Alma Mater Studiorum · Universitá di Bologna

scuola di ingegneria e architettura
campus di cesena
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

ARCHITETTURE PER SMART HEALTH: IL CASO FITSTADIUM

Elaborata nel corso di: Sistemi Distribuiti

Relatore:
Prof.Ing.
ANDREA OMICINI

Presentata da:
ROBERTO D'ELIA

Correlatore:
Ing.
STEFANO MARIANI

Sessione II
Anno Accademico 2014-2015

Desidero ringraziare il Prof. Andrea Omicini, l'Ing. Stefano Mariani e tutti i ragazzi di FitStadium per l'aiuto, la disponibilitá e la simpatia dimostrata. Ringrazio, inoltre, tutti coloro che mi hanno sostenuto in questo percorso.

Indice

Elenco delle figure

Introduzione

L'information communication technology (ICT) ha influenzato profondamente il modo di vivere e il modo di rapportarsi delle persone. Le caratteristiche dei servizi e come vengono offerte le varie funzionalità sono profondamente mutate nel corso degli anni; tutto ciò grazie alla ricerca e ai tentativi di soddisfare nel migliore modo possibile le esigenze che via via andavano naturalmente nascendo. Lo sviluppo di nuove tecnologie e il loro radicalizzarsi nelle più disparate discipline ha aperto inevitabilmente nuovi orizzonti che un tempo, forse, era addirittura difficile da immaginare. Non solo, quindi, la richiesta di nuove funzionalità così come la necessità di colmare le nuove esigenze influenza il progresso dell'ICT, ma è vero anche l'incontrario, ossia i due aspetti si influenzano a vicenda in un processo senza fine.
Su questa linea i servizi user centric sono tra le tipologie che maggiormente stanno sperimentando un grande sviluppo e una estrema diffusione. I servizi user centric sono servizi orientati principalmente all'utente finale i cui bisogni, desideri e caratteristiche influenzano le funzionalità sin dalla fase di progettazione.
Non a caso, infatti, nell'ultimo periodo il mercato è stato invaso da un grande numero di device portatili, tutti con peculiarità e dimensioni diverse. Escludendo le loro risorse limitate se confrontate con i moderni *personal computer* (ma oramai neanche più di tanto), la caratteristica fondamentale di questi dispositivi è proprio quella di essere mobili e costantemente vicino all'utente. Questo porta gli sviluppatori alla realizzazione di sistemi distribuiti pervasivi estremamente personalizzabili, costituiti da componenti che, per l'appunto,

sono inseriti al'interno di environment soggetti a continui cambiamenti. La mancanza di un controllo amministrativo diretto e la loro sporadica connessione alla rete, inoltre, rende questi sistemi spesso instabili.
L'ultima ondata fresca nell'ambito dei dispositivi mobili è rappresentata dai *wearable devices*, ossia vestiti o accessori che incorporano computer e tecnologia elettronica avanzata, consentendo di avere a disposizione una capacità computazionale. I dispositivi wearable sono anche un buono esempio dell'*Internet of Things* in quanto sono composti da oggetti fisici che si inseriscono all'interno dei sistemi pervasivi poco prima citati.
Per sfruttare al massimo il contesto nel quale vengono inseriti, questi dispositivi sono dotati di sensori che hanno il compito di recuperare i dati relativi all'ambiente o all'utente che lo sta utilizzando. Esempi di questi dispositivi sono gli smart watch e gli activity trackers. Proprio quest'ultimi sono stati progettati per la rilevazione di dati legati all'attività fisica ed, in generale, per il fitness.
I servizi per il fitness e per la salute hanno caratteristiche tipiche dei servizi user-centric di cui si è parlato poco prima; sono nati forse da una forte necessità di mantenere tacciate le prestazioni degli utenti e la loro effettiva realizzazione è stata resa possibile grazie ai wearable devices e alle architetture pervasive.
Incrociando il mondo delle applicazioni mobili relative al fitness/salute con quello delle *smart city* si aprono infinite strade che portano alla definizione di un nuovo concetto, indicato dal termine *smart health*. Le smart city prelevano, infatti, una gran quantità di dati provenienti da fonti diverse come ad esempio i sensori ambientali. Questi dati possono essere resi disponibili secondo la logica dei *Open Data* e quindi utilizzabili dai servizi che da essi hanno una forte dipendenza, proprio come quelli del fitness e salute.
D'altro lato le architetture software devono essere in grado di gestire e abilitare questo tipo di scenari: tramite l'architettura, infatti, si esprimono le proprietà cardini del sistema, le quali derivano dai vincoli dovuti all'applicazione dei fodamentali principi ingegneristici.
L'obiettivo della tesi è progettare un'architettura abilitante per scenari smart

health, concentrandosi sulla parte end-user (non sulla parte server-cloud), ossia sperimentando l'ambito dei wearable devices e facendo riferimento al binomio fitness-Apple Watch, comodo perchè presente nell'azienda FitStadium che ci fornisce motivazioni, requisiti e goals.
Nel primo capitolo si analizzeranno le soluzioni offerte attualmente dal mercato per la realizzazione di servizi legati al fitness, focalizzandosi in particolare sulle architetture proposte e come quest'ultime possano convivere con l'ecosistema FitStadium.
Il secondo capitolo è riservato invece all'approfondimento delle tecnologie Apple, che verranno utilizzate concretamente per la realizzazione del caso di studio. Ancora una volta si farà attenzione alle possibilità architetturali offerte da queste tecnologie.
Nel terzo capitolo viene trattato nella sua interezza il caso di studio, analizzandone in particolare lo stato pre e post tesi. Verrà cioè descritta l'applicazione implementata insieme alla presentazione di un'architettura abilitante anche per i scenari smart health. Infine, all'interno del capito 4 viene descritto più precisamente il concetto di smart health e il percorso che ha condotto alla sua definizione.

Capitolo 1

Analisi delle piattaforme

Apple e Google sono due tra i principali competitor nel mondo delle applicazioni e dei dispositivi mobile. Entrambi i colossi tecnologici offrono sui relativi store (*Google play* e *Apple store*) un'ampia gamma di applicazioni tra loro eterogenee.
Allo scopo di favorire la creazione e quindi l'aumento del numero di app scaricabili, le due organizzazioni mettono a disposizione degli strumenti dedicati agli sviluppatori. Tra questi sono anche presenti piattaforme con l'obiettivo di abilitare i servizi orientati alla salute ed, in particolare, al fitness. Queste piattaforme (*Google fit* per Google e *Health* per Apple) sono caratterizzate da specifiche qualità e vincoli.
In questo capitolo verranno quindi analizzate le architetture delle due piattaforme con particolare attenzione alle proprietà che da esse derivano.

1.1 Piattaforma Apple

Apple propone un'architettura centralizzata sul dispositivo iPhone. Gli altri dispositivi (fisici) utilizzati sono l'Apple Watch e i vari activity tracker che ospitano i sensori.

Tutti i dati che riguardano le attività di fitness e la salute dell'utente, possono

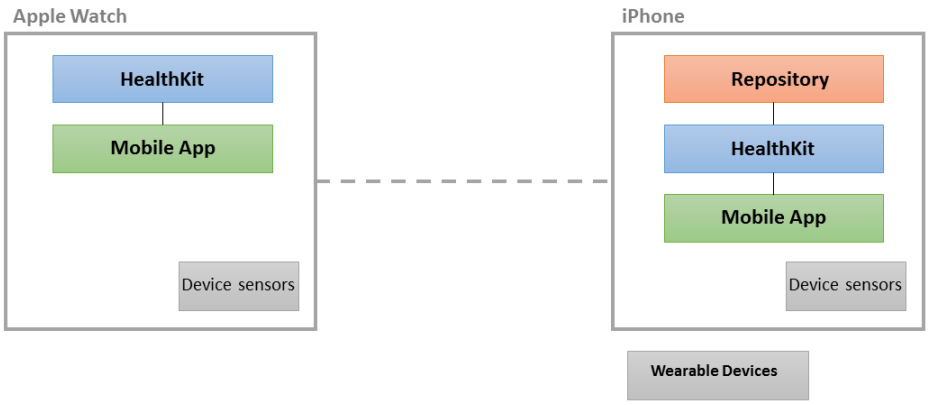

Figura 1.1: Piattaforma Apple

essere salvati su un componente che funge da repository, che si trova localmente sull'iPhone. Il repository è gestito dal framework HealthKit che, quindi, rappresenta il riferimento comune a tutte le applicazioni che necessitano di questo tipo di dati. Esse potranno, infatti, andare a recuperare qualsiasi dato memorizzato al suo interno, previa autorizzazione dell'utente. I dati memorizzati, quindi, possono provenire da diverse fonti e possono essere letti anche da chi non ne è il creatore originale.

HealthKit fornisce delle precise astrazioni per rappresentare i dati interessati. Il framework verrà trattato più nel dettaglio in seguito, all'interno del capitolo 2.

Particolarmente significativa è la relazione che sussiste tra l'Apple Watch e l'iPhone. L'Apple Watch è in grado di recuperare un insieme di dati attraverso i propri sensori e salvarli sul repository HealthKit. I dati, grazie al framework, vengono trasmessi in modo trasparente allo sviluppatore tramite la tecnologia Bluetooth. Per recuperare i dati precedentemente catturati attraverso i sensori dell'Apple Watch, quindi, è sufficiente eseguire da codice una specifica query sul repository.
Healthkit gestisce anche i sensori dell'Apple Watch a seconda del tipo di allenamento da eseguire (vedi anche API per la creazione di un workout session): in particolare sceglierà quali sensori attivare al fine di disporre dei dati relativi. Con la nuova versione del sistema operativo WatchOS, Apple permette di accedere ai dati memorizzati in HealthKit sia dall'iPhone sia dall'AppleWatch, sempre in seguito all'autorizzazione dell'utente.
Per gestire la comunicazione tra iPhone e Apple Watch bisogna fare riferimento al Watch connectivity framework descritto successivamente nell'apposito paragrafo.

Per quanto riguarda i dispositivi di terze parti come i vari smart band, fitness band o fitness tracker, essi non sono programmabili ma mettono a disposizione delle applicazioni, solitamente sia per il sistema operativo di Apple sia per il sistema operativo Android. In questo caso, quindi, è responsabilità di queste applicazioni salvare correttamente i dati sul repository HealthKit, rendendoli quindi fruibili a tutte le altre app.

L'architettura messa in piedi da Apple, perciò, consente di recuperare i dati provenienti da diversi sensori ed immagazzinarli in un database su iPhone. In seguito questi dati sono messi a disposizione per una computazione locale da parte delle applicazioni iOS interessate, oppure possono essere trasferiti in un database remoto grazie alla capacità di connessione dell'iPhone o dell'Apple Watch. L'utilizzo di un repository locale sul quale salvare dati precedentemen-

te trasmessi via Bluetooth è particolarmente vantaggioso rispetto al consumo della rete Internet che si avrebbe nel caso in cui tutti i dati dei sensori venissero trasferiti direttamente ad un repository remoto.
Un altro particolare vantaggio è sicuramente legato alla sicurezza: con questa architettura i dati rimangono memorizzati in locale e vengono perciò meno esposti a certi tipi di attacchi come l'intercettazione sulla rete Internet o attacchi su architetture cloud. Per gli altri dettagli sulla sicurezza e privacy garantita da Apple, fare riferimento alla sezione su HealthKit di questa tesi.

Il repository locale su iPhone fornisce, quindi, le seguenti proprietà:

- *Distribution transparency*: rende la distribuzioni fisica dei sensori irrilevante.
- *Access transparency*: Fornisce un accesso uniforme ed omogeneo ai dati.
- *Concurrency transparency*: Gestione dell'accesso concorrente, in maniera trasparente.

Di base su iPhone è pre-installata l'applicazione *Health* la quale consente di accedere facilmente ai dati salvati nel repository. Tramiti essa, inoltre, possono essere memorizzati i dati personali dell'utente considerati stabili, come il sesso e la data di nascita.

In riferimento al caso di studio (che verra presentato nel capitolo 3) la figura "Flussi dati sensibili Apple" mostra i possibili flussi di dati sensibili (quelli relativi alla salute e al fitness) tra i vari nodi dell'architettura proposta da Apple.

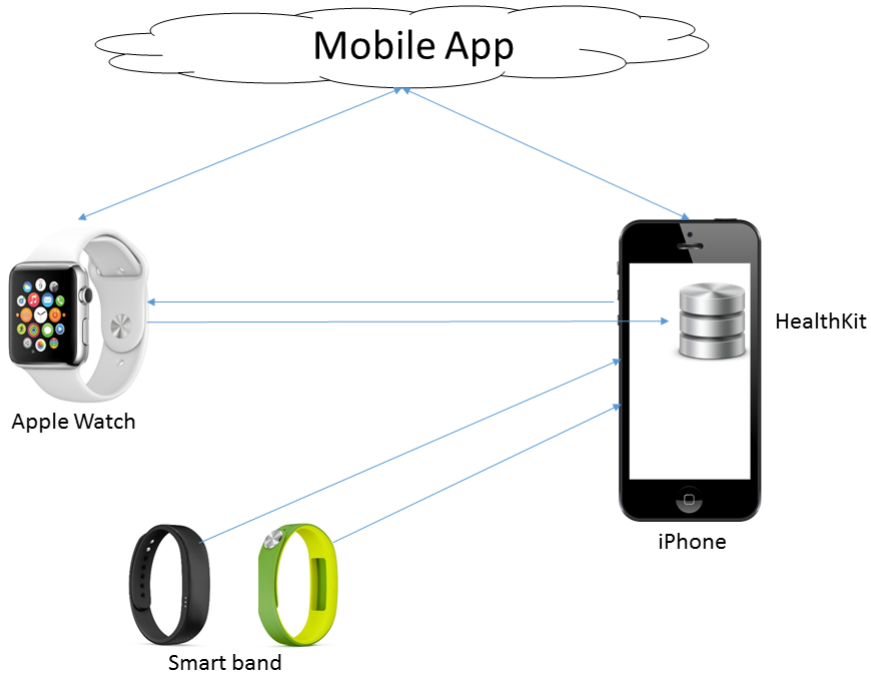

Figura 1.2: Flussi dati sensibili Apple

1.2 Piattaforma Google

Google propone una piattaforma (*Google Fit*) diversa da quella di Apple, schematizzabile nella figura "Piattaforma Google".

Google Fit permette agli sviluppatori di memorizzare le informazioni che

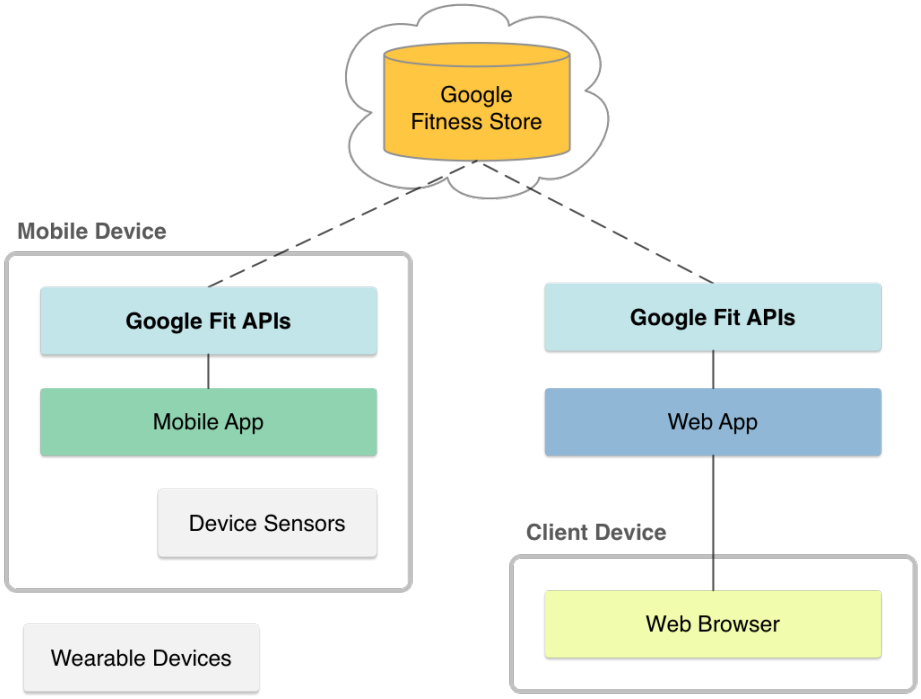

Figura 1.3: Piattaforma Google

riguardano il fitness all'interno del fitness store remoto, al quale gli utenti possono accedere attraverso diversi device e applicazioni.
Similmente alla piattaforma Apple quindi:

- Le applicazioni fitness possono memorizzare dati provenienti da diversi sensori e dispositivi *wearable*.
- Le applicazioni fitness possono accedere ai dati memorizzati da qualsiasi app su autorizzazione dell'utente.

Le prime differenze consistono, invece, nel fatto che Google Fit è pensato per raccogliere solo i dati riguardanti il fitness, escludendo ad esempio quelli relativi alla salute come avveniva nel caso Apple.
Il repository, inoltre, non è locale su smartphone Android ma in remoto, ospitato dall'infrastruttura Google.

Con *Wearable devices* Google indica, oltre ai diversi smart band e fitness tracker, anche i smart watch con il sistema operativo *Android Wear*.
Come già detto in precedenza, spesso i fitness tracker mettono a disposizione applicazioni sia per i device Apple sia per i device Android, che solitamente salvano i dati quindi o su HealthKit o su Google Fit.
Per quanto riguarda Android Wear, similmente all'Apple Watch è possibile creare applicazioni di terze parti con le quali andare a recuperare i dati dai sensori inclusi. Android Wear permette un'accesso diretto ai dati del sensore, senza l'obbligo, quindi, di passare dalla piattaforma Google Fit: memorizzare qui i dati, però, è consigliabile per renderli disponibili a tutte le applicazioni e ai diversi dispositivi.
Gli smart watch con il sistema operativo di Google, inoltre, attraverso delle apposite API comunicano con lo smartphone Android tramite tecnologia Bluetooth.

Le *Google Fit API* permettono l'accesso al fitness store in maniera trasparente. Esse si dividono in API per Android e in *REST* API: quest'ultime permettono l'utilizzo dello store anche da Web app. E' possibile, quindi, creare applicazioni che supportano Google Fit su multiple piattaforme e device, come Android, iOS, e Web App.

Un repository remoto accessibile dal Web, quindi, fornisce le seguenti proprietà:

- *Distribution transparency*: rende la distribuzioni fisica dei sensori irrilevante.
- *Access transparency*: fornisce un accesso uniforme ed omogeneo ai dati.

- *Concurrency transparency*: Gestione dell'accesso concorrente, in maniera trasparente.
- *Making resources available*: Rende disponibili le risorse ai componenti eterogenei distribuiti.

Oltre alle prime tre proprietà presenti anche nella piattaforma di Apple, Google Fit garantisce una sincronizzazione dei dati tra dispositivi eterogenei.
In Apple, l'Apple Watch era in grado di accedere al repository solo grazie al framework HealthKit. In Google, i vari smart watch possono usufruire dei dati dei loro sensori e quelli in Google Fit rimanendo disaccoppiati dal device Android.

Per quanto riguarda la sicurezza, trasferendo i dati su cloud lontani dall'utente attraverso la rete Internet, Google li espone maggiormente a possibili attacchi. Importante precisare che i dati dovrebbero essere relativi alla sola attività fisica, escludendo quelli sulla salute o medicinali che sono considerati più sensibili.
I dati vengono trasferiti attraverso una connessione criptata, ma non viene specificato come vengono memorizzati i dati sul cloud.
Per accedere a Google Fit, inoltre, bisogna utilizzare il proprio account Google: perciò particolare attenzione deve essere rivolta alla sicurezza del proprio account attivando, ad esempio, la verifica a due passaggi.
Per quanto riguarda la privacy, siccome i dati si trovano sull'infrastruttura Google, l'azienda potrà utilizzarli per diversi scopi come ad esempio:

- Miglioramento dei servizi offerti.
- Risultati di ricerca più rilevanti.
- Combinazione dei dati con altri servizi Google.
- Condivisione in forma anonima con partner pubblicitari .
- Condivisione per ragioni legali con agenzie governative e di intelligence.

Per l'uso dei dati da parte di applicazione di terze parti, Google non ne è responsabile, perciò attenzione alle autorizzazioni concesse a queste applicazioni.

Per ciò che riguarda i dati provenienti da eventuali sensori per l'ambiente, come rilevatori di temperatura o inquinamento, nè HealthKit nè Google offrono un supporto.

La seguente immagine mostra i possibili flussi di dati sensibili nella piattaforma Google Fit.

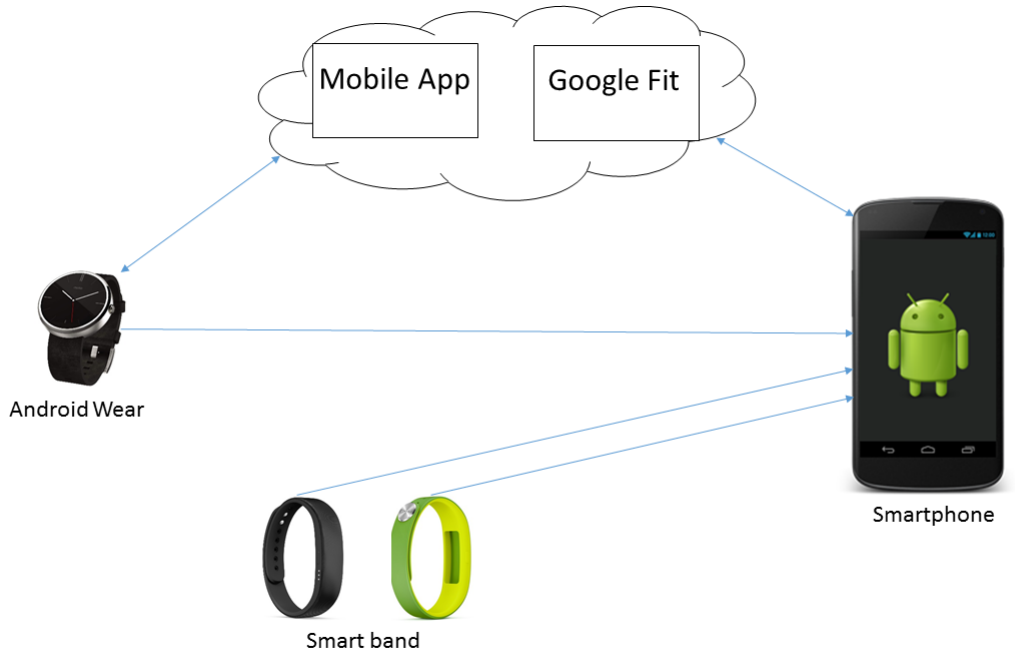

Figura 1.4: Flussi dati sensibili Google

Dispositivi

Il mercato propone al giorno d'oggi una grande varietà di smart watch con sistema operativo Android Wear e tantissimi fitness tracker di diversi produttori.

Quasi tutti gli smart watch hanno sensori quali accelerometro, sensore di luce ambientale, giroscopio e connettività bluetooth. I più sofisticati possono includere anche il sensore del battito cardiaco, GPS, bussola e connettività Wi-Fi. Gli smart band sono solitamente dotati di meno sensori, tra i quali l'accelerometro è sempre presente così come la connettività bluetooth.

Capitolo 2

Tecnologie Apple

In questo capitolo verrano introdotte le tecnologie che Apple mette a disposizione e che verranno utilizzate per il caso di studio di questa tesi. Si farà in particolar modo riferimento agli strumenti utili alla realizzazione di applicazioni per l'Apple Watch, compreso il framework HealthKit per la creazione di servizi legati al fitness. Verranno descritte inoltre le architetture delle applicazioni in riferimento alle diverse versioni del sistema operativo WatchOS e analizzati i concetti base dello sviluppo per il device di Apple.

2.1 Apple Watch

Figura 2.1: Apple Watch

L'*Apple Watch* è l'ultimo dispositivo mobile di casa *Apple*, presentato il 9 settembre 2014. Il dispositivo permette di accedere a funzionalità smart integrate a quelle di un orologio digitale da polso. La maggior parte delle funzionalità richiedono anche la presenza di un *iPhone*.

2.1.1 Funzionalità Principali

Le funzionalità principali sono le seguenti:

- Notifiche e accesso delle funzionalità delle applicazioni iPhone.
- Feedback utente tramite sistema di vibrazione *TapticEngine* o autoparlante integrato.
- Risposta alle chiamate ricevute tramite il microfono integrato.
- Ricezione e risposta ai messaggi di testo attraverso diverse modalità.
- Riproduzione file audio e video utilizzando cuffie *Bluetooth*.
- Assistente vocale *SIRI*.
- Rilevazione dati tramite sensori incorporati

- Registrazione e calcolo dati per il fitness.

Le applicazioni su Apple Watch sono intrinsecamente legate alla loro versione su iPhone, almeno per quanto riguarda quelle progettate per il sistema operativo *WatchOS1*. Attraverso delle interazioni brevi ed immediate, è possibile ricevere notifiche ed utilizzare alcune (solitamente un sottoinsieme) delle funzionalità messe a disposizione dalla loro controparte su iPhone. Con il sistema operativo WatchOS2, invece, molte applicazioni sono native, cioè vengono eseguite interamente su Apple Watch e necessitano della presenza dell'iPhone solo durante la fase di installazione o nel caso in cui prevedano una comunicazione con l'applicazione su iOS.

Il sistema di feedback permette di utilizzare diversi tipi di vibrazioni, ad esempio a seconda del tipo di notifica ricevuta, mentre l'autoparlante integrato emette brevi segnali che possono essere disattivati anche attraverso particolari movimenti del polso.

Il microfono integrato permette di rispondere alle chiamate, dettare brevi messaggi o interagire con l'assistente vocale SIRI. Le funzionalità di SIRI sono simili a quelle su iPhone.

Per la riproduzione di file audio, invece, è necessario utilizzare cuffie bluetooth.

Grazie ai sensori integrati e la raccolta di dati provenienti da più sorgenti, l'Apple Watch è in grado di monitorare l'attivita fisica dell'utente. I dati rilevati potranno in seguito essere utilizzate per altre applicazioni e per fornire informazioni di sintesi.

2.1.2 Connettività

L'Apple Watch può connettersi tramite tecnologia *Bluetooth*. A differenza di alcuni modelli meno recenti di iPhone è inoltre dotato di tecnologia *NFC*

tramite la quale è possibile effettuare pagamenti contactless con *Apple Pay*. Il dispositivo è anche in grado di accedere ad hotspot *WI-FI*; nonostante ciò, solo a partire dalla versione 2 del sistema operativo WatchOS saranno disponibili, per le applicazioni di terze parti, nuove API per eseguire richieste ad URLs (protocolli ftp, http e https compresi) e sfruttare quindi questa tecnologia.
L'Apple Watch, quindi, è in grado di connettersi ad un iPhone vicino attraverso la tecnologia Bluetooth, oppure se entrambi hanno accesso alla stessa rete WI-FI.

2.1.3 Sensoristica

- Accelerometro.
- Giroscopio.
- Cardiofrequenzimetro.
- Sensore luce ambientale.

2.1.4 Cardiofrequenzimetro

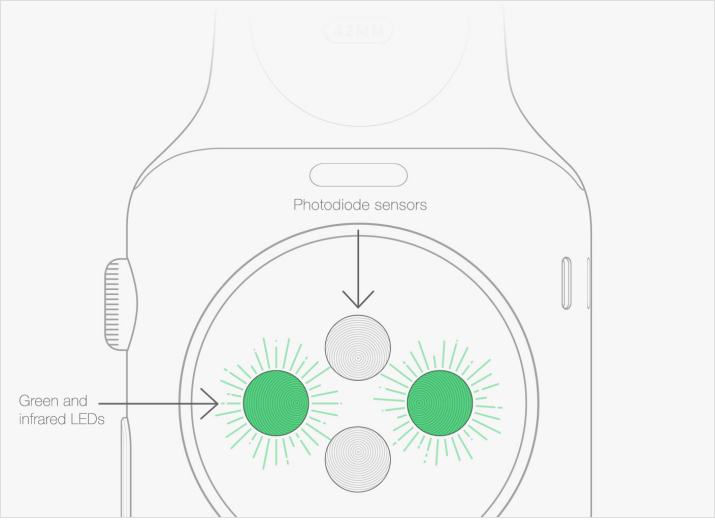

Figura 2.2: Cardiofrequenzimetro

L'Apple Watch per rilevare il battito cardiaco utilizza la *photoplentysmography*. Questa tecnologia sfrutta la proprietà del sangue di riflettere la luce rossa e di assorbire la luce verde. Quando il cuore batte, aumenta sia il flusso di sangue che scorre nelle vene sia, di conseguenza, la luce verde che viene assorbita. Attraverso due *LED* di luce verde accoppiati a fotodiodi sensibili alla luce viene calcolata la quantità di sangue che scorre nelle vene in un dato momento. Facendo quindi lampeggiare numerose volte i LED viene calcolato il numero di volte che il cuore batte in un minuto. In caso di basso segnale, le luminosità dei LED e la frequenza di campionamento viene aumentata di conseguenza.
In alcuni soggetti e a causa di diversi fattori, l'Apple Watch non è in grado di misurare la frequenza cardiaca in maniera affidabile.
Movimenti ritmici, come la corsa o il ciclismo, danno risultati migliori rispetto a movimenti irregolari come quelli nel tennis o nella boxe.
Modifiche permanenti o temporanee della pelle, come alcuni tatuaggi, possono rendere la luce del sensore inefficace.

2.2 WatchKit

WatchKit è il *Software Development Kit* (SDK) che permette di sviluppare le applicazioni per WatchOS. WatchOS è il sistema operativo dell'Apple Watch che con la versione 2 (autunno 2015) introduce interessanti novità architetturali.
Le applicazioni per Apple Watch sono costituite da due parti: la WatchKit Extension e la Watch App stessa. La Watch App contiene la *storyboards* e le risorse necessarie per visualizzare l'interfaccia grafica, mentre la WatchKit Extension gestisce i contenuti, risponde alle interazioni dell'utente e aggiorna l'interfaccia.

Come si può notare dall'immagine, nella prima versione di WatchOS le due

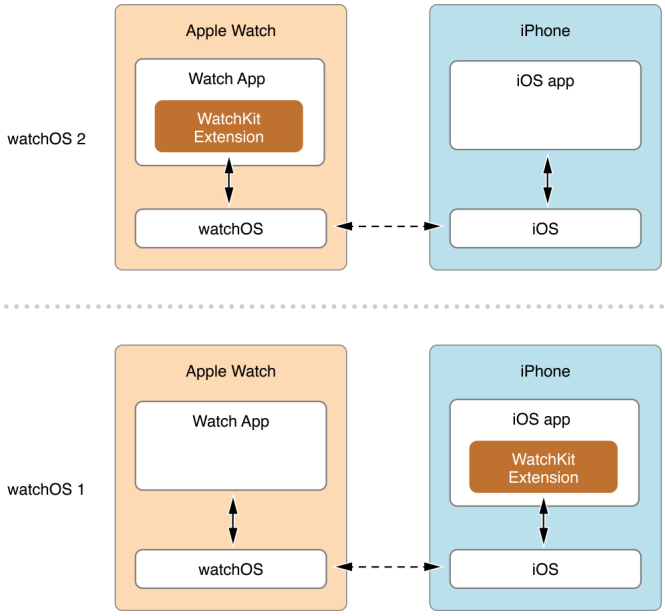

Figura 2.3: Architettura WatchOS1 vs. WatchOS2

parti erano collocate in ambienti di esecuzione differenti: Watch App sull'Apple Watch e WatchKit Extension su iPhone. In questa architettura, ad ogni interazione dell'utente i comandi e i dati devono essere inviati in modalità wireless da Apple Watch a iPhone e viceversa. In WatchOS2 la WatchKit Extension è stata, invece, integrata direttamente all'interno della Watch App.

Eliminando la latenza del trasferimento wireless si beneficia di una maggiore velocità di esecuzione e reattività, ma ciò implica anche una diversa gestione delle risorse richieste dall'applicazione, le quali possono essere memorizzate sull'Apple Watch, sull'iPhone, su entrambi, o reperibili attraverso la rete Internet. Con questa novità architetturale, quindi, le applicazioni per Apple Watch hanno la possibilità di essere più consistenti ed autonome, senza la necessità di fare affidamento sulla presenza di un iPhone. Per sviluppare un nuovo progetto per l'Apple Watch (all'interno dell' IDE *XCode*) è comunque necessario partire da un progetto iOS (per iPhone) esistente o crearne uno nuovo.

In WatchOS2, la WatchKit Extension può comunicare con la app iOS su iPhone con il *Watch Connectivity Framework*, il quale verrà descritto in maniera più dettagliata in seguito in questa tesi. Il framework fornisce una comunicazione bidirezionale per scambiare dati tra i due processi. Supporta, inoltre, lo scambio di dati anche in background.

Per lo sviluppo di applicazioni iOS, OSX, e WatchOS, Apple ha creato il linguaggio di programmazione *Swift* sulle basi del *C* e di *Objective-C*. Swift è un linguaggio moderno che affianca alle caratteristiche tipiche del paradigma ad oggetti anche quelle del paradigma funzionale. In particolare sono presenti costrutti come le guardie, chiusure, tipi che indicano valori opzionali e numerose funzionalità avanzate.
Per la guida completa del linguaggio si consiglia di consultare il manuale gratuito scaricabile da iBooks Store o la guida ufficiale online.

2.2.1 User Interface

Il punto di partenza per implementare la Watch App è definire l'insieme delle scene che faranno parte della storyboard. La scena definisce una porzione dell'interfaccia utente e può essere personalizzata per le diverse dimensioni dell'Apple Watch.
Xcode fornisce un editor grafico tramite il quale vengono definite e posizionate

le varie scene dell'interfaccia, compresi i componenti in essa contenuti. I vari componenti possono essere allineati e modificati personalizzando gli attributi dall'Inspector.
I Group Object sono particolari oggetti non visibili che permettono di definire diversi layout tra loro innestati, all'interno dell'interfaccia.
Una buona pratica è quella di evitare, quando possibile, di vincolare i vari componenti a dimensioni prefissate, ma permettere un ridimensionamento a seconda dello spazio disponibile al fine di creare un'interfaccia compatibile con le vari dimensioni dell'Apple Watch. L'Editor permette comunque di specificare dimensioni per ogni tipo di device, nel caso fossero necessari degli aggiustamenti.
L'Editor grafico è un buon strumento per definire un'interfaccia a tempo di progettazione. Nel caso siano richiesti cambiamenti a tempo di esecuzione, ad esempio per far evolvere l'interfaccia dinamicamente con l'aggiunta o la modifica di componenti, bisogna agire a livello di codice ed, in particolare, sull'oggetto istanza della classe *WCInterfaceController*. L'oggetto interface controller è situato sulla WatchKit Extension ed è relativo ad un unica interfaccia tra quelle inserite nella storyboard. Durante la navigazione dell'utente all'interno dell'applicazione, il WatchKit si impegna a caricare l'interface controller attualmente selezionato.
Ogni componente che fa parte di una interfaccia è collegato ad un *Interface Object*. Per motivi di latenza, non è possibile ottenere e manipolare il valore corrente di una view attraverso l'Interface Object: ciò causerebbe, infatti, una doppia comunicazione wireless tra l'orologio e il dispositivo portatile. Ciò che è possibile fare, invece, è assegnargli un determinato valore.
All'interno dell'interface controller è possibile fare riferimento agli interface object in esso contenuti, creando un *Outlet* ossia un collegamento con il relativo oggetto creato dall'editor grafico.

2.2.2 Application Life Cycle

Quando l'applicazione viene avviata, viene messa in esecuzione sia la Watch App sia la WatchKit Extension ed apparirà la scena designata come quella principale. Le due parti si scambieranno i dati finchè l'utente smetterà di interagire con l'applicazione: a questo punto anche la WatchKit Extension verrà sospesa in attesa della prossima interazione.
In WatchOS2 vengono instanziati automaticamente l'oggetto *extension object* ed *extension delegate*. L'extension object è un oggetto condiviso dal quale si può ricavare l'extension delegate. Quest'ultimo gestisce dei comportamenti che sono centrali per l'applicazione e che un tempo era compito dell'interface controller della main interface. L'extension delegate gestisce il ciclo di vita degli eventi dell'applicazione: quando l'applicazione viene avviata vengono emessi degli eventi, catturati dall'extension delegate, che indicano la transizione tra differenti stati. La figura "Applicazione life cycle" mostra i diversi metodi che vengono richiamati per ogni transizione.

Quando l'applicazione viene lanciata, si passa dallo stato *Not running* fino

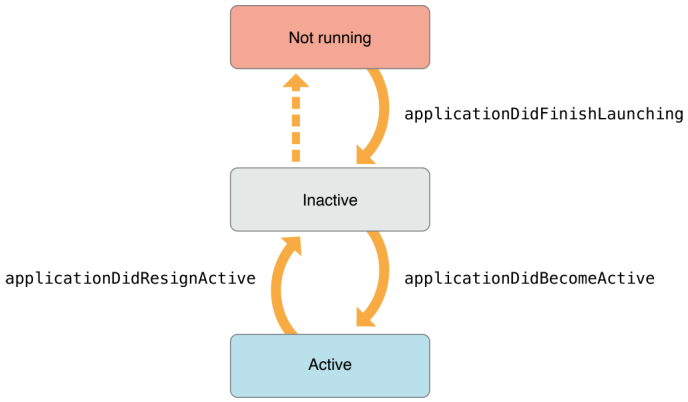

Figura 2.4: Applicazione life cycle

allo stato *Active*. Se l'utente esce dall'applicazione avviene la transizione allo stato *Inactive* e, qualche istante dopo, allo stato Not running.
L'extension delegate può essere utilizzato anche per implementare azioni in risposta alle notifiche ricevute sull'Apple Watch.

In particolare, all'interno del metodo *applicationDidFinishLaunching* possono essere attuate le procedure di configurazione dell'interfaccia da visualizzare, mentre nel metodo *applicationDidBecomeActive* possono essere avviati i vari task messi in pausa quando l'applicazione passò allo stato Inactive.

Un altro life cycle riguarda le transizioni delle scene che costituiscono la storyboard dell'interfaccia, gestite questa volta all'interno dell'interface controller (figura "Interfaccia life cycle").
Quando la Watch App carica una scena, il framework WatchKit fa una richiesta alla WatchKit Extension di caricare l'oggetto interface controller corrispondente.
Utilizzare i metodi *init* e *awakeWithContext* per caricare i dati richiesti, im-

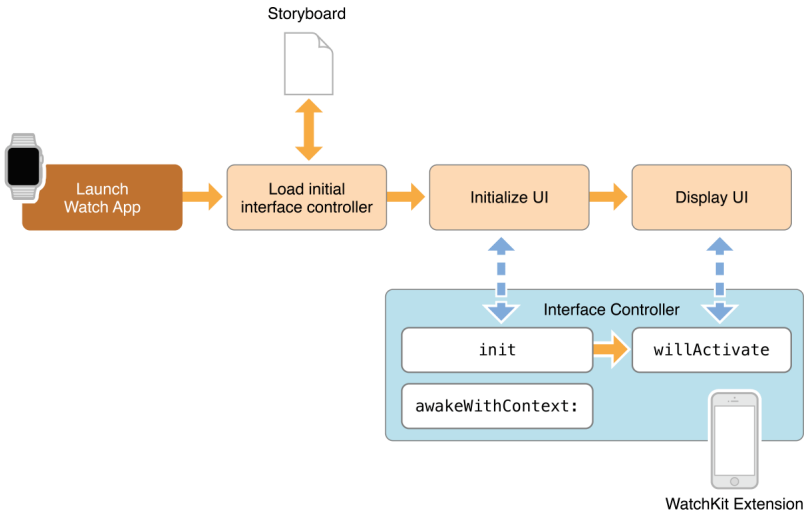

Figura 2.5: Interfaccia life cycle

postare i valori dei componenti, preparare l'interfaccia ad essere visualizzata. Il metodo *willActivate* viene richiamato in avanzamento, appena prima che l'applicazione venga visualizzata: non deve essere quindi utilizzato per impostare l'interfaccia ma per eseguire gli ultimissimi cambiamenti.
Ad esempio nello stile *Page based*, ossia un particolare modo di navigare le scene di una app, l'interfaccia è costituita da una serie di scene delle quali solo una

per volta viene visualizzata. L'utente, a runtime, è in grado di navigare tra le scene eseguendo degli swipe a destra e a sinistra dello schermo. In questo caso gli oggetti interface controller vengono caricati inizialmente tutti, e di mano in mano richiamato il metodo willActivate della scena richiesta.
In WatchOS2 viene richiamato il metodo *didAppear* una volta che l'interfaccia è visibile sullo schermo. Similmente, WatchKit chiama i metodi *willDisappear* e *didDeactivate* nel momento in cui l'interfaccia si disattiva.

L'emissione degli eventi dell'interface controller è intramezzata da quelli emessi dall'extension delegate. Quando viene avviata l'applicazione, quindi, non c'è alcuna garanzia sul quale, tra i metodi will activate dell'interface controller e applicationDidBecomeActive dell'extension delegate, venga richiamato per primo.

Mentre l'utente interagisce con i vari componenti visualizzati (button, label...), WatchKit richiama i relativi *action methods* nell'interface controller per fornire la risposte. Gli action methods devono essere inseriti nell'interface controller tramite i riferimenti (Outlets) dallo strumento grafico.
Le interazioni dell'utente cn l'Apple Watch sono pensate per essere brevi ed immediate: per questo gli interface objects devono essere leggeri e mai eseguire tasks lunghi.

Prima di sospendere la Watchkit Extension, viene richiamato il metodo *did-Deactivate*.

Le applicazioni progettate per WatchOS1 non vengono mandate in esecuzione del tutto finchè l'utente non sblocca l'iPhone accoppiato. In WatchOS2, se correttamente progettate, le applicazioni sono in grado di essere lanciate anche senza lo sblocco del relativo iPhone, ma esse non potranno comunicare con l'applicazione "compagna" iOS se essa si trova nello stato not running (ciò succede se l'applicazione non è stata mai mandata in esecuzione o se, pe qualche

motivo, è stata terminata dal sistema operativo).

2.3 Watch Connectivity Framework

Per gestire la comunicazione tra Apple Watch e iPhone è stato creato il Watch connectivity framework. Creando il relativo oggetto sessione su entrambi i device, è possibile scambiare messaggi in ambe due le direzioni.

Vengono quindi messe a disposizione delle primitive con le quali è possibile inviare sia messaggi immediati, ossia messaggi che richiedono che il processo destinatario sia in foreground e i due device abbastanza vicini, sia messaggi che vengono memorizzati in un buffer in attesa del risveglio del destinatario (il trasferimento avverrà anche se l'applicazione corrente viene sospesa o terminata).
In particolare quando un messaggio deve essere spedito alla sua controparte, viene inserito all'interno di una coda e consegnato nell'ordine con cui sono stati inviati. Similmente i messaggi ricevuti sono recuperati dalla coda di ricezione e consegnati ai delegate nell'ordine in cui sono stati ricevuti. Tutti i messaggi sono consegnati ai delegate serialmente su un thread secondario.

Alla ricezione di un messaggio, invece, verranno richiamati i relativi metodi del delegate aggiunto alla sessione, ossia l'oggetto che implementa il protocollo WCSessionDelegate (simile all'interfaccia in Java).

2.4 HealthKit

Il framework HealthKit fornisce servizi di salute e fitness attraverso i dati memorizzati all'interno di una struttura condivisa locale. La struttura rappresenta il centro di immagazzinamento per ciò che riguarda la salute; con essa le applicazioni saranno in grado di leggere e scrivere i dati di cui necessitano. Quest'ultimi possono provenire, ad esempio, da diversi sensori o dispositivi *wearable*.
Tutti i dati saranno quindi disponibili, previa autorizzazione dell'utente, alle applicazioni interessate compresa la *Health App*: applicazione inclusa all'interno dell'iPhone che mostra informazioni e dati statistici sommari.
L'intera piattaforma offerta da Apple, e in particolare la sua architettura, è stata descritta più nel dettaglio all'interno del capitolo 1.

A causa della particolare sensibilità dei dati, la sicurezza e la privacy viene particolarmente curata da parte di Apple. Innanzitutto il database è locale su iPhone. I dati, inoltre, sono criptati e non accessibili alla lettura se l'iPhone risulta essere bloccato. Ciò significa che le applicazioni non possono accedere ai dati quando vengono eseguite in background. Possono comunque continuare a scrivere: i dati verranno prima salvati in una cache e poi, appena viene sbloccato il telefono, sulla struttura criptata.
HealthKit garantisce, inoltre, un controllo fine sulle informazioni che possono essere condivise. In particolare l'utente può acconsentire o negare il permesso di lettura in base al tipo di dato: ad esempio permettere la lettura del numero dei passi eseguiti e negare il livello di glucosio nel sangue.

Il database HealthKit è considerato thread-safe, perciò può essere in generale utilizzato in ambienti multithread.

2.4.1 Health data

Per rendere i dati da memorizzare il più possibile interoperabili e significativi, il framework vincola il tipo dei dati e le unità ad una lista predefinita. Come risultato i dati sono compatibili e usufruibili da molte applicazioni ma, di contro, gli sviluppatori non possono creare dei nuovi tipi.

Tutti gli oggetti nel database derivano dalla classe *HKObject*. Ognuno di essi possiede almeno le seguenti proprietà:

- **UUID**: identificatore unico per la entry
- **Source**: La sorgente del dato. Può essere una app o un device che salva i dati direttamente nel database.
- **Metadata**: Dizionario contenente informazioni aggiuntive. Le chiavi utilizzate possono essere sia custom sia predefinite.

Gli oggetti memorizzati si dividono in due gruppi: *characteristic* e *samples*. Gli oggetti characteristic rappresentano informazioni che tipicamente non cambiano come la data di nascita e il sesso. Questo tipo di informazioni vengono inserite solamente dall'utente attraverso la Health App. Gli oggetti samples rappresentano dati catturati in un certo momento e sono caratterizzati dalle seguenti proprietà:

- **Type**: ad esempio step count o height sample.
- **Start date**: data del campione.
- **End date**: Se il dato rappresenta un valore calcolato in un intervallo di tempo, indica la data di fine. Altrimenti coincide con lo Start Time.

I samples sono ulteriormente partizionati in quattro tipi:

- **Category samples**: samples classificabili all'interno di un insieme finito di categorie. Al momento ne fa parte solo la categoria *sleep analysis*.

- **Quantity samples**: memorizzano un valore numerico. Sono i più comuni tipo di dato in HealthKit (height, weight, pulse rate...).
- **Correlations**: rappresenta una composizione di dati samples. Al momento viene usato per *food* e *energy burned*.
- **Workouts**: rappresenta un particolare tipo di attivita fisica come *running* o *swimming*. Ad un workout possono essere collegati più samples per fornire informazioni riguardo ad un specifico intervallo del workout. A differenza del tipo correlations, i samples non vengono memorizzati all'interno del workout ma possono essere recuperati tramite una query.

WatchOS2 porta importanti novità anche ad HealthKit: nella prima versione, in particolare, non c'era modo di ricavare i dati provenienti direttamente dai sensori dell'Apple Watch, come ad esempio il cardiofrequenzimetro o l'accelerometro, con il fine di avere informazioni real time.
Le nuove *API* consentono di creare ed eseguire dei workout session specificando il tipo di attività e il tipo di luogo dove viene eseguita (all'aperto o al chiuso). A seconda del workout, l'Apple Watch attiverà gli ingressi secondo lui più appropriati e inizierà a monitorare l'utente salvando i dati sul repository. Se, ad esempio, l'attività è una corsa all'aperto, l'Apple Watch tenterà di ricevere informazioni anche dal GPS del'iPhone, oppure attiverà l'accelerometro se il workout è eseguito al chiuso.
Eseguendo delle particolari query sul workout si possono recuperare i dati real time se disponibili, o venire notificati in caso di modifica degli stessi.
Durante il workout l'applicazione deve essere necessariamente in foreground. Inoltre, è possibile eseguire solo un workout alla volta.

Capitolo 3

Caso di Studio

PEr eseguire correttamente un particolare esercizio fisico, entrano in gioco numerose variabili come ad esempio una corretta impostazione iniziale, il giusto posizionamento del corpo durante l'esecuzione dell'esercizio, una respirazione eseguita correttamente in tutte le fasi e così via. Ciò è particolarmente vero quando si cerca di eseguire esercizi che lasciano molta libertà dei movimenti, come quando si utilizza il peso dei manubri o si cerca di sfruttare quello del proprio corpo. Chiaramente la soluzione migliore consisterebbe nella presenza di un personal trainer, che segue l'utente durante tutte le fasi dell'allenamento e che propone i giusti esercizi per raggiungere lo scopo finale.
Quando tutto ciò però non è possibile, è necessario trovare altri modi per aiutare e sostenere l'utente durante la sua attività.

Il capitolo tratta il caso di studio attinente alla tesi. Viene innanzitutto introdotta l'azienda *FitStadium*, presso la quale è stata svolta l'attività di tirocinio: viene quindi descritta l'azienda e i servizi che essa intende offrire. In particolare si pone l'attenzione sull'architettura software sulla quale si basa il core business della società.
In seguito si presenterà il contesto e le motivazioni dell'attività svolta e, quindi, la realizzazione dell'applicazione.
Infine verrà mostrata un'architettura pensata per le applicazioni legati al fitness

e per la smart health.

3.1 FitStadium

Figura 3.1: Logo Fitstadium

FitStadium è un'azienda nata in Romagna, che si occupa di servizi legati al fitness.
Muove i primi passi nel 2012 con la costruzione di un social network dedicato agli amanti dell'esercizio fisico, per poi concentrarsi, negli anni successivi, sullo sviluppo di un'applicazione per dispositivi mobili.
L'idea iniziale era, infatti, quella di creare una comunity social dedicata agli amanti del fitness, dove pubblicare e condividere i propri workout giornalieri. Nasce così una rete virtuale dove tutti gli appassionati potevano incontrarsi e confrontarsi.
Il social network va online nel settembre del 2012 in versione beta.
Per un certo periodo di tempo la società sosta in *Cesena Lab*, un'incubatore di giovani *startup* inserito nel territorio cesenate, per poi diventare sempre più indipendente.
Nel 2013 FitStadium si aggiudica il bando Spinner per le idee imprenditoriali

innovative e, in particolare, arriva il finanziamento di *Technogym* che decide di interessarsi del progetto e di scommettere sul suo potenziale.
Il core business dell'azienda si sposta da un social network ad un'applicazione per smartphone con l'obiettivo di offrire uno strumento smart per l'allenamento e per raggiungere uno stato di forma ideale.

3.1.1 L'applicazione

L'applicazione FitStadium è disponibile sia per la piattaforma *Android* di *Google* sia per il sistema operativo *iOS* di Apple.

L'obiettivo dell'applicazione è quello di offrire un modo semplice ed efficace per mantenersi in forma. Per fare questo, svolge il ruolo di un personal trainer tascabile, fornendo schede di esercizi personalizzati ad hock per l'utente.
Gli esercizi sono illustrati tramite pratici video e vengono inoltre visualizzate

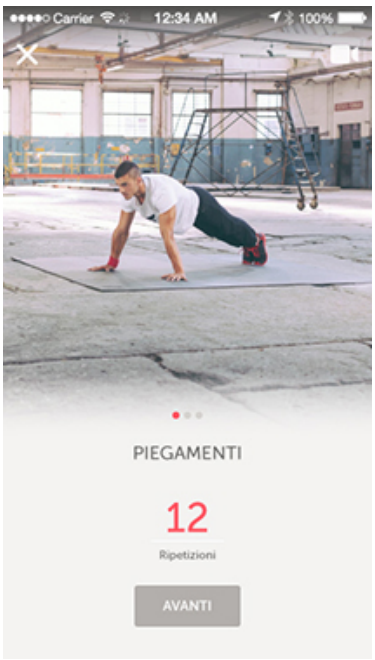

Figura 3.2: Applicazione FitStadium

le ripetizioni e i tempi di recupero.

Ogni attività è pensata per essere eseguita a corpo libero e, quindi, nel modo più immediato possibile: ci si può allenare ovunque, senza la richiesta di attrezzi.
Prima di tutto bisogna impostare un obiettivo, come la perdita di peso o la tonificazione della muscolatura. All'inizio di ogni workout, inoltre, viene richiesto di selezionare la parte del corpo su cui focalizzare gli esercizi, il proprio peso e il tempo a disposizione.

La caratteristica più importante è sicuramente l'adattabilità dell'allenamento proposto, in funzione delle caratteristiche di chi andrà ad eseguire l'esercizio. L'allenamento, infatti, è progettato per essere su misura dell'utente oppure è in grado di cambiare dinamicamente a fronte dei feedback alla fine di ogni esercizio.

3.1.2 Architettura

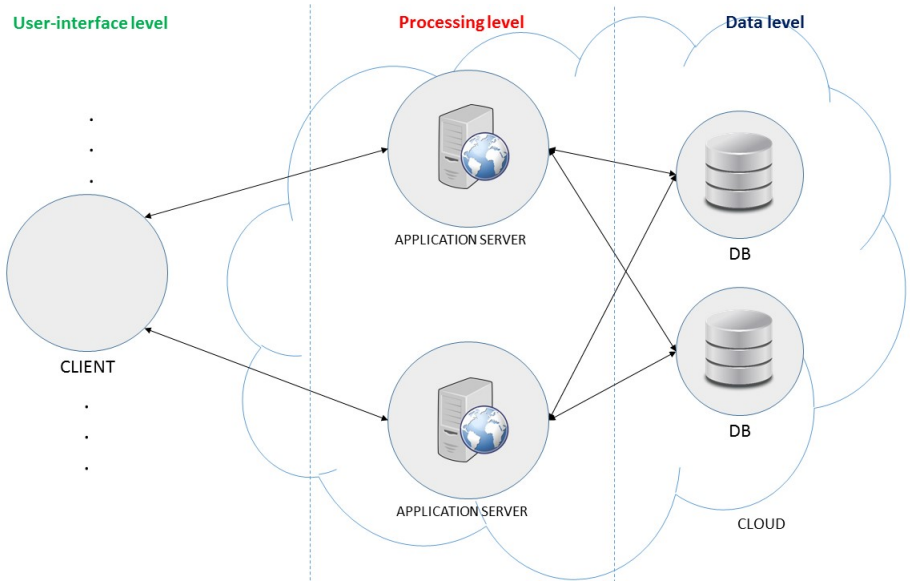

Figura 3.3: Architettura FitStadium

Il sistema distribuito FitStadium presenta un'architettura di tipo centralizzata, secondo il modello *Client - Server*.
Nelle architetture di questo tipo il client richiede specifici servizi che vengono implementati e offerti dal server, il quale rimane sempre in attesa delle richieste.
Il componente client, in particolare, modella l'applicazione presente sui dispositivi portabili: per questo motivo è un componente mobile che si connette al sistema in maniera sporadica.
La parte server si divide, invece, in due componenti logiche: l'*application server* e il *database server*. L'application server modella i servizi offerti ai client mentre il database server è il luogo in cui vengono memorizzati i dati.
Come tipicamente avviene in questo tipo di architetture, si possono individuare almeno tre layer logici nei quali si organizzano i vari componenti:

- **User-interface layer**: contiene l'interfaccia con la quale l'utente può interagire.
- **Processing level**: contiene la logica e le funzionalità principali dell'applicazione.
- **Data level**: contiene i dati e i file necessari.

Ovviamente non sempre l'organizzazione logica rispecchia quella fisica. Nel caso di FitStadium, però, il server è effettivamente separato in una macchina fisica per il database ed in una macchina fisica per l'application server.
L'application server è stato, inoltre, replicato in un secondo application server per motivi di bilanciamento del carico. In questo modo, nel caso pervenissero troppe richieste in un determinato istante di tempo, un *load balancer* provvederà a distribuirle ai due application server in modo tale da evitare qualsiasi sovraccarico che potrebbe altrimenti causare l'arresto del servizio.
L'infrastruttura lato server è gestita tramite cloud: grazie questo servizio i server possono essere facilmente gestiti, aumentando o diminuendo sia la loro capacità sia il numero di unità, in maniera molto flessibile e *on demand*.

Dalle caratteristiche appena esposte, ed in particolare a causa della presenza dei dispositivi mobili utilizzabili dagli utenti, possiamo classificare l'architettura di FitStadium come un sistema distribuito pervasivo.
Una proprietà fondamentale dei sistemi pervasivi è quella di essere a stretto contatto con l'ambiente circostante. I dispositivi mobili sono, infatti spesso dotati di sensori o attuatori, tramite i quali acquisire o modificare i valori dei parametri che caratterizzano il loro environment. Questi dati, in particolare, possono influenzare o stabile il comportamento stesso del servizio: per questi motivi, i sistemi pervasivi sono difficilmente predicibili o instabili.

3.1.3 Tecnologie utilizzate

FitStadium, quindi, è un'applicazione *Web Service-oriented*: implementa dei servizi accessibili dai dispositivi mobile, attraverso il middleware al giorno d'oggi più utilizzato: il *World Wide Web*.
L'approccio scelto è quello *RESTful*, cioè l'interazione tra i vari componenti avviene tramite l'invio di messaggi *Http*, con il payload in formato *JSON*.

L'applicazione front-end è stata realizzata tramite la piattaforma *Apache Cordova*. Apache Cordova consente la costruzione di applicazioni mobile multipiattaforma, utilizzando le tecnologie tipiche del WEB. Invece di utilizzare il linguaggio nativamente supportato dallo specifico device, quindi, viene utilizzato *Html*, *CSS*, *Javascript*, *AJAX*, *jQuery mobile* (in questo caso viene mantenuto tutto sul device e non su un server remoto come nel caso del WWW).
Javascript, quindi, viene esteso con API che permettono l'accesso alle funzioni del dispositivo come la fotocamera o l'accelerometro.
Tutto ciò ha il vantaggio di risparmiare tempo (e denaro) nello sviluppo e rendere immediatamente disponibile la propria applicazione ad un pubblico più ampio ed eterogeneo.
I possibili svantaggi in termini di prestazioni sono: una minore efficenza nel rendering grafico e nell'accedere alle risorse locali. La grafica dell'interfaccia, inoltre, potrebbe discostarsi molto dagli standard tipici della piattaforma.

I servizi sono realizzati con script *PHP* su server *Apache*. In particolare, viene utilizzato il framework *elgg* per l'interazione con il database.
Il framework elgg era particolarmente utile per lo sviluppo del social network FitStadium, al momento non è stato scartato ma viene ancora utilizzato.

3.1.4 Interazione e flusso dati

Per ciò che riguarda l'interazione tra il client e il server è importante distinguere la prima volta che viene eseguito l'allenamento da quelle successive. Quando l'utente si iscrive a FitStadium, infatti, la sua profilazione risulta essere incompleta: è necessario, quindi, recuperare le informazioni durante la prima seduta di allenamento.
In questo caso, infatti, viene richiesto di inserire sesso, data di nascita, altezza, obiettivo (forza o dimagrimento) e livello di condizione (bassa, media o alta). Questi dati vengono inviati al service il quale si occuperà di salvarli a sua volta all'interno del database.
Successivamente e ogni volta che si vuole iniziare un nuovo allenamento, vengono inviati tre ulteriori dati: l'area corporea da allenare (Upper, Core, Lower o tutte e tre), il peso e il tempo a disposizione (15, 30, 45 minuti): anche in questo caso il service procede a salvare i dati ricevuti.
Il service con i dati ora a disposizione, è grado di eseguire l'algoritmo per il calcolo dell'allenamento da proporre al client: in particolare vengono selezionati un sottoinsieme di esercizi completi di ripetizioni, serie e tempi di recupero da svolgere all'interno dell'intervallo di tempo disponibile.
L'allenamento viene quindi salvato nel database e infine inviato al client.
Da questo momento in poi, l'applicazione cercherà di mantenere sempre aggiornato lo stato dell'allenamento in corso anche lato server. In questo senso, alla fine di ogni esercizio il client richiede al service di salvare l'avanzamento all'interno del database.
Al termine di ogni esercizio, inoltre, è richiesto all'utente di inserire un feedback sul livello di difficoltà incontrato: troppo facile, ottimo, troppo difficile.

In entrambi i casi, tutto ciò corrisponde ad un computazione lato server, con l'obiettivo di cambiare dinamicamente il piano dell'allenamento per renderlo, quindi, adatto all'utente. Questo avviene, in particolare, anche nel caso del feedback "ottimo": il server, infatti, cercherà di impegnare maggiormente l'utente proponendo, ad esempio, un numero maggiore di ripetizioni.
Nel caso il feedback risulti essere "troppo difficile" al server viene inviato come ulteriore parametro il numero di ripetizioni correttamente eseguite.

3.2 Allenamento dinamico

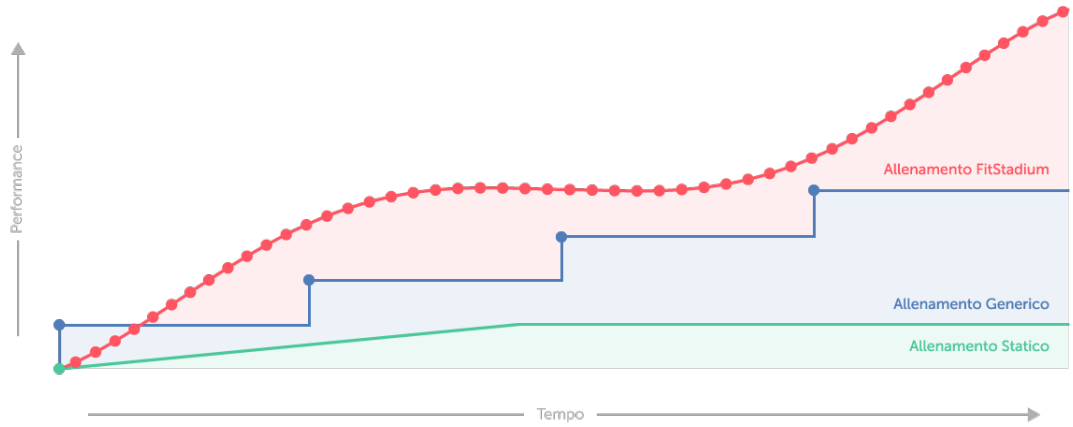

Figura 3.4: Allenamento dinamico

Riprendendo il discorso dell'introduzione di questo capitolo, oltre alla corretta esecuzione dell'esercizio è fondamentale offrire all'utente un'esperienza completamente compatibile con le sue abilità e fare in modo, inoltre, di adattare l'allenamento in modo dinamico, in base alle capacità acquisite o il grado di energie residue.
La maggior parte dei competitor FitStadium, in particolare, propone un tipo di allenamento statico, che non si basa sull'evoluzione dell'abilità dell'utente, ma che cambia dopo un certo intervallo di tempo (solitamente ogni settimana). La diretta conseguenza è che l'allenamento statico propone una certa soglia di difficoltà che raramente è quella più ottimale, rivelandosi alla fine o troppo alta o troppo bassa. Nel caso in cui l'allenamento proposto fosse troppo difficile, l'utente è portato ad eseguirlo male ottenendo così scarsi benefici ed un'alta probabilità di infortunio. Nel caso di un allenamento troppo semplice, invece, l'utente non riuscirà ad ottenere alcun miglioramento per la propria condizione di forma.
L'allenamento dinamico, invece, cambia continuamente cercando di seguire, in modo approssimato, la condizione di chi si allena adattandosi anche alle situazioni particolari come: giorni in cui ci si sente particolarmente stanchi, miglioramento della muscolatura, esercizio appena terminato che ha richiesto

un particolare consumo di energie.
L'allenamento proposto, quindi, non solo risulta essere personalizzato per l'utente, ma cambia dinamicamente durante l'allenamento stesso per far fronte alle esigenze dell'utente.

Su questa linea FitStadium ha adottato un sistema di feedback, attraverso il quale l'utente, alla fine di ogni esercizio, esprime il grado di difficoltà che ha incontrato tramite un'apposita interfaccia grafica. In questo modo, oltre a proporre un allenamento giusto, quest'ultimo può essere modificato in qualsiasi momento secondo particolari logiche che tengono conto, oltre che del feedback, del tempo residuo totale a disposizione (che dipende dalla scelta fatta dall'utente), del tipo di esercizi fatti, del numero di ripetizioni eseguite e così via.
Il grado di soddisfazione che l'utente può esprimere attraverso il sistema di feedback corrente consiste in tre livelli: troppo semplice, perfetto e troppo difficile.

Non sempre, però, questa soluzione risulta essere efficace. In primo luogo, spesso l'utente stesso non è in grado di stabilire con il giusto grado di approssimazione se ha eseguito correttamente l'esercizio oppure no, o se ha fatto la giusta fatica che ci si aspettava.
Inoltre, per certi tipi di esercizi come ad esempio quelli eseguiti a circuito, sarebbe opportuno, al fine di assistere al meglio l'utente, avere dei feedback continui durante l'esecuzione stessa invece che soltanto alla sua conclusione.

3.2.1 Allenamento a circuito

L'allenamento a circuito è una tipologia di allenamento che consiste nell'eseguire una serie di esercizi, senza pausa o con pause molto brevi tra un esercizio e l'altro. Il numero di ripetizioni per ogni esercizio solitamente è medio o alto, utilizzando un carico basso.
Nella pratica l'esecutore si sposta tra una stazione e l'altra con piccoli intervalli, eseguendo le ripetizioni della durata di 15 - 45 secondi.

In genere l'allenamento a circuito è pensato per coinvolgere in un'unica seduta tutto il corpo, riuscendo a generare i benefici tipici dell'allenamento anaerobico e aerobico. Il battito cardiaco durante l'allenamento è mantenuto ad una frequenza superiore a causa del tempo di riposo ridotto.
I vantaggi dell'allenamento a circuito si riscontrano nell'incremento della forza e della resistenza fisica.

Nel caso di FitStadium, l'allenamento viene proposto per gli utenti con obiettivo "dimagrimento" in quanto ha come scopo principale la perdita di peso.

3.2.2 Feedback Automatico

Il caso di studio affrontato, consiste nello sviluppare un sistema di feedback automatizzato: il sistema deve essere in grado di rilevare l'intensità con la quale l'utente esegue l'esercizio ed allo stesso tempo fornire un feedback durante l'attività.
Il feedback potrà essere rivolto sia all'utente, al fine di suggerire la giusta intensità con la quale svolgere gli esercizi, sia in futuro all'applicazione stessa, al fine di avere un responso immediato e automatico utilizzabile per correggere e migliorare l'allenamento proposto.

In questo modo:

- L'utente avrà a disposizione un feedback quanto più possibile "real-time" durante lo svolgimento dell'esercizio.
- L'utente potrà eseguire l'esercizio con la giusta intensità prevista.
- L'utente non dovrà interagire direttamente con l'applicazione, ma i dati vengono rilevati in automatico e ed in maniera continua.

Questo meccanismo verrà in particolar modo adottato durante gli allenamenti a circuito.

Per rilevare i dati necessari al caso di studio è necessario disporre della giusta tecnologia (ad esempio dei sensori). Sono già state analizzate, inoltre, le possibilità e le soluzioni offerte attualmente dal mercato, in particolare si è posta l'attenzione sulle architetture e come quest'ultime possano convivere con l'ecosistema FitStadium.
Per quel che riguarda l'allenamento a circuito, l'informazione più importante è la frequenza del battito cardiaco: mantenere la giusta frequenza durante l'allenamento, garantisce il massimo in termini di benefici oltre che evitare infortuni o situazioni di pericolo.

La possibilità di avere dati provenienti da sensori diversi sui quali svolgere analisi ed elaborazioni, inoltre, permetterà di estrapolare nuove informazioni prima inaccessibili, le quali possono essere utilizzate per offrire un migliore servizio all'utente, ad esempio migliorando l'allenamento proposto (ma da questo punto di vista, le vie possono essere davvero tantissime).

3.3 Realizzazione

In riferimento al caso di studio di questa tesi, l'Apple Watch mette a disposizione il sensore cardiofrequenzimetro, il quale può essere sfruttato per recuperare la frequenza cardiaca dell'utente durante un particolare esercizio. I dati saranno quindi disponibili e recuperabili tramite il repository HealthKit. Questi dati dovranno essere innanzitutto resi visibili all'utente, preferibilmente direttamente sull'Apple Watch, e bisognerà eseguire un confronto con il "range ottimale" previsto, per ottenere così il massimo dall'allenamento. Il range target, come vedremo, può dipendere da diversi fattori. Quest'ultima fase permetterà quindi di fornire un feedback all'utente, suggerendogli di aumentare o diminuire il ritmo di allenamento per eseguirlo al meglio.

Per il feedback inverso, ossia la rilevazione delle prestazioni dell'utente al fine di modificare dinamicamente l'allenamento, può essere sfruttato ancora una volta la frequenza cardiaca più eventualmente i dati di altri sensori sempre presenti all'interno del repository. Questi dati verranno quindi inviati all'infrastruttura FitStadium remota dove viene solitamente eseguito l'algoritmo per la creazione o modifica dell'allenamento.

Per stabilire il range ottimale per la corretta esecuzione dell'allenamento, invece, sono necessari oltre ai dati personali dell'utente anche i dati sull'esercizio: quindi il tipo, il numero di serie, ripetizioni, carico e tempo di riposo.

Ricapitolando, le informazioni richieste sono:

- Dati dell'allenamento da eseguire.
- Dati dell'utente (ad esempio l'età, le condizioni di salute).

Questi dati verrano utilizzati come input di un algoritmo che avrà il compito di calcolare il range target della frequenza cardiaca. Questo range verrà quindi confrontato con la frequenza cardiaca attuale.
Il numero dei fattori che influenzano la frequenza target è in realtà altissimo.

Chiaramente la quantità di dati effettivamente sfruttata dipende dalla bontà e dalla qualità dell'algoritmo utilizzato. In fase di progettazione, comunque, è bene prevedere l'uso di tutti i dati possibili nel migliore dei modi, indipendentemente dallo stato attuale dell'algoritmo.

A questo punto la computazione potrà avvenire, una volta raccolti tutti i

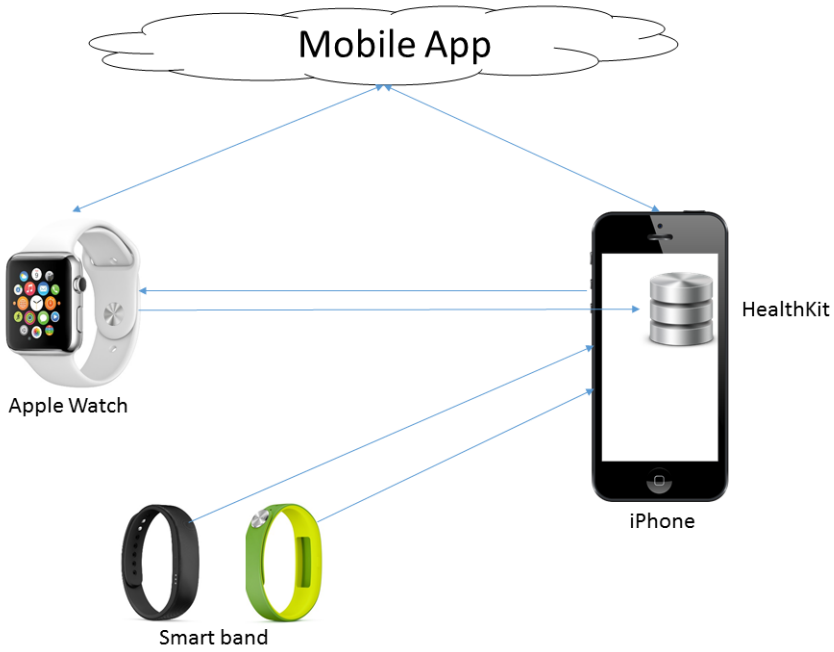

Figura 3.5: Flussi dati sensibili Apple

dati utili, in remoto, localmente su iPhone o localmente su Apple Watch.
La figura "Flussi dati sensibili Apple" (uguale alla 1.2) mostra i possibili flussi di dati sensibili (quelli relativi alla salute e al fitness) tra i vari nodi dell'architettura proposta da Apple.

Per le caratteristiche della rete e del servizio da offrire non conviene eseguire una computazione remota: i dati di cui l'algoritmo necessita, infatti, sono tutti disponibili localmente sull'iPhone ed, inoltre, non si vuole appesantire il server con un carico computazionale aggiuntivo, necessario per rispondere a

tutte le richieste dei client. Il vantaggio dell'eseguire un algoritmo in remoto è in questo caso legato alla possibilità di mantenerlo segreto. Le applicazioni mobile se non adeguatamente protette, infatti, sono facilmente smontabili con il rischio di perdere il "knowhow". Rimane quindi da stabilire se l'algoritmo deve essere eseguito su iPhone o direttamente su Apple Watch.

Ipotizzando un algoritmo localizzato sull'Apple Watch, devono essere trasferiti via Bluetooth i dati relativi l'utente e dell'allenamento. Questi dati, oltre che dall'iPhone, potrebbero essere richiesti anche all'infrastruttura FitStadium, ma solo se presente una connessione Wi-Fi.
La frequenza cardiaca attuale viene rilevata dal cardiofrequenzimetro dell'Apple Watch, ma deve essere necessariamente memorizzata prima su HealthKit per poi essere recuperata tramite una query. La frequenza cardiaca deve essere quindi inviata nuovamente all'orologio per essere visualizzata all'utente ed eseguire il confronto con il range target. Fortunatamente Apple permette, tramite il framework HealthKit, di eseguire un query direttamente dall'Apple watch, rendendo trasparente allo sviluppatore l'implementazione di quest'ultimo trasferimento.

Nel caso di un algoritmo sull'iPhone, non è necessario implementare neppure il trasferimento dei dati di input per l'algoritmo, in quanto i dati dell'esercizio sono già stati ricavati da Fitstadium, mentre i dati relativi all'utente sono memorizzati e recuperabili dal repository locale HealthKit. L'unico dato da trasferire sarà la frequenza target dell'esercizio, l'output dell'algoritmo che verrà utilizzato per eseguire il confronto con la frequenza attuale già presente sull'Apple watch.

Quest'ultima soluzione sembra essere quella più ottimale se consideriamo anche il fatto che il carico computazionale viene spostato sull'iphone, lasciando all'Apple watch, che possiede caratteristiche hardware meno performanti, il solo compito di confrontare i risultati e mostrarli all'utente.

A causa delle proprietà e vincoli della piattaforma Apple, era comunque necessaria la presenza dell'iphone per il corretto funzionamento dell'applicazione sul watch, al fine di recuperare i valori della frequenza cardiaca dal repository.

Ricapitolando, il progetto prevede l'utilizzo della piattaforma Apple ed del linguaggio di programmazione Swift. In particolare è prevista la realizzazione di un'applicazione prototipo per Apple Watch ed un'estensione della relativa applicazione di FitSTadium su iPhone. L'applicazione per Apple Watch dovrà confrontare l'attuale frequenza cardiaca dell'utente con quella "target" calcolata per lo specifico allenamento. Dovrà perciò comprendere un'interfaccia minimale per consentire l'interazione dell'utente e mostrare i risultati; inoltre dovrà interagire con l'applicazione su iPhone che ha, invece, il compito di eseguire l'algoritmo per il calcolo della frequenza di riferimento.

Come prima cosa, è necessario richiedere all'utente l'autorizzazione per la lettura dei dati sensibili come la frequenza cardiaca. Una volta effettuata la richiesta sarà possibile accedere ai dati memorizzati sul repository HealthKit. Il framework WatchKit permette di eseguire questa richiesta direttamente dal dispositivo Apple Watch, suggerendo all'utilizzatore di avviare l'applicazione "compagna" sull'iPhone al fine, quindi, di concedere o negare l'autorizzazione per ogni tipo di dato richiesto.

Con la nuova versione del sistema operativo WatchOS2, è possibile recuperare i dati dal cardiofrequenzimetro in maniera quasi real-time. Per fare questo, si fa affidamento sul framework HealthKit. In particolare, è sufficiente creare un nuovo "Workout" attraverso le API fornite, specificando la location (interna o esterna) e il tipo di allenamento: ciò suggerisce al dispositivo di attivare gli opportuni sensori ed iniziare a registrare i dati. I dati rilevati verrano memorizzati nel repository e saranno recuperabili attraverso specifiche query.
La query è realizzata tramite il framework HealthKit sulla data di inizio del workout e sullo specifico tipo di dato della frequenza cardiaca. Come già ricor-

dato, eseguendo la query direttamente dal watch, il conseguente passaggio dei dati provenienti dall'iPhone attravero la tecnologia Bluetooth sarà trasparente allo sviluppatore. In particolare la query può essere personalizzata al fine di essere notificati ogni qual volta siano presenti nuovi dati dal sensore.
Ad ogni notifica di un nuovo dato, la frequenza cardiaca rilevata viene prima visualizzata all'interno dell'interfaccia grafica e infine confrontata con il range target previsto. A seconda del risultato ottenuto, l'interfaccia assumerà quindi un colore diverso per un feedback immediato all'utente.

Per permettere all'applicazione sul watch di comunicare con la contropar-

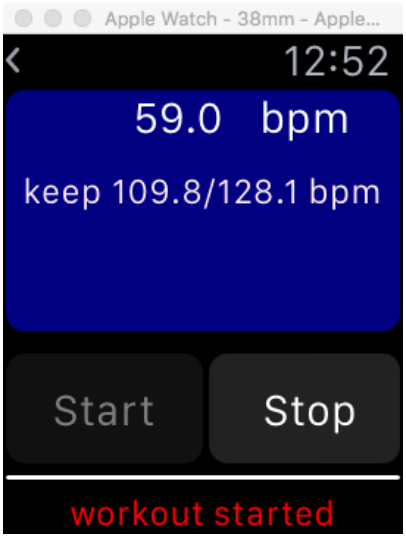

Figura 3.6: Workout Applicazione

te sull'iPhone, è stato utilizzato il watch connectivity framework. Il framework permette di realizzare un'interazione tra le due entità secondo il paradigma dello scambio di messaggi. Appena una delle due parti riceverà un messaggio, il framework richiamerà l'apposito handler implementato sul delegate.
Ad esempio, quando viene selezionata la voce workout sull'Apple Watch, l'applicazione invia un messaggio di richiesta per la frequenza target consigliata. Se l'app non è stata terminata, ossia si trova nello stato active o inactive, eseguirà la richiesta e risponderà con un ulteriore messaggio contenente il dato

richiesto. Se la comunicazione (per qualche motivo) non dovesse avvenire, l'utente sarà comunque in grado di visualizzare la propria frequenza cardiaca ma non beneficerà del supporto previsto.

Attualmente, la ricerca ha teorizzato diverse formule attraverso le quali è possibile ricavare un valore molto approssimato della frequenza cardiaca ottimale. La frequenza cardiaca ottimale è relativa all'esecuzione di una certo tipo di allenamento o al raggiungimento di un determinato obiettivo come la perdita dei grassi ecc. Spesso queste formule si basano sul calcolo della frequenza massimale che un certo utente può raggiungere: anche questa frequenza rappresenta un'approssimazione. Per fornire un migliore servizio, più personalizzato e su misura dell'utente, bisogna fornire un modo per calcolare la frequenza cardiaca massima in modo tale che sia il più vicino possibile a quella reale.
A questo punto la frequenza massima può essere quindi recuperata mediante delle formule (frequenza teorica), attraverso un analisi dei dati presenti in HealthKit, oppure proponendo all'utente di eseguire un test massimale e leggere la frequenza attraverso i sensori. La capacita di raggiungere la frequenza cardiaca massima è influenzata dalla condizione di forma dell'utente che sta eseguendo il test.
Il test può essere una variante di quello formulato dal medico Conconi: l'esercizio consiste nell'eseguire uno sforzo di intensità crescente in ogni intervallo, per poi arrivare allo sforzo massimo nel tratto finale, fino a che le prestazioni non iniziano a calare.
L'applicazione implementata consente quindi di scegliere se eseguire l'allenamento oppure il test per il calcolo della frequenza cardiaca massima.

Una volta ottenuta la frequenza massimale, essa viene memorizzata all'interno dell'Apple Watch e viene offerto all'utente la possibilità di eseguire il workout con la frequenza ottimale ottenuta attraverso il calcolo sul valore reale o su quello teorico. In ogni momento l'utente può rieseguire il test per un nuovo calcolo della frequenza massima.

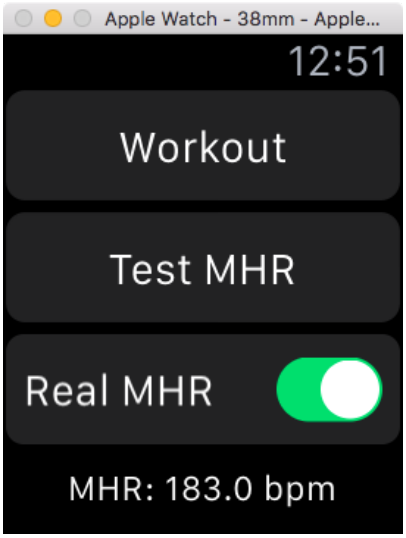

Figura 3.7: Applicazione Main Interface

Ci sono vari motivi che possono portare a non eseguire il test ed affidarsi ad una frequenza più approssimata: ad esempio essendo un esercizio volto a rilevare le capacità massimali, e quindi piuttosto faticoso, potrebbe risultare addirittura pericoloso se affrontato da persone anziane.

3.4 Un'architettura per più tecnologie

Indipendentemente dal corrente caso di studio, è fondamentale fare affidamento sulla giusta architettura software per la realizzazione delle proprie applicazioni. Quest'ultima permette, infatti, di gestire ed offrire nel migliore dei modi l'insieme dei servizi per i quali è stata progettata.
Nel caso di servizi destinati al mondo dei dispositivi mobili e legati al fitness, non si possono ignorare del tutto le soluzioni e le tecnologie messe a disposizione dei due colossi tecnologici (Apple e Google) analizzati precedentemente. Le ragioni di questo sono abbastanza ovvie: il mercato è attualmente orientato principalmente su queste due piattaforme, perciò i dati abilitanti a questi tipi di servizi sono localizzati sulle loro infrastrutture. Molti brand, inoltre, sviluppano applicazioni e dispositivi che a loro volta si appoggiano o sono compatibili con queste tecnologie.
Tutto ciò ha ovviamente sia aspetti positivi sia aspetti negativi, quest'ultimi in particolar modo sono legati ai limiti delle piattaforme proposte.

Relativamente al processo di produzione, sarebbe comodo utilizzare strumenti sullo stampo di *Cordova* o *RoboVM* i quali permettono, partendo da un determinato linguaggio, di ottenere un'applicazione per smartphone sia per la tecnologia Apple che per quella di Google. Come abbiamo analizzato e avuto modo di sperimentare, però, i due brand si basano su due architetture diverse per questo tipo di sevizi. Queste architetture impongono una progettazione diversa, in quanto entrambe spingono verso precise logiche applicative.
Al fine di ovviare ai costi di produzione e rendere più efficiente lo sviluppo del software in termini di velocità, riuso del codice e manutenzione, si potrebbe pensare ad un unica architettura di sistema la quale si presti ad essere istanziata su più tecnologie diverse, pur mantenendo un core condiviso.
Ciò che corrisponde al core, ossia ciò che viene spesso indicato come la parte specifica di un'applicazione, è la *business logic*: fattorizzando il più possibile questo tipo di codice, il successivo trasferimento su tecnologie differenti risulterà essere meno dispendioso.

Ma non è solo una questione di efficienza e risparmio nel processo di produzione: si tratta anche di avere a disposizione l'architettura giusta per fornire le specifiche proprietà richieste da questo genere di applicazioni

Già dall'analisi precedente delle piattaforme per il fitness di Apple e Google, sorgono delle proprietà e delle differenze che suggeriscono le potenzialità dell'adottare un'architettura rispetto ad un'altra.
Ad esempio è importante evidenziare che la raccolta dei dati provenienti dai sensori in aggiunta a quelli inseriti dell'utente in specifici repository (HealthKit o in remoto con Google Fit), nasconde la distribuzione dei vari device, ossia garantisce la *distribution transparency*.
Un componente remoto come quello di Google Fit permette l'accesso uniforme (*access transparency*) ai dati a tanti dispositivi tra loro eterogenei. Per fare questo Google mette a disposizione sia delle API specifiche per certi linguaggi di programmazione, sia si attiva come web-service RESTful e quindi accessibile attraverso il middleware WWW.
Un limite della piattaforma di Google è quella di restringere la tipologia dei dati a solo quelli relativi al fitness: in realtà l'utente, a differenza di quanto succede in Apple, può crearsi dei dati custom, ma Google vieta che questi siano rivolti a finalità diverse da quella del fitness. Per il servizio offerto da FitStadium, sarebbe utile permettere l'acceso remoto a dati relativi anche alla salute o ai valori dei sensori ambientali. Fitness, salute e ambiente, infatti, condividono tra loro numerose dipendenze che possono rilevarsi centrali nello sviluppo di un algoritmo o di un certo servizio. L'unione tra il servizio mobile per il fitness (e quindi anche per la salute) con i dati offerti dall'infrastruttura delle così dette *smart city* apre le porte verso la *smart health*. Il concetto di smart health e le numerose possibilità ad esso collegate verranno trattate nel prossimo capitolo.
Affiancando a Google Fit un componente aggiuntivo per la raccolta dei dati relativi alla salute degli utenti ed un nuovo componente per i dati ambientali, sarà possibile effettuare una sincronizzazione automatica dei dati per una

grande varietà di dispositivi (*Making resources available*) indipendentemente dalla loro tecnologia. L'effetto di tutto ciò è un aumento della scalabilità del sistema.
Per quanto riguarda il repository per i dati ambientali probabilmente, per gestire la grande quantità di dati proveniente dalle smart city sparse per il globo, sarà a sua volta composto da più componenti tra loro distribuiti.
Seguendo la linea tracciata da Google Fit, i servizi offerti dai nuovi componenti potrebbero essere accessibili dal WWW e quindi come Web Service. Questo, certamente, garantirebbe la compatibilità con la stragrande maggioranza dei device, ma non è detto rappresenti il metodo migliore.
Mantenere i dati sensibili lontano dall'utente su una infrastruttura cloud, inoltre, richiede di stabilire meccanismi e politiche di sicurezza e privacy adeguate.

Oltre che in remoto, per motivi di efficienza o per l'accesso offline, sarebbe interessante memorizzare una copia dei dati richiesti vicino all'utente, ad esempio sul device.
Tutto ciò sarebbe in particolar modo utile se la business logic dell'applicazione risiedesse sul client, come nel corrente caso di studio: un componente installato sul dispositivo che svolge il ruolo di buffer dei dati può essere un fattore abilitante per una computazione locale.
La presenza di questo tipo di componente non solo per Apple ma anche per la piattaforma di Google (e quindi per i dispositivi Android), all'interno del quale memorizzare e leggere i dati, permette di progettare una logica comune per entrambe le tecnologie e velocizzare il processo di produzione.
Questo è possibile solo se all'interno del repository è possibile recuperare i dati proveniente da diversi sensori o che altre applicazioni hanno contribuito a reperire, esattamente come succedeva su iPhone. Il componente dovra essere quindi sincronizzabile con il repository remoto Google Fit, all'interno del quale gli utenti Google sono soliti memorizzare i dati relativi al fitness, o con il componente Health per la gestione dei dati relativi alla salute.
Il nuovo repository, inoltre, potrebbe essere istanziato sia come componente

interno alla mobile app sia, se il sistema Android lo permette, come componente esterno come mostrato nella figura "Architettura generale". In questo secondo caso il componente offrirà le sue funzionalità a tutte le applicazioni che vogliono beneficiare di questi dati. Ovvio che, essendo un componente di terze parti, solo le applicazioni che sono a conoscenza della sua esistenza inseriranno o leggeranno dati da esso.
Al momento nè l'iPhone, e di conseguenza neppure il nuovo repository per Android, dedicano un componente alla memorizzazione locale dei dati provenienti dai sensori ambientali, che rimarranno accessibili solo attraverso la rete Internet.

A seconda di come verrà realizzato il repository, i dati potranno essere memorizzati in forma criptata e prevedere dei meccanismi con il fine di aumentare la sicurezza e la privacy, così come succedeva sulla piattaforma Apple.
Le funzionalità saranno quindi accessibili tramite l'utilizzo di un framework sulla falsa riga di HealthKit.

Il nuovo componente rappresenta una sorta di cache locale da parte del client che potrebbe causare problemi di consistenza. La mobile app stessa, inoltre, potrebbe memorizzare dei dati locali privati creando un ulteriore livello di cache. In definitiva, si aggiunge una nuova proprietà:

- *Replication transparency*: le repliche dei dati dovrebbero essere accessibili nello stesso modo, in maniera trasparente, attraverso lo stesso nome e possibilmente nello stesso stato.

In questo caso, la consistenza fa riferimento anche allo stato (intero o parziale) del repository.
E' sufficiente immaginarsi, infatti, una query che richiede ad esempio il numero totale dei passi eseguiti in una settimana. Se il repository locale non è consistente relativamente al periodo di tempo considerato o al tipo di dato, il risultato potrebbe essere in difetto dei dati provenienti da certe sorgenti.

Parte della computazione potrebbe essere trasferita direttamente sul nuovo componente, togliendo la responsabilità all'applicazione di certe implementazioni. Ad esempio si potrebbero fornire risultati già parzialmente raggruppati secondo particolari criteri, come avviene nei più diffusi database o su Google Fit stesso.

Inoltre, più flussi di controllo potrebbero attraversare il nuovo componente, in particolar modo se viene istanziato in maniera tale che sia accessibile da più applicazioni. Su questa linea, quindi, deve essere garantito anche l'accesso concorrente (*concurrency transparency*) ed essere prevista una gestione dei conflitti riguardo alla sincronizzazione con i componenti remoti.

La sincronizzabilità del repository potrebbe essere bidirezionale, scaricando la responsabilità dell'upload dei dati dalla mobile app al nuovo componente.
Per disporre di questa caratteristica anche in Apple e quindi mettere a disposizione questi dati ad un gran numero di dispositivi, è necessario istanziare un nuovo componente che avrà lo scopo di recuperare i dati attraverso HealthKit e trasferirli in remoto.
Saranno così, ad esempio, disponibili su device Android i dati che provengono non necessariamente solo dai sensori locali o da Android Wear, ma anche quelli dei vari dispositivi Apple.
All'interno di questo processo ci si dovrà preoccupare di gestire le diverse rappresentazioni dei dati rispetto ai vari componenti. Il nuovo componente sarà intrinsecamente legato all'organizzazione che intende sfruttare questa architettura: in questo caso FitStadium sincronizzerà i dati con un componente inserito nella propria infrastruttura (vedi figura).
Bisogna inoltre ricordare che attualmente, a causa della Privacy di Apple, non è possibile memorizzare i dati fitness di HealthKit in Google Fit in quanto verrebbero utilizzati anche per scopi pubblicitari.

Come già annunciato, per sfruttare una business logic comune a più tecno-

logie, saranno necessari strumenti simili a Cordova o *RoboVM* attraverso i quali ricavare il codice specifico per più tecnologie. Questi strumenti dovranno perciò adottare una tecnologia di programmazione (un linguaggio) con integrate delle API. Quest'ultime dovranno garantire l'accesso ai nuovi componenti indipendentemente dalla piattaforma Apple o Google.

Il componente "Health" per la raccolta dei dati relativi alla salute è stato

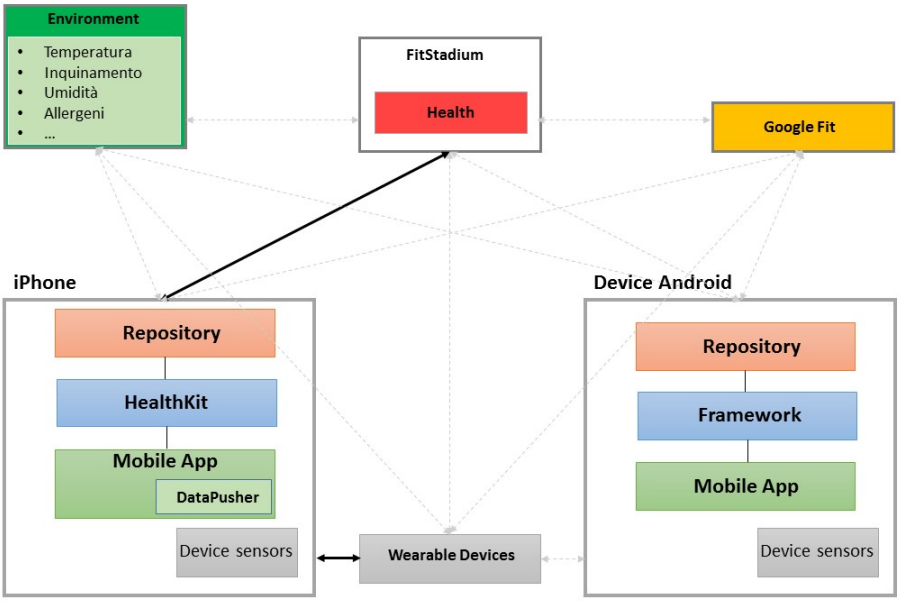

Figura 3.8: Architettura generale

aggiunto come nuovo componente dell'infrastruttura aziendale. Grazie ad esso, vengono raccolti solo i dati relativi agli utenti registrati.
Come già spiegato in precedenza, è presumibile che il componente "Environment" consenta l'accesso ai propri dati attraverso dei web services secondo la logica dei *Open Data*. A sua volta anche FitStadium potrebbe avere la possibilità di contribuire inserendo dati al suo interno.
Nella figura vengono inoltre mostrati i flussi di dati sensibili che possono essere attivati a seconda delle necessità dell'applicazione da realizzare. In questo caso viene mostrato, dalle linee più spesse, il percorso dei dati relativi al caso di

studio della tesi.

Tutti i componenti di questa architettura finale rappresentano dei "tasselli" che abilitano non solo la realizzazione di applicazioni mobile per servizi di Fitness ma anche un'infinità di scenari che rientrano nell'ambito della smart health.

Capitolo 4

Il Ruolo dell'ICT nell'Healthcare e nel Fitness

NEl corso degli anni l'età media dell'essere umano, così come la qualità di vita nei paesi avanzati, è migliorata grazie al progresso scientifico.
Un importante contributo è stato sicuramente apportato dall'*information and communication technology* (ICT).
Il ruolo dell'ICT all'interno del settore *healthcare*, è stato ed è particolarmente importante per la riduzione dei costi e l'aumento dell'efficenza dei servizi offerti dalla nostra società.
Nonostante questo il progresso tecnologico ed informatico non si ferma ma continua ad introdurre novità e, nei casi più fortunati, aprire nuove strade per sviluppi futuri.
Basta pensare alla diffusione degli smartphone, dei più variegati dispositivi mobili, o ancora prima l'avvento di Internet e del World Wide Web: in un modo o nell'altro queste tecnologie hanno influenzato il modo di vivere della comunità.
Risulta di fondamentale importanza, quindi, fare ordine e chiarezza sui principali trend, cercando di individuare i concetti base sui quali focalizzare gli sviluppi futuri. Così facendo gli sforzi dei ricercatori e dei sviluppatori, saranno più efficienti e orientati verso la giusta direzione, nonostante tutte le complicazioni e sfide inevitabili come ad esempio la difficile interazioni fra le

varie discipline, la sicurezza o la privacy.

In questo capitolo si cercherà di descrivere i principali concetti che sono nati grazie all'avvento delle tecnologie informatiche nel settore della salute e quindi nel fitness, fino ad arrivare al concetto, ancora in fase consolidazione, di smart health.

4.1 E-health e M-health

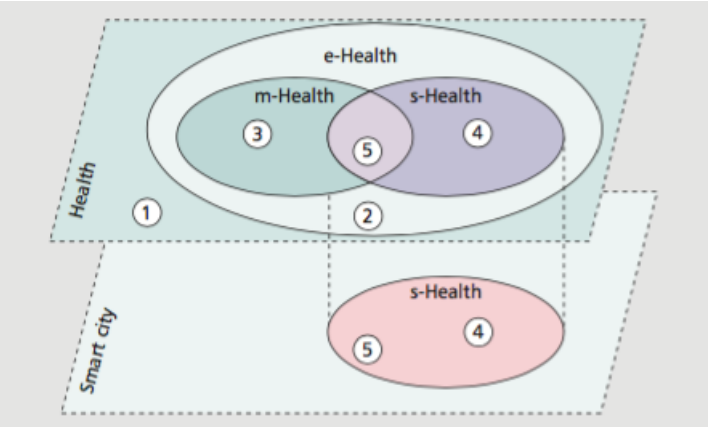

Figura 4.1: Set per l'ambito della salute e delle smart city

Il primo concetto introdotto, segna l'ingresso dell'ITC nel settore e prende il nome di *Electronic Health* (*e-health*): consiste fondamentalmente nell'adozione dei sistemi informativi informatici e delle reti di telecomunicazione.
Un esempio è l'attività del medico di base, che dopo aver visitato il paziente, salva tutti dati all'interno di un database digitale.
Se propriamente sfruttato, l'e-health contribuisce significativamente alla riduzione dei costi di gestione ed aumenta l'efficienza: riduce in modo sostanziale la necessità di spostare professionisti o pazienti, migliora le comunicazioni, cataloga tutte le informazioni dei pazienti e velocizza la sorveglianza dei parametri di salute.
Su questa linea, inoltre, il costo delle risorse medicinali subiscono un generale abbassamento mentre i trattamenti e le procedure diventano più comode per i pazienti.
Nel complesso , l'e-health può essere considerata una rivoluzione in quest'area. Comunque, probabilmente una più importante rivoluzione sta avvenendo grazie alla diffusione dei mobile devices.

Seguendo il consolidamento dell'e-health, infatti, il generalizzato uso dei dispositivi mobili con funzionalità di localizzazione apre le porte al concetto di

mobile health (*m-health*).
Con m-health possiamo intendere l'utilizzo di comunicazioni mobili nel campo della salute oppure, più precisamente, la consegna di servizi sanitari attraverso l'uso di dispositivi mobili.
La m-health, pur muovendo tutt'ora i suoi primi passi, possiede elevate potenzialità in quanto aggiunge all'e-health tutti i benefici offerti da questi dispositivi come: funzionalità di monitoraggio, ampia disponibilità, immediatezza.
Ad esempio l'utente è in grado di visualizzare le informazioni sui medicinali somministrategli dal medico direttamente dal proprio cellulare.
M-health ridefinisce i servizi healthcare in tre principali aspetti:

- Permette un facile accesso ad un grande numero di servizi e informazioni.
- Può essere user-oriented.
- Può essere personalizzato.

M-health diventa, inoltre, uno strumento potente in molti campi di ricerca, come il continuo monitoraggio dello stato del paziente, pre-rilevamento di situazioni di emergenza e rilevamento di cambiamento dei parametri di salute.

4.2 Smart cities

Parallelamente a tutto questo, si sta sviluppando un nuovo promettente fenomeno che ha come obiettivi contrastare i problemi locali del territorio, i problemi economici e del trasporto pubblico, migliorare la qualità della vita e la consapevolezza cittadina sulle attività governative: questo concetto, ancora in realtà non completamente definito ma con molti punti in comune con la m-health, prende il nome di *Smart Cities*.
Una definizione di questo concetto è stata data da *Caragliu* e poi successivamente estesa da *Pèrez et al.*:

> Smart cities are cities strongly founded on information and communication technologies that invest in human and social capital to improve the quality of life of their citizens by fostering economic growth, participatory governance, wide management of resources, sustainability, and efficient mobility, whilst they guarantee the privacy and security of the citizens.

In questo senso molti governi stanno già investendo in ICT per equipaggiare le proprie città con infrastrutture tecnologiche, per creare un ambiente intelligente, al fine di aumentare la responsabilità sociale e tutelare il cittadino con servizi migliori. Le possibilità, in realtà, sono praticamente infinite.
Le smart cities si basano principalmente su una fitta rete di sensori, attraverso i quali vengono recuperati dati di diverso tipo e natura che includono: temperatura, umidità, concentrazione di allergeni, inquinamento, condizioni del traffico e così via.
Questi dati possono essere raccolti, manipolati e possono quindi determinare il comportamento di applicazioni e servizi. Allo stesso tempo, dopo aver ricevuto questi dati le applicazioni posso incidere a loro volta sull'ambiente.
Ci sono numerosi campi nei quali le smart cities potrebbero svolgere un ruolo chiave: alcuni di questi sono la sicurezza pubblica, lo sviluppo economico, educazione, servizi sociali, healthcare ed altri.
Ad esempio è possibile immaginare un utente che consulta un pannello informativo, nel quale sono presenti i dati sulla concentrazione di sostanze allergiche. A questo punto l'utente può considerare la possibilità di percorrere una via più sicura per la propria salute.

4.3 Smart Health

Il concetto di m-health e quello di smart cities conducono in modo naturale alla coniazione di un nuovo concetto: la *Smart health* (*s-health*).
La s-health può essere intesa come il naturale complemento della m-health, attraverso la consapevolezza delle condizioni ambientali apportata dalla smart

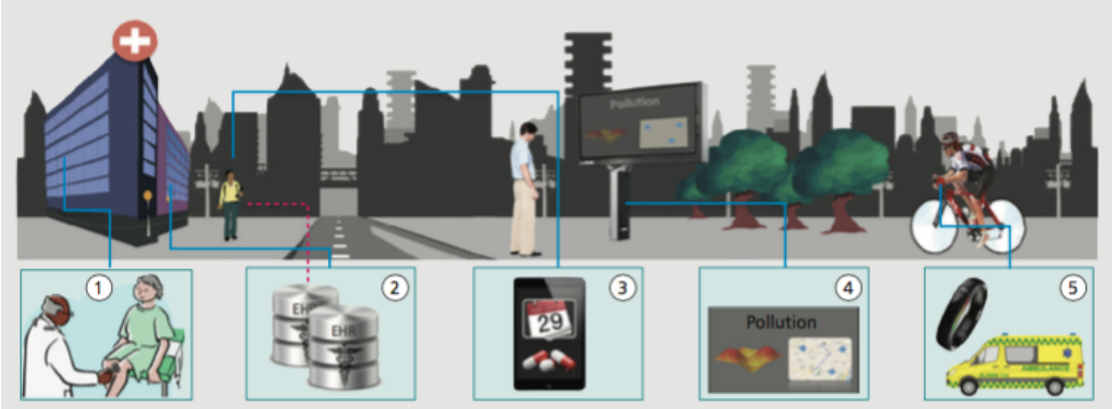

Figura 4.2: Esempi di diverse applicazioni per la salute all'interno delle smart city

city.
Nonostante questi ultimi due concetti siano stati studiati (e continuano ad esserlo) in maniera indipendente, recentemente si è verificata una convergenza su diversi punti in comune.

> Smart health (s-health) is the provision of health services by using the context-aware network and sensing infrastructure of smart cities.

In accordo con questa definizione, s-health è una specializzazione dell'e-health, in quanto è basata sull'infrastruttura ICT della smart cities. Inoltre, s-health è differente dalla m-health nel senso che questa infrastruttura non è detto sia mobile: in effetti, in molti casi fa riferimento a dei sensori fissi.
L'adozione di un nuovo concetto, comunque, è giustificato da ulteriori differenze significative, dal concetto di m-health:

- **Sorgenti dell'informazioni**: I dati nella m-health provengono dai pazienti. Nella s-health i dati posso provenire non solo dai pazienti ma da più sorgenti tra loro indipendenti (cioè tramite la rete di sensori della smart city).
- **Flussi dell'informazioni**: la m-health è *user-centric* ed i servizi possono essere personalizzati per ogni utente. Nella s-health i servizi possono essere anche *city-centric*.

Come esempio consideriamo un ciclista che indossa un braccialetto per il fitness, con accelerometro ed altre funzionalità che permettono di rilevare un incidente.
I sensori percepiscono la caduta ed inviano un segnale di alert all'infrastruttura della città. Una volta ricevuto il segnale, il sistema analizza le condizioni del traffico e viene spedita un'autoambulanza lungo la direzione migliore.
In aggiunta, i semafori della città vengono dinamicamente configurati al fine di ridurre il tempo di attesa dei soccorsi per il ciclista.

Con la definizione e il consolidamento del nuovo concetto di s-health, viene favorito lo sviluppo in sinergia di tecniche, modelli, interazioni che contribuiscono a ridurre i costi, incrementare la qualità di vita, collezionare grandi quantità di dati per la ricerca e una globale evoluzione della società in generale.

4.4 Privacy e sicurezza

A causa dell'elevata sensibilità dei dati gestiti dalle tecnologie informatiche nell'ambito del fitness e dell'healthcare, il problema della protezione della privacy e della sicurezza dell'infrastruttura, risulta diventare di fondamentale importanza.
Dai dati collezionati dalla smart city, ad esempio, potrebbe essere possibile inferire le abitudini dei cittadini, lo stato sociale e persino la religione. Tutte queste variabili sono davvero sensibili, e se combinate con quelle del fitness e della salute, il risultato è ancora di più delicato.
L'utilizzo di dispositivi mobili e la grande quantità di dati da gestire, inoltre, porta ad adottare soluzioni cloud-based. Grazie all'infrastruttura cloud, infatti, è possibile mantenere sincronizzati tra loro i dispositivi più eterogenei e beneficiare di servizi come infrastructure as a cloud (IaaC), platform as a cloud (PaaC) e software as a service (SaaS) senza i costi di gestione e mantenimento richiesti. La sicurezza e la privacy, però, è anche in questi casi un fattore da tenere bene in considerazione, soprattuto a causa degli attacchi di cui questo

servizio è particolarmente affetto.

Conclusioni

Giunti alla conclusione di questa tesi, è possibile trarre delle considerazioni generali sui servizi e sulle tecnologie sperimentate.
Il mercato delle tecnologie informatiche si sta sempre più dirigendo verso sevizi user centric, ossia funzionalità che ruotano attorno alle necessità e ai bisogni richiesti dall'utente. Le conseguenze di ciò sono riscontrabili già dalla fase di progettazione, nella quale l'utente finale risulta essere fondamentale.
I wearable devices e la loro attuale diffusione ne sono la prova lampante, in quanto aumentano la pervasività di questi sistemi e mettono a disposizione dati estremamente personali, i quali possono teoricamente influenzare il comportamento stesso delle applicazioni, creando funzionalità personalizzate e "su misura" dell'utente.
Tutto ciò è stato sperimentato grazie al caso di studio propostoci dall'azienda FitStadium.
Lo svolgimento di questa tesi ha permesso prima di tutto di analizzare le proprietà architetturali delle piattaforme Apple e Google per la realizzazione di applicazioni mobile per il fitness. Lo studio ha messo in risalto anche quali siano state le scelte dei due colossi tecnologici nel momento in cui si sono approcciati a questo tipo di servizi. I sforzi di Apple si sono concentrati sull'aspetto della privacy e sicurezza dei dati; scelta che è stata in qualche modo obbligata nel momento in cui si è deciso di integrare i dati sull'attività dell'utente con quelli più sensibili relativi alla salute. Google, invece, ha scelto la strada della sincronizzazione, rendendo disponibili e utilizzabili i dati raccolti ad un vasto insieme di dispositivi. Per quanto riguarda i dati sulla salute Google mise a

suo tempo a disposizione un ulteriore servizio chiamato *Google health* che però è attualmente sospeso a causa della timida accoglienza da parte degli utenti.
L'attenzione si è poi spostata sullo specifico caso di studio.
La realizzazione dell'applicazione ha evidenziato l'importanza del ruolo svolto dall'architettura della piattaforma utilizzata. Quest'ultima con i suoi vincoli e proprieta è il principale fattore abilitante per la scelta di una precisa logica applicativa e dalla quale dipenderanno poi le prestazioni finali.
Nel caso particolare della tecnologia Apple, i meccanismi logici sono stati spesso vincolati anche dalle caratteristiche dei framework che lo sviluppatore è obbligato ad utilizzare per accedere a determinate funzionalità. Da un lato questo permette ad Apple di garantire certe caratteristiche alle applicazioni sviluppate, dall'altro limita la libertà e la creatività dello sviluppatore. In futuro, ad esempio, sarebbe auspicabile introdurre la possibilità di accedere direttamente ai dati dei sensori dell'Apple Watch o decidere quali di questi attivare per il monitoraggio di un'attività fisica (così come avviene in Android).
Sono necessari, inoltre, miglioramenti nell'algoritmo per il calcolo della frequenza target ottima, cercando di sfruttare maggiormente i dati messi a disposizione dai repository. In questo modo si terrà conto di più fattori che incidono poi sul calcolo finale, ottenendo quindi un'approssimazione migliore. Stesso discorso vale per il calcolo della frequenza massima teorica dell'utente.
Dal caso di studio, inoltre, è nata la necessità di concepire un'architettura attraverso la quale realizzare un'unica logica applicativa che sia in seguito facilmente trasportabile su entrambe le tecnologie.
Le motivazioni di fare questo sono diverse, in particolare avere la giusta architettura per fornire le proprietà richieste da questo tipo di servizi (coniugando quelle di Google e di Apple), ma anche una migliore gestione e manutenibilità del prodotto ed un processo di produzione più efficiente.
Tra i componenti, si è ritenuto ragionevole prevedere anche l'inserimento di un sistema per la gestione dei dati provenienti dai sensori delle smart city. Questo permette di creare un eco sistema per la smart health, all'interno del quale il comportamento delle applicazioni viene modificato in funzione dell'environ-

ment e viceversa.
E' risultato subito chiaro che, per le caratteristiche dei device e delle applicazioni, per permettere un vasto accesso a questi dati la soluzione più comoda è farlo attraverso il www. Questa tesi, in particolare, si è focalizzata sul lato end-user ma è auspicabile per sviluppi futuri condurre uno studio lato server per analizzare i middleware o i paradigmi che più si adattano alle caratteristiche di questo componente.
Ad esempio, come si evince anche dal caso di studio, risulta essere molto utile ottenere i dati in maniera più possibile real time, ad esempio per avvertire il prima possibile l'utente. Gli sviluppi futuri possono perciò muoversi anche in questa direzione, progettando una rete di sensori ideale a questi scopi.

Bibliografia

[1] Apple Watch *http://www.apple.com/it/watch/*.

[2] WatchKit - Apple Developers *https://developer.apple.com/watchkit/*.

[3] HealthKit - Apple Developers *https://developer.apple.com/healthkit/*.

[4] Tanenbaum, A. S. and van Steen, M. (2007) *Distributed Systems. Principles and Paradigms* Pearson Prentice Hall, Upper Saddle River, NJ USA, 2nd edition.

[5] A.Omicini *Introduction to Distributed Systems.*

[6] A.Solas, C.Patsakis, M.Conti, I.S.Vlachos, V.Ramos, F.Falcone, O.Postolache, P.A.Pèerez-martinez, R.Di Pietro, D.N.Perrea, A.Martinez-Ballestè *Smart Health: A Context-Aware Health Paradigm within Smart Cities.*

[7] Wikipedia *Allenamento a circuito.*

[8] Google Fit - Google Developers *https://developers.google.com/fit/*.

[9] FitStadium *https://www.fitstadium.com.*

[10] Wikipedia *Wearable technology.*

[11] Calcolo frequenza cardiaca *www.my-presonaltrainer.it.*