guardando il lato performance invece si ha una preferenza verso l'approccio nativo, in quanto i framework cross-platform solitamente possiedono un livello in più che permette la traduzione del codebase nel codice nativo di ogni singolo sistema operativo, questo porta a un rallentamento delle performance.

[1] https://www.statista.com/statistics/1020964/apple-app-store-app-releases-worldwide/

4.2 Scelta del framework

Dopo aver scelto di sviluppare l'applicazione seguendo un approccio cross-platform si sono analizzati vari framework che permettono lo sviluppo sia su Android che su iOS utilizzando un solo codice sorgente.

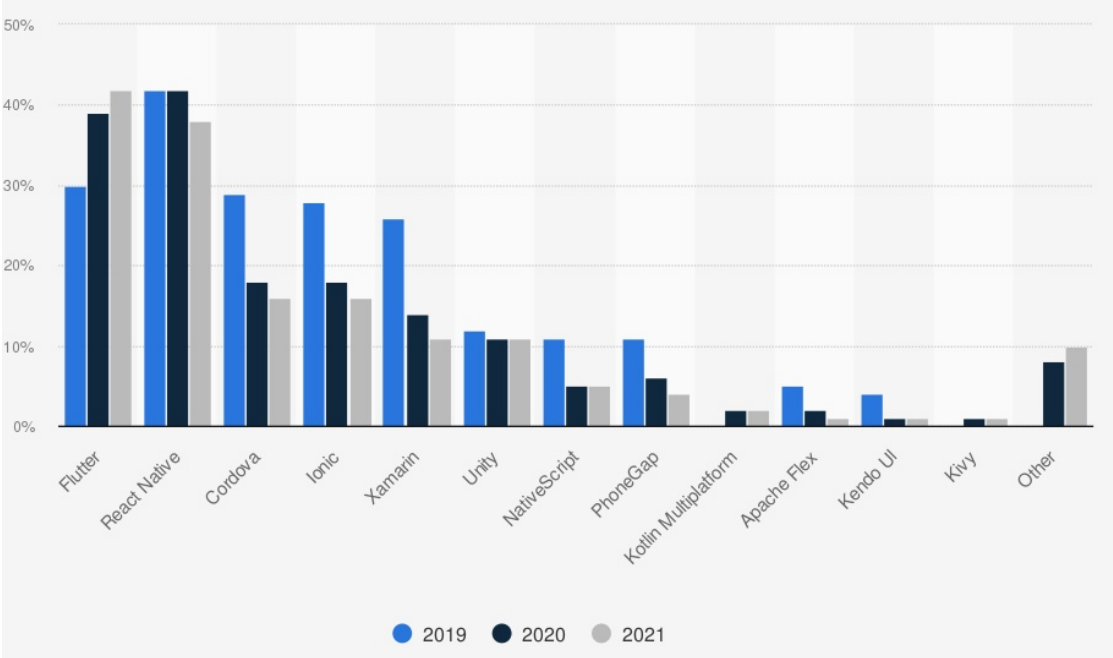

Figura 4.1: Framework cross-platform utilizzati da sviluppatori software dal 2019 al 2021 [14]

Come mostra la figura 4.1 i leader di questo settore sono React Native, Flutter e Ionic. Il primo è un framework sviluppato da Facebook che utilizza Javascript come linguaggio di programamzione, il secondo è sviluppato da Google e utilizza Dart come linguaggio mentre con il terzo si possono creare app attraverso HTML,CSS e Javascript. Per capire quale fosse più adatto allo sviluppo di HealthApp si sono valutate altre caratteristiche, come performance, community e utilizzo.
Dopo aver analizzato le caratteristiche di ognuno, si è deciso di scegliere React Native sia per il linguaggio di programmazione, Javascript, che è il più utilizzato dagli sviluppatori software [2], per le performance e inoltre poichè essendo che HealthAppWeb utilizza React come framework, si agevola la manutenzione del codice da ambo le parti.

[2] `https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages`

	React Native	**Ionic**	**Flutter**
Linguaggio	Javascript + Swift Objective-C, Java	HTML, CSS, Javascript Typescript	Dart
Riutilizzo codice	Facilmente riutilizzabile	Facilmente riutilizzabile	Facilmente riutilizzabile
Performance	Molto simile a quelle di applicazioni native	Non al livello delle altre poiché utilizza tecnologie web per il render dell'applicazione	La migliore poiché non ci sono ponti con i componenti nativi
Community	Vastissima poiché utilizza React, che è una delle librerie più conosciute e Javascript come linguaggio che è il più utilizzato dai software developers.	Community robusta poiché utilizza HTML, CSS e Javascript come linguaggi.	Community in via di sviluppo poiché Flutter è nata da poco (2017), ma comunque già in vetta alla classifica per utilizzo.
Utilizzo (2021)	38%	16%	42%
App popolari	Facebook, Instagram	Sanvello, Sworkit	Google Pay, Stadia, iRobot
Prezzo	Gratis (Open-source framework)	Gratis + piano a pagamento (Ionic Pro)	Gratis (Open-source framework)

Tabella 4.1: Confronto tra React Native, Ionic e Flutter

4.3 React Native

React Native è un framework per lo sviluppo di app cross-platform che si basa interamente sul linguaggio JavaScript. Presentato e sviluppato nel 2015 da Facebook è diventato una delle soluzioni più versatili e utilizzate per lo sviluppo di applicazioni mobili. React Native usa un ponte per interpretare o viceversa trasformare il codice nativo e renderlo compatibile con i propri componenti.

4.3.1 Componenti

L'elemento basico di React Native è il componente, il quale ha la potenzialità di essere riutilizzabile, con scopi diversi attraverso tutta l'applicazione. I componenti possono contenere al loro interno altri componenti, essi possono comunicare e passare informazioni sia ai componenti figli che al proprio genitore. I principali componenti base di React Native sono:

- **View**: è il contenitore più ad alto livello dell'interfaccia grafica, se si dovesse fare un paragone è il corrispondente dell' elemento div in HTML. Esso viene arricchito con uno stile e solitamente contiene al suo interno altri componenti.

- **ScrollView**: ha la stessa funzionalità di contenitore di View, ma si distingue dal sopracitato per la capacità di far scrollare lo schermo per permettere la vista di molteplici elementi al suo interno.

- **Text**: è il componente che permette di inserire un testo al suo interno. Solitamente viene inserito all'interno di un View.

- **StyleSheet**: fornisce uno stile alla vista, è simile al CSS ma possiede una sintassi diversa.

- **Pressable**: questo componente permette di catturare la pressione dell'utente su un'area dello schermo attraverso la funzione `onPress` e scatenare un evento a seguito della pressione.

Oltre ai componenti principali ce ne sono altri e sopratutto l'utente ha la possibilità di creare i propri, personalizzati, in base alle proprie necessità. I componenti di React sono controllati da due tipologie di dato: state e props. Lo stato è un parametro mutabile e viene dichiarato come const all'inizio del componente e settato con la funzione setState. Ogni volta che questo cambia e nella vista c'è un riferimento ad esso, la vista verrà renderizzata con il valore nuovo. Con le props invece i componenti riescono a passarsi i dati tra di loro. Le props, a differenza degli state, sono immutabili dal componente che le riceve e sono gestite dal componente padre insieme ad altre funzionalità. All'interno delle props è possibile passare qualsiasi tipo di valore, da stringhe a funzioni fino ad arrivare anche ad un componente.

4.3.2 Vantaggi

- Fast refresh: L'aggiornamento rapido consente agli sviluppatori di eseguire l'app mentre la aggiornano a nuove versioni e modificano l'interfaccia utente. Le modifiche sono visibili immediatamente e lo sviluppatore viene risparmiato dalla ricostruzione dell'intera app. Ciò porta a due vantaggi significativi: il risparmio di tempo, poiché i programmatori risparmiano tempo sulla compilazione e l'incremento della produttività.

- Interfaccia intuitiva: dato l'utilizzo di JavaScript per costruire l'interfaccia dell'applicazione, essa è veloce e reattiva, innalzando il livello di esperienza per l'utente.

- Performance: nonostante lo sviluppo di app native sia sicuramente più veloce, questa differenza di performance è difficilmente visibile ad occhio umano [3]

[3] `https://www.netguru.com/blog/swift-vs-react-native`

4.3.3 Svantaggi

- Scalabilità: nonostante questo framework venga utilizzato da colossi come Facebook e Instagram, alcune compagnie hanno deciso di fare un passo indietro e virare su altre soluzioni. Un esempio è quello di Airbnb, che, ha deciso di utilizzare il framework per poi abbandonarlo e scegliere lo sviluppo di due applicazioni native.

4.3.4 Pacchetti utilizzati

Oltre ai componenti base che React Native offre all'utente, come quelli sopracitati, è possibile utilizzarne altri attraverso librerie esterne. Queste possono essere importate nel progetto React Native e installate attraverso il registro `npm` usando un manager di pacchetti basato su `Node.js`. Per installare una libreria esterna basterà lanciare il comando `npm install package-name` e automaticamente la libreria verrà importata all'interno del nostro progetto e aggiunta al file di configurazione *package.json*.

- **Expo**

  Expo [4] è un framework usato per lo sviluppo di applicazioni mobile che si basano su React Native e permette allo sviluppatore di testare molto velocemente la propria applicazione. Con Expo si è in grado di emulare l'applicazione e lanciarla o sui simulatori di device o direttamente sui device fisici. Uno dei grandi vantaggi di Expo è la possibilità di simulare la propria applicazione sul proprio smartphone, con sistema operativo Android o iOS, attraverso l'app Expo Go che permette di emulare l'app attraverso l'uso di un QR-Code o di un link. Lo sviluppatore in questo modo può lavorare sul codice e vedere le modifiche in tempo reale sul proprio smartphone. Infine attraverso la Expo Cli e la EAS Cli è possibile generare un file con formato apk, standard utilizzato da Android per le applicazioni mobili, che permette all'utente di installare l'applicazione sul proprio smartphone e testarla.

- **React Navigation**

  React Navigation è una libreria che permette di avere una navigazione attraverso schermate lungo un' applicazione sviluppata su React Native. Il concetto principale è quello di suddividere le macro-sezioni in diverse schermate e inserirle all'interno di un componente padre che gestisce la navigazione di queste. Il componente padre è il `NavigationContainer`, il quale al suo interno contiene uno `Stack.Navigator`, che a sua volta contiene una o più

[4] `https://docs.expo.dev`

schermate che vengono rappresentate dal componente `Stack.Screen`. Lo Stack.Screen è definito da diversi attributi quali `name`, `component` e `options`, che rappresentano rispettivamente il nome della schermata che verrà utilizzato da chi la vorrà chiamare, il componente associato alla schermata e delle opzioni che possono includere il titolo della schermata da mostrare in cima allo schermo. Lo `Stack.Navigator` al momento della dichiarazione viene corredato con l'attributo initialRouteName che definisce qual è il primo screen ad essere renderizzato. Per muoversi attraverso le schermate occorre invece utilizzare la funzione `navigation.navigate(route` che permette all'utente di navigare in una schermata definita all'interno dello Stack Navigator. Se questa non è definita verrà riportato un errore. Per tornare indietro alla schermata precedente si utilizza la funzione `navigation.goBack()`, che di default renderizza il componente della schermata precedente. Un'altra funzione che è stata utilizzata anche all'interno dello sviluppo di HealthApp è la `navigation.replace(route)`, che permette di sostituire la uno screen con un altro, a differenza della `navigation.navigate` però non è più possibile ritornare indietro alla schermata precedente, che viene rimossa dallo stack di navigazione.

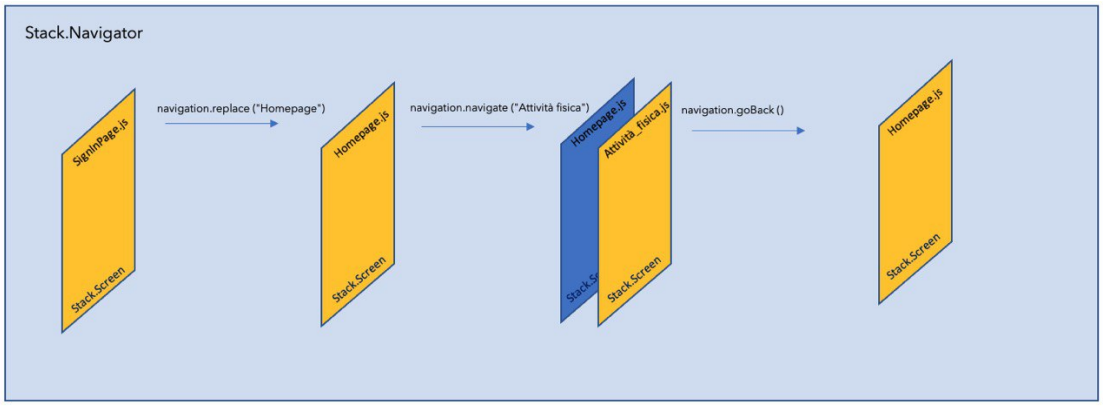

Figura 4.2: Navigazione in React Navigation

Come si può vedere in Figura 4.2, si sono definiti in arancione gli Stack.Screen renderizzati e in blu quelli presenti nello Stack ma non renderizzati e invisibili all'utente. Un'altra funzionalità di React Navigator è quella di poter passare parametri agli Screen durante la navigazione.

I parametri vengono ricavati dal nuovo Screen attraverso l'attributo `route.params` e quindi poi possono essere utilizzati all'interno del componente.

```
<View style={styles.button}>
 <CustomButton onPress={()=>navigation.navigate("Questionario", {
    nomequestionario : props.titolo,
    domande_e_risposte : props.domande_e_risposte,
    compiledAnswers:props.compiledAnswers,
    username:props.username,
    user:props.user,
    compilato:props.compilato,
    questionnaireTemplateId: props.questionnaireTemplateId
  })} text={props.compilato !== undefined ? "VISUALIZZA" : "COMPILA"} fontSize="small"/>
  </View>
</View>
```

CopertinaQuestionario.js

Figura 4.3: Parametri passati durante la navigazione

```
export default function Questionario({route,navigation}) {

  const {nomequestionario,domande_e_risposte,update,compilato,compiledAnswers,questionnaireTemplateId} = route.params;
  const [n_domanda,setNumeroDomanda] = useState(0);
```

Questionario.js

Figura 4.4: Parametri ereditati navigazione

4.4 Diagramma architetturale

L'applicazione mobile HealthApp comunica con il Fitbit indossato dal paziente attraverso il FitBit Server, il quale viene interrogato dal backend che a sua volta comunica con HealthApp attraverso chiamate API. Lo smartphone del paziente avrà al suo interno installata anche un'altra applicazione: Fitbit Connect App, che verrà utilizzata dal medico per effettuare la prima iscrizione dell'utente ai server di Fitbit e sarà responsabile della connessione con il fitness tracker. I dati verranno così inviati dal fitness tracker di Fitbit all'applicazione Fitbit Connect App e allo stesso tempo il backend di HealthApp comunicherà con il server Fitbit per salvarli nel proprio database. L'applicazione mobile, attraverso delle API, comunicherà con il backend HealthApp e mostrerà i dati all'utente.

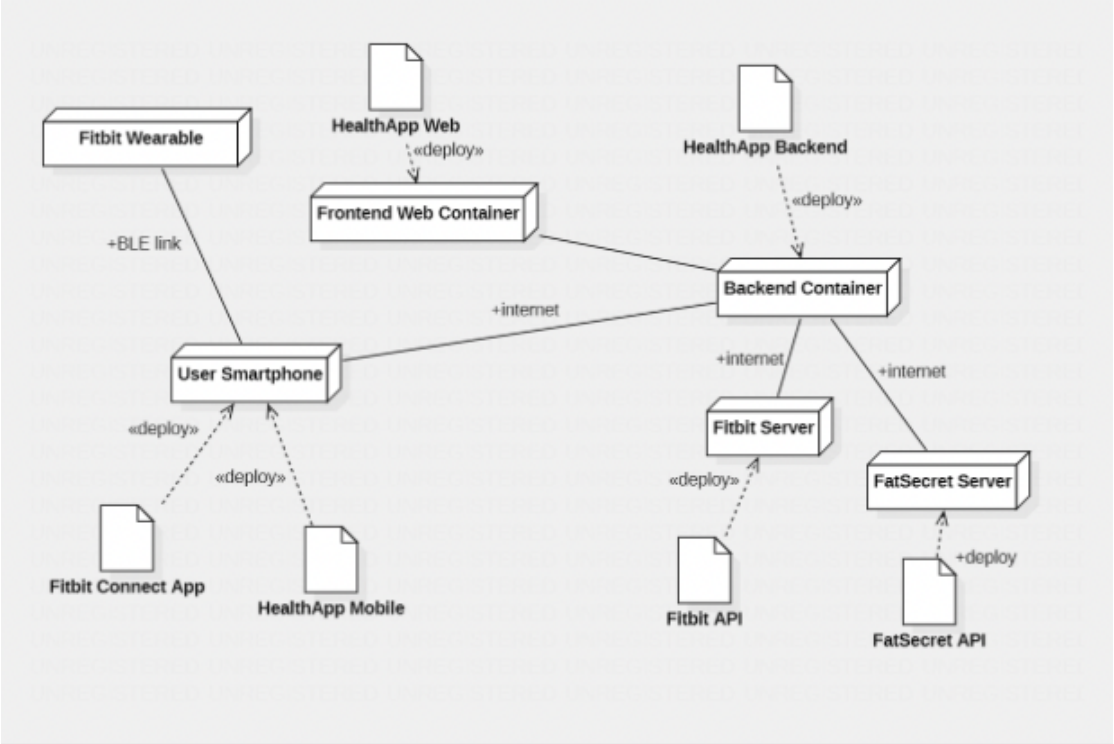

Figura 4.5: Diagramma architetturale HealthApp

4.5 Prototipo

Prima di iniziare lo sviluppo dell'applicazione, si è deciso di costruire un prototipo a bassa fedeltà con l'utilizzo del tool Balsamiq (https://balsamiq.cloud), affinchè si potesse avere una prima idea di design e di interazione dell'applicazione. La costruzione di un prototipo è molto importante poiché attraverso questo, il cliente può avere una prima bozza del prodotto dopo poco tempo. In questo modo il cliente può fornire dei feedback che verranno implementati nel prototipo e successivamente nello sviluppo dell'applicazione. Inizialmente è stato prodotto un prototipo a mano, disegnando sketch su carta e successivamente, dopo ogni schermata costruita si è passati ad implementarla su Balsamiq. Come si può notare nei paragrafi successivi, durante lo sviluppo dell'applicazione si è cercato di rimanere il più attinenti possibili al design delle schermate e alla navigazione implementata nel prototipo. Nel corso dello sviluppo però per alcune funzionalità si è deciso di scegliere una implementazione differente. Due esempi possono essere la sezione del peso, che inizialmente nel prototipo era stata pensata come schermata singola, poi, invece in fase di sviluppo è stata implementata come Modal, evitando di aggiungere una pagina di navigazione in più e la sezione dei questionari, che

inizialmente, prevedeva di rappresentare ogni singolo questionario all'interno di un cerchio che fosse cliccabile e poi in fase di sviluppo si è deciso di optare per una soluzione grafica differente utilizzando un bottone per accedere alla compilazione. E' stato creato inoltre un prototipo di alcune schermate per funzionalità che non sono state sviluppate nella versione 1.0 ma che verranno prese in considerazione per gli aggiornamenti futuri, per esempio la sezione dell'inserimento manuale dell'alimentazione.

4.5.1 Prototipo a bassa fedeltà

Figura 4.6: Prototipo a bassa fedeltà dell'applicazione

4.6 Pagine e navigazione

4.6.1 Login

Per la schermata di Login si è deciso di utilizzare una schermata semplice e diretta per l'utente. Attraverso il componente personalizzato *CustomInput* l'utente è in grado di inserire le proprie credenziali in due *TextInput* differenti, attraverso la propria tastiera. Il *CustomInput* adibito all'inserimento della password è contrassegnato con una *prop* di nome eye affinché l'utente possa nascondere per ragioni di sicurezza la propria password o visualizzarla nel caso di un eventuale controllo prima di confermarla. L'unico bottone presente nella schermata ha un ruolo ben preciso, una volta premuto, invierà le informazioni al server attraverso un'API e se la risposta avrà successo permetterà all'utente di accedere alla pagina iniziale dell'applicazione altrimenti l'utente verrà notificato attraverso un *Alert*, un componente di React Native, che c'è stato un errore di autenticazione, l'username e/o la password inseriti sono incorretti. Una volta completata la fase di accesso all'applicazione, l'utente viene reindirizzato alla schermata di homepage, attraverso la funzione navigation.navigate.

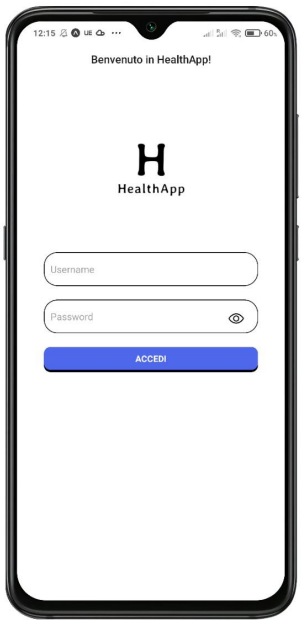

Figura 4.7: Schermata Login

Figura 4.8: Alert autenticazione fallita

SignInPage.js

```
return (
    <SafeAreaView style={styles.container}>
        <View style={styles.container_header}>
            <Image
                style={styles_.logo}
                source={require('../../../assets/logo.png')}
            />
        </View>
        <View style={styles.root}>
            <CustomInput placeholder="Username" value={username} setValue={setUsername}
                         onFocus={() => handleError(null, 'username')} error={errors.username}>
            </CustomInput>
            <CustomInput placeholder="Password" value={password} setValue={setPassword} eye={true}
                         onFocus={() => handleError(null, 'password')} error={errors.password}/>
          <View style={styles.loginButton}>
            <CustomButton  login={true} style={{width:"100%"}}onPress={
                validate
            } button={"first"} text={isLoading ? "Accedi" :
                (loggedUser?"Loggato":<ActivityIndicator/>)
            }
            />
          </View>
        </View>
    </SafeAreaView>
);
```

Figura 4.9: SignInPage.js

CustomInput.js

```
const [showPassword,setShowPassword] = useState(true);

const handleShowPassword = () => {

    setShowPassword(!showPassword);

}

    return (
            <View style={styles.container}>
                <TextInput
                    autoCapitalize='none'
                    onFocus={onFocus}
                    placeholder={placeholder}
                    selectionColor={"grey"}
                    style={numericInput? s.input_num : styles.input}
                    value={value}
                       onChangeText={setValue}
                       secureTextEntry={eye ? showPassword : false}
                       error={error}
                       keyboardType={keyboardType}
                    />
                {eye ?
                <TouchableOpacity
                      activeOpacity={0.8}
                      style={styles.visibilityBtn}
                      onPress={handleShowPassword}>
                    <Ionicons
                       name={showPassword ? "eye-outline" : "eye-off-outline"}
                       style={styles.btnImage}
                       size={24}
                    />
                </TouchableOpacity>
                :
                <></>
                }
            </View>
    )
}
```

Figura 4.10: CustomInput.js

```
setIsLoading(true);
    navigation.navigate('HomePage_s',{ username:loggedUser.user.firstName,
        ip_add:ip_add,user:loggedUser.user
    } )
```

Figura 4.11: Navigazione homepage

4.6.2 Homepage

Il corpo della homepage è composto da quattro macro-sezioni:

- **Sonno**
- **Attività fisica**
- **Alimentazione**
- **Progressi**

Quando il componente Homepage viene montato, all'interno dell'hook *useEffect*, vengono fatte quattro chiamate al server:

- **getTodaySteps(currentDateApi)**: richiede al server il numero di passi giornalieri completati e aggiorna lo stato `steps_daily_done`. La card dell'attività fisica si aggiorna con il valore ogni volta che questo cambia.
- **getTodayHrValue(currentDateApi)**: richiede al server l'ultima misurazione cardiaca effettuata dall'utente manualmente dal braccialetto FitBit collegato e aggiorna lo stato `hr_daily_done`. La card dell'attività fisica si aggiorna con il valore dei battiti cardiaci (bpm), ogni qual volta viene fatta una nuova misurazione cardiaca dall'utente.
- **getTodaySleepValue(currentDateApi)**: richiede al server il numero di ore dormite nell'ultima notte passata dall'utente e aggiorna lo stato `sonno_daily_done`. La card del sonno si aggiornerà con le ore dormite, ogni volta che la chiamata al server ritornerà i dati richiesti.
- **getQuestionnairesAvailable()**: questa funzione richiede al server tramite l'API */api/questionnaires/templates* i vari template disponibili e ricevuti questi si aggiorna lo stato `allQuests`, all'interno della stessa funzione viene chiamata un'altra funzione, *getQuestionnairesCompiled*,relegata all'aggiornamento dello stato `questsTodo`, che conterrà i questionari che l'utente ancora deve compilare. La card corrispondente all'alimentazione verrà quindi aggiornata con il dato dei questionari ancora da compilare.

`questsTodo` e `questsCompilati` che rispettivamente contengono i questionari da compilari e quelli già compilati dall'utente. Ogni sezione è rappresentata da una *Card*, un componente ottenuto dalla libreria *react-native-shadow-cards*. Sia con queste Card e attraverso la barra di navigazione posta a fondo dello schermo, l'utente è in grado di navigare e muoversi attraverso queste sezioni.

Figura 4.12: Schermata Homepage

4.6.3 Alimentazione

Questa sezione ha lo scopo di tenere sotto controllo l'alimentazione del paziente così che il dottore possa monitorarla dalla piattaforma HealthApp Web. In prima battuta si è pensato di adoperare questo controllo attraverso dei questionari che il paziente è sottoposto compilare ogni settimana. Successivamente, l'alimentazione verrà gestita attraverso l'inserimento manuale della quantità di cibo assunta dal paziente per ogni pasto consumato al giorno. Questa soluzione verrà implementata nella versione successiva dell'applicazione e quindi non oggetto di questo lavoro di tesi. I questionari vengono inizializzati dai dottori attraverso la piattaforma HealthApp Web e associati ai pazienti. Quando il paziente apre l'applicazione mobile HealthApp e naviga nella sezione Alimentazione vedrà il numero di questionari che

dovrà compilare. Ogni settimana, questi si aggiornano e dovranno essere ricompilati per permettere al dottore di avere un'analisi precisa sull'alimentazione del paziente. Ad ogni questionario corrisponde un template, identificato dalla classe *QuestionnaireTemplate*, che contiene al suo interno identificatore come l'id e il nome e la lista di domande. La domanda è associata alla classe *Question*, che è arricchita dagli attributi id, testo e un vettore di possibile risposte. L'utente in questa schermata può vedere i questionari compilati e quelli ancora da compilare. Una volta che il questionario viene compilato e inviato attraverso l'API */api/patients/id/questionnaires*, l'utente viene reindirizzato alla pagina di ricapitolo dei questionari e può verificare l'ora della compilazione e controllare le risposte che ha mandato cliccando sul bottone "VISUALIZZA", senza però poter più modificare le risposte. In fase di studio del prototipo si è pensato di migliorare l'user experience dell'utente notificandolo su quante domande sono da compilare prima di sottomettere il questionario. Ciò è stato fatto in fase di sviluppo attraverso la costruzione di una barra dei progressi costituita da rettangoli che corrispondono ad ogni domanda, appena l'utente compila la domanda il rettangolo si colora di arancione, indicando che si può procedere con la prossima domanda. Un cursore a forma di cerchio indica invece la domanda a cui si sta rispondendo.

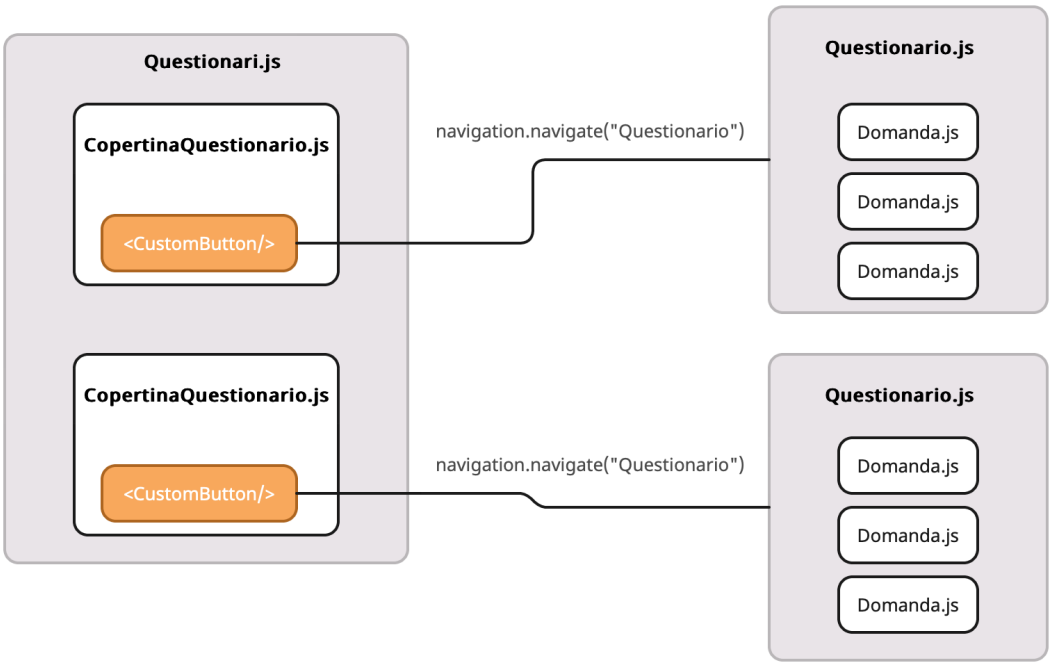

Figura 4.13: Schema componenti questionari

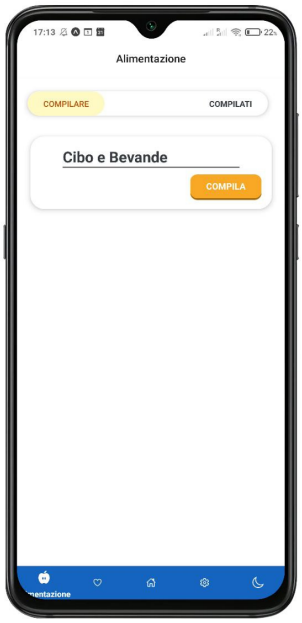

Figura 4.14: Questionari da compilare

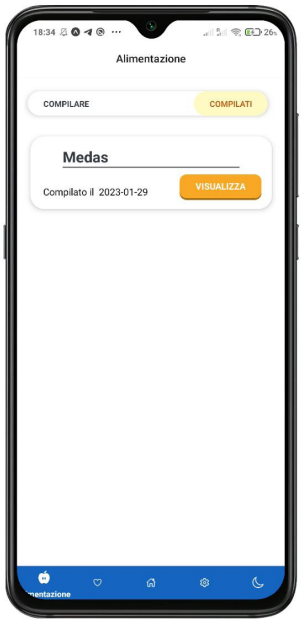

Figura 4.15: Questionari compilati

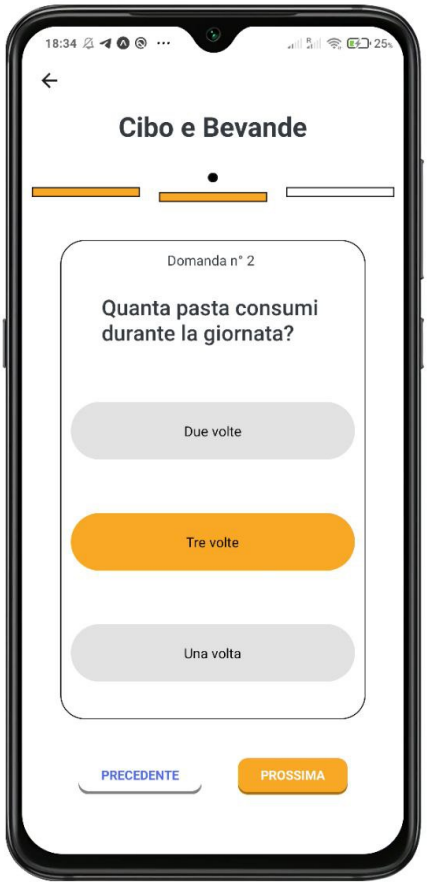

Figura 4.16: Domanda questionario

4.6.4 Attività fisica

In questa sezione è tracciata l'attività fisica dell'utente e misurata con due parametri: passi e battiti per minuto a riposo. La granularità dei dati è divisa in dati giornalieri, settimanali e mensili. L'utente può quindi, navigare tra le date attraverso le frecce o sfruttando la sezione dedicata a fondo pagina. Per mostrare i dati riguardanti i passi completati dall'utente si è deciso di utilizzare un istogramma e invece per il battito cardiaco un grafico ad area combinato con un grafico a linee, più ottimali quando vengono trattati dati che non hanno cambiamenti drastici lungo il tempo. L'istogramma per i passi mostra quanti passi completati ha effettuato l'utente in quella settimana o in quel mese.

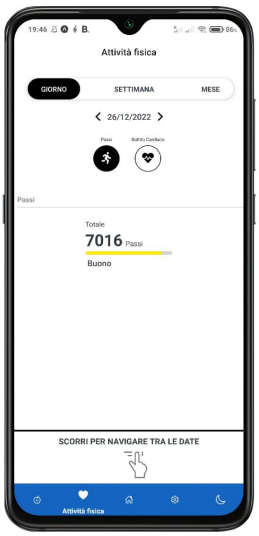

Figura 4.17: Passi - Sezione giorno

Figura 4.18: Passi - Sezione settimana

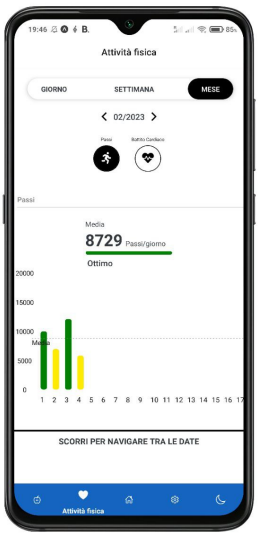

Figura 4.19: Passi - Sezione mese

Se il paziente è di tipo controllo, tutte le barre dell'istogramma avranno un colore grigio, questo per non dare nessun feedback, mentre se il paziente segue un approccio sperimentale, ogni barra viene colorata con un colore che è associato al feedback ricevuto. Il grafico indica anche con una linea tratteggiata orizzontale il valore medio di passi completati durante il periodo di tempo selezionato. L'istogramma viene popolato con i dati ogni volta che l'utente cambia periodo di tempo. Al primo rendering del componente, all'interno dell'hook *useEffect* viene utilizzata l'API: **/api/patients/id/activities/steps?startDate=$startDate&endDate=$endDate**, che fornisce un vettore di oggetti composti da due attributi: date e steps. Il vettore

contiene quindi i dati da mostrare all'interno del grafico e dello specchietto riassuntivo della media passi. Appena l'utente decide di cambiare temporizzazione, cliccando su 'Settimana' o 'Mese' viene effettuata una chiamata al server attraverso la stessa API ma con parametri startDate ed endDate diversi. Nei casi di periodo di tempo di settimana e mese, successivamente all'acquisizione dei dati corrisponde una traduzione di questi in un vettore di oggetti composti da tre attributi: value, frontColor e label, che contengono al loro interno rispettivamente il numero di passi effettuati dall'utente, il colore con cui viene colorata la barra all'interno dell'istogramma e il giorno della settimana/numero del mese. Per mostrare il grafico si è deciso di utilizzare un pacchetto npm chiamato *react-native-gifted-charts* che permette di usufruire di grafici a barre, a linea, a torta e tanti altri.[5] Un'altro indicatore presente per il paziente di tipo sperimentale è la barra dei progressi posta al di sotto del valore medio di passi per periodo di tempo. Per costruire la barra dei progressi si è utilizzato il pacchetto npm *react-native-progress* [6] che mostra all'utente il colore rispettivo all'indice di feedback e quanto manca per raggiungere l'obiettivo.

Figura 4.20: Indici feedback

Gli indici di feedback sono divisi per numero di passi completati giornalieri:

- 0-999: **Scarso**
- 1000-3999: **Discreto**
- 4000-7999: **Buono**
- 8000+: **Ottimo**

Queste soglie possono essere impostate e cambiate dai medici attraverso l'applicazione web HealthApp Web.

Per quanto riguarda la sezione dedicata alla misurazione del battito cardiaco si è deciso di mostrare i dati attraverso un grafico ad area combinato con un grafico a linee.

[5] `https://www.npmjs.com/package/react-native-gifted-charts`

[6] `https://www.npmjs.com/package/react-native-progress`

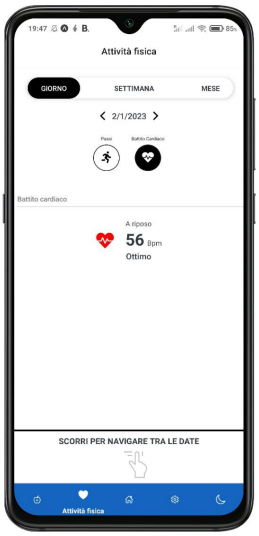

Figura 4.21: Battito cardiaco - Giorno

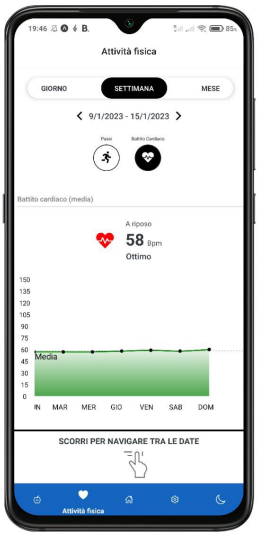

Figura 4.22: Battito cardiaco - Settimana

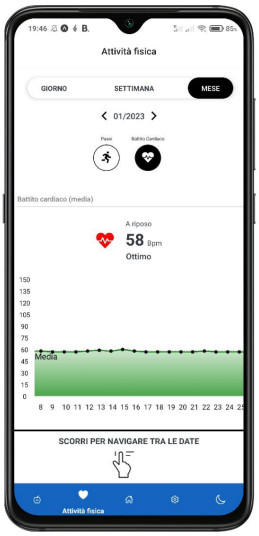

Figura 4.23: Battito cardiaco - Mese

Per lo stesso criterio seguito per il passi, il colore del feedback è rappresentato all'interno del grafico e l'acquisizione dei dati avviene attraverso lo stesso flusso, ma questa volta utilizzando l'API **/api/patients/id/hrs/rest?startDate=$startDate&endDate=$** che fornisce un vettore di oggetti composti da due attributi: date e battito cardiaco (in bpm).

4.6.5 Sonno

Nella sezione dedicata al sonno del paziente, si è scelto lo stesso design utilizzato nella sezione dell'attività fisica cambiando solamente i colori della CustomNavbar per essere coerenti con la Card del sonno presentata nella pagina iniziale.

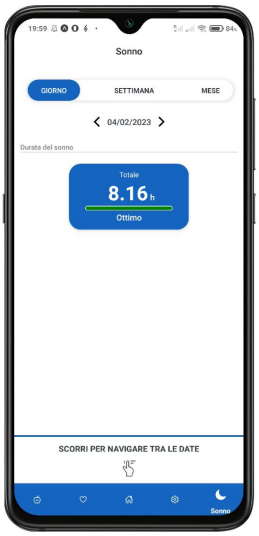

Figura 4.24: Durata sonno - Giorno

Figura 4.25: Durata sonno - Settimana

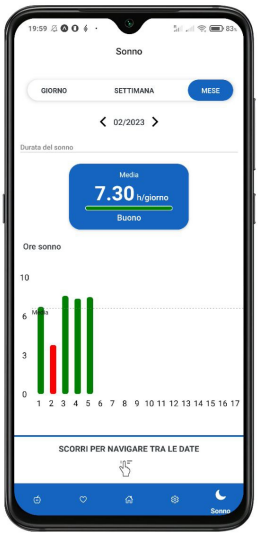

Figura 4.26: Durata sonno - Mese

4.6.6 Impostazioni

Un'altra sezione posta nella barra di navigazione è quella delle impostazioni. Questa contiene a sua volta altre sezioni quali, il profilo e la gestione delle analisi del sangue. Oltre a queste anche la possibilità di effettuare il logout dalla applicazione.

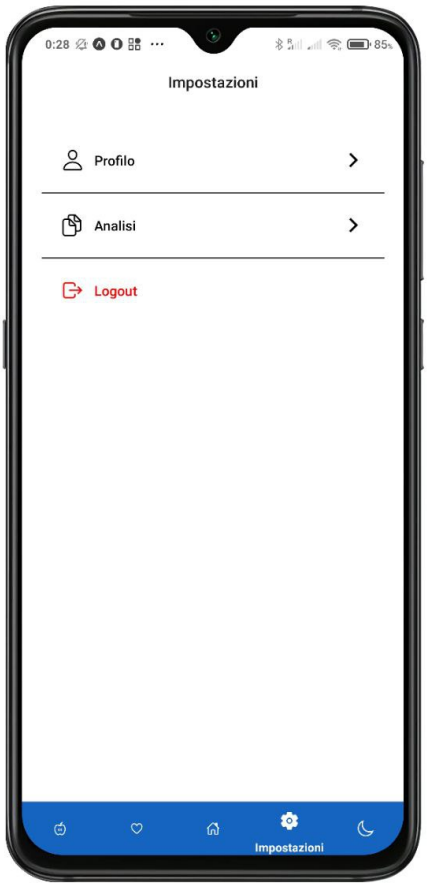

Figura 4.27: Impostazioni

4.6.7 Profilo e Peso

Attraverso questa schermata l'utente può controllare le proprie generalità immesse inizialmente dal medico attraverso l'HealthApp Web e modificare attraverso un form il proprio peso. Corredata alla schermata è presente anche un grafico ad area che mostra tutti i pesi inseriti dal paziente dall'inizio della supervisione del medico.

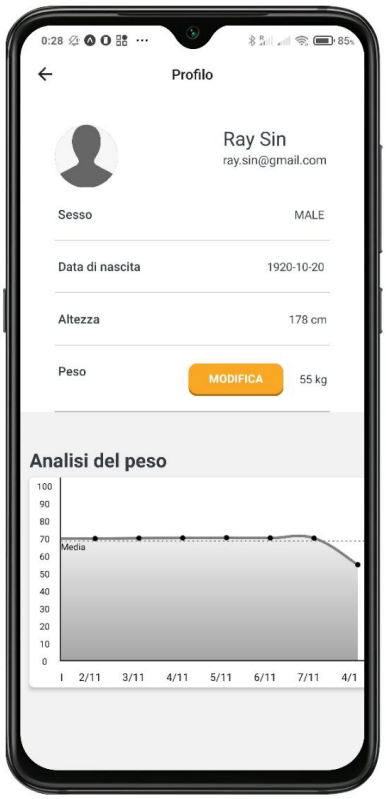

Figura 4.28: Profilo

Figura 4.29: Inserimento peso

4.6.8 Analisi del sangue

Tra le altre funzionalità presenti nella applicazione HealthApp si trova anche quella dell'inserimento di analisi del sangue eseguite dal paziente durante tutto il periodo di sperimentazione presso il centro ospedaliero. Navigando attraverso le impostazioni l'utente può accedere alla sezione analisi del sangue e da qui il paziente può o visualizzare le analisi del sangue inserite nel passato o aggiungerne una nuova.

Figura 4.30: Analisi

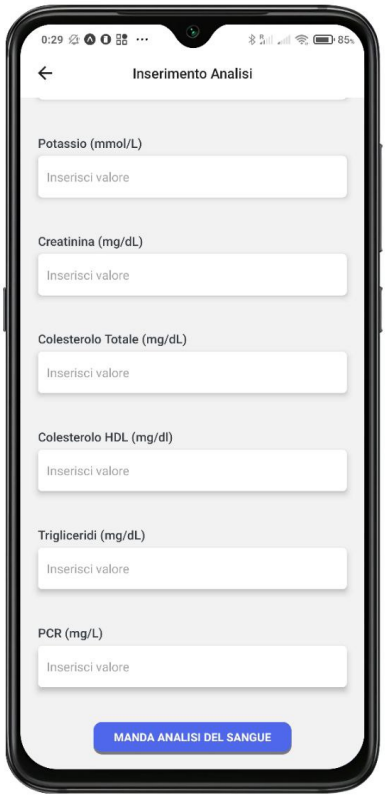

Figura 4.31: Inserimento analisi

Capitolo 5

Principi design mobile app

5.1 Principi design mobile app

Lungo tutta l'implementazione dell'applicazione si sono seguiti dei principi, che hanno come scopo quello di migliorare l'user experience dell'utente e rendere l'applicazione facilmente utilizzabile e intuitiva-

5.1.1 Schermate semplici e individuali

Uno dei problemi da affrontare quando si costruisce il design di una mobile app è lo spazio limitato dello schermo. Sebbene la dimensione di questi sia aumentata negli anni, bisogna anche evitare di popolare lo schermo di troppe informazioni, per non confondere l'utente. Per questo motivo la soluzione è quella di creare numerose schermate ma semplici. Durante la creazione di un account spesso all'utente vengono richieste molte informazioni e sebbene attraverso una interfaccia web questo potrebbe non risultare così vincolante, quando invece si parla di interfacciarsi con uno smartphone ciò potrebbe diventare frustrante e non produttivo per chi ha sviluppato l'app. L'utente potrebbe abbandonare l'applicazione appena incorre in schermate piene di informazioni da riempire. Per esempio l'applicazione YAZIO divide la fase di registrazione in schermate multiple e semplici, permettendo all'utente di conoscere il progresso di compilazione di queste, attraverso una barra che aumenta il suo valore ogni volta che l'utente compila una domanda. Questa implementazione è molto importante poiché l'utente non deve perdersi durante la navigazione delle pagine e deve essere sempre al corrente di dove si trova.
Si è seguito questo principio nella sezione alimentazione, all'interno di ogni questionario le domande sono presentate all'utente una schermata alla volta, così da non aumentare il peso di informazioni in essa. Come si può notare, si è implementata anche una barra dei progressi, che ha la funzionalità di mostrare all'utente quante domande ha compilato e quante invece da compilare. La barra dei progressi

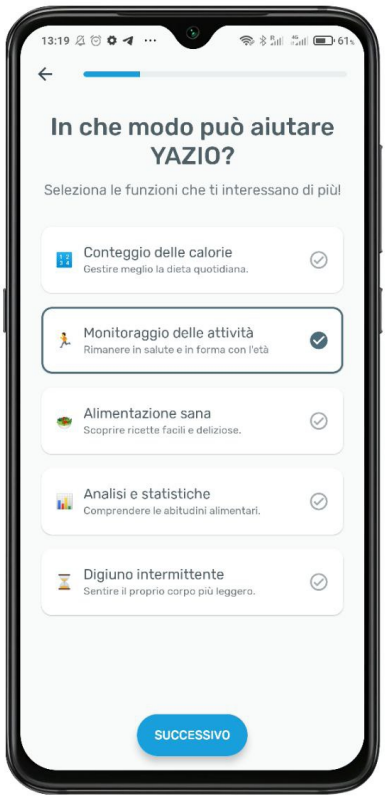

Figura 5.1: Schermate YAZIO

Figura 5.2: Schermate HealthApp

indica anche se la domanda è stata compilata o meno, colorando il rettangolo corrispondente della domanda corrente.

5.1.2 F/Z-Shape pattern

Uno degli obiettivi principali per il design di un'applicazione è riuscire a captare l'occhio e l'attenzione dell'utente. Questo è molto importante poiché la maggior parte degli utenti non andrà a leggere tutto ciò che è presente sullo schermo ma farà, inconsciamente, uno scanning molto veloce della schermata. In fase di progettazione del design quindi è importante chiedersi dove si vuole far puntare il focus dell'occhio dell'utente e in che modo controllarlo. Ciò è stato oggetto di studio fin dagli anni 80, dove attraverso uno strumento, l'eye-tracker, molte aziende di marketing riuscivano a scoprire la reale efficacia delle loro pubblicità. L'eye-tracker, è un device che misura il movimento degli occhi e di solito sono degli occhiali o un software che

attraverso l'uso di una webcam riesce a captare il movimento. Una ricerca condotta nel 2006 dal Nielsen Normal Group [1] dimostrò attraverso l'uso dell'eye-tracking, che il pattern più utilizzato dagli utenti che guardavano una pagina internet era l'F-pattern. Quest'ultimo è caratterizzato da punti fissi a sinsitra e nella parte alta della pagina. Si è scoperto infatti che dapprima, l'utente legge la parte alta della pagina con un movimento orizzontale, poi conduce lo sguardo verso metà della pagina ed effettua sempre un movimento orizzontale ma più breve del primo e per finire fa uno scanner verticale della parte sinistra della pagina.
Anche nel mondo del design per mobile app questo pattern è ricorrente quando ci si trova ad avere una pagina con grandi blocchi di testo, se invece si ha un misto tra contenuti visivi e testo si tende a preferire lo Z-pattern. Questo pattern viene utilizzato quando si vuole spingere l'utente a cliccare in certe determinate aree dello schermo. In questo caso il design in cui vengono disposte le sezioni principali deve formare la lettera Z. Partendo in alto a sinistra l'utente scorre con la vista in modo orizzontale, visualizzando i contenuti in cima allo schermo. Di seguito scende lungo una diagonale verso la parte sinistra dello schermo e infine con un movimento orizzontale raggiunge l'ultimo punto i basso a destra. Nell'implementazione di HealthApp si è deciso, per la schermata principale, di scegliere un design che seguisse questo pattern per far identificare nel minor tempo possibile all'utente le sezioni più importanti.

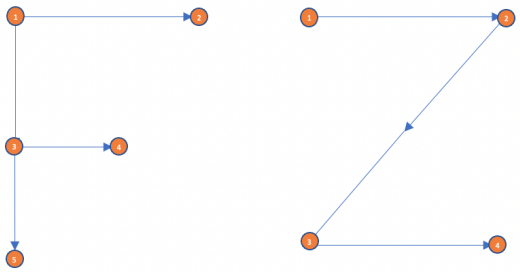

Figura 5.3: F-pattern e Z-Pattern

[1] `https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content-discovered/`

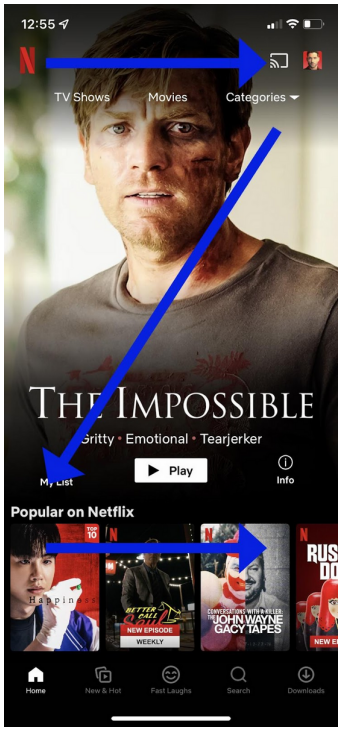

Figura 5.4: Z-pattern Netflix

Figura 5.5: Z-pattern HealthApp

5.1.3 Design per schermi grandi

Sebbene gli schermi degli smartphone continuino a diventare sempre più grandi, questo non si può dire per le mani di una persona. Più dell'85% degli smartphone venduti nel 2022 ha uno schermo superiore ai 5,5" [2], questo poiché negli ultimi anni lo smartphone è diventato uno strumento imprescindibile nella vita di quasi tutte le persone. Molti utenti lo utilizzano al posto di un PC grazie anche alla possibilità di svolgere operazioni attraverso uno schermo sempre più grande. Quando si pensa al design di una mobile app quindi bisogna riflettere anche alla facilità di raggiungere con le dita i contenuti che mostriamo sul display dello smartphone. Uno studio condotto nel 2013 [3] da Steven Hoober ha mostrato che su 1.333 persone osservate navigare con il proprio smartphone, il 49% di questi utilizzavano lo smartphone con una sola mano, il 36% poggiando una mano dietro lo schermo per tenerlo e l'altra per digitare e il 15% a due mani, con entrambi i pollici pronti per digitare.

[2] https://www.statista.com/statistics/684294/global-smartphone-shipments-by-screen-size/

[3] https://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php

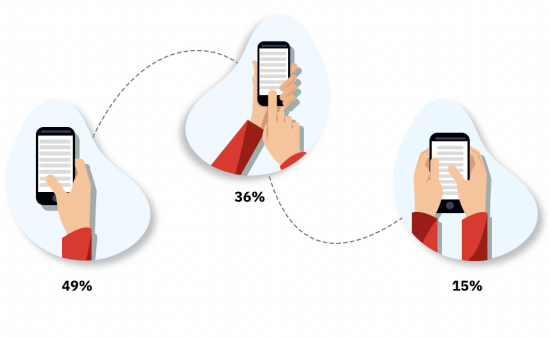

Figura 5.6: Come le persone tengono in mano uno smartphone

Di tutti questi, il 67% usavano il pollice destro per toccare lo schermo mentre il restante utilizzava il pollice sinistro. Utilizzare uno smartphone con una sola mano è certamente più comodo quando l'utilizzo di questo è rapido e non richiede molta attenzione, per esempio quando si è sui mezzi di trasporto o al supermercato, in questi casi lo smartphone viene utilizzato per brevi momenti e magari non tutte e due le mani sono libere per tenerlo. A fronte di questa comodità però sorge anche una difficoltà di navigazione dello smartphone, infatti, quando si utilizza questo con una solo mano, non tutti i contenuti mostrati sul display sono accessibili allo stesso modo. I contenuti che sono posti in alto sono difficilmente raggiungibili, mentre quelli che stanno in mezzo e non sul lato opposto del dito hanno una facilità maggiore di utilizzo.

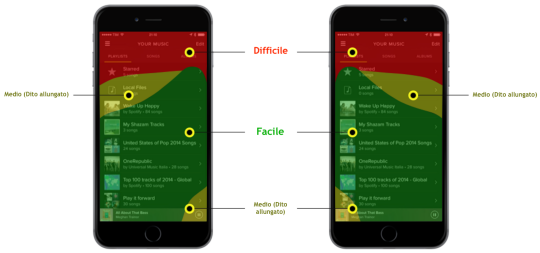

Figura 5.7: Facilità di tocco su schermo

Per questi motivi all'interno dell'applicazione HealthApp si è pensato di implementare una barra di navigazione al fondo dello schermo, per facilitare la navigatione dell'utente, sopratutto quando questo utilizza l'applicazione con una mano sola. Il principio è stato rispettato anche nelle sezioni di attività fisica e sonno, dove si è deciso di riservare uno spazio appena sopra la barra di navigazione, il cui scopo è quello di facilitare l'utente nello scorrimento delle date.

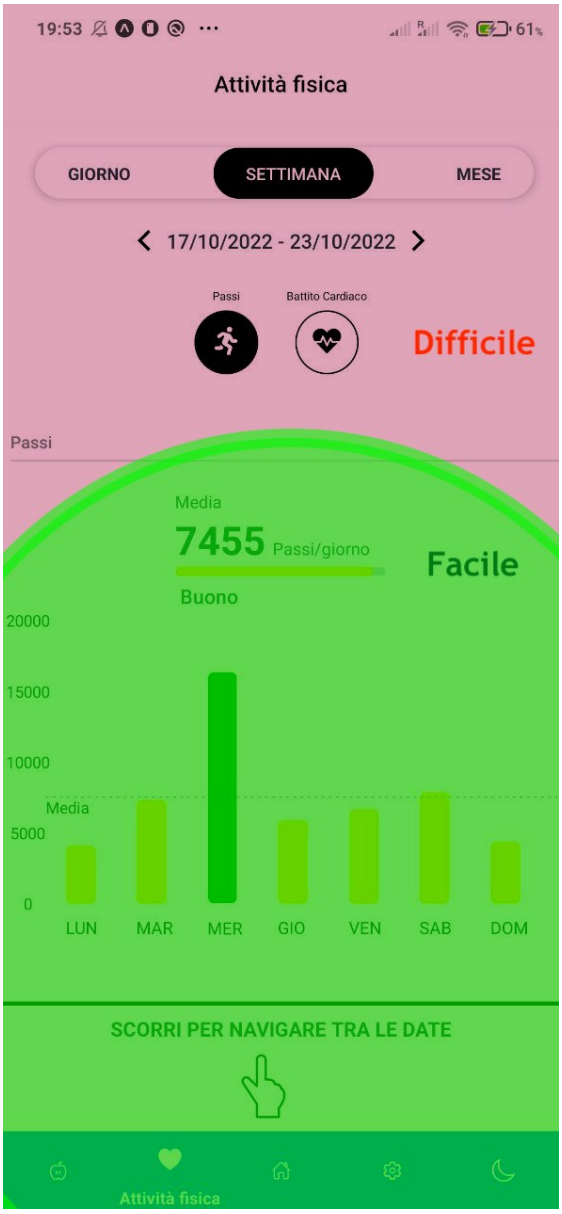

Figura 5.8: Facilità di tocco HealthApp

Capitolo 6

Conclusioni

L'obiettivo di questo lavoro di tesi è stato quello di creare una prima versione dell'applicazione mobile HealthApp e curarla dal punto di vista grafico e funzionale. Come punto di partenza si è approfondito il tema della longevità e di come poter migliorare questa, principalmente attraverso la lettura del libro "Path to longevity" di Luigi Fontana. Dopo aver appreso queste conoscenze si è potuto iniziare a pensare al miglior modo per comunicare con un fitness tracker. Inizialmente si è scelto di sviluppare una mobile app che non avesse bisogno di API create dal produttore del braccialetto, per essere più indipendenti. È seguito quindi uno studio del protocollo di comunicazione BLE per avere le basi per poter creare una connessione con il braccialetto e ricavarne i dati. Si è riusciti a stabilire una connessione tra il fitness tracker che si è scelto, il Mi Band 3, e la nostra applicazione nativa, sviluppata su Android, riusciva a catturare i dati che si volevano, quali i passi e il battito cardiaco, però sono stati trovati ostacoli, quali la velocità di sviluppo, l'obsolescenza del dispositivo e la mancanza di compatibilità con altri dispositivi, che hanno portato a cambiare approccio. Con il cambio di soluzione, si è cambiata tecnologia di sviluppo, prediligendo una soluzione cross-platform (React Native) e si è cambiato fitness tracker (FitBit). Questo cambio di approccio ha portato a un significativo aumento della velocità di sviluppo, facendo in modo di avere pronta la prima versione dell'app. A seguito di ciò si è costruito il primo prototipo dell'applicazione, prima a mano e poi creando un prototipo a bassa fedeltà. La costruzione del prototipo è stata oggetto di vari feedback che hanno portato all'aggiunta o modifiche della user interface. Una volta che il prototipo è stato confermato si è passati all'implementazione dell'app attraverso la tecnologia React Native e utilizzando il linguaggio Javascript. Ad ora l'applicazione presenta le funzionalità primarie che sono state richieste dai medici dell'Azienda Ospedaliera di Verona, quali il tracciamento dell'attività fisica e una visione generale dell'andamento settimanale dell'alimentazione dei pazienti. In futuro l'applicazione verrà migliorata e nuove funzionalità verranno aggiunte come per esempio la sezione consigli che ora nella versione 1.0 è vuota ma sarà riempita

con le raccomandazioni che verranno generate in base alle risposte dei questionari. Un'altra miglioria che è già in vista per la versione successiva e di cui è stato già fatto un prototipo grafico, è la possibilità di inserire i pasti giornalieri non più attraverso dei questionari ma attraverso un'interfaccia che permetta di selezionare la quantità e la tipologia del cibo mangiata giorno per giorno. Attraverso questo metodo la precisione delle raccomandazioni, sia automatiche, che quelle fatte dai medici saranno molto più precise rispetto a quelle prodotte in base alle risposte dei questionari. Ad ora l'applicazione è stata testata all'interno del team di sviluppo e il prossimo step sarà quello di farla testare ad utenti esterni (amici, medici) che possano fornire feedback utili per il costante miglioramento dell'app.

Bibliografia

[1] *Life Expectancy by Country and in the World*. URL: https://www.worldometers.info/demographics/life-expectancy/ (cit. a p. 1).

[2] L. Fontana. *The Path to Longevity: The Secrets to Living a Long, Happy, Healthy Life*. Hardie Grant, 2020. ISBN: 9781743795965. URL: https://books.google.fr/books?id=OuF6zQEACAAJ (cit. alle pp. 1, 3).

[3] *Chronic diseases*. URL: https://www.cdc.gov/nchs/health_policy/adult_chronic_conditions.htm (cit. a p. 1).

[4] J. Huang, L. M. Liao, S. J. Weinstein, R. Sinha, B. I. Graubard e D. Albanes. «Association Between Plant and Animal Protein Intake and Overall and Cause-Specific Mortality». In: *JAMA Intern Med* 180.9 (set. 2020), pp. 1173–1184 (cit. a p. 2).

[5] C. Thomsen, O. W. Rasmussen, C. Christiansen, F. Andreasen, P. L. Poulsen e K. Hermansen. «The glycaemic index of spaghetti and gastric emptying in non-insulin-dependent diabetic patients». In: *Eur J Clin Nutr* 48.11 (nov. 1994), pp. 776–780 (cit. a p. 2).

[6] S. Yusuf et al. «Effect of potentially modifiable risk factors associated with myocardial infarction in 52 countries (the INTERHEART study): case-control study». In: *Lancet* 364.9438 (2004), pp. 937–952 (cit. a p. 4).

[7] Ahmed Merghani, Aneil Malhotra e Sanjay Sharma. «The U-shaped relationship between exercise and cardiac morbidity». In: *Trends in Cardiovascular Medicine* 26.3 (2016), pp. 232–240. ISSN: 1050-1738. DOI: https://doi.org/10.1016/j.tcm.2015.06.005. URL: https://www.sciencedirect.com/science/article/pii/S1050173815001711 (cit. a p. 5).

[8] S. Sabia, A. Fayosse, J. Dumurgier, V. T. van Hees, C. Paquet, A. Sommerlad, M. ki, A. Dugravot e A. Singh-Manoux. «Association of sleep duration in middle and old age with incidence of dementia». In: *Nat Commun* 12.1 (apr. 2021), p. 2289 (cit. a p. 6).

[9] R. N. Aurora, J. S. Kim, C. Crainiceanu, D. O'Hearn e N. M. Punjabi. «Habitual Sleep Duration and All-Cause Mortality in a General Community Sample». In: *Sleep* 39.11 (nov. 2016), pp. 1903–1909 (cit. a p. 6).

[10] D. R. Mazzotti, C. Guindalini, W. A. Moraes, M. L. Andersen, M. S. Cendoroglo, L. R. Ramos e S. Tufik. «Human longevity is associated with regular sleep patterns, maintenance of slow wave sleep, and favorable lipid profile». In: *Front Aging Neurosci* 6 (2014), p. 134 (cit. a p. 6).

[11] *Digital Health Trends 2021*. URL: `https://www.iqvia.com/insights/the-iqvia-institute/reports/digital-health-trends-2021` (cit. a p. 7).

[12] F. Piccinini, G. Martinelli e A. Carbonaro. *Accuracy of Mobile Applications versus Wearable Devices in Long-Term Step Measurements*. Nov. 2020 (cit. a p. 8).

[13] *Ble GATT data organization*. URL: `https://microchipdeveloper.com/wireless:ble-gatt-data-organization` (cit. a p. 20).

[14] JetBrains. *The State of Developer Ecosystem 2021*. URL: `https://www.jetbrains.com/lp/devecosystem-2021/miscellaneous/` (cit. a p. 28).

Ringraziamenti

Con questo lavoro di tesi si chiude un capitolo molto importante della mia vita. Nonostante gli ostacoli, le difficoltà e i dubbi sono arrivato alla fine. Il viaggio non è stato uno dei più semplici ma mi ha insegnato tanto, sia dal punto di vista tecnico che da quello umano. Porterò con me i successi ma soprattutto i fallimenti, dai quali ho imparato molto di più. Fino all'ultimo c'è stato bisogno di sudare per arrivare all'obiettivo e in merito a questo vorrei ringraziare il mio relatore di tesi Maurizio Morisio per avermi accompagnato al traguardo finale. Un grazie va al prof.Cortese, che sin dal primo anno mi ha accolto mostrandomi la sua umanità e la sua disponibilità, conserverò le sue parole che mi sono state di aiuto nei momenti più difficili. Ringrazio inoltre la prof.ssa Graziano per avermi, attraverso le sue lezioni, regalato una bussola, magari tra qualche tempo la utilizzerò. Grazie a chi mi è stato vicino, chi ha sempre avuto una parola di conforto o di incoraggiamento per me, amici miei, grazie per le chiacchierate infinite in macchina sotto casa, per le promesse, le freccette, il polo del '900, la siga della buonanotte, i lavori trovati alle 23 e lo svago che fa parte dell'equilibrio delle cose. Un grazie poi non basterebbe per la persona che sta al mio fianco e che è stata la mia compagna di viaggio sin dall'inizio di questo percorso, da tutti i posti che mi tenevi in aula studio fino al supporto che costantemente mi hai dato e mi dai tutti i giorni. Vorrei ringraziare mio zio Agostino per avermi seguito durante tutto il percorso, sarai sempre una fonte di ispirazione per me.
Infine a voi, mamma, papà e sorellina, dedico tutto questo, nulla sarebbe stato possibile senza i vostri sforzi e il vostro infinito amore.