

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Scuola di ingegneria e architettura
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

UTILIZZO DEL WEB SEMANTICO PER
LINTEGRAZIONE E LA CONDIVISIONE DEI DATI
IOT SUL FITNESS E LA SALUTE

Elaborata nel corso di: Web semantico

Tesi di Laurea :
CHIARA SILVESTRO

Relatore:
Prof. ANTONELLA
CARBONARO

Co-relatori:
Dott ROBERTO REDA

ANNO ACCADEMICO 2017–2018
SESSIONE I

PAROLE CHIAVE

Web semantico
Linked Open Data
Internet of Things

Indice

Introduzione	vii
1 Web semantico	1
1.1 L'evoluzione del web	1
1.2 Introduzione al web semantico	3
1.3 Cos'è il web semantico	4
1.3.1 Architettura del web semantico	5
1.3.2 Ontologie	7
1.4 I linguaggi del web semantico	9
1.4.1 XML	9
1.4.2 RDF	9
1.4.3 RDFS	11
1.4.4 OWL	12
1.4.5 SPARQL	12
2 Internet of Things	15
2.1 Cos'è l'Internet of Things	15
2.1.1 Architettura dell'Internet of Things	17
2.1.2 Criticità dell'IoT	18
2.1.3 Ambiti di utilizzo dell'Internet of Things	19
2.1.4 Applicazioni IoT nel settore medico	21
2.2 Wearables	22
2.2.1 Wearables per Fitness Sport e Benessere	23
2.2.2 Vantaggi e svantaggi dell'utilizzo quotidiano dei we- reables	28
2.2.3 Il futuro dei wearables	30

3	Linked Open Data e portali LOD	33
3.1	Open data	33
3.1.1	Data.gov, esempio di open data	34
3.2	Linked data	35
3.2.1	Linked data application	38
3.3	Linked open data	40
3.3.1	Settori di utilizzo dei LOD	43
3.3.2	Esempi di portali LOD	44
4	Caso di studio	49
4.1	Casi d'uso	50
4.2	Architettura	53
4.2.1	Tecnologie utilizzate	54
4.3	Implementazione della piattaforma	67
4.3.1	Progetto di partenza	67
4.3.2	Modifiche apportate al progetto di partenza	71
4.4	Ontologia applicativa	73
4.5	Diagramma della piattaforma	76
4.5.1	Pagine della piattaforma	80
4.5.2	Dettagli implementativi	88
4.5.3	Avvio	96
5	Conclusioni	99
5.0.1	Miglioramenti	100
5.0.2	Sviluppi futuri	101

Introduzione

Se un tempo erano sufficienti delle semplici cuffiette per fare sport, oggi risulterebbe strano allenarsi senza dei dispositivi che permettano di monitorare l'andamento della propria attività fisica, come i battiti cardiaci, la distanza percorsa, la velocità media e tanto altro ancora. I dispositivi indossabili più utilizzati attualmente nell'healthcare sono i braccialetti e gli smartwatch. L'Internet Of Things sta spopolando sempre di più e, in particolare nell'ambito del fitness, dove ci si aspetta una crescita costante per gli anni a venire. A questo proposito è prevista, inoltre, una settorializzazione sempre maggiore per ogni specifico tipo di allenamento. Ci si aspettano anche ulteriori incrementi di funzionalità dei dispositivi wearables, i quali ad oggi sono in grado di raccogliere dati attraverso dei sensori, caricarli e renderli disponibili su Cloud agli utenti.

Tuttavia, i device IoT incentrati sul fitness e in generale sul wellness, producono dati di formati diversi. Tale eterogeneità e la mancanza di standard a cui uniformarsi, fanno sì che i dati rimangano limitati all'interno del sistema, senza la possibilità di avere una visione integrata di essi. Ciò porta ad una riduzione della loro potenzialità e della possibilità di estrarre nuove informazioni.

Scopo della presente tesi è dunque quello di creare una piattaforma che permetta di integrare tali dati utilizzando un formato comune, in modo che l'utente che utilizza diversi device possa visualizzare grafici contenenti le informazioni di un certo allenamento (come farebbe sulle altre piattaforme), ma con il vantaggio di avere a disposizione i dati di tutti i dispositivi. Questo permette all'utilizzatore una visione più completa e generale dell'attività fisica e della sua condizione di salute. Per esempio l'utente, utilizzando tale piattaforma, sarà in grado di confrontare il battito cardiaco rilevato da un certo dispositivo con la pressione sistolica e diastolica ottenuta da un altro, ottenendo un quadro più completo del proprio stato fisico. La piattaforma

dovrà essere un vero e proprio portale linked open data, permettendo quindi all'utente di interrogare direttamente i dati memorizzati attraverso un apposito endpoint.

Il progetto è stato realizzato cercando di sfruttare al massimo le tecnologie del web semantico assegnando la semantica ai dati raccolti dai vari dispositivi, cosicché le macchine possano comprenderne il significato, ed eliminando il problema dell'eterogeneità dei formati, convertendoli secondo un certo vocabolario in modo tale da poter unire i dati e trarre nuove informazioni.

Nel primo capitolo della tesi viene introdotto il web semantico, descrivendo la storia del web, le nozioni principali di questo campo e i linguaggi che lo caratterizzano. Nel secondo capitolo invece viene data una definizione al concetto di Internet of Things: sono descritti i suoi maggiori campi di utilizzo, soffermandoci in particolare sui wearables in ambito del fitness, fornendone esempi pratici reali. Nel terzo capitolo vengono definiti i concetti di open data, linked data e linked open data, delineando alcuni dei principali portali LOD. Il quarto capitolo invece è incentrato sul caso di studio: qui sono dettagliate le tecnologie utilizzate e le modalità di realizzazione del progetto.

Capitolo 1

Web semantico

1.1 L'evoluzione del web

Nel 1989 Tim Berners Lee inventò il World Wide Web (WWW), un sistema che permetteva la consultazione di pagine web raggiungibili con iperlink attraverso l'utilizzo di Internet, e nel 1991 pubblicò il primo sito web al mondo.

Il decennio che va dal 1990 al 2000 è stato considerato come quello del Web 1.0: secondo lo stesso Berners sono stati gli anni dell'”only read web”. Con ”1.0” si intende definire la prima diffusione del web: era l'Internet dei contenuti, caratterizzato da siti web statici, formati quindi da un insieme di pagine statiche con testo e immagini concatenate da semplici link, realizzati in HTML. Solo gli esperti del settore, i quali avevano le competenze tecniche e gli strumenti necessari, potevano aggiornare le pagine di un sito. Le pagine erano dunque realizzate una ad una e allo stesso modo venivano modificate dagli sviluppatori quando necessario. Dal momento che le pagine potevano essere solo consultate, gli utenti potevano soltanto usufruire dei contenuti, senza la possibilità di creare interazioni.

Negli anni 2000 – 2006 si entrò poi nella fase del “read-write web”, il web 2.0. Con l'introduzione dei linguaggi di programmazione dinamici, gli sviluppatori hanno permesso anche a utenti non tecnici di interagire con i contenuti dei siti. Si iniziò a dare importanza all'usabilità e al modo di condividere i contenuti. Questa fase è caratterizzata da una partecipazione attiva degli utenti per quanto riguarda la costruzione dei contenuti, la loro classificazione e distribuzione.

Tim Berners-Lee si è espresso in merito al passaggio dal web 1.0 al 2.0 in questo modo: “[...] Il web 1.0 era già connettere le persone. E io penso che il web 2.0 sia di fatto solo un’espressione gergale, nessuno sa neanche cosa significhi. Se il web 2.0 per voi sono blog e wiki, allora questo consiste in persone che parlano a persone (people to people). Ma questo è ciò che il web si supponeva che fosse fin dall’inizio. E infatti, vedete che questo cosiddetto web 2.0 significa utilizzare gli standard che sono stati prodotti da coloro che lavorano sul web 1.0. [...] Il web 2.0 non è nulla di nuovo, [...] Tutte le componenti del cosiddetto web 2.0 c’erano già alla nascita del web, e quindi possiamo piuttosto parlare di una sua naturale evoluzione”.

Nel 2006 si è cominciato a parlare di web 3.0, l’epoca del “read-write-execute web”. Le parole d’ordine, qui, sono dati e semantica. La differenza di questa era, rispetto alle precedenti, non è data dall’introduzione di nuove tecnologie, ma da differenti fattori, quali:

- la rete come enorme database: l’introduzione prima degli RSS e dei file XML, successivamente dei rich snippet e dei meta data, offre la possibilità di utilizzare Internet come appunto un enorme database. Si parla quindi di Data Web;
- le intelligenze artificiali: al giorno d’oggi si hanno diversi esempi di programmi di questo tipo. Uno di questi sono gli algoritmi di Google, che analizzano la rete per comprendere come posizionare i contenuti a seconda della qualità e della pertinenza, in base a determinate parole chiave;
- il web semantico: caratterizzato da motori di ricerca capaci di comprendere il “significato”;
- un web adattabile ai diversi dispositivi: il Responsive Web Design e la grafica vettoriale correlata permettono di poter visualizzare su dispositivi diversi i contenuti presenti in rete.
- ...

Dopo il web 3.0 si sta già iniziando a parlare di Web 4.0. Per adesso non si ha ancora una definizione certa, ma alcune delle parole chiave sono “spazio” e “big data”. I fattori che porteranno a questa evoluzione potrebbero essere [29]:

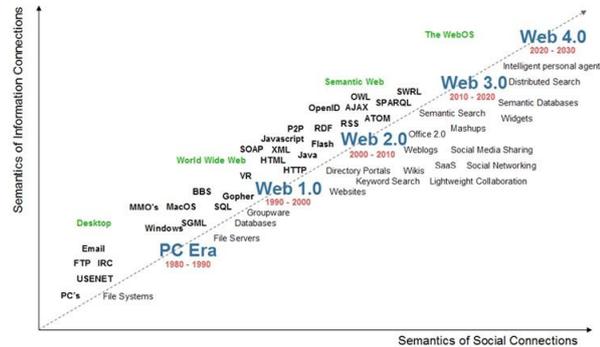


Figura 1.1: L'evoluzione del web, from [5]

- la realtà aumentata dei Google Glasses (occhiali di Google per la realtà aumentata in fase di sviluppo) o gli smartwatch, i quali forniscono un'interfaccia veloce di comunicazione con il proprio smartphone. Ci permetteranno in futuro (e in alcuni casi già oggi) di interagire in tempo reale con il web, sovrapponendo il mondo che ci circonda con la rete.
- creazione di un alter ego digitale. Pian piano che i nostri documenti si aggiornano e collegano fra loro, e man mano che popoliamo la rete con i nostri contenuti personali, andremo a creare un vero e proprio alter ego virtuale. Esso ci permetterà di far interagire le due identità (quella reale e quella digitale) in tempo reale.
- le nuove interfacce: la domotica, che si sta diffondendo negli elettrodomestici e nelle nuove automobili intelligenti, permetterà di scambiare i dati relativi al mondo reale con il nostro alter ego digitale. La trasmissione dei dati è possibile grazie ad internet e ai dispositivi elettronici che ci circondano.

1.2 Introduzione al web semantico

Lo scenario attuale del World Wide Web è quello di un enorme insieme di testi collegati tra loro, e una delle sue caratteristiche più importanti che ne sta alla base è l'universalità. Il potere di un link ipertestuale deriva dal fatto

che qualunque cosa può essere collegata a qualunque altra da chiunque. Il successo del web lo si deve proprio a questi collegamenti ipertestuali, che permettono di accedere e saltare da un documento all'altro. Questo stesso pregio però rappresenta anche un limite, poiché è possibile “linkare” solo a un documento, senza la presenza di un'informazione semantica a ciò che questi documenti contengono.

Inoltre i testi sono creati ad uso e consumo dei soli utenti umani, gli unici attualmente in grado di comprendere i contenuti delle pagine che stanno visitando [21].

Gli utenti infatti utilizzano la loro esperienza di navigazione e la capacità di riconoscere parole ed espressioni chiave. L'esperienza è un aspetto molto importante di cui tutti noi ci serviamo: si impara per esempio che determinati contenuti possono essere reperiti sotto certi portali e che il layout di un sito può dirci qualche cosa sul genere delle informazioni in esso contenute.

Il web è quindi un insieme di risorse e link, e si basa sul linguaggio markup HTML, il quale descrive come le informazioni vengono presentate e connesse, ma non il significato. Di conseguenza, se questo mondo è perfetto per un utente umano, alla domanda “quanta informazione estrae un agente software da una pagina web?” la risposta è sicuramente poca. Solitamente un numero limitato di informazioni infatti sono machine-readable e il significato dei links è riconoscibile solo dal contesto che li circonda.

1.3 Cos'è il web semantico

Il web semantico ha come obiettivo quello di fare in modo che gli esseri umani e i computer cooperino così come fanno tra loro gli esseri umani.

“The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.” – Tim Berners-Lee

È così che l'inventore del World Wide Web, Tim Berners-Lee, definisce il web semantico, *“...un'estensione del web attuale in cui le informazioni sono strutturate con un senso compiuto, migliorando il lavoro tra le persone e i computer”*. [24]

Il termine “web semantico” è quindi stato associato all'idea di un Web nel quale agiscono agenti intelligenti, ovvero delle applicazioni in grado di

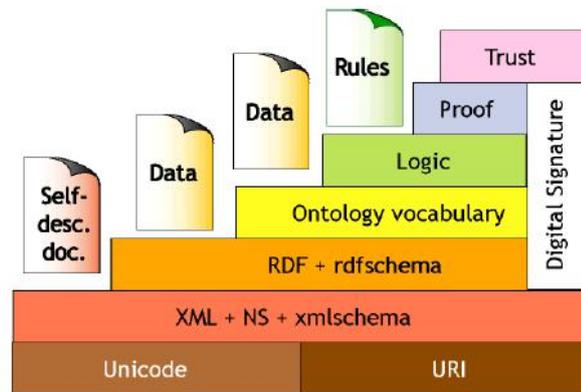


Figura 1.2: Architettura web semantico, from [21]

comprendere il significato dei testi presenti in rete. Grazie a questa caratteristica, gli agenti sono in grado di guidare l'utente direttamente verso l'informazione ricercata, oppure di sostituirsi a lui nello svolgimento di alcune operazioni.

1.3.1 Architettura del web semantico

Il Semantic Web è un'estensione del Web esistente (un "Web sintattico") in cui la semantica viene aggiunta alle risorse.

Come mostrato in figura 1.2 il web semantico è caratterizzato da un'architettura a livelli dove ogni livello fornisce un certo insieme di funzionalità. I livelli più bassi sono costituiti da dati e metadati e forniscono una rappresentazione standard per le informazioni, in modo tale che i dati possano essere facilmente scambiati tra sistemi eterogenei e applicazioni.

- UNICOD: un sistema di codifica che assegna una combinazione di bit ad ogni carattere in maniera indipendente dal programma, dalla piattaforma e dalla lingua;
- URI: fornisce un mezzo per identificare in maniera univoca risorse astratte o fisiche, come pagine Web o contenuti multimediali [41]
- XML (eXtensible Markup Language): sintassi standard per la rappresentazione delle informazioni nel Web. Questo linguaggio consente di

strutturare i dati attraverso tag definiti dall'utente[56], e fornisce un insieme di regole sintattiche per modellare la struttura dei documenti e dei dati. XML Schema fornisce un metodo per comporre vocabolari XML. Un Namespace è un insieme di nomi di elementi e/o attributi identificati in modo univoco da un identificatore;

- Resource Description Framework (RDF): descrive le informazioni contenute in una risorsa Web fornendo metodi non ambigui per esprimerne la semantica. [43] RDF Schema (RDFS) consente di definire i vocabolari usati in RDF [25].

Nei livelli più alti dell'architettura si trovano:

- Ontologie: mezzo per esprimere concetti di un certo dominio e le relazioni tra i concetti, specificando anche vincoli complessi sui tipi di risorse e le loro proprietà. Le ontologie sono importantissime, costituiscono infatti il fulcro del Web semantico. Il linguaggio di ontologia più popolare è OWL ed è un'estensione di RDFS. OWL ha tre sottolinguaggi: OWL Lite, OWL DL e OWL Full [65]. I linguaggi delle regole consentono di scrivere regole di inferenza in un modo standard per effettuare reasoning in un particolare dominio. Tra questi linguaggi troviamo RuleML e SWRL (Semantic Web Rule Language). SPARQL è un linguaggio di interrogazione standardizzato per dati RDF, fornisce sia un protocollo che un linguaggio per interrogare i grafi RDF tramite pattern matching.

Le tecnologie dei livelli dell'architettura appena descritti sono alla base del processo di rappresentazione della conoscenza. I livelli successivi invece sono caratterizzati da tecnologie che sono tuttora in evoluzione. Sono infatti necessari ulteriori progressi per sviluppare soluzioni complete per valutare e garantire l'affidabilità, la sicurezza e la privacy dei contenuti.

- La logica del primo ordine e descrittiva sono utilizzate per supportare il sistema di reasoning, in grado di fare inferenze ed estrarre nuove conoscenze basate sul contenuto della risorsa, affidandosi a una o più ontologie. L'obiettivo è quindi quello di realizzare inferenze tra molteplici dati al fine di fornire all'utente le effettive informazioni ricercate;

- Trust e proof: questi layers vengono utilizzati per verificare l'attendibilità e la validità di una certa risorsa;
- Firma digitale: garantisce l'autenticità delle asserzioni e permette di scoprirne la provenienza. In questo modo chi pubblica materiale sul Web se ne assume le responsabilità.

1.3.2 Ontologie

Il Web è intrinsecamente distribuito, occorre quindi un linguaggio che non solo permetta di esprimere dati e regole sui dati, ma che consenta anche di esportare questa conoscenza (ontologia) per renderla disponibile a qualunque applicazione. Il W3C ha definito, per questa esigenza, il Web Ontology Language (OWL). Infatti, per poter effettuare reasoning, per definire le classi e per altre esigenze, RDF Schema da solo non è sufficiente: occorre quindi un modo per rappresentare la conoscenza e delle regole che permettano di dedurre ulteriore conoscenza, le ontologie. [68] Partendo dal concetto di ontologia, a seguito verrà spiegato il web ontology language.

Tim Berners-Lee nel suo articolo [24], evidenziava già all'ora un problema, ovvero la necessità di confrontare informazioni contenute in due database che, però, utilizzano identificatori diversi per uno stesso concetto.

Si pensi, per esempio, all'indirizzo della residenza di una persona: un database può memorizzare i dati riguardanti via, numero civico, CAP, città, ecc. in campi separati, mentre un altro database può memorizzare il tutto in un unico attributo. Le ontologie, ipotizza ancora Berners Lee, dovrebbero porre rimedio a questo problema.

Un'altra criticità che si può riscontrare è l'ambiguità del linguaggio naturale, come nelle ricerche di informazione in rete. La ricerca di un termine, quale ad esempio "Jaguar", effettuata tramite un qualsiasi motore di ricerca, produrrebbe come risultati un'enorme quantità di dati, che possono spaziare dalla macchina all'animale. Se le pagine web si riferissero alle ontologie, la ricerca potrebbe essere semplificata. Specificando infatti il contesto dove è utilizzata la parola "jaguar", i risultati risulterebbero essere più precisi.

Sottolinea ancora Berners-Lee che "la vera potenza delle ontologie si otterrà quando i computer le utilizzeranno in maniera sofisticata per correlare le informazioni contenute in una pagina con le relative strutture di conoscenza e regole di deduzione".

Così facendo si potrebbero reperire informazioni che non sono contenute in una singola pagina, ma in più pagine anche non collegate sintatticamente tra loro [24].

Dando una definizione filosofica, un'ontologia è un'area della metafisica che studia come è realmente fatto l'universo che ci circonda attraverso lo studio dell'essere, le categorie fondamentali e le relazioni fra esse. Da un punto di vista informatico prendiamo alcune definizioni:

- Neches ('91) "L'insieme dei termini basilari e delle relazioni, che costituiscono il vocabolario di un'area specifica, e delle regole per combinare termini e relazioni per determinare estensioni del vocabolario"
- Gruber ('93) "L'ontologia è una specificazione esplicita di una concettualizzazione"
- Borst, Gruber ('97) "L'ontologia è una specificazione formale ed esplicita di una concettualizzazione condivisa"
- Guarino ('97) "Una teoria logica che spiega il significato inteso di un vocabolario formale"
- Swartout ('97) "Un'ontologia è un insieme di termini descrittivi un dominio strutturato in maniera gerarchica che può essere usato come fondamento di una knowledge base"

Un'ontologia deve essere formale e condivisa, cioè organizzata in modo preciso affinché sia comprensibile alle macchine, e le informazioni da essa rappresentate devono essere accettate dalla comunità e non il frutto di un'idea o visione di un singolo. Un'altra importante caratteristica delle ontologie è la possibilità di essere incorporate o estese da altre ontologie. Il cambiamento di un'ontologia si ripercuote su ogni altra collegata a quella modificata, e le nuove non devono preoccuparsi di ridefinire concetti già esistenti o di utilizzare esattamente le stesse parole delle altre ontologie per definire lo stesso concetto.

Le ontologie del Web semantico consentono alle macchine di interpretare ed elaborare le informazioni sul Web, fornendo un modello comune che può essere compreso sia dagli esseri umani che dai computer. In questo modo è possibile poter condividere, scambiare e riutilizzare i dati in base ai loro significati previsti.

Riassumendo, l'uso delle ontologie riduce le ambiguità semantiche offrendo una singola risorsa interpretativa, e il contenuto informativo viene reso disponibile per il consumo da parte della macchina.

1.4 I linguaggi del web semantico

Fin dal principio il Web è cresciuto intorno a HTML (HyperText Markup Language), linguaggio che rappresenta lo standard per strutturare i documenti, cosicché i browser possano interpretarli e tradurli uniformemente. Le informazioni per la formattazione nell'HTML sono racchiuse nei markup.

Il WWW è un ambiente in continua evoluzione e per un utente umano può essere difficile trovare ciò di cui ha bisogno: un agente software potrebbe essergli d'aiuto, ma non sarebbe in grado di comprendere ed interagire con i dati, poiché non sono elaborabili essendo espressi in linguaggio naturale. Per risolvere questa incapacità di capire la semantica dei concetti presenti in una pagina Web devono essere utilizzati degli strumenti, i quali devono permettere di associare ad ogni concetto il suo significato. Solo così infatti l'agente software sarà in grado di utilizzare tali concetti.

1.4.1 XML

XML è il primo linguaggio che separa il markup dei contenuti da quello di presentazione: è uno standard dove le informazioni non solo sono formattate per facilitarne il reperimento per un utente umano, ma anche elaborabili in modo più semplice da agenti software. A tale scopo XML permette la creazione di documenti strutturati ma non si occupa della semantica dei contenuti. Con questo linguaggio quindi un programma è in grado di riconoscere i contenuti ma non di attribuire loro un significato. Inoltre non fornisce meccanismi di classificazione o reasoning e dovrà essere quindi affiancato da linguaggi più potenti.

1.4.2 RDF

A supporto del problema precedentemente descritto è nato RDF (Resource Description Framework). RDF è stato sviluppato dal W3C come uno standard per la gestione dei metadati relativi alle risorse nel WWW, con lo scopo di aggiungere semantica formale al Web.

Rdf è basato su tre differenti tipi di dati:

- Uniform Resource Identifier (URI): sono riferimenti utilizzati per identificare le risorse come ad esempio `http://www.w3.org/2000/01/rdf-schemaClass`. La parte che segue il simbolo “#” è detta Fragment Identifier [62] Dal momento che gli URI possono condividere prefissi comuni, la notazione QName (Qualified Name) è spesso utilizzata per rendere più breve e leggibile il documento: si definisce un elemento prefisso che sostituisce un namespace, quindi la radice dell’URI (la parte precedente al simbolo “#”) è rimpiazzata con una stringa seguita da “:”. Ad esempio, nell’URI prima indicato, la parte a sinistra del “#” può essere sostituita con “rdfs:”, e il risultato sarà “rdfs:Class”.
- Literal: si tratta di una stringa di caratteri e numeri delimitata da apici, la quale rappresenta un valore concreto. La stringa può essere seguita da una parte opzionale, la quale specifica il tipo di dato contenuto: stringa, intero o data;
- Blank Nodes: sono utilizzati per rappresentare concetti che non sono noti o non specificati. Essi definiscono dunque risorse anonime che non sono identificate da un URI: il prefisso in questo caso è espresso tramite la notazione “_:”. Per la definizione stessa di blank node, esso non può essere utilizzato per identificare globalmente una risorsa ma solo all’interno del documento RDF in cui è specificato. [67]

RDF è un mezzo per descrivere le relazioni tra le risorse in termini di proprietà e valori. Esso descrive le risorse per mezzo di triple, le quali hanno la forma soggetto, predicato, oggetto e forniscono il modo di fare affermazioni sulle cose.

- Soggetto: esso indica la risorsa a cui l’oggetto si riferisce. Per evitare ambiguità, il soggetto è un IRI (Internationalized Resource Identifier) (ad esempio `http://example.org/titoloLibro`) o un blank-node;
- Predicato: il predicato codifica il significato semantico del rapporto tra il soggetto e l’oggetto ed è rappresentato da un IRI (per esempio `http://example.org/Autore`);
- Oggetto: l’oggetto indica la risorsa associata al soggetto tramite il predicato. Questa risorsa può essere un valore (es. Mario Rossi), un IRI (es. `http://example.org/MarioRossi`) o un blank-node. [66]



Figura 1.3: Tripla RDF

Un esempio di tripla è: Leonardo autoreDi Gioconda.

Gli statement sulle risorse sono rappresentabili attraverso un grafo orientato dove i nodi rappresentano le risorse (o i tipi primitivi) e gli archi le proprietà (predicati). Uno statement è rappresentato quindi da un grafo composto da due nodi, soggetto ed oggetto, collegati da un arco orientato dal soggetto all'oggetto rappresentante il predicato, come mostrato in figura 1.3.

1.4.3 RDFS

RDFS [25] è un linguaggio che permette di definire semplici vocabolari (delle sorti di ontologie) di termini che possono essere usati per costruire statement RDF che siano in accordo con tali ontologie. RDF Schema è un'estensione di RDF, la sintassi infatti è la stessa. Questo strumento, proposto da W3C, permette di descrivere relazioni ben definite tra classi e proprietà in una struttura gerarchica.

Ogni documento RDFS viene identificato da un URI, utilizzabile come namespace XML. Questo meccanismo permette il riutilizzo e l'estensione di vocabolari RDFS. I Costrutti principali di RDFS sono Class e Property.

- **Class:** è utilizzato per indicare che una risorsa ne contiene altre (es. Persona). È possibile anche creare gerarchie di classi tramite il predicato "subClassOf" (es. Artista, subClassOf, Persona). In RDFS una classe C è definita dalla seguente tripla: C rdfs:type rdfs:Class. Per dire ad esempio che la classe autore è sottoclasse di persona, ex:Autore rdfs:subClassOf ex:Person
- **Property:** vengono usate per descrivere la relazione tra un soggetto e un oggetto. Una property può essere dichiarata con un dominio e un range. Per esempio la Property "èScrittoDa" può avere come

soggetto (il dominio) una risorsa del tipo “Libro” e come oggetto (il range) una risorsa del tipo “Person”. Anche le Property possono essere organizzate gerarchicamente.

1.4.4 OWL

Nell’ambito del Semantic Web, il W3C ha sostenuto lo sviluppo di OWL, il linguaggio per la creazione di ontologie più comune. Si tratta di un linguaggio di markup che permette di rappresentare esplicitamente significato e semantica dei termini attraverso l’uso vocabolari e relazioni tra termini. Rispetto a RDFS, OWL permette di:

- avere costrutti più complessi, come vincoli di cardinalità, definizione di sinonimi e relazioni tra classi;
- modellare la conoscenza del dominio di interesse e non la struttura con la quale essa viene codificata nel documento.

Vi sono tre tipi di OWL:

1. OWL Lite: consente la costruzione di gerarchie e semplici regole.
2. OWL DL (Description Logic): questa versione supporta la massima espressività, garantendo la decidibilità e la completezza computazionale (ogni conclusione può essere computata). OWL DL include tutti i costrutti di OWL con alcune restrizioni (per esempio una classe non può essere l’istanza di un’altra classe);
3. OWL Full: fornisce la massima espressività e la libertà sintattica di RDF ma non fornisce garanzie in fase di computazione.

Le ontologie permettono quindi di creare un modello preciso della realtà presa in esame e, tramite RDF/S, di creare reti di dati collegati tra loro da legami logici e ben definiti, i quali possono essere sfruttati per interrogare efficacemente questa enorme base di dati.

1.4.5 SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) è un linguaggio che permette di effettuare interrogazioni sulle informazioni contenute nei

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX RDF:< <http://www.w3.org/1999/02/22-rdf-syntax-ns#>>
SELECT ?name ?surname ?email
WHERE {
    ?person rdf:type foaf:Person .
    ?person foaf:surname ?surname .
    ?person foaf:name ?name .
    ?person foaf:mbox ?email .
}
```

Figura 1.4: Esempio query SPARQL

documenti RDF. È stato standardizzato dal “RDF Data Access Working Group (DAWG)” della W3C ed è considerato una tecnologia chiave del Web Semantico.

Sia SPARQL che SQL sono linguaggi dichiarativi utilizzati per effettuare interrogazioni. SQL è usato per ricercare informazioni memorizzate in forma tabellare all’interno di database relazionali: attraverso questo linguaggio è possibile selezionare all’interno di una tabella una specifica riga e i valori contenuti nelle sue colonne. In un contesto come il web però, dove è possibile che le stesse informazioni siano memorizzate in maniera differente, non è pensabile utilizzare le stesse regole per ottenere informazioni.

SPARQL può essere utilizzato per esprimere query in maniera trasversale tra diverse sorgenti dati, sia che essi siano memorizzati nel formato RDF o che siano visti come RDF grazie all’aiuto di “middleware”, cioè di applicazioni che si interpongono tra la sorgente di dati e l’applicazione che li utilizza. SPARQL ha la capacità di effettuare query su vari modelli di grafo con operazioni quali l’unione. I risultati delle interrogazioni SPARQL possono risultare essere insiemi di dati o grafi RDF. È possibile anche effettuare dei test sui valori ed operazioni di conversione di graph pattern. SPARQL supporta anche il test dei vincoli sul grafo RDF sorgente.

Nella figura 1.4 è mostrato un esempio di query SPARQL effettuata sull’ontologia FOAF. Essa restituisce nome, cognome e indirizzo e-mail di ogni persona. Se una persona ha più di una mail vi saranno più righe con lo stesso nome, ognuna con un indirizzo diverso. Il risultato è possibile vederlo nella tabella che segue.

Risultato query SPARQL		
n	ame	surname
email		
mario	rossi	mariorossi@gmail.com
claudio	verdib	claudioverdi@hotmail.com

Uno dei vantaggi dell'utilizzo di SPARQL deriva dal fatto che esso permette agli utenti di scrivere query prive di ambiguità, in quanto ogni identificatore (URI) in SPARQL è globalmente non ambiguo. I grandi dataset solitamente offrono un endpoint SPARQL, usando tale protocollo: un esempio è l'endpoint SPARQL di Dbpedia [28].

Capitolo 2

Internet of Things

2.1 Cos'è l'Internet of Things

Esistono diverse definizioni di Internet of Things (IOT). Una tra queste è quella fornita dal National Intelligence Council statunitense: *"L'Internet of Things è l'idea generale delle cose, in particolare oggetti quotidiani, che sono leggibili, riconoscibili, localizzabili, indirizzabili e controllabile via Internet - tramite RFID, wireless Local Area Network, Wide-Area Network, o altri mezzi"*.

Il concetto IOT è molto più ampio rispetto ai classici dispositivi collegati a Internet come computer, laptop, smartphone e tablet (ad es. iPads, ecc.). In particolare, oggetti quotidiani che prima non sembravano affatto elettronici, stanno iniziando ad essere connessi con sensori e microprocessori integrati, comunicando tra loro e attraverso Internet [54]. Il termine Internet of Things (IoT) indica quindi una combinazione di software e hardware che produce dati attraverso la connessione di più dispositivi e sensori. Si tratta di una vera e propria rete di oggetti interconnessi individuati in modo univoco, i quali sono in grado di comunicare tra loro e di interagire con il mondo reale. In questo modo l'IOT permette di avere un collegamento tra il mondo virtuale e quello reale: qualsiasi cosa può essere un dispositivo IoT se può trasmettere e ricevere dati attraverso la rete.

Il numero di dispositivi su Internet è in continuo aumento. Sta emergendo un ecosistema di Internet of Things (IOT) ad ampio raggio per supportare il processo di connessione agli oggetti del mondo reale ad internet. Per oggetti del mondo reale si intendono ad esempio edifici, strade, elet-

trodomestici, o addirittura corpi umani attraverso l'uso di sensori e chip a microprocessore che registrano e trasmettono dati quali onde sonore, temperatura, movimento e altre variabili. L'esplosione dei sensori connessi a Internet implica la creazione di nuove classi di capacità tecniche e applicazioni. Si stanno sviluppando nuovi "comportamenti" sui dati, come la valutazione della correlazione, il rilevamento delle anomalie e l'elaborazione dei dati ad alta frequenza. Questo sviluppo aumenta e continuerà ad aumentare man mano che gli umani si adatteranno ai diversi tipi di flussi di dati dell'IOT. Le tre categorie principali in cui si è diffuso l'IOT sono: monitoraggio e controllo delle prestazioni di case e edifici, applicazioni automobilistiche e di trasporto, monitoraggio della salute e monitoraggio dell'ambiente personale. [54]

Internet è in continua evoluzione, e, dal solo collegamento tra persone e computer, oggi ci si sta muovendo anche verso la connessione tra cose e oggetti. Questo è reso possibile grazie ad una tecnologia che è in continua evoluzione, come la connettività Internet a banda larga, la quale sta diventando sempre più economica e onnipresente, o la dimensione dei dispositivi, che sta diventando sempre più ridotta.

Questi device prendono il nome di smart object o smart things: essi, a differenza dei normali dispositivi, hanno un ruolo attivo in quanto interagiscono all'interno del sistema in cui si trovano. Per "cosa" o "oggetto" si intendono categorie di apparecchiature che possono andare da semplici dispositivi a complessi impianti e sistemi. Il fatto che tutti questi oggetti siano o possano essere collegati in rete permette di creare una mappa intelligente di tutte le cose, del loro funzionamento e delle informazioni che sono in grado di rilevare e di trasmettere, creando nuove forme di conoscenza. [11]

Seguono alcune caratteristiche di questi device:

- sono degli oggetti caratterizzati da costo, forma, peso.;
- hanno risorse limitate in termini di capacità computazionale, memoria, approvvigionamento energetico e routing;
- sono identificati univocamente attraverso un ID e possono individuare altri dispositivi nella rete e a loro volta essere individuati. Ad essi è associato anche un nome, in modo tale che l'uomo lo possa riconoscere;

- possono essere influenzati e influenzare la realtà che li circonda (per esempio attraverso gli attuatori).

2.1.1 Architettura dell'Internet of Things

L'architettura dell'Internet of Things può essere suddivisa su quattro livelli: ognuno di essi è reciprocamente indispensabile e prevede l'utilizzo di certe tecnologie. I quattro livelli che si possono identificare sono [42]:

1. Object sensing and information gathering: livello riguardante le modalità con cui i dispositivi diventano intelligenti (smart) e quindi capaci di collezionare informazioni sull'ambiente o su altri dispositivi connessi. La tecnologia più utilizzata è Radio-Frequency Identification, l'RFID, la quale permette di identificare e memorizzare automaticamente le informazioni riguardanti oggetti, animali o persone. Essa si basa sull'utilizzo di TAG, etichette elettroniche, e sulla loro capacità di rispondere all'interrogazione a distanza da parte di appositi reader, apparecchi che possono essere fissi o portatili;
2. Information delivering: l'informazione prodotta dalle tecnologie dei dispositivi deve essere comunicata all'esterno. Le tecnologie più utilizzate per questo livello sono quelle senza fili, per esempio: il Wifi, la più utilizzata, il Bluetooth, wireless sensor network (WSNs), body area network (BANs), il GPRS e la linea mobile cellulare;
3. Information processing and handling: l'informazione creata dai sensori che si trovano sui dispositivi e che viene condivisa dalle reti deve essere immagazzinata e analizzata. Una tecnologia importantissima per la gestione dei dati è il cloud computing. Con questo termine si indica un insieme di tecnologie che permettono l'elaborazione e l'archiviazione dei dati attraverso l'utilizzo di risorse (hardware e software) distribuite in rete;
4. Application and services: con il passare del tempo le prestazioni della rete sono migliorate, sia in termini di utilizzo della larghezza di banda e capacità di calcolo che da un punto di vista di efficienza energetica. I software che permettono la realizzazione di interfacce utente sono un esempio di applicazioni front-end per l'utilizzo dei dispositivi.

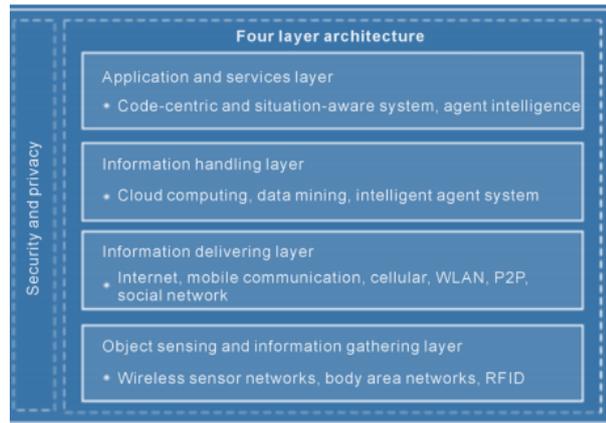


Figura 2.1: Architettura IoT, fonte Chen Min

Gli ostacoli principali che devono essere superati per una completa realizzazione dell'Internet of Things sono la privacy e la sicurezza. Come si può vedere anche dalla figura 2.2 rappresentante i vari strati, questi due fattori sono comuni a tutti i livelli dell'architettura. Questa problematica verrà dettagliata nei paragrafi successivi.

2.1.2 Criticità dell'IoT

Il numero di dispositivi IoT è in continuo aumento ed anche il loro utilizzo da parte degli utenti. Nonostante il grande successo, oggi l'Internet of Things si trova nella sua fase iniziale. Vi sono infatti ancora degli aspetti che devono essere risolti quali scalabilità, eterogeneità di diversi dispositivi, sicurezza e altro. Barnaghi evidenzia quattro problemi di interoperabilità presenti nell'Internet of [50]:

1. l'interoperabilità tecnica comporta l'eterogeneità dei componenti hardware e software e dei relativi protocolli di comunicazione;
2. l'interoperabilità sintattica coinvolge i formati di dati e la rappresentazione dei dati. Essa è fondamentale per interpretare i dati dell'Internet of Things e costruire sistemi intelligenti;

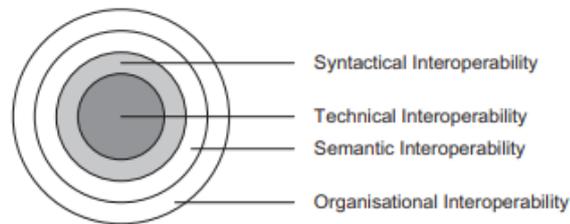


Figura 2.2: Architettura IoT

3. l'interoperabilità semantica implica l'interpretazione del significato dei dati scambiati;
4. l'interoperabilità organizzativa coinvolge l'eterogeneità delle diverse infrastrutture, e dipende dall'interoperabilità tecnica, sintattica e semantica.

I dati raccolti dai sensori presenti nei dispositivi, affinché siano utili, devono essere analizzati e compresi. Prendendo come esempio il settore del fitness, ci si trova di fronte a un numero elevato di dispositivi provenienti da fornitori diversi, che raccolgono lo stesso tipo di dati ma li memorizzano e li scambiano in modi diversi, portando inevitabilmente a problemi sintattici e semantici quando li si vuole unire o confrontare. Le tecnologie web semantiche sono strumenti che sono in grado di risolvere questo problema.

2.1.3 Ambiti di utilizzo dell'Internet of Things

Il frigorifero di casa, l'orologio, il semaforo, le automobili, i *wereables*, potrebbero essere considerati tutti come casi dell'Internet of Things. Caratteristica essenziale affinché siano considerati tali, è che questi oggetti siano connessi alla rete, con la possibilità di trasmettere e ricevere dati [42]. Vediamo in dettaglio alcuni esempi:

- una videocamera non ha più solo la capacità di inviare dati e immagini, ma di farlo in modo intelligente, in funzione ad esempio di quello che riprende. In questo modo è in grado di modificare il proprio comportamento in base ai parametri di interesse che possono evolvere nel corso del tempo, adattandosi alle esigenze;

- l'orologio può ricordarci appuntamenti e verificare se li rispettiamo, controllando se siamo o meno in un certo luogo a una certa ora o addirittura se siamo in ritardo;
- le confezioni di prodotti alimentari possono fornire importanti informazioni sulla qualità del prodotto, sul modo in cui è stato realizzato e su tutti coloro che hanno partecipato alla produzione.
- la confezione di un farmaco ci può avvertire se non lo stiamo assumendo come previsto, se ci stiamo dimenticando di prenderlo o come risolvere una dimenticanza;
- automobili che possono dialogare continuamente con l'ambiente circostante e possono facilitare la guida, aumentando comodità e sicurezza;
- ...

Altri temi interessanti nell'ambito dell'Internet of Things sono [11]:

- casa, smart home, domotica;
- edifici intelligenti, smart building, bulding automation;
- monitoraggio in ambito industriale, Robotica, Robotica collaborativa;
- industria automobilistica, automotive, self driving car;
- smart health, sanità, mondo biomedicale;
- tutti gli ambiti della sorveglianza e della sicurezza;
- smart city, smart mobility;
- nuove forme di digital payment tramite oggetti;
- zootecnia, wearable per animali;
- ...

Nei paragrafi che seguono vediamo l'IoT applicato in ambito medico, e, in particolare nei wereables, nel nel settore del fitness.

2.1.4 Applicazioni IoT nel settore medico

Un ambito importante dove l'IoT è utilizzato è quello medico. Wearables come gli smart glass, hanno trovato ampio utilizzo anche in questo settore. In Italia ad esempio sono stati utilizzati dall'Istituto di ricovero e cura Humanitas: i dispositivi venivano utilizzati per rendere partecipi in tempo reale l'equipe medica durante un'operazione chirurgica, permettendo di chiarire dubbi insorgenti rimanendo concentrati sull'operazione senza levare lo sguardo dal lavoro che si stava compiendo [32]. I wearables sono usati anche per monitorare la fase post-operatoria del paziente, cosicché il medico lo possa controllare in tempo reale, essendo aggiornato in modo automatico sul suo stato di salute. Un esempio è la Galvanic skin resistance, GSR, tecnica utilizzata per monitorare la variazione di resistenza elettrica della pelle del paziente provocata da stimoli emozionali. Tale variazione deriva dalla quantità di umidità prodotta dalle ghiandole sudoripare delle dita. Vi sono due tipi di attività misurabili attraverso la GSR:

- l'attività tonica: essa esprime un indice di attivazione del sistema nervoso dell'organismo. Se il valore è alto l'individuo è in una situazione di rilassatezza, se invece si abbassa, la sudorazione aumenta e il paziente sarà in uno stato di agitazione.
- l'attività fasica: si tratta di feedback provocati da emozioni che il paziente prova durante la fase di riabilitazione.

Gli strumenti più moderni permettono di misurare contemporaneamente sia l'attività tonica che quella fasica, mostrandole su un display. In campo clinico il GSR fasico permette per esempio di compilare in un primo tempo una gerarchia di situazioni-stimolo nei soggetti fobici, e di somministrare il feedback al fine di ridurre la risposta fobica in presenza dello stimolo. [26]. Per un monitoraggio più accurato in fase post-operatoria, i pazienti possono indossare anche dispositivi in grado di misurare il battito cardiaco, la pressione del sangue o la frequenza respiratoria. L'affidabilità di questi dispositivi deriva però in gran parte dall'attenzione che il paziente ha nel seguire le istruzioni di utilizzo indicate dal medico. Infatti, attraverso un utilizzo corretto e costante dei device da parte del paziente, il lavoro del medico può



Figura 2.3: IoT nel settore medico, fonte [19]

essere notevolmente agevolato, al fine di poter individuare la miglior cura [12].

2.2 Wreables

I wereables stanno diventando i dispositivi più personali per gli utilizzatori. Due su cinque utenti proprietari di dispositivi indossabili dicono di sentirsi nudi quando non lo indossano, e un quarto dorme addirittura con esso [10]. Nell'ambito IoT quindi i dispositivi indossabili stanno spopolando a vista d'occhio. Oggi, grazie al loro utilizzo, si ha la possibilità di monitorare la realtà che ci circonda e le informazioni che ne derivano. Sono tantissimi i settori di applicazione di questa tecnologia, dal fitness, wellness, al sanitario sino addirittura alla moda. Attualmente le aziende tecnologiche stanno sviluppando tecnologie sempre più innovative e alla portata di tutti per espandere il mercato. I dispositivi indossabili orientati alla salute e al fitness riescono addirittura a rilevare misure biometriche come la frequenza cardiaca, i livelli di sudorazione e misurazioni complesse come i livelli di ossigeno nel sangue. I progressi tecnologici possono consentire addirittura di effettuare la rilevazione dei livelli di alcol o altre misurazioni simili. La capacità di rilevare, memorizzare e monitorare le misure biometriche nel tempo

e quindi analizzare i risultati, è solo una possibilità interessante. Monitorare la temperatura corporea, ad esempio, potrebbe fornire un'indicazione precoce di un raffreddore o di un'influenza. [14]

2.2.1 Wereables per Fitness Sport e Benessere

Uno degli utilizzi principali dei wereables è relativo al monitoraggio del corpo durante le attività sportive, e in generale al wellness, ovvero alla cura del benessere dell'utente. Nel mercato dei dispositivi IoT, device fitness indossabili come Fitbit o Apple Watch sono sempre più venduti e utilizzati. Questi dispositivi, come vedremo meglio successivamente, oltre alle esigenze richieste dagli utenti, quali velocità media o calorie bruciate, offrono delle funzionalità che permettono di tenere traccia dello stato salute del proprietario. In particolare, i wereables sono sempre più utilizzati dalle persone, sportivi e non: grazie alla loro comodità e praticità gli utenti li indossano durante quasi tutto l'arco della giornata. Di conseguenza, raccogliendo tutte queste informazioni, i wereables possono essere visti anche come un ottimo strumento per gestire e prevenire alcune malattie.

I wereable e le applicazioni relative alla salute presenti ormai su tutti gli smartphone, come Salute di iOS, stanno pian piano cambiando addirittura l'assistenza sanitaria, permettendo agli utenti di tenere traccia e di controllare la propria condizione fisica, sia in termini di allenamento che di stato di salute.

I componenti elettronici dei dispositivi con il passare del tempo sono diventati di dimensioni sempre più piccole. Ciò ha permesso alla tecnologia di essere utilizzata in modo sempre più semplice in generale nello sport ma anche nel monitoraggio dei parametri vitali durante lo svolgimento di un'attività fisica (come il battito cardiaco). Per rilevare queste informazioni i dispositivi indossabili sono dotati di sensori: in questo modo, gli utilizzatori saranno in grado di monitorare in modo autonomo le proprie prestazioni di allenamento.

Tra i wereables più utilizzati in ambito sportivo si possono trovare:

- fasce toraciche: esse permettono di controllare la frequenza del battito cardiaco durante l'attività sportiva;
- occhiali: consentono a chi sta facendo un'attività all'aperto (per esempio bici o corsa) di esplorare nuovi percorsi avendo sempre in sovrim-

pressione il percorso che si vuole seguire, rimanendo concentrati sull'attività che si sta svolgendo in modo da evitare pericoli che possono presentarsi, come le buche. Un esempio possono essere i Google Glass;

- braccialetti: permettono di ricavare il numero di passi fatti e la velocità media;
- ...

Il completo monitoraggio della salute dell'utente deriva dal fatto che i dispositivi indossabili controllano il corpo umano non solo quando si esegue un'attività fisica, ma durante tutto l'arco della giornata. Alcune delle informazioni che è possibile ricavare da questi device sono la quantità di calorie bruciate, il numero passi effettuati, o la velocità media. I dati ottenuti risultano essere anche abbastanza precisi. Esistono dei dispositivi che permettono addirittura di scandire i cicli del sonno di una persona, permettendo di creare un vero e proprio storico dell'utente, al fine di fornire dei consigli per migliorare le proprie abitudini e indicare il momento più opportuno per svegliarsi [16]. Nei sottoparagrafi successivi saranno descritti più in dettaglio alcuni dei *wereables* più diffusi.

Smartwatch

Per *smartwatch*, o orologio intelligente, si intende un orologio con funzionalità che vanno oltre al semplice cronometraggio. Mentre i primi modelli erano in grado di eseguire solo operazioni di base, come ad esempio calcoli, traduzioni, e giochi, gli orologi intelligenti più moderni sono dei veri e propri computer indossabili. Gli *smartwatch* moderni sono dotati di sensori quali microfoni, sensori di luce ambientale, sistemi GPS. Inoltre, grazie al continuo contatto sulla pelle, questo dispositivo è in grado di monitorare con precisione, oltre alle attività fisiche dell'utente, anche alcuni parametri vitali come la temperatura del corpo.

Nel settore del fitness e del wellness uno dei prodotti più utilizzati e venduti è l'Apple Watch: esso è dotato del sistema operativo WatchOS, il quale si interfaccia attraverso l'applicazione Salute di iOS trasmettendogli i dati dell'utente dell'intera giornata e dell'attività sportiva. I dati sono resi disponibili per l'utente stesso, ma anche in forma anonima, con lo scopo di aiutare studi scientifici nella prevenzione di malattie. A tal proposito Apple ha creato ResearchKit, una piattaforma open source attraverso la



Figura 2.4: Smartwatch, fonte [10]

quale i ricercatori possono studiare i dati raccolti da Apple. Essi osservano i comportamenti degli utenti e il conseguirsi dello sviluppo di certe malattie. Un altro dispositivo noto e molto diffuso è Android Wear: le funzionalità sono analoghe a quelle di Apple Watch. Esso può essere sincronizzato sia con dispositivi iOS che Android e comunica con la piattaforma Google Fit.

Oggi in molti stanno iniziando a investire anche sull'abbigliamento smart per lo sport, come Under Armour. Under Armour è un'azienda produttrice di abbigliamento sportivo che ha acquistato di recente due piattaforme per realizzare indumenti che permettano di tenere traccia dei movimenti dell'utente e diano dei consigli su come raggiungere gli obiettivi prefissati [16].

Vediamo ora le caratteristiche di alcuni orologi:

- Pebble watch: fornisce applicazioni connesse a Internet come la notifica di chiamate in arrivo, e-mail e messaggi di avviso tramite Bluetooth per connettersi agli smartphone;
- Basis watch: è un orologio di auto-localizzazione, una piattaforma multi-sensore con accelerometro 3D, cardiofrequenzimetro, sensore di temperatura e sensore GSR. Come Fitbit, non si sincronizza in tempo reale ma solo quando è collegato a un computer;
- Wimm Labs Contour Watch: è concepito per consentire un'ampia gamma di applicazioni tra cui sport, salute, moda, finanza, elettronica di consumo e tante altre. In una possibile estensione del Google Project Glass (occhiali per realtà aumentata), Google ha brevettato la tecnologia smartwatch per uno smartwatch con realtà aumentata con due schermi mobili, un touchpad e connettività wireless.

Come tutti i wearable device anche gli smartwatch hanno alcune limitazioni, quali l'usabilità e l'autonomia delle batterie, causate dalla ridotta dimensione dello schermo e del dispositivo. Inoltre, Kamdar osserva che lo scorretto posizionamento sul polso del dispositivo indossabile è una limitazione degli smartwatch per alcuni tipi di sensori. Per esempio, se l'orologio intelligente non è indossato correttamente, la frequenza cardiaca non potrà essere raccolta quando il sensore non è a diretto contatto con la pelle [40]. Ulteriori limitazioni dei wearables saranno descritte nei paragrafi successivi.

Wristbands

I wristbands sono un predecessore degli smartwatch e rimangono una categoria di prodotti di successo per conto proprio. Essi sono dei dispositivi da polso in grado di raccogliere e inviare dati riguardanti l'indossatore, quali il conteggio dei passi o frequenza cardiaca. Il trasferimento dei dati avviene al server dell'azienda attraverso un gateway. Uno dei primi esempi di wristband utilizzava gli accelerometri per misurare i passi compiuti con prodotti come Nike Sense [54].

Gli accelerometri basati su braccialetti, come per esempio Fitbit, sono uno dei tipi dei wearable più utilizzati oggi. Essendo economici e in grado di rilevare molte delle attività giornaliere essi sono riusciti a spopolare facilmente.

I dati raccolti da tali dispositivi possono:



Figura 2.5: Wristbands, fonte [13]

- essere trasmessi in modalità wireless per un feedback in tempo reale;
- caricati sul cloud.

Alcuni dei più famosi wristbands sono [54]:

- Nike fuelband, per il conteggio dei passi;
- il braccialetto Jawbone UP e l'app per iPhone, tracciamento dei passi, distanza, calorie bruciate, ritmo, livello di intensità, tempo attivo inattivo e inattivo e GPS;
- Adidas MiCoach, che fornisce il monitoraggio della frequenza cardiaca, coaching digitale in tempo reale, allenamento interattivo e analisi post-allenamento di ritmo, distanza e velocità del passo.
- Mio Active aggiunge la frequenza cardiaca, con o senza una fascia toracica;
- LarkLife identifica il tipo di attività, consente il monitoraggio della dieta con un solo pulsante, misura il sonno e utilizza le metriche combinate per fornire raccomandazioni personalizzate sui cambiamenti che un utente può fare per sentirsi meglio;
- il braccialetto e il fermaglio Amiigo misurano anche il tipo di esercizio, oltre alla temperatura corporea e ai livelli di ossigeno nel sangue attraverso un sensore a infrarossi;

- altre piattaforme di sensori si concentrano anche sul fitness e sull'allenamento atletico, ad esempio Somaxis con sensori ECG e EMG per muscoli e cuore e GolfSense, dove gli utenti collegano un sensore da polso a un guanto da golf. L'unità dispone di due accelerometri e altri sensori che raccolgono e trasmettono dati in modalità wireless per un feedback in tempo reale.

In generale quindi, gli Smart Watches e i Fitness-Tracker fanno parte di quella categoria di dispositivi *wearable* che si sono affermati per essere alla portata di tutti. Gli orologi "intelligenti" consentono la connessione con il proprio smartphone permettendo di gestire chiamate, riproduzione musicale, indicazioni stradali, senza togliere il proprio smartphone dalla tasca. I Fitness-Tracker, invece consentono di monitorare i movimenti, le calorie bruciate, i metri percorsi e la qualità del sonno direttamente sul display del bracciale, oppure sullo smartphone [4].

2.2.2 Vantaggi e svantaggi dell'utilizzo quotidiano dei *wearables*

L'utilizzo di *wearables* presenta per gli utilizzatori una serie di vantaggi. Uno di quelli più importanti è l'immediatezza: spesso quando l'utente utilizza uno smartphone, si trova un insieme di applicazioni ma nessuna informazione immediata, a differenza di un *wearable* come può essere uno smartwatch. Ad esempio, un utente può monitorare in tempo reale l'andamento del suo allenamento direttamente sulla lente degli smart glass o sul display dello smartwatch senza dover utilizzare lo smartphone. L'utilizzo dei *wearables* quindi comporta un aumento dell'efficienza, in quanto essi consentono di visualizzare velocemente per esempio un messaggio o una mail, e nel caso rispondere tramite voce. Con uno smartphone invece bisognerebbe estrarlo dalla tasca, sbloccarlo, cercare l'applicazione, e infine cercare l'informazione desiderata dall'utente.

Un altro importante vantaggio è il coinvolgimento dell'utente con l'ambiente esterno. Per esempio, con l'utilizzo di uno smartphone l'utente si concentra esclusivamente su ciò che succede sullo schermo, escludendosi così da ciò che lo circonda. I *wearables* invece riescono a dare delle informazioni utili all'utente che li indossa rimanendo discreti e concentrati su quello che si sta facendo. I dati raccolti dai dispositivi indossabili potranno

essere utilizzati per le ricerche mediche, andando a vantaggio di tutta la popolazione che, automonitorandosi, involontariamente aiuta ricercatori a combattere malattie che da sempre affliggono le persone di tutto il mondo, come ad esempio i disturbi cardiovascolari. Altri vantaggi, magari scontati ma importanti, sono le loro dimensioni ridotte, la loro facilità di utilizzo e le informazioni facilmente reperibili.

Nonostante i numerosi vantaggi i *wereables* presentano anche degli svantaggi. Uno di questi è sicuramente il limite tecnico causato dalle batterie: le ridotte dimensioni infatti obbligano i produttori all'uso di batterie con un numero ridotto di ampere, causando un'autonomia non troppo elevata. In presenza di uno schermo la durata media è di un giorno e mezzo. Per superare questo limite tecnico i produttori stanno ottimizzando i sistemi operativi degli smartwatch e ricorrendo all'utilizzo di display Amoled, che potrebbero aumentare la durata della batteria [3]. Questi dispositivi inoltre contengono informazioni riguardanti il nostro stile di vita e le nostre abitudini, di conseguenza, smarrire un device di questo tipo, potrebbe comportare un problema importante per la privacy dell'utilizzatore. Molti dispositivi inoltre sfruttano GPS e telecamera, tecnologie che tracciano ogni istante della nostra vita: esse infatti sono in grado di indicare in ogni momento il luogo in cui ci troviamo o cosa stiamo facendo. Se qualche mal intenzionato viola i sistemi di sicurezza può avere accesso a tutte queste informazioni. Il problema della privacy è quindi sicuramente uno dei più importanti, di conseguenza, le aziende che utilizzano questi dati sensibili per scopi scientifici o altro, devono indicare nel modo più chiaro possibile le proprie politiche riguardanti la privacy [9].

"Affinchè i wearable contribuiscano a plasmare la New Health Economy, i dispositivi di nuova generazione dovranno essere interoperabili, social e orientati ai risultati", spiega Vaughn Kauffman, direttore, PwC Salute Industries. "I dati dei wearable possono essere utilizzati dagli assicuratori e datori di lavoro per gestire meglio i costi per la salute, il benessere e la sanità e dalle aziende farmaceutiche per eseguire studi clinici più robusti, e dagli operatori sanitari per supportare i risultati basati rimborso. Ma sarà fondamentale per affrontare le preoccupazioni dei consumatori gestire i costi, la privacy, e la facilità d'uso" [20].



Figura 2.6: Il futuro dei wereables, fonte [10]

2.2.3 Il futuro dei wereables

Dai robusti orologi da polso per cellulari della fine degli anni 2000 alle attuali fitness band, la tecnologia indossabile si è notevolmente evoluta negli ultimi anni. Oggi i wearables stanno cambiando il modo in cui i consumatori interagiscono con l'ambiente, e la loro popolarità sta crescendo a vista d'occhio.

Paragonata all'analisi di Ericsson ConsumerLab del 2015, la ricerca attualmente indica che i proprietari di dispositivi indossabili tra gli utenti di smartphone in Brasile, Cina, Corea del Sud, Regno Unito e Stati Uniti sono raddoppiati. Il mercato dei dispositivi indossabili è ancora nelle prime fasi di espansione ed è attualmente dominato da dispositivi di tracciamento di salute, wellness e attività fisica, ma si cercano sviluppi del settore anche a un numero crescente di casi d'uso.

I dispositivi connessi hanno superato le persone connesse nel 2008. Cisco stima che entro il 2020 ci saranno 50 miliardi di dispositivi connessi, 7 volte la popolazione mondiale. Analogamente, l'iniziativa Connected Life sponsorizzata dalla GSMA (GSM Association, un'associazione di settore per

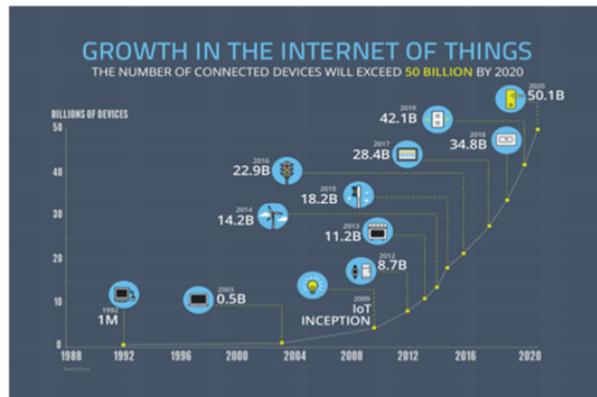


Figura 2.7: Crescita prevista dell'IoT, fonte [15]

gli operatori mobili mondiali) ha rilevato che nel 2011 c'erano 9 miliardi di dispositivi connessi a Internet, di cui due terzi (6 miliardi) erano mobili, e stima che nel 2020 ci saranno 24 miliardi di dispositivi connessi a Internet, 12 miliardi di dispositivi mobili [22]. Inoltre, si stima che questi 24 miliardi di dispositivi connessi a Internet avranno un impatto economico di oltre 4,5 trilioni di dollari nel 2020 [54].

Capitolo 3

Linked Open Data e portali LOD

3.1 Open data

Con il termine “open” si intende che chiunque può liberamente accedere, utilizzare, modificare e condividere per qualsiasi scopo (soggetto, al massimo, a requisiti che ne preservano la provenienza e l’apertura). ” Con il termine Open Data (dati aperti, di libero utilizzo) si indica l’operazione di rendere accessibile tali dati a chiunque, abbattendo, per quanto possibile e ragionevole, le restrizioni tecnologiche ed imponendo vincoli legali minimi sul riutilizzo dei dati. Con open data si intende quindi la pubblicazione di una qualsiasi raccolta di dati in un formato leggibile dalla macchina, senza licenze o restrizioni sui brevetti, in modo che tutti siano liberi di usare, riutilizzare e ridistribuire i dati per qualsiasi scopo. [8] .

I dati normalmente sono parte di una informazione o conoscenza strutturata che può essere codificata e archiviata in un formato digitale. Alcuni esempi di open data sono le informazioni geografiche, formule matematiche, dati medici e biologici, i dati in possesso dalle pubbliche amministrazioni come gli atti ufficiali, e tanto altro. Questi dati presi singolarmente possono avere un certo valore, ma una volta aggregati in modo efficace, sicuramente questo valore aumenterà di molto.

La disponibilità degli Open Data è preziosa sia per la società che per le imprese, ma per valorizzare del tutto l’informazione, l’apertura da sola non basta. È infatti necessario [70]:

1. rendere gli Open Data autodescrittivi e poter inferire conoscenza dall'aggregazione e dalla correlazione di dataset differenti;
2. favorirne la facilità di utilizzo, il reperimento e il consumo sia per gli esseri umani che per i software automatici.

3.1.1 Data.gov, esempio di open data

Tra le sorgenti di Open Data più famose troviamo le pubbliche amministrazioni con l'Open Government [46].

L'Open Government, OGP, è un movimento che prevede la pubblicazione dei dati dei vari governi nazionali. Data.gov è un sito web del governo degli Stati Uniti lanciato a fine maggio 2009 dall'allora Chief Information Officer (CIO) degli USA, Vivek Kundra. Data.gov mira a migliorare l'accesso pubblico a set di dati di elevato valore, leggibili meccanicamente, generati dal ramo esecutivo del governo federale. Il sito è un deposito di informazioni governative federali, statali e locali, messe a disposizione del pubblico. Attraverso questa piattaforma è stata introdotta nel governo federale degli Stati Uniti la filosofia degli open data, un approccio che porterà molti benefici, tra cui "la ricostruzione della fiducia nel governo e negli affari" [53]. Data.gov è passato da 47 dataset al momento del lancio a oltre 180.000.

L'Open Government Partnership è un'iniziativa internazionale che mira a ottenere impegni concreti dai governi in termini di promozione della trasparenza, di sostegno alla partecipazione civica, di lotta alla corruzione e di diffusione di nuove tecnologie (dentro e fuori le Pubbliche Amministrazioni) a sostegno dell'innovazione[47]. L'apertura dei dati pubblici quindi risponde a molteplici finalità [46]:

1. trasparenza: i cittadini devono poter accedere a tutte le informazioni necessarie al fine di conoscere il funzionamento e l'operato delle pubbliche amministrazioni. I dati devono essere diffusi in formato aperto per garantirne il riutilizzo e la rielaborazione;
2. partecipazione: cittadini, organizzazioni della società civile e imprese devono essere coinvolti nei processi decisionali e nella definizione delle politiche nazionali e locali contribuendo con idee, conoscenze e abilità al bene comune e all'efficienza delle amministrazioni;

3. cittadinanza digitale e innovazione: le nuove tecnologie rendono l'amministrazione più efficiente e facilitano la trasparenza e la partecipazione civica. L'OGP promuove lo sviluppo di piattaforme pubbliche per la fornitura di servizi, l'espansione della cittadinanza digitale e la condivisione di idee e informazioni.

Lo scopo del portale data.gov è quello di diventare l'unico punto d'accesso e di riferimento per tutte le informazioni pubbliche che vengono prodotte dal governo. Questa piattaforma permette alle aziende, ai singoli e alle altre istituzioni di prelevare o estrarre i dati grezzi attraverso strumenti, quali applicazioni o siti web. L'obiettivo principale dell'Open Government è quello di diminuire la distanza fra i cittadini e l'apparato pubblico, mettendo così al primo posto il tema della trasparenza. Altri obiettivi che l'amministrazione potrebbe raggiungere, attraverso l'uso di questo portale, sono un risparmio sui fondi pubblici e un nuovo slancio economico dato dalle opportunità di utilizzo dei dati da parte delle aziende.

3.2 Linked data

Il concetto di Linked Data consiste nell'utilizzare il Web per creare collegamenti tipizzati tra dati provenienti da fonti diverse. Si tratta di un insieme di best-practice relative alla pubblicazione e interconnessione dei dati sul Web, in modo tale che possano essere machine-readable [55].

Una caratteristica fondamentale affinché i dati siano facilmente riutilizzabili è che siano ben strutturati, infatti, migliore è la definizione della struttura, più facile sarà creare strumenti per poterli riutilizzare. I Linked Data descrivono un metodo per pubblicare dati strutturati in modo tale da poter essere interconnessi con altri dati attraverso i metadati [63]. La struttura dei dati così creata ha come fondamentale caratteristica di essere facilmente navigabile anche dalle macchine.

Tecnicamente, i Linked Data si riferiscono ai dati pubblicati sul Web in modo tale che:

- siano machine readable;
- il loro significato sia esplicitamente definito;
- siano collegati ad altri set di dati esterni e che possano a loro volta essere collegati a dataset esterni.

Mentre le unità primarie del Web ipertestuale sono documenti HTML collegati da hyperlink non tipizzati, il concetto di Linked Data si basa su documenti contenenti dati in formato RDF. Quindi, anziché semplicemente collegare questi documenti, i Linked Data utilizzano RDF per creare statement tipizzati che collegano cose arbitrarie nel mondo. Il risultato è il “web of data”, il quale può essere visto come una rete di cose nel mondo descritta dai dati che si trovano sul Web.

Il web semantico quindi non si occupa solo di mettere i dati sul web, ma su questi dati devono essere creati dei collegamenti, in modo che una persona o una macchina possa esplorare il web of data. Il vantaggio derivante dall'utilizzo dei linked data è che, quando se ne ha uno, si possono trovare in modo semplice e veloce tutti gli altri dati correlati.

Effettuando quindi un confronto tra il web ipertestuale e il web of data si può notare che entrambi sono costituiti da documenti sul web, ma, a differenza del primo, dove i collegamenti sono ancora di relazioni in documenti ipertestuali scritti in HTML, nel web of data i dati linkano tra cose arbitrarie descritte in RDF, dove gli URI identificano qualsiasi tipo di oggetto o concetto.

Tim Berners-Lee coniò il termine ‘linked data’ nel 2006, e definì una serie di ‘regole’ per pubblicarli. Attenendosi a queste indicazioni i dati vengono inseriti nel Web of Data. Le regole prendono il nome di “Linked Data Principles” e si compongono di quattro punti:

1. gli URI come nome per le cose;
2. usa gli URI HTTP in modo che le persone possano cercare quei nomi;
3. quando qualcuno cerca un URI, fornisce informazioni utili, usando gli standard (RDF, SPARQL);
4. includere link ad altri URI, in modo che si possano scoprire più cose.

Se i database relazionali utilizzano delle tabelle per rappresentare la conoscenza, i linked data la organizzano come un grafo interconnesso. In questo modo si otterrà una rappresentazione più completa, flessibile e facilmente riorganizzabile nel caso in cui sia necessario un cambiamento dello schema rappresentante la base di conoscenza.

Riassumendo, le entità sono identificate da URI che utilizzano lo schema `http://`, così da poter essere cercate e trovate de-referenziando l'URI con il

protocollo http. In questo modo il protocollo HTTP diventa un meccanismo semplice, ma universale, per il recupero sia delle risorse serializzabili in flussi di byte che delle descrizioni delle risorse che non possono essere inviate attraverso la rete in questo modo. Ad esempio, de-referenziando l'URI `http://dbpedia.org/resource/Berlin`, e chiedendo il contenuto `application/rdf+xml`, dopo il reindirizzamento, si ottiene la descrizione associata, equivalente alla descrizione RDF di `http://dbpedia.org/resource/Berlin` contenente l'informazione sulla risorsa `http://dbpedia.org/data/Berlin`. URI e HTTP sono integrati da una tecnologia fondamentale per il Web of Data, il linguaggio RDF: mentre HTML fornisce un mezzo per strutturare i documenti e i link sul Web, RDF fornisce un modello di dati graph-based, con cui strutturare i dati e i link che descrivono le cose nel mondo [27].

L'utilizzo del data model RDF nel contesto dei Linked Data porta una serie di vantaggi: [57]:

- utilizzando HTTP URI come identificatori univoci globali per gli elementi dei dati, nonché per i termini del vocabolario, RDF è destinato ad essere utilizzato su scala mondiale, permettendo a chiunque di fare riferimento a qualcosa;
- gli utenti possono cercare qualsiasi URI in un grafo RDF sul Web per recuperare informazioni aggiuntive. Ogni tripla RDF è parte del Web of Data e può essere utilizzata come punto di partenza per esplorare questo spazio dei dati;
- consente di impostare i collegamenti RDF tra i dati provenienti da fonti diverse;
- consente di rappresentare informazioni in un unico grafo anche se espresse attraverso schemi diversi. È possibile quindi combinare le definizioni di vocabolari diversi per rappresentare i dati;
- combinando RDF con linguaggi quali RDF-Schema e OWL, il modello di dati consente la rappresentazione dei dati che si desiderano (dati strutturati e semi-strutturati).
- ...

Alcuni tra i principali vantaggi dei Linked Data: [57]:

- forniscono un modello unificante per i dati: i Linked Data utilizzano come unico formato di dati RDF, il quale è stato progettato appositamente per la condivisione globale dei dati, al contrario degli altri metodi per la pubblicazione di dati sul Web che, basandosi su una varietà di modelli, devono gestire l'eterogeneità che ne deriva attraverso un processo di integrazione;
- forniscono un meccanismo standardizzato di accesso ai dati: a differenza delle Web API dove le fonti dei dati sono accessibili solo tramite interfacce proprietarie, i Linked Data sono legati all'utilizzo del protocollo HTTP, il quale consente l'accesso alle sorgenti tramite browser generici e la loro scansione attraverso l'uso di motori di ricerca;
- una discovery dei dati di tipo hyperlink-based: differentemente delle web API, che utilizzate come depositi di dati in formati proprietari risultano dei contenitori di dati isolati, i Linked Data usano URI come identificatori globali per le diverse entità, consentendo la creazione di link ipertestuali tra entità in diverse fonti di dati. In questo modo si crea un unico spazio globale di dati e si permette alle applicazioni di scoprire nuove fonti di dati a run-time.
- ...

3.2.1 Linked data application

Dato l'alto numero di Linked Data pubblicati sul Web, si stanno facendo numerosi sforzi per la creazione di applicazioni che siano in grado di sfruttare il Web of Data. Queste applicazioni possono essere classificate in tre categorie: Linked Data browsers, Linked Data search engines, e domain-specific Linked Data applications. Le prime due possono essere viste come applicazioni generiche, le quali sono in grado di processare dati appartenenti a qualsiasi argomento, come le pubblicazioni di un autore o le formule matematiche [27].

Linked data browser

Come i tradizionali web browser i linked data browser consentono agli utenti di navigare tra le pagine HTML seguendo i link ipertestuali, ma permettendo la navigazione anche tra fonti di dati attraverso collegamenti espressi

come triple RDF. Ad esempio, un utente può visualizzare la descrizione RDF di DBpedia della città di Birmingham, seguire un link "luogo di nascita" alla descrizione del comico Tony Hancock (che è nato in città), e da lì in poi nei dati RDF si può arrivare sino alla BBC che descrive le trasmissioni in cui recitava Hancock. Ogni utente in questo modo può iniziare la navigazione in un'origine dati e attraversare progressivamente il Web seguendo, anziché link HTML, RDF. Alcuni esempi di questi browser sono:

1. Disco hyperdata browser: esso segue questo approccio e può essere visto come un'applicazione diretta del paradigma di navigazione ipertestuale sul Web dei dati;
2. Tabulator5: dispone di diverse modalità per l'analisi dei dati risultanti come linee temporali o mappe;
3. Marbles6: tiene traccia della provenance dei dati mostrando le diverse fonti;
4. altri progetti come Fenfire7 o VisiNav8 , che permettono ricerche con query molto articolate.
5. ...

Linked data search engine

Nel tradizionale Web dell'ipertesto, la navigazione e la ricerca sono spesso viste come le due modalità di interazione dominante. Mentre i browser forniscono i meccanismi per navigare nello spazio delle informazioni, i motori di ricerca sono spesso il luogo in cui inizia il processo di navigazione. Sono stati sviluppati numerosi motori di ricerca che eseguono la scansione dal Web dei linked data: essi sono in grado di aggregare dati provenienti da migliaia di sorgenti e forniscono la possibilità di interrogarli eseguendo delle query.

In generale, questi servizi possono essere suddivisi in due categorie:

1. human-oriented search engines: applicazioni come Falcons9 e Semantic Web Search Engine, le quali permettono agli utenti la ricerca dei dati attraverso l'utilizzo di parole chiave;

2. application-oriented indexes: Sindice, Swoogle e Watson, sono invece orientati alla ricerca automatizzata, quella necessaria alle applicazioni costruite sui Linked Data, e che utilizzano API per trovare i documenti RDF che fanno riferimento ad un certo URI o che contengono certe parole.

Applicazioni di dominio specifico

Sebbene i linked data browser e i motori di ricerca forniscano in gran parte funzionalità generiche, sono stati sviluppati anche numerosi servizi che offrono funzionalità più specifiche per un certo dominio, "mescolando" i dati da varie fonti di Linked Data [27]. Un esempio di queste applicazioni è Revyu, un sito di revisione e valutazione basato sui principi dei Linked Data e sullo stack tecnologico del web semantico. Oltre a pubblicare linked data, Revuy utilizza quelli presenti sul Web per migliorare la user experience. Ad esempio, quando i film vengono revisionati su Revyu, il sito tenta di abbinarli con la relativa voce in DBpedia, e, in caso di corrispondenza, vengono recuperate ulteriori informazioni sul film (come il nome del regista o il poster del film) e visualizzate nelle pagine HTML del sito. Un altro esempio è DBpedia Mobile, un location-aware Linked Data browser progettato per essere eseguito su un dispositivo mobile. Esso è stato pensato per un turista che visita una città. DBpedia Mobile consente agli utenti di pubblicare sul web immagini, posizione o recensioni come linked data, in modo che possano essere utilizzati anche da altre applicazioni. Invece di essere semplicemente taggati con coordinate geografiche, il contenuto pubblicato è interconnesso con una risorsa DBpedia vicina, contribuendo così all'aumentare la vastità e la ricchezza del Web of Data.

3.3 Linked open data

Come visto, mentre in generale gli Open Data abbattano le barriere culturali, legali ed economiche relative al riutilizzo dei dati, i Linked Data si concentrano sulla messa a punto di strumenti che permettono di dare ai dati (aperti o non) un'identità e di renderli interoperabili e collegati tra loro, rimuovendo gli ostacoli tecnologici per una libera condivisione dei dati sul web. I Linked Data infatti facilitano l'integrazione dei dati provenienti da fonti diverse in quanto sono basati su vocabolari condivisi, le cui definizioni sono recuperabili e i cui termini sono collegati fra loro tramite link.

La fusione del concetto di Open Data con quello di Linked Data dà quindi vita alla nozione di Linked Open Data (LOD). Si tratta di un metodo di pubblicazione di dati aperti leggibili da una macchina, che possa essere interconnesso tra diversi dataset sul Web, consentendo l'integrazione dei dati e l'interrogazione semantica [27].

È possibile classificare la qualità dei dati in funzione alla facilità di utilizzo e alla loro aderenza ai principi del web semantico. Tale classificazione, mostrata anche in figura 3.1, proposta per la prima volta da Tim Berners-Lee nel 2006, è composta da 5 stelle [17]:

- * 1 stella, On the Web and Open License: quando il dato è messo a disposizione di tutti sul web (in qualsiasi formato) con licenza libera;
- ** 2 stelle, Machine-readable data: quando il dato è presente in modo strutturato e machine-readable (es. una tabella di un foglio di calcolo Excel);
- *** 3 stelle, Non-proprietary formay: quando il dato è presente come nel caso delle due stelle, ma in formato non proprietario;
- **** 4 stelle, RDF Standards: quando il dato del caso precedente usa gli standard Linked Data del W3C per identificare i dati;
- ***** 5 stelle, Linked RDF: quando i dati sono disponibili come in tutti i quattro punti precedenti, ma con in più la possibilità che essi siano collegati ad altri, utilizzando gli standard W3C, per fornire un contesto.

Questi punti rappresentano anche la trasformazione che dovrebbe avvenire sui dati affinché siano considerati a tutti gli effetti come Linked Open Data.

Benché nell'Information Retrieval la tecnica dell'aggregazione dei dati al fine di trovare l'informazione desiderata non risulti essere innovativa, se inserita in un contesto LOD rappresenta un'innovazione. Nell'IR le best practices della pubblicazione delle risorse hanno spinto i fornitori delle informazioni a pubblicare dei dati grezzi. Per consentire il loro riuso occorre poter combinare e mescolare liberamente i dataset, ed è quindi necessario collegare tra loro i dati, stabilendo un collegamento diretto quando i dati sono relazionati o si riferiscono a oggetti identici. Questo collegamento diretto dà la possibilità di saltare da un dataset all'altro. È proprio grazie a



Image taken from w3.org

Figura 3.1: Le cinque stelle dei LOD, from 3.1

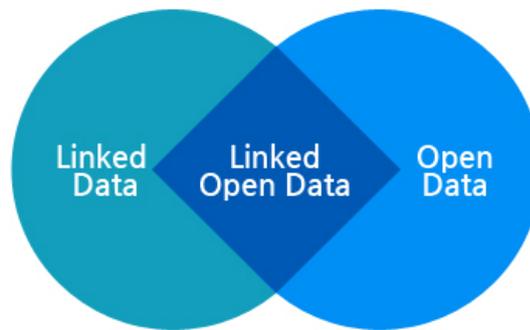


Figura 3.2: Linked open data, from [59]

questo scenario che le possibilità offerte dall'aggregazione dei dati risultano ampliate, in un modo che non ha precedenti[31].

3.3.1 Settori di utilizzo dei LOD

I linked open data si stanno diffondendo sempre di più, e sono ormai utilizzati in molti settori. Quelli descritti di seguito sono solo alcuni dei campi in cui i LOD vengono utilizzati.

Geografia

Uno tra i fornitori di dati geografici più importanti è Geonames: esso consiste in un database geografico open license, che pubblica e collega i dati di circa otto milioni di località. Un altro importante progetto è LinkedGeoData, il quale utilizza le informazioni raccolte dal progetto OpenStreetMap (successivamente descritto) e le rende disponibili come conoscenza di base RDF secondo i principi dei Linked Data. LinkedGeoData collega i dati con altre basi di conoscenza che utilizzano i Linked Open Data. I dati RDF risultanti comprendono circa 20 miliardi di triple, sono disponibili in base ai principi dei linked data e sono interconnessi con DBpedia e GeoNames [30].

Media

Nel settore dei media, una delle fonti più importanti di Linked Data è la British Broadcasting Corporation (BBC). I siti dei programmi e della musica della BBC forniscono dati relativi a episodi di programmi TV e radio. Questi dati vengono interconnessi con DBPedia (descritto successivamente) e MusicBrainz, un database musicale open-license, consentendo alle applicazioni di recuperare e combinare i dati. In questo modo la BBC potrebbe utilizzare i dati in tempo reale per fornire al proprio pubblico un contesto più ampio sulla musica che viene riprodotta sulle varie reti radio gestite da essa. Inoltre, l'uso dei dati di MusicBrainz offre alla BBC l'opportunità di sostenere le sue trasmissioni con un'offerta online completa e basata sui dati, la quale consente di scoprire nuova musica, sia all'interno che all'esterno dei siti Web della BBC [44].

Scienza

Bio2RDF [23] è un database biologico che utilizza tecnologie del web semantico per fornire dati relativi a scienze biologiche interconnesse. Bio2RDF collega circa 35 dataset biologici ampiamente utilizzati, tra cui clinicaltrials.gov, dbSNP, GenAge, GenDR, LSR, OrphaNet, PubMed, SIDER e

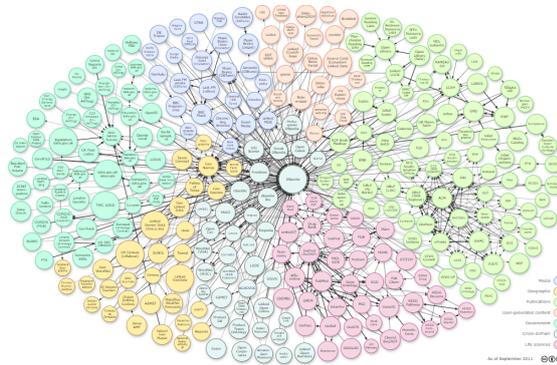


Figura 3.3: Dati DBpedia 2011, from [69]

WormBase. In Bio2RDF si contano più di due miliardi di triple RDF. All'interno del W3C inoltre si trova Linking Open Drug Data [18], dove le società farmaceutiche Eli Lilly, AstraZeneca e Johnson Johnson, stanno cooperando al fine di collegare gli open data relativi ai farmaci con quelli delle sperimentazioni cliniche al fine di sostenere ed aiutare la ricerca.

3.3.2 Esempi di portali LOD

Come già visto, al fine di raccogliere informazioni da poter pubblicamente rilasciare, sono nati numerosi portali linked open data. Nei paragrafi successivi sono descritti alcuni dei più importanti.

DBpedia

DBpedia è stato uno dei primi esempi di portali Linked Open Data. È stato creato nel 2007 per estrarre informazioni strutturate presenti su Wikipedia al fine di pubblicarle sul web come Linked Open Data in formato RDF. Il grande vantaggio è che, quando un utente cerca un'informazione su questo portale, riesce facilmente ad accedere a tante altre tipologie di informazioni che sono collegate a quella cercata. Dalle figure 3.3 e 3.4 si può vedere come i collegamenti dei dati siano aumentati dal 2011 al 2018.

DBpedia inoltre, attraverso un apposito endpoint 3.5, consente di eseguire query SPARQL sui dati memorizzati. Questo portale è considerato da Tim Berners Lee come una delle parti più importanti del progetto Linked Data basato su RDF.

OpenStreetMap

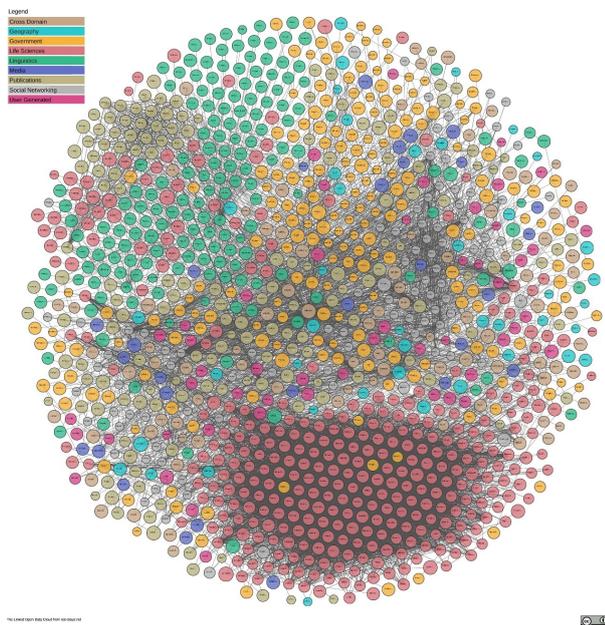


Figura 3.4: Dati DBpedia oggi, from [7]

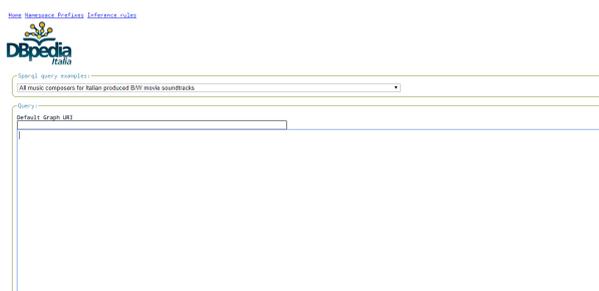


Figura 3.5: Endpoint DPBEDIA

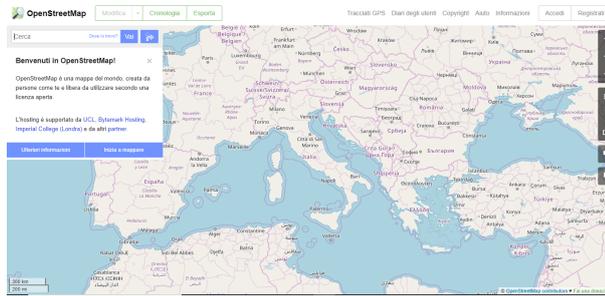


Figura 3.6: OpenStreetMap

OpenStreetMap è un'iniziativa aperta alla creazione di dati geografici ed alla loro fornitura gratuita a chiunque li desideri. Essa conta una centinaia di migliaia di utenti registrati in tutto il mondo. Il progetto OpenStreetMap, ha come obiettivo quello di incoraggiare la crescita, lo sviluppo e la distribuzione di dati geospaziali e fornirli a chiunque, per l'uso e la condivisione. OpenStreetMap [48] è quindi un progetto collaborativo il cui scopo è quello di creare mappe a contenuto libero, al fine di incoraggiare la crescita e lo sviluppo dei dati Geospaziali disponibili, utilizzabili e condivisibili da chiunque. Per ottenere le mappe OpenStreetMap, descrive il sito ufficiale, il miglior punto di partenza è uno snapshot del database OpenStreetMap. I dati di tutto il mondo sono disponibili ed aggiornati regolarmente. Per estrarre ciò di cui si ha bisogno si possono scrivere semplici programmi e script, oppure utilizzare alcuni dei tools e procedure sviluppati direttamente dalla comunità OpenStreetMap. Non ci sono vincoli su chi può usare i dati: singoli, club, collettivi, istituzioni benefiche, comunità accademiche, istituzioni pubbliche, società commerciali. Non ci sono vincoli neanche sul dove usare i dati: privatamente o pubblicamente, commercialmente o meno. Come gli altri portali LOD mette a disposizione un endpoint SPARQL per l'interrogazione dei dati.

ISTAT

La piattaforma Linked Open Data dell'ISTAT consente di accedere e navigare dati in formato open, sulla base di tecnologie e standard del web semantico. I LOD, direttamente interrogabili, rispondono alle esigenze espresse dalle comunità di utilizzatori di disporre di dati standardizzati e interoperabili. Il primo insieme di Linked Open Data pubblicato dall'Istat è costi-

tuito da dati provenienti dal Censimento della popolazione e delle abitazioni 2011. In particolare, sono state definite le seguenti ontologie:

1. ontologia delle Basi Territoriali: l'ontologia delle Basi Territoriali formalizza e descrive il territorio italiano analizzandolo sia da un punto di vista amministrativo che statistico-geografico;
2. ontologia dei Dati Censuari: questa ontologia formalizza e descrive i metadati delle variabili censuarie. Trattandosi di dati statistici è stata utilizzata la meta ontologia Data Cube Vocabulary creata appositamente per la rappresentazione di dati multi-dimensionali. Per verificare la provenienza dei dati, affinché siano affidabili e di qualità, l'ISTAT sfrutta l'ontologia PROV-O che consente di descrivere in dettaglio la provenienza dei dati.

L'ISTAT [33] dichiara *"Negli ultimi tempi ci stiamo avvicinando ad una fase di più matura consapevolezza riguardo ai potenziali vantaggi derivanti dall'utilizzo degli Open Data. Si sta cominciando a fare più attenzione alla qualità dei dati rilasciati e questo, assieme ad una cultura del dato sempre più diffusa, diventa il presupposto per la nascita di casi di riuso sempre più interessanti"*. I dati aperti stanno quindi cominciando a contribuire in modo fondamentale alla nascita di idee e progetti, fornendo servizi e nuove opportunità per i cittadini, per le imprese e per le stesse pubbliche amministrazioni.

Capitolo 4

Caso di studio

Come accennato nei capitoli precedenti, oggi il mercato IoT del fitness è dominato da wearables e da applicazioni mobile health per smartphone. La potenza di questi device risiede nella loro capacità di raccogliere dati riguardanti gli allenamenti e i parametri vitali. Il contesto dell'IoT però, da un punto di vista dei dati, è caratterizzato da un'elevata eterogeneità di formati. Tra i vari fornitori di dispositivi IoT riguardanti il fitness, gli stessi concetti sono infatti rappresentati attraverso tipi e formati diversi. Tale eterogeneità e la mancanza di standard a cui uniformarsi, fanno sì che i dati rimangano limitati all'interno del sistema, senza possibilità di avere una visione integrata di essi. Ciò porta ad una riduzione della loro potenzialità e della possibilità di estrarre nuove informazioni. Queste problematiche potrebbero essere risolte attraverso un sistema in grado di integrare i dati raccolti dai dispositivi IoT, consentendo così analisi su di essi.

Lo scopo del progetto di tesi è quello di creare una piattaforma, utilizzando le tecnologie del web semantico, che permetta di integrare dati provenienti dai diversi dispositivi attraverso un formato comune, RDF, in modo che l'utente utilizzatore di diversi fitness device, possa visualizzare grafici contenenti le informazioni di un certo allenamento come farebbe sulle altre piattaforme, ma con il vantaggio di avere a disposizione i dati di tutti i suoi dispositivi. Questo permetterà all'utilizzatore di avere una visione più completa e generale sia dell'attività fisica che della sua condizione di salute.

La piattaforma creata dovrà essere inoltre un vero e proprio portale Linked Open Data, mettendo a disposizione un endpoint SPARQL per poter

interrogare i dati. Questo permetterà di usufruire di tutti i vantaggi derivanti dall'utilizzo dei Linked Open Data, tra cui fornire un meccanismo di accesso standardizzato ai dati e un modello unificante di essi, in quanto, l'utilizzo di un unico formato (RDF), permette di condividerli globalmente.

Prima di passare alle scelte implementative e alla descrizione della realizzazione del progetto sono definiti due concetti fondamentali, che saranno utili al fine di comprendere meglio i paragrafi successivi.

Triplestore

Le triple definite attraverso il linguaggio RDF vengono solitamente memorizzate in uno storage detto triplestore. Il triplestore è un database pensato e costruito appositamente per la memorizzazione e il recupero di triple. Esso è simile a un database relazionale in quanto le informazioni sono memorizzate in una base dati e vengono recuperate attraverso un apposito linguaggio di interrogazione. I triplestore RDF vengono utilizzati con successo per gestire i dataset Linked Open Data, come DBPedia e GeoNames. Essi permettono di interrogare dati diversi e in evoluzione provenienti da più fonti, e sono più efficienti in termini di costi e di tempo [45].

Endpoint SPARQL

Al fine di eseguire interrogazioni sui dati memorizzati nel triplestore è necessario un endpoint SPARQL. Esso può essere definito come un web service attraverso il quale è possibile accedere ai dati con interrogazioni SPARQL. Gli endpoint possono essere suddivisi in due macro-categorie [61]:

- endpoint specifici: è possibile interrogare solo un insieme predefinito di dataset precaricati nell'endpoint;
- endpoint generici: è possibile eseguire l'interrogazione su qualsiasi fonte dati RDF accessibile via Web.

4.1 Casi d'uso

Per comprendere meglio le funzioni e i servizi offerti dalla piattaforma, e come sono percepiti dagli utilizzatori che interagiscono con esso, è stato realizzato il diagramma dei casi d'uso mostrato in figura 4.1.



Figura 4.1: Diagramma dei casi d’uso

L'utente può eseguire le seguenti azioni:

- registrazione: permette la registrazione dell'utente alla piattaforma;
- login: attraverso questa operazione l'utente potrà loggarsi fornendo le proprie credenziali, username e password. L'azione di login potrà essere effettuata solo a seguito della registrazione dell'utente;
- logout: questa operazione permette di disconnettersi dalla piattaforma e può essere effettuata solo a seguito del login;
- visualizzazione dei dati personali: questa operazione potrà essere effettuata dopo il login e permette la visualizzazione dei dati inseriti dall'utente al momento della registrazione;
- caricamento dei dati rilevati dai dispositivi: questa azione permette all'utente di caricare i dati relativi agli allenamenti, e in generale alla salute, raccolti dai dispositivi di cui è in possesso. La piattaforma, attraverso un processo di mapping, provvederà a convertire i dati dal formato di input in RDF, in modo tale da poter essere memorizzati e quindi utilizzati insieme, al fine di realizzare grafici relativi agli allenamenti e al proprio stato fisico;
- inserimento query per creazione grafico: l'utente, attraverso l'inserimento di query SPARQL, può creare nuovi grafici, e decidere se memorizzarli o meno. A seguito dell'esecuzione della query l'utente potrà quindi visualizzare il grafico risultante, il quale verrà sempre riprodotto sulla dashboard nel caso in cui si sia scelta la sua memorizzazione;
- visualizzazione del risultato query: a seguito dell'inserimento ed esecuzione della query SPARQL l'utente potrà visualizzare il grafico risultante;
- visualizzazione dei grafici: l'utente può visualizzare sulla dashboard tutti i grafici da lui creati, attraverso le query SPARQL, e memorizzati. Al momento della registrazione vengono assegnate automaticamente due query all'utente: esso inizialmente vedrà quindi sulla dashboard i due grafici vuoti, i quali si popoleranno nel momento in cui caricherà i dati necessari;

- eliminazione dei grafici: l'utente ha la possibilità di eliminare i grafici (di conseguenza le query) nel caso in cui ne avesse necessità;
- interrogazione attraverso l'endpoint: l'utente potrà interrogare l'intero insieme di dati attraverso l'endpoint SPARQL fornito. Trattandosi di un portale Linked Open Data, l'utilizzo dell'endpoint è possibile anche senza l'operazione di login;
- inserimento di un nuovo dispositivo: affinché i dati possano essere memorizzati devono essere convertiti in RDF, e, per poterlo fare, è necessaria una mappatura. Per ogni dispositivo, i cui dati vengono sottoposti al processo di mapping, si necessita quindi del relativo file di mappatura. L'utente potrà inserire nuovi fornitori di dispositivi oltre a quelli già presenti, caricando i dati e i file necessari al mapping. Questo inserimento può essere svolto anche senza effettuare il login.

4.2 Architettura

La figura 4.2 mostra lo scheletro dell'architettura dal quale si è partiti per realizzare il progetto. Qui non sono ancora fatti riferimenti alle tecnologie utilizzate, ma viene indicato solo il funzionamento generale del sistema.

Il client effettua delle richieste al server, il quale:

- si relaziona direttamente con il triplestore in caso di: esecuzione e memorizzazione di query SPARQL per il recupero dei dati necessari alla realizzazione dei grafici, esecuzione di query SPARQL effettuate sull'endpoint, visualizzazione dei dati personali, registrazione, login ed eliminazione dei grafici;
- provvede alla memorizzazione dei dati provenienti dai dispositivi dell'utente sul triplestore solo a seguito della conversione del formato di origine dei dati in RDF attraverso il sistema di mapping.

Il triplestore è quindi l'elemento che memorizza tutti i dati di fitness relativi agli utenti: esso viene popolato con le triple RDF ottenute dal mapping e interrogato per ottenere i dati richiesti e utili agli utenti.

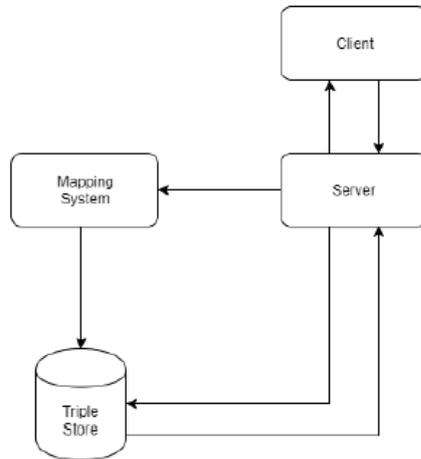


Figura 4.2: Architettura piattaforma IFO

4.2.1 Tecnologie utilizzate

Tenendo a mente l'architettura mostrata in figura 4.2, sono ora descritte le tecnologie scelte per l'implementazione della piattaforma.

Web Server – Apache Tomcat

Un web server è un'applicazione software che, in esecuzione su un server, è in grado di gestire le richieste di trasferimento di pagine web di un client, tipicamente un web browser. Il Web Server scelto è Tomcat. Apache Tomcat è un web server (nella forma di contenitore servlet) che implementa le specifiche JavaServer Pages (JSP) e servlet, fornendo una piattaforma software per l'esecuzione di applicazioni Web sviluppate in linguaggio Java. Il software Apache Tomcat è quindi un'implementazione open source delle tecnologie Java Servlet, JavaServer Pages, Java Expression Language e Java WebSocket. Apache Tomcat alimenta numerose applicazioni Web mission-critical su larga scala in una vasta gamma di industrie e organizzazioni. Esso è scritto interamente in Java e può quindi essere eseguito su una qualsiasi architettura su cui sia installata una JVM [58].

JSP-SERVLET

Un'applicazione web è un'applicazione client-server a cui è possibile accedere attraverso un web browser. Essa è quindi caratterizzata da un insieme di pagine web generate in risposta alle richieste degli utenti. In un'applicazione web le componenti risiedono sia sul client che sul server. Il browser usato per accedere all'applicazione gira sul client e trasforma codice HTML nell'interfaccia utente. L'applicazione vera e propria invece risiede sul server, dove si trova un web server responsabile dell'invio delle pagine web al browser.

In un'applicazione web Java è previsto un servlet/jsp engine, o servlet/jsp container, che consente al web server di lavorare con servlet e JSP:

- le servlet sono classi Java eseguite lato server per esaudire richieste provenienti dai web server e supportano un modello di programmazione "request and response". Esse vengono usate per elaborare e salvare dati provenienti da form HTML, provvedere alla creazione di pagine HTML dinamiche, gestire informazioni con stato... Quando un client effettua una richiesta al server, esso la invia alla servlet, la quale costruisce una risposta che il server invia al client. La servlet per prima cosa determina se la richiesta è una GET o una POST, e, di conseguenza, chiama uno dei seguenti metodi: doGet o doPost. È da notare che le servlet vengono eseguite in un contenitore e che le API forniscono la gestione del ciclo di vita di oggetti e sessioni. Quando si utilizzano le servlet, si ottengono tutti i vantaggi derivanti dall'utilizzo di una piattaforma Java: la sandbox (sicurezza), l'API di accesso al database tramite JDBC e la portabilità multipiattaforma delle servlet.
- per JSP si intende una tecnologia che consente la creazione di pagine HTML dinamiche, e che permette di mescolare codice HTML, componenti riusabili, applicazioni remote (servlet), codice java e script java-like. L'estensione del file di una pagina JSP può essere .jsp, .html o .htm: ciò indica al server che la pagina richiede una gestione speciale che verrà eseguita da un'estensione del server o da un plug-in. Quando viene chiamata una pagina JSP essa viene compilata dal JSP engine in una servlet Java. A questo punto la servlet viene gestita dal

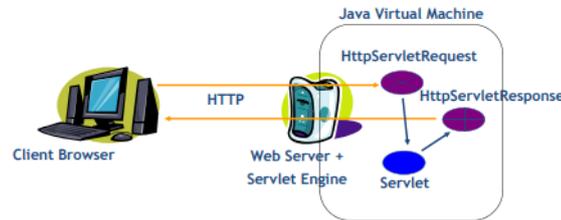


Figura 4.3: Architettura piattaforma IFO

servlet engine, proprio come qualsiasi altra servlet. Il servlet engine carica quindi la classe servlet e la esegue per creare HTML dinamico da inviare al browser. Quando viene effettuata una richiesta successiva della pagina:

- se la pagina JSP non è stata modificata il JSP engine esegue la servlet già caricata;
- altrimenti la pagina JSP verrà automaticamente ricompilata in una servlet ed eseguita.

Questa tecnologia mette a disposizione una serie di tag e costrutti specifici, tra cui:

- `< % statement % >`: all'interno di questo tag viene scritto codice java che verrà interpretato dalla Java Virtual Machine;
- `<%=espressione%>`: all'interno del tag viene calcolata l'espressione indicata e ne viene restituito il valore;
- `<%!dichiarazione%>`: all'interno di questo tag sono presenti frammenti di codice che vengono riportati al di fuori del metodo principale di gestione della richiesta, ovvero a livello di classe;

Uno dei vantaggi derivanti dall'utilizzo delle JSP è che permettono la generazione dinamica di contenuti HTML, oltre che l'interazione con componenti di complesse Web Application, permettendo quindi di creare sia pagine statiche che dinamiche. Inoltre [49]:

- le tecnologie sono abbastanza facili da imparare;
- se usato in modo efficace seguendo le best practice, servlet e pagine JSP aiutano a separare la presentazione dal contenuto;
- il linguaggio di programmazione è robusto e Object Oriented;
- il Web Container fornisce servizi aggiuntivi, come la gestione degli errori e la sicurezza;
- tradurre o far comunicare un'applicazione Java con pagine JSP è estremamente facile.

D'altro canto, però, uno degli svantaggi più forti di questa tecnologia, è che le pagine JSP richiedono più tempo quando vi si accede per la prima volta, in quanto devono essere compilate sul server.

Linguaggi del web semantico utilizzati

Avendo già descritto nel primo capitolo il linguaggio di interrogazione SPARQL, viene ora introdotto SPARQL UPDATE, un linguaggio di aggiornamento RDF con sintassi SPARQL. SPARQL Update è un linguaggio complementare e il cui uso è previsto in combinazione con il linguaggio di interrogazione SPARQL. Esso è pensato per essere un linguaggio standard per la specifica e l'esecuzione di aggiornamenti dei grafi RDF in un Graph Store. In particolare, il linguaggio offre le seguenti funzionalità [64]:

- inserimento di triple in un grafo RDF nel Graph Store;
- eliminazione di triple da un grafo RDF nel Graph Store;
- caricamento di un grafo RDF nel Graph Store;
- eliminazione di un grafo RDF nel Graph Store;
- creazione di un nuovo grafo RDF in un Graph Store;
- spostamento, aggiunta o copiatura del contenuto di un grafo RDF da un Graph Store a un altro;
- esecuzione di un gruppo di operazioni di aggiornamento come una singola azione.

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ns: <http://example.org/ns#> .
<http://example/book1> ns:price 42 .
```

Figura 4.4: Dataset prima dell’inserimento, from [64]

Per Graph Store si intende un contenitore mutevole di grafi RDF. Vediamo più in dettaglio le operazioni di aggiornamento [64]:

- **INSERT DATA:** permette di aggiungere alcune triple in un grafo. Se il grafo di destinazione non esiste allora viene creato. Nel caso in cui il grafo non esista, e non possa essere creato per un qualsiasi motivo, viene restituito errore;
- **DELETE DATA:** ha come obiettivo la rimozione di alcune triple nel caso in cui siano presenti nel grafo richiesto;
- **INSERT, DELETE:** sono le azioni fondamentali per gli aggiornamenti di un grafo. Permettono di eliminare o aggiungere gruppi di triple. La differenza tra INSERT / DELETE e INSERT DATA / DELETE DATA è che INSERT DATA e DELETE DATA non sostituiscono i binding in un template da un pattern. Le forms DATA richiedono dati concreti: avere operazioni specifiche per dati concreti significa che una richiesta può essere trasmessa in streaming in modo da poter eseguire aggiornamenti di dati puri e di grandi dimensioni;
- **LOAD:** legge il contenuto di un documento rappresentante un grafo in un grafo presente nel Graph Store;
- **CLEAR:** rimuove tutte le triple in uno o più grafi.

Sono ora mostrati due esempi forniti dal w3c, rispettivamente per l’inserimento e l’eliminazione di triple. La figura 4.4 mostra il dataset prima dell’operazione di inserimento, la quale è indicata in figura 4.5. Il risultato è raffigurato nella 4.6.

Invece, la figura 4.7 mostra il dataset prima dell’operazione di cancellazione, la quale è indicata in figura 4.8. Il risultato è raffigurato nella 4.9.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
INSERT DATA
{
  <http://example/book1> dc:title "A new book" ;
                        dc:creator "A.N.Other" .
}
```

Figura 4.5: Operazione di insert from [64]

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ns: <http://example.org/ns#> .

<http://example/book1> ns:price 42 .
<http://example/book1> dc:title "A new book" .
<http://example/book1> dc:creator "A.N.Other" .
```

Figura 4.6: Dataset dopo l'inserimento from [64]

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ns: <http://example.org/ns#> .

<http://example/book2> ns:price 42 .
<http://example/book2> dc:title "David Copperfield" .
<http://example/book2> dc:creator "Edmund Wells" .
```

Figura 4.7: Dataset prima della cancellazione from [64]

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
DELETE DATA
{
  <http://example/book2> dc:title "David Copperfield" ;
                        dc:creator "Edmund Wells" .
}
```

Figura 4.8: Operazione di cancellazione from [64]

```
# Default graph
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ns: <http://example.org/ns#> .

<http://example/book2> ns:price 42 .
```

Figura 4.9: Dataset dopo la cancellazione from [64]

Jena

Jena è il più diffuso framework Java gratuito e open source per la creazione di applicazioni web semantiche e Linked Data. Include un motore di inferenza basato su regole per effettuare reasoning e una varietà di strategie di archiviazione per memorizzare triple RDF in memoria o su disco [60].

Questo framework fornisce librerie Java per aiutare gli sviluppatori a scrivere codice che gestisca RDF, RDFS, RDFa, OWL e SPARQL in linea con le raccomandazioni del W3C. Jena archivia le informazioni come triple RDF in grafi e consente, attraverso il codice, di aggiungere, rimuovere, modificare, archiviare e pubblicare tali informazioni. I grafi e le triple così generati sono accessibili tramite l'API RDF di Jena. I triplestore offerti da Jena sono:

- SDB: triplestore persistente che utilizza database relazionali per l'archiviazione e l'interrogazione dei dati RDF. Non è però consigliabile usare SDB per le nuove applicazioni, TDB infatti è più scalabile e efficiente. Si tratta di un componente solo di mantenimento [38];
- TDB: triplestore nativo. Un dataset TDB dovrebbe essere direttamente accessibile da una singola JVM alla volta, altrimenti potrebbe verificarsi un danneggiamento dei dati: dalla versione 1.1.0 in poi TDB include la protezione automatica contro l'utilizzo di più JVM. Se si desidera condividere un dataset TDB tra più applicazioni, è necessario il server Fuseki, il quale permette di utilizzare TDB per l'archiviazione persistente e fornisce i protocolli SPARQL per query, aggiornamenti e REST su HTTP [39].

La gestione di SPARQL, sia per le query che per gli aggiornamenti, è responsabilità dell'API SPARQL.

In Jena tutte le informazioni di stato fornite da una raccolta di triple RDF sono contenute in una struttura dati detta "model". Il model indica quindi un grafo RDF, così chiamato in quanto contiene una raccolta di nodi RDF collegati tra loro da relazioni etichettate. Ogni relazione va solo in una direzione, quindi la tripla:

```
example:ex foaf:name "Chiara"
```

può essere letta come "la risorsa example: ex ha una proprietà foaf:name con valore "Chiara". Chiaramente il contrario non è vero. Matematicamente, questo rende il modello un'istanza di un grafo diretto.

In termini Java, la classe `Model` rappresenta un contenitore delle informazioni RDF, le quali si trovano sotto forma di grafo. Il model è progettato per avere un'API ricca, con molti metodi volti a semplificare la scrittura di programmi e applicazioni basati su RDF. `Model` fornisce inoltre un'astrazione per i diversi modi di memorizzare i nodi e le relazioni RDF: strutture di dati in memoria, archivi persistenti basati su disco e motori di inferenza. In particolare, i tre concetti di contenitori RDF in Jena sono [34]:

- `graph`: una vista matematica di relazioni orientate tra nodi in una struttura connessa;
- `Model`: una API Java per gli sviluppatori;
- `Graph`: libreria che ha come scopo quello di estendere alcune funzionalità di Jena.

Le informazioni RDF sono contenute in un grafo di nodi connessi, i quali possono essere URI o valori letterali: questi denotano rispettivamente alcune risorse su cui si vogliono fare asserzioni ed i valori di dati concreti che appaiono in quelle asserzioni. Una risorsa rappresentata come un URI rappresenta una cosa nominata, caratterizzata da un'identità. È quindi possibile utilizzare quell'identità per fare riferimento direttamente alla risorsa. Il valore letterale invece rappresenta solo un valore di dati, come la stringa "dieci" o il numero 10.

Quando non è nota l'identità (cioè l'URI) della risorsa, RDF consente anche un caso speciale. Citando l'esempio del sito ufficiale di Jena, consideriamo la frase "Ho dato cinque dollari al mio amico". Da questa affermazione si ricava che io ho un amico, ma non si sa chi sia. Si conosce anche una proprietà dell'amico, ovvero che ha cinque dollari in meno rispetto a prima. In RDF è possibile modellare questa situazione attraverso l'uso di una risorsa "speciale" detta risorsa anonima. Le risorse anonime in RDF vengono rappresentata con identità vuote: esse sono indicate come `blank identities` o `blank node` del grafo, in genere abbreviate in `bNodes`. Da un punto di vista del codice:

- l'interfaccia `Java Resource` rappresenta sia le normali risorse URI che i `bNode`;
- l'interfaccia `Java Literal` rappresenta i letterali.

Jena permette di supportare anche il reasoning. In particolare, il sottosistema di inferenza Jena è progettato per consentire a una gamma di reasoning di inferenza di essere collegati a Jena. Essi sono utilizzati per derivare ulteriori asserzioni RDF implicate da alcune basi RDF combinate insieme a una qualsiasi informazione ontologica opzionale e agli assiomi e regole associati al reasoning. Questo meccanismo permette di supportare l'uso di linguaggi come RDFS e OWL, i quali consentono di dedurre ulteriori fatti dai dati dell'istanza e dalle descrizioni delle classi. Come mostrato in figura 4.12, l'architettura di Jena è molto articolata e presenta molteplici funzionalità. In particolare, noi siamo interessati alle API SPARQL per l'interrogazione e l'update dei dati. Le classi di interesse per quanto riguarda l'esecuzione e la gestione del risultato di una query sono [36]:

- Query: classe che rappresenta la query dell'applicazione, è un contenitore per tutti i dettagli relativi alla query;
- Query Factory: metodi che forniscono l'accesso ai vari parser;
- DatasetFactory: per creare dataset;
- QueryExecution: classe che rappresenta l'esecuzione di una query;
- QueryExecutionFactory: usata per ottenere un'istanza di QueryExecution a partire da un oggetto Query e un riferimento al servizio endpoint SPARQL;
- Nel caso di una query SELECT:
 - “QuerySolution”: oggetto che contiene le soluzioni della query;
 - “ResultSet”: iteratore su tutte le risposte alla query;
 - “ResultSetFormatter”: classe in grado di trasformare un ResultSet in più modi, come per esempio un testo, formato CSV, JSON, grafo RDF, XML e altro.

Per gli aggiornamenti dei dati presenti sul triplestore invece viene utilizzato SPARQL Update. Una richiesta SPARQL Update è composta da un certo numero di operazioni di aggiornamento, quindi in una singola richiesta è possibile creare grafi, popolarli con dati RDF e modificarli. Le principali classi API sono [37]:

- UpdateRequest: un elenco di aggiornamenti da eseguire;
- UpdateFactory: crea oggetti UpdateRequest analizzando le stringhe o analizzando il contenuto di un file;
- UpdateAction: esegue gli aggiornamenti.

Fuseki

La pubblicazione di dati su Internet è un requisito comune nelle moderne applicazioni: per soddisfarlo sono nate numerose tecnologie, tra cui Fuseki, il quale permette di presentare e aggiornare i modelli RDF sul web utilizzando SPARQL e HTTP. Apache Jena Fuseki è un server SPARQL e può essere eseguito come servizio del sistema operativo, applicazione Web Java (file WAR) o come server autonomo. Fuseki fornisce sicurezza attraverso Apache Shiro ed è caratterizzato da un'interfaccia utente per il monitoraggio e l'amministrazione dei server.

Fornisce i protocolli SPARQL 1.1 per query e aggiornamenti, nonché il protocollo SPARQL Graph Store. Fuseki è strettamente integrato con TDB per fornire un solido livello di storage persistente transazionale e incorpora query di testo e spaziali di Jena. Esso fornisce REST-style SPARQL HTTP Update, SPARQL Query, e SPARQL Update [35].

Per memorizzare le triple in modo persistente Fuseki utilizza TDB, un componente delle Jena API che permette la memorizzazione e la ricerca di triple RDF. TDB fornisce la possibilità di gestire transazioni che soddisfano le proprietà ACID per la memorizzazione e la ricerca delle triple RDF, sfruttando il meccanismo di write-ahead-logging. Esso può essere eseguito in modalità safe, dove vengono accettate solo operazioni di lettura, oppure in modalità normal, nella quale vengono eseguite anche le operazioni di aggiornamento. Per avviare Fuseki su Windows è necessario utilizzare il comando: `java -jar fuseki-server.jar [parametri]`. In [parametri] si possono scegliere le varie configurazioni offerte:

- `--mem`: crea in memoria (non persistente) un dataset vuoto;
- `--file=FILE`: crea in memoria (non persistente) un dataset vuoto e poi carica il file;
- `--loc=DIR`: crea un database TDB esistente;



Figura 4.10: Fuseki: Manage Dataset

- `--desc=assemblerFile`: costruisce un dataset basato su una descrizione in assembler;
- `--config=ConfigFile` costruisce uno o più endpoint basati su un file di configurazione.

Fuseki gira sulla porta 3030, è quindi necessario accedervi con `http://localhost:3030`. La schermata così ottenuta è mostrata in figura 4.10: in particolare in “Manage dataset” si trovano tutti i dataset creati e viene fornita la possibilità di crearne di nuovi.

Sui dataset presenti, Fuseki mette a disposizione le seguenti operazioni:

1. `remove`: per eliminare il dataset;
2. `backup`: per effettuare il backup del dataset;
3. `file upload`: per caricare file in formato Turtle.

Nella schermata “Dataset” mostrata in figura 4.11 invece è possibile interrogare e aggiornare i dataset. Ogni operazione supportata da Fuseki ha il proprio endpoint:

- `http://*host*/dataset/query`, per eseguire le operazioni di ricerca;
- `http://*host*/dataset/update`, per eseguire le operazioni di aggiornamento.

Di conseguenza, quando da Jena viene richiesto un aggiornamento la richiesta dovrà essere effettuata all’endpoint `http://*host*/dataset/update`, mentre, se si vuole effettuare un’interrogazione, a `http://*host*/dataset/query`.

Avendo introdotto i concetti principali di Jena e Fuseki, è ora possibile comprenderne meglio l’architettura mostrata in figura 4.12.

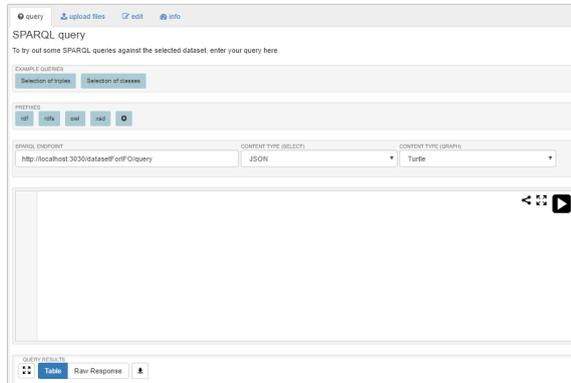


Figura 4.11: Fuseki: endpoint

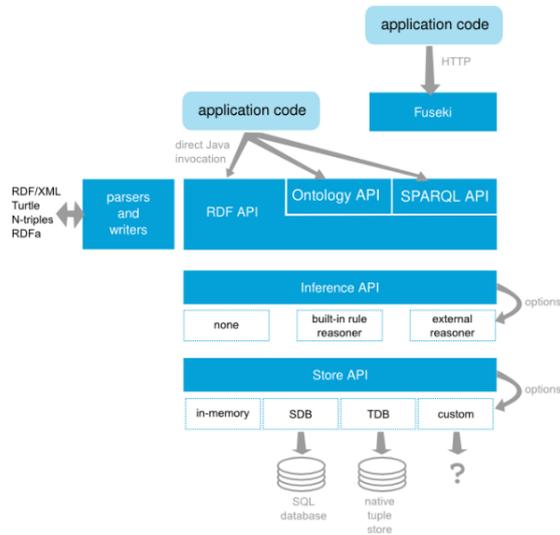


Figura 4.12: Architettura Jena from [60]

RML

Il linguaggio RDF Mapping (RML) è un linguaggio di mappatura definito per esprimere regole di mapping personalizzate per strutture dati eterogenee, al fine di convertirle in un modello dati RDF. RML è definito come superset del linguaggio di mappatura standardizzato W3C R2RML, con l'obiettivo di estenderne l'applicabilità e ampliarne l'ambito, aggiungendo supporti per dati in altri formati strutturati (come CSV, XML, JSON e altri). R2RML è lo standard W3C per esprimere associazioni personalizzate da database relazionali a RDF, ed il suo limite è proprio quello di poter gestire in input solo essi. RML segue esattamente la stessa sintassi di R2RML, di conseguenza i mapping RML sono essi stessi grafi RDF [52]. Dato che RML, a differenza di R2RML, si occupa di diverse serializzazioni di dati, sono necessari linguaggi di query specifici per poter fare riferimento al contenuto di una certa risorsa (per esempio JSONPATH per i file JSON).

Una mappatura RML definisce la mappatura di un qualsiasi dato in un certo formato sorgente strutturato nel formato RDF. L'input per un mapping RML prende il nome di Logical Source, mentre l'output è detto output dataset e consiste nell'insieme di dati RDF generati. Una mappatura RML è composta da una o più triples maps. Una triple map è composta da tre parti: la logical source, la object map, zero o più predicate-object map o referencing object maps.

- Logical Source: utilizzata per determinare i dati di input che devono essere mappati.
 - Reference formulation: consente di specificare quale standard o linguaggio di query viene utilizzato per fare riferimento ai dati della logical source. Le reference formulation sono: ql:CSV, ql:XPath, ql:JSONPath e ql:CSS3;
 - Iterator rml: consente di definire il pattern di iterazione sulla sorgente di input e di specificare quali dati mappare durante ogni iterazione;
 - rml:reference: per fare riferimento alle singole parti dell'input di dati. Sia il valore dell'iteratore che il valore del riferimento devono essere indicati attraverso un'espressione valida secondo la reference formulation definita nella logical source.

- Subject map: definisce il criterio con il quale vengono generati identificatori univoci (URI) per le risorse da mappare. Gli stessi URI sono anche usati come soggetto per ogni tripla RDF prodotta dalla Triple Map;
- Predicate-object map: predicate e object map generano rispettivamente i predicati e gli oggetti per il soggetto generato dalla subject map;
- Referencing Object Maps: essa esegue un'operazione di join tra più mapping. La referencing object map è rappresentata da una risorsa che:
 - ha esattamente una proprietà `rr:ParentTriplesMap`, il cui valore deve essere una triple map (triples map principale);
 - può avere una o più proprietà `rr:joinCondition`, i cui valori devono essere condizioni di join.

La referencing object map collega quindi insieme i valori prodotti da una subject map agli oggetti delle triple prodotte da un'altra map.

L'esecuzione di una mappatura RML richiede una sorgente di input e un documento rappresentate la specifica di mappatura, il quale descrive le triples maps e punta alla sorgente di input. Il processore RML applica ai dati in input le regole di mappatura, seguendo le specifiche indicate dal documento. Per ogni punto di riferimento ai dati presente all'interno della sorgente di input, i valori vengono estratti valutando le corrispondenti espressioni di destinazione e vengono così generate le triple. RML non fornisce dei mezzi per la pulizia dei dati, di conseguenza, sarà necessario procedere con il data cleaning prima di salvare le triple RDF su un triplestore, così da poterle memorizzarle correttamente.

4.3 Implementazione della piattaforma

4.3.1 Progetto di partenza

La piattaforma è stata creata partendo dal progetto realizzato dal collega dottor Roberto Reda. Prima di spiegare i dettagli della presente tesi, è

quindi necessario descrivere quanto da lui svolto. L'ontologia IFO e il mapping RML sono stati da lui realizzati e, successivamente, da me modificati in base alle esigenze di cui il progetto necessitava. Nel paragrafo che segue è riassunto il suo progetto di tesi, per ulteriori approfondimenti consultare il documento da lui redatto [51].

Ontologia IFO

L'ontologia IFO mira a rappresentare i concetti più comuni e importanti nell'ambito del dominio dei dispositivi IoT di fitness e dei wellness appliance. Al fine di identificare i concetti descritti all'interno di questa ontologia, sono state considerate e analizzate attentamente le caratteristiche e le funzionalità fornite dai diversi *wereables* IoT e wellness appliance e dalle applicazioni mobili per la salute disponibili sul mercato. L'elenco di prodotti e fornitori presi in considerazione durante il processo di progettazione include: Apple HealthKit, Microsoft HealthVault, Google Fit, Fitbit, Jawbone, Strava, Runtastic, iHealth e Nokia Health. L'ontologia è stata costruita attorno alla nozione di episodio. Un episodio rappresenta l'insieme di tutti gli eventi possibili che possono essere misurati dai dispositivi IoT e dai sistemi di wellness. Ad esempio, un episodio potrebbe essere la frequenza cardiaca misurata durante una sessione di allenamento da un *wearable wrist worn heart rate monitor* o il peso corporeo della persona misurato da una scala smart. Ad ogni episodio è associato un riferimento temporale e un valore numerico con la relativa unità di misura (figura 4.13). L'ontologia IFO organizza gli episodi in una struttura gerarchica basata sull'ereditarietà singola. Lungo la gerarchia si possono distinguere due principali categorie di episodi:

- le attività fisiche, le quali comprendono qualsiasi tipo di attività che coinvolge il movimento del corpo come camminare, correre, nuotare;
- le misurazioni del corpo, le quali sono relative ai parametri fisiologici di una persona (come il peso corporeo o l'altezza del corpo), o ai segni vitali di essa (come la frequenza cardiaca o la pressione sanguigna);
- categorie minori di episodi che l'ontologia IFO definisce riguardano il sonno e la meditazione.

Altri componenti fondamentali dell'ontologia IFO sono la classe *OWL Measure* e la classe *TimeFrame*, che rispettivamente modellano la misurazione e il riferimento temporale. Queste due classi sono associate alla classe

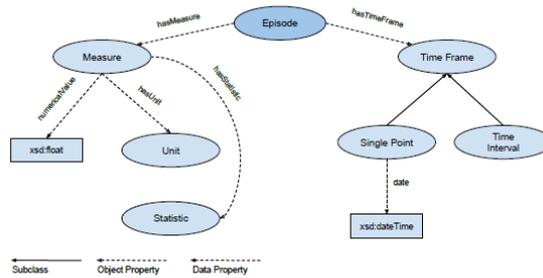


Figura 4.13: Ontologia IFO, from [51]

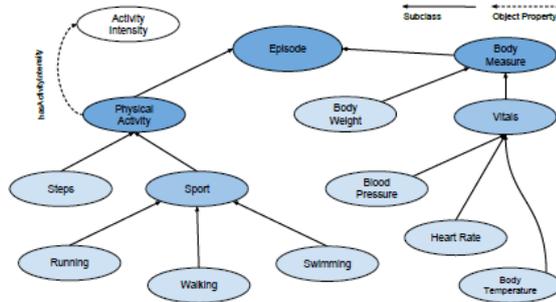


Figura 4.14: Gli episode dell'ontologia IFO, from [51]

Episode tramite le proprietà OWL `hasMeasurement` e `hasTimeFrame` come mostrato in figura 4.14.

Sistema di mappatura

Il sistema di mappatura è stato implementato utilizzando RML, in particolare le specifiche di mappatura sono state definite per quattro sistemi IoT tra quelli che sono stati utilizzati per costruire l'ontologia (Fitbit, Garmin, Apple Health e Nokia Health). Come esecutore di un processo di mappatura è stato utilizzato RML Mapper, implementazione Java di un processore di mappatura RML che supporta i formati di dati XML, JSON e CSV e

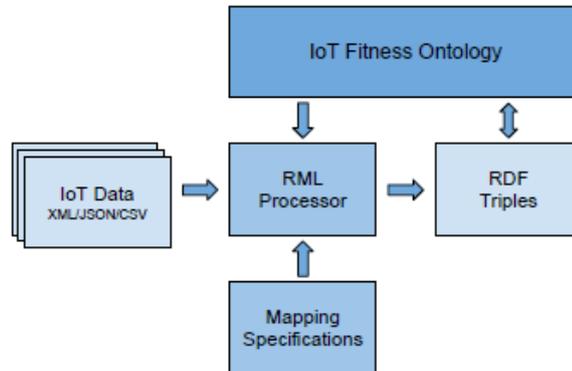


Figura 4.15: Framework del sistema di mapping, from [51]

pertanto non necessita di estensioni o modifiche del software esistente. I dati grezzi raccolti dai dispositivi IoT possono essere recuperati manualmente quando i sistemi sono dotati di funzionalità di esportazione dati (ad esempio, Apple Health): normalmente l'esportazione avviene in formati di serializzazione XML o CSV. Quando una funzione di esportazione dei dati non è direttamente disponibile all'interno del dispositivo o dell'applicazione mobile, i dati raccolti dai sistemi IoT possono essere scaricati dal Cloud, solitamente in formato JSON, tramite API RESTful fornite dal produttore del dispositivo (come per esempio Fitbit). Per annotare semanticamente i dati IoT secondo l'ontologia IFO, ovvero tradurre i dati di input in un grafo RDF, il processore RML richiede specifiche di mappatura per i vari target di dati. Il linguaggio RML consente definizioni di mappatura che possono essere riutilizzate in diverse implementazioni per diversi formati sorgente, riducendo i costi di implementazione.

Architettura generale

La figura 4.15 mostra l'architettura generale del framework realizzato da Roberto Reda. I due componenti principali dell'intero sistema sono l'ontologia IoT Fitness Ontology (IFO) e il processo di mappatura appena descritti.

- il ruolo principale dell'ontologia IFO è quello di fornire una rappresentazione formale dei concetti principali all'interno del dominio IoT del fitness. L'ontologia è un componente essenziale per raggiungere l'interoperabilità, analizzare, integrare, archiviare e trasferire i dati IoT nel modo più preciso e sicuro;
- il processore RML, fornito con le specifiche di mappatura per le varie fonti, consuma i dati grezzi IoT e li trasforma in un grafo RDF, che è lo stesso dato di input semanticamente annotato secondo l'ontologia IFO.

Questa architettura mostra in dettaglio il componente “mapping system” di quella mostrata in figura 4.2.

4.3.2 Modifiche apportate al progetto di partenza

Problema dei blank nodes

I blank node hanno lo scopo di rappresentare concetti che non sono noti o specificati. Essi permettono di definire risorse anonime che non sono identificate da un URI, di conseguenza non possono essere utilizzati per identificare globalmente una risorsa, ma solo all'interno del documento RDF in cui è specificato. Questi nodi indicano semplicemente l'esistenza di una cosa, senza dire nulla di più. Per questo motivo sono anche definiti come variabili esistenziali di un grafo RDF.

A causa dell'assenza di un nome (URI), la manipolazione dei dati contenenti blank node è molto più difficile: essi infatti rendono le operazioni che sarebbero normalmente banali molto più complesse, complicando la vita dei consumatori di dati, soprattutto se le informazioni sono soggette a cambiamenti futuri e sono scarsamente comprensibili.

Questo è un argomento spesso discusso e controverso che divide la comunità del web semantico. Richard Cyganiak ha scritto una breve analisi riguardante il problema dei blank nodes nel post del blog intitolato “Blank nodes considered harmful”. La domanda posta è “I nodi vuoti sono dannosi?”, E risponde: “*Non sempre. A volte sono tollerabili, a volte sono un'ultima risorsa necessaria e talvolta sono abbastanza buoni. Ma non sono mai buoni*”. Richard discute poi diverse situazioni in cui i questi nodi possono essere tollerati, come i dati transitori che non si intende memorizzare,

o le risorse ausiliarie non importanti. Richard conclude: *“Maggiore è la percentuale di nodi vuoti in un set di dati, meno utile è”* [2].

Modifiche apportate all’ontologia IFO e alla mappatura RML

L’ontologia IFO e la mappatura RML realizzate da Roberto Reda sono state modificate in base alle esigenze del progetto.

Il sistema di mappatura realizzato dal mio collega, attraverso l’utilizzo di RML, generava triple aventi come soggetti dei blank nodes. Per eliminare gli svantaggi derivanti dal loro utilizzo e per fare in modo che i dati caricati da un certo utente siano ad esso associati sono state utilizzate le seguenti soluzioni.

Con l’obiettivo di creare un identificatore univoco per le triple RDF in sostituzione dei blank node, è stato necessario modificare l’ontologia IFO inserendo una nuova data property alla classe Episode, così da mantenere l’identificatore sul triplestore. Quando l’utente carica i nuovi dati, al fine di definire il soggetto delle triple attraverso un identificatore univoco, e quindi memorizzare correttamente le triple sul triplestore:

1. viene eseguita una query per recuperare l’id;
2. viene incrementato l’id recuperato, in modo tale che sia diverso da quello usato precedentemente;
3. ai dati degli allenamenti, contenuti nei file json/xml/csv, viene aggiunto l’id appena recuperato e incrementato, in modo tale che RML lo possa utilizzare come soggetto delle triple;
4. update dell’id sul triplestore così da renderlo disponibile per il successivo caricamento dei dati di un utente.

Per risolvere invece il secondo problema, è stata aggiunta all’ontologia IFO una data property `hasUsername` alla classe `Person`. In questo modo quando l’utente effettua una query per la creazione di un grafico, è possibile recuperare dal triplestore i dati da lui precedentemente memorizzati. Nell’ontologia IFO infatti la classe `Episode` e la classe `Person` sono legate dalla object property `isRelativeToAPerson`, la quale permette di sapere che un certo episodio è relativo a una certa persona, la quale ha un certo username indicato dalla data property aggiunta `hasUsername`. La modifica viene

meglio mostrata nella figura 4.16. Oltre all'identificatore, è stato quindi necessario aggiungere anche lo username ai dati degli allenamenti presenti nei file json/xml/csv.

Infine, per unire queste due soluzioni, sono state apportate modifiche al file di mappatura RML, in modo tale da:

- assegnare un identificatore univoco come soggetto delle triple ad ogni episode che deve essere memorizzato sul triplestore, eliminando il problema dei blank node;
- indicare lo username dell'utente che ha caricato i dati, in modo che un utente possa recuperare quelli da lui memorizzati.

4.4 Ontologia applicativa

Con lo scopo di gestire gli utenti della piattaforma e i relativi dati di registrazione, è stata creata un'ontologia applicativa. La figura 4.17 ne mostra la struttura.

L'ontologia applicativa per la gestione degli utenti è composta da:

- tre classi:
 1. User: con data properties `hasName` e `hasSurname` per il nome e cognome dell'utente;
 2. User_Query: con data propertie `hasQuery` per le query inserite dall'utente;
 3. User_Credentials: con data properties `hasUsername` e `hasPassword` per username e password dell'utente;
- due object properties:
 1. `user_has_userCred`, che collega lo user alle sue credenziali;
 2. `user_has_query`, che collega lo user alle sue query.

Su Fuseki sono quindi stati creati due dataset, uno per i dati caricati degli allenamenti e uno per quelli degli utenti.

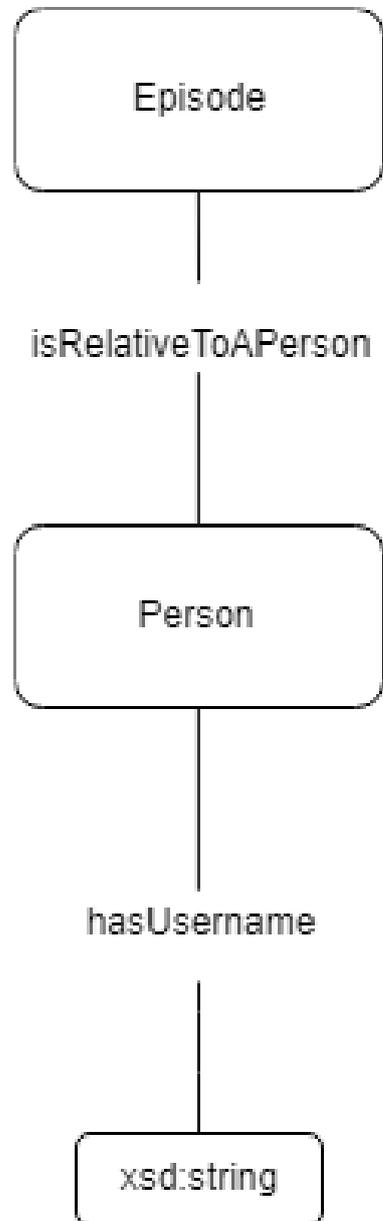


Figura 4.16: Modifica ontologia IFO

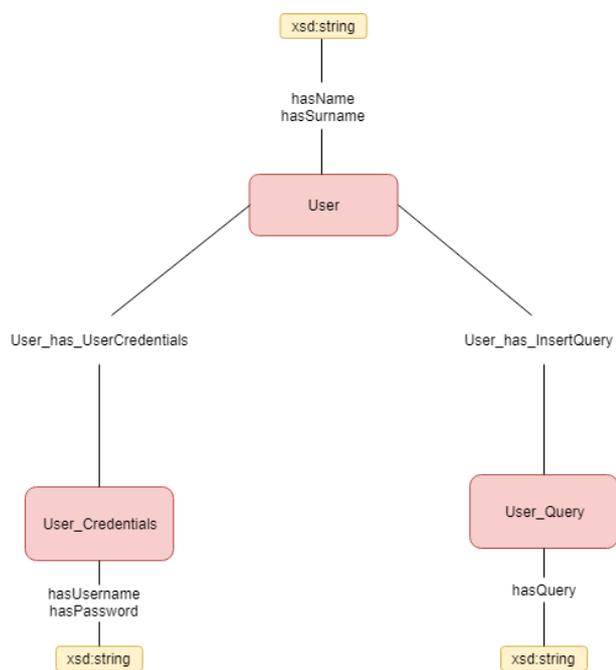


Figura 4.17: Ontologia applicativa

4.5 Diagramma della piattaforma

Per comprendere meglio il funzionamento della piattaforma, i componenti principali e le interazioni tra essi, è stato realizzato il diagramma mostrato in figura 4.18.

Osservandolo è possibile notare un gestore per le query e uno per gli update: il primo permette di gestire le esecuzioni delle query sui dati presenti sul triplestore, mentre il secondo l'inserimento, la modifica e la cancellazione delle triple. Vengono ora descritte più dettagliatamente tutte le operazioni e interazioni.

Signin

Per effettuare la registrazione alla piattaforma è presente una pagina attraverso la quale l'utente può inserire nome, cognome, username e password. La servlet ricaverà questi dati e li passerà al gestore degli update che provvederà a memorizzarli sul triplestore. In automatico per l'utente verranno memorizzate anche due query per la creazione dei grafici relativi al battito cardiaco e al peso.

Login

Al momento del login l'utente, attraverso l'apposita pagina, inserisce username e password. La servlet di login recupera i dati inseriti e si rivolge al gestore delle query per eseguire quella di login. Il risultato sarà un boolean: true se l'utente è già registrato, false altrimenti.

Caricamento dei dati

L'utente può caricare i dati rilevati dai sensori dei dispositivi in due modalità:

- con l'upload del file ottenuto a seguito del download dal sito del fornitore;
- attraverso la piattaforma, la quale provvede direttamente a scaricare i file dal sito del fornitore.

Sulla piattaforma sono presenti entrambe le modalità in quanto la seconda non è ancora stata implementata per tutti i dispositivi per i quali è

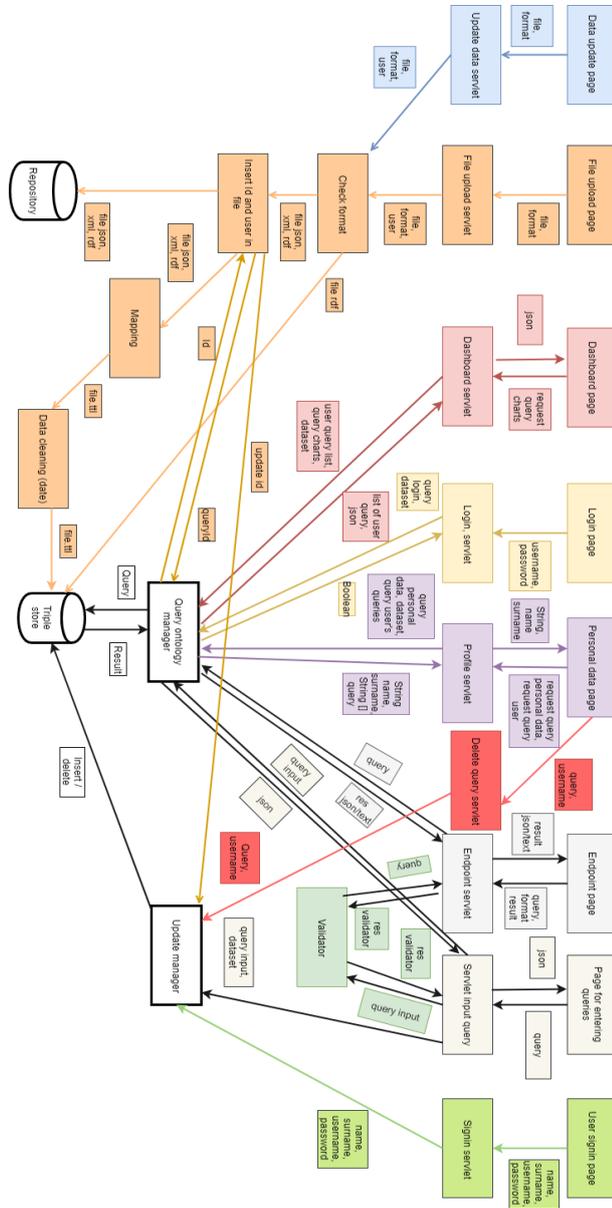


Figura 4.18: Struttura della piattaforma

possibile memorizzare i dati sul triple store. Per una questione di comodità per l'utente, l'idea è che, una volta terminata questa implementazione, la prima opzione venga totalmente rimossa.

Upload del file

Una volta che l'utente carica il file e ne indica il formato attraverso la pagina del sito, la servlet recupera i dati inseriti e li passa al componente "Check format", il quale controlla il formato: se il file è RDF allora le triple vengono memorizzate direttamente sul triplestore, altrimenti devono essere eseguite alcune operazioni. Per prima cosa i dati del file dovranno essere aggiornati, inserendo lo username dell'utente che ha caricato il file e l'id per definire l'identificatore univoco delle triple. Bisognerà quindi:

- eseguire la query SPARQL per recuperare l'identificatore;
- incrementare l'identificatore;
- effettuare l'update dell'identificatore in modo che sia pronto per il caricamento successivo.

Per tenere traccia dei file caricati, essi vengono memorizzati su un repository, il quale contiene una directory per ogni utente. Il file aggiornato viene poi sottoposto al mapping RML, il quale restituirà come risultato un file RDF. Dal momento che RML non fornisce funzionalità per la pulizia dei dati, sull'output ottenuto dalla mappatura viene effettuato il data cleaning per la modifica del formato delle date, in modo tale da memorizzarle correttamente sul triplestore.

Update dei dati

L'utente può caricare i propri dati direttamente attraverso la piattaforma, senza dover prima scaricare i file dal sito del fornitori. I dati verranno mappati in formato RDF e memorizzati sul triple store. Le operazioni che verranno poi effettuate sono esattamente le stesse che sono state descritte nell'alternativa precedente.

Personal data

Nella pagina relativa ai personal data l'utente può visualizzare i dati personali inseriti al momento della registrazione e tutte le sue query. La servlet profile richiederà quindi al gestore delle query di recuperare i dati personali e le query per esso memorizzate. Una volta eseguite le query per il recupero delle informazioni il risultato relativo ai dati personali è convertito in una stringa, mentre le query in un array di stringhe, in modo da poter visualizzare le informazioni sulla pagina della piattaforma. L'utente attraverso questa pagina ha anche la possibilità di eliminare le query per i grafici che non sono più di suo interesse.

Input query

Per creare un nuovo grafico l'utente deve inserire una query SPARQL, e, se desidera, può scegliere di memorizzarla, in modo tale da poter visualizzare sempre sulla propria dashboard i grafici delle query già effettuate. La query inserita dall'utente ha però alcune limitazioni dovute all'impossibilità di creare grafici per qualsiasi tipologia di query. Di conseguenza, la servlet, una volta recuperata la query inserita e la scelta di memorizzazione, provvede al controllo della validazione sintattica e al rispetto delle limitazioni indicate. Se entrambi i controlli vengono superati allora la query viene eseguita e, se richiesto dall'utente, memorizzata. Una volta ottenuto il risultato viene creato e visualizzato il grafico. Nel caso in cui la query non producesse alcun risultato (ovvero un grafico vuoto), essa non verrà memorizzata anche se indicato dall'utente.

Dashboard

Sulla pagina relativa alla dashboard vengono visualizzati i grafici dell'utente. La servlet della dashboard quindi deve indicare al gestore delle query di recuperare l'insieme delle interrogazioni precedentemente memorizzate per l'utente, e di eseguirle al fine di ottenere i dati necessari per la realizzazione dei grafici. I dati ottenuti sono quindi passati in formato json alla pagina che, attraverso una funzione javascript, provvede alla creazione dei grafici.



Figura 4.19: Pagina di benvenuto

Endpoint

Non essendo disponibile una funzionalità di Fuseki che permetta di fornire l'interfaccia relativa all'endpoint, è stata creata una pagina attraverso la quale l'utente potrà digitare la query e scegliere il formato in cui vuole visualizzare il risultato (json o testuale). La servlet dell'endpoint controllerà sintatticamente la query attraverso un validatore: se è corretta viene passata al gestore delle query per l'esecuzione e viene restituito il risultato nel formato richiesto, altrimenti verrà indicato l'errore.

4.5.1 Pagine della piattaforma

Vediamo ora più in dettaglio le pagine che l'utente potrà navigare attraverso la piattaforma e le loro funzionalità.

Pagina di benvenuto

La pagina mostrata in figura 4.19 è quella che l'utente visualizza nel momento in cui accede al sito. Essa avrà come scopo quello di spiegare in che cosa consiste il portale, descrivendo in particolare l'ontologia IFO e cosa l'utente può fare attraverso questa piattaforma. Da qui l'utente potrà quindi decidere di registrarsi al servizio e successivamente effettuare il login, o procedere direttamente con il login nel caso in cui fosse già registrato. Quando l'utente effettua il logout sarà indirizzato a questa pagina. È possibile inoltre passare alla pagina dell'endpoint successivamente descritta.

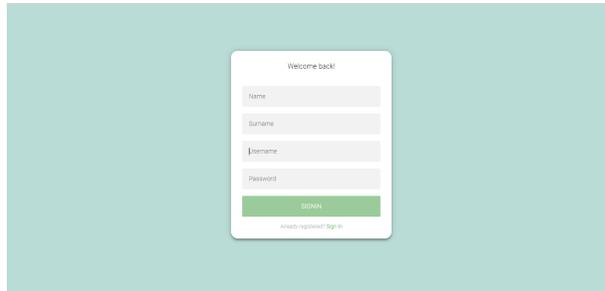


Figura 4.20: Pagina di registrazione

Pagina di registrazione

La pagina di registrazione (4.20) è quella a cui l'utente accede quando decide di registrarsi al sito. Questa pagina è raggiungibile sia da quella di benvenuto che da quella di login. In particolare, è presente una form contenente i campi richiesti all'utente per la registrazione: nome, cognome, username e password. Una volta terminata la loro compilazione, l'utente conferma la registrazione con un bottone, il quale effettuata una submit inviando i dati alla servlet di signin attraverso una POST. La servlet verificherà che non esista già un utente registrato con lo stesso username, e, in caso positivo, provvederà all'inserimento dei dati del nuovo utente sul triplestore. Al momento della registrazione in automatico verranno memorizzate per l'utente due query per la creazione dei grafici relativi al battito cardiaco e al peso. Inizialmente i grafici visualizzati sulla dashboard (4.29) saranno vuoti in quanto sul triple store non sono presenti informazioni sul nuovo utente, ma si disegneranno nel momento in cui egli caricherà i dati necessari relativi al proprio peso e battito cardiaco.

Pagina di login

Quando l'utente decide di accedere al sito deve effettuare il login: questa pagina (4.21) è raggiungibile da quella di benvenuto e richiede username e password di registrazione. A tale scopo nella pagina è presente una form contenente i campi richiesti all'utente necessari al login. L'utente a questo punto conferma l'azione con un bottone, il quale effettua una submit, inviando i dati alla servlet di login attraverso una POST. La servlet di login

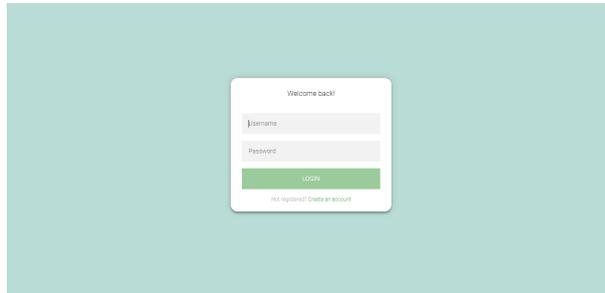


Figura 4.21: Pagina di login



Figura 4.22: Pagina dei dati personali

controlla che l'utente sia già registrato, verificandone l'esistenza sul triplestore: in caso positivo viene mostrata la pagina relativa alla dashboard, altrimenti un messaggio di errore.

Pagina dei dati personali dell'utente

Questa pagina (4.22) permette all'utente di visualizzare i dati da esso inseriti al momento della registrazione. Il recupero dei dati personali avviene attraverso la servlet `profile`, la quale provvede ad eseguire la query per il recupero delle informazioni dell'utente che ha effettuato il login. Tra i dati personali sono presenti anche le query per lui memorizzate: l'utente avrà la possibilità di eliminarle, ovvero di cancellare dalla dashboard quei grafici che non sono più di suo interesse. Attraverso il bottone "Elimina" l'utente sceglie il grafico che desidera eliminare, e, attraverso una POST, viene passata alla servlet la query di cui deve essere effettuata la delete.

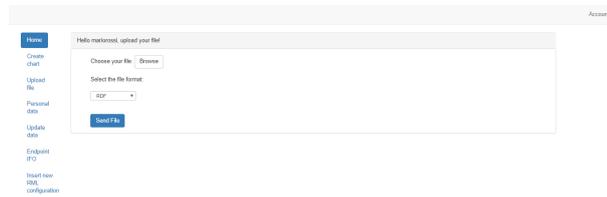


Figura 4.23: Pagina di upload del file

Pagina upload del file

La pagina di upload del file (4.23) permette all'utente di caricare un file contenente i dati che si vogliono memorizzare. L'utente dovrà selezionare il file e indicare il dispositivo di provenienza (Nokia Health, Fitbit, Garmin, Health kit): a questo punto si procederà con il mapping RDF sui dati caricati e le triple così ottenute saranno memorizzate sul triplestore. L'utente avrà anche la possibilità di caricare file che sono già in formato RDF: in questo caso le triple in esso contenute verranno memorizzare direttamente sul triplestore, in quanto non vi è necessità di mapping. Una volta che l'utente clicca il bottone Send File, dopo aver selezionato file e formato, viene effettuata una submit inviando i dati alla servlet di upload attraverso una POST. La servlet provvederà ad eseguire tutte le operazioni necessarie.

Pagina update dei dati

Questa pagina (4.24) permette all'utente di caricare i propri dati direttamente sul triple store. Qui non è necessario che l'utente scarichi prima i file dal sito dei fornitori dei dispositivi: i file vengono recuperati direttamente dalla piattaforma, effettuando la richiesta ai fornitori dei dispositivi selezionati dall'utente attraverso la pagina. L'utente dovrà cliccare il bottone rappresentante il dispositivo di cui vuole memorizzare i dati, e verrà reindirizzato alla pagina di login del device scelto: una volta inserite le credenziali si ritornerà alla piattaforma IFO con i dati memorizzati sul triple store.

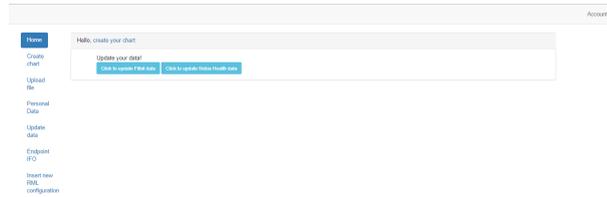


Figura 4.24: Pagina di update dei dati

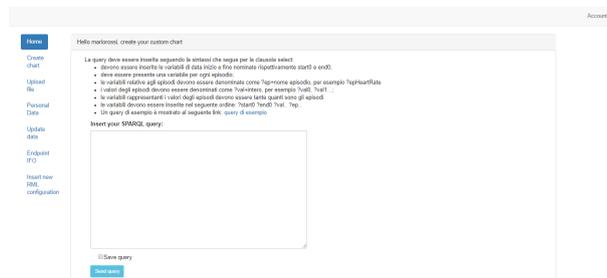


Figura 4.25: Pagina per la creazione dei grafici

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ifo: <http://purl.org/ifo/#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX op: <http://www.w3.org/2005/speleth-Functions#>
SELECT ?start0 ?end0 ?val0 ?epBodyWeight
WHERE {
  ?epBodyWeight rdf:type ifo:BodyWeight .
  ?measure rdf:type ifo:Measure .
  ?person rdf:type ifo:Person .
  ?timeFrame rdf:type ifo:TimeInterval .
  ?epBodyWeight ifo:hasTimeInterval ?timeFrame .
  ?epBodyWeight ifo:isRelativeToAPerson ?person .
  ?person ifo:hasSurname ?name .
  ?epBodyWeight ifo:hasMeasure ?measure .
  ?measure ifo:hasNumericalValue ?val0 .
  ?timeFrame ifo:startDate ?start0 .
  ?timeFrame ifo:endDate ?end0 .
  FILTER( REGEX(str(?name),"mariorossi") ) && ((?start0 >= "2017-06-10T11:00:00"^^xsd:dateTime)
&& ((?end0 <= "2017-06-10T21:00:00"^^xsd:dateTime) && (?start0=?start0) && (?end0=?end0) ) )
ORDER BY ?start0

```

Figura 4.26: Esempio di query

Pagina creazione grafici

La pagina (4.25) permette all'utente di inserire query SPARQL per la creazione dei grafici. L'utente quindi deve scrivere la query nell'apposita area, e scegliere se memorizzarla o meno. Con il bottone Send Query viene effettuata una submit inviando le informazioni tramite POST. La servlet recupera la query inserita dall'utente, la scelta sulla sua memorizzazione, e lo user che l'ha inserita, in modo tale da poterla memorizzare, se richiesto, sul triplestore per quel preciso utente. Per prima cosa si effettua la validazione sintattica della query e si verifica che siano rispettate le limitazioni. Nel caso in cui vengano superati questi due controlli si procede con l'esecuzione della query:

- se si ottiene un risultato verrà visualizzato il grafico e, se l'utente ha scelto di memorizzarla, la query verrà salvata sul triplestore;
- se non si ottiene nessun risultato verrà visualizzato un grafico vuoto e la query non verrà memorizzata anche se richiesto dall'utente.

Nel caso in cui non venga superato anche solo uno dei due controlli verrà segnalato l'errore all'utente.

Pagina di visualizzazione del grafico creato

Questa pagina permette all'utente di visualizzare il grafico ottenuto dai risultati dalla query inserita attraverso la pagina appena descritta. Ad esempio, inserendo la query rappresentata in figura 4.26, il grafico che si ottiene è mostrato in figura 4.27.



Figura 4.27: Pagina che mostra il risultato della query

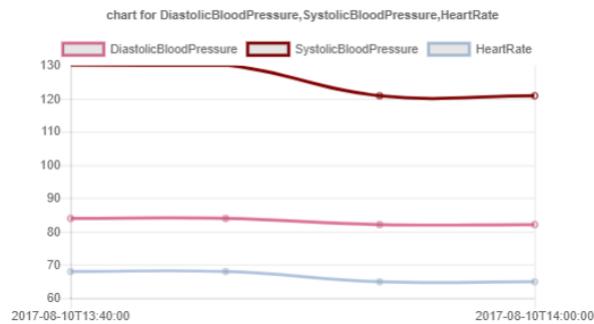


Figura 4.28: Grafico

Da esso è possibile visualizzare come il peso è cambiato in un certo intervallo di tempo indicato dall'utente attraverso la query.

Eseguendo un'altra query è invece possibile ottenere il grafico 4.28, il quale mostra la pressione sistolica e diastolica, e il battito cardiaco.

In questo preciso esempio i dati della pressione sistolica e diastolica sono ottenuti da Nokia Health mentre quelli del battito cardiaco da Garmin. Questo è uno dei grandi vantaggi derivanti dall'utilizzo di questa piattaforma, la quale, trasformando i dati in RDF, permette di analizzarli e utilizzarli insieme, cosa che senza un formato comune non si sarebbe potuta fare. Se le query inizialmente possono sembrare lunghe e complesse, dopo qualche prova risulteranno più familiari. In essa dovranno essere indicati gli episodi (in questo caso battito cardiaco, pressione sistolica e diastolica) che si vogliono visualizzare nel grafico e l'intervallo di tempo: la struttura della



Figura 4.29: Pagina dashboard

query sarà sempre la stessa e cambierà solo in base al numero degli episodi richiesti.

Dashboard

La pagina contenente la dashboard (4.29) verrà visualizzata dall'utente appena effettuato il login: essa contiene i grafici che l'utente ha precedentemente creato e memorizzato. Per visualizzare i grafici sulla dashboard è stata utilizzata una servlet, la quale si occupa di recuperare tutte le query memorizzate per l'utente ed eseguirle, recuperando così i dati necessari alla realizzazione dei grafici. Al fine di creare i grafici, i risultati delle query SPARQL vengono convertiti in formato JSON e passati a una funzione javascript.

Endpoint

La pagina relativa all'endpoint SPARQL (4.30) permette all'utente di interrogare il contenuto del triplestore attraverso delle query SPARQL. La servlet recupera la query inserita dall'utente, verifica che sia sintatticamente corretta e, in caso positivo la esegue, altrimenti segnala l'errore in modo tale che possa essere corretta.

Pagina per l'inserimento di un nuovo dispositivo

Attraverso questa pagina (4.31) l'utente può inserire un nuovo fornitore caricando il relativo file di mappatura RML. In questo modo, anche i dati

Figura 4.30: Pagina endpoint

Figura 4.31: Pagina per l'inserimento di un nuova mappatura RML

raccolti dal nuovo device possono essere convertiti in RDF e memorizzati sul triplestore insieme a quelli degli altri dispositivi. Così facendo il numero di dati RDF presenti sul triplestore aumenterà sempre di più e la conoscenza che si potrà estrarre sarà sempre maggiore.

4.5.2 Dettagli implementativi

Uno tra gli altri obiettivi di questo progetto era quello di renderlo il più possibile modulare, in modo tale da semplificarne lo sviluppo, il test, la manutenzione, e fare in modo che fosse, per quanto possibile, indipendente dalle scelte tecnologiche effettuate. Per esempio, se in un futuro si decidesse di non utilizzare più Fuseki come server SPARQL, esso deve poter essere sostituito senza sconvolgere l'intero lavoro. Questo perché il progetto dovrà essere modificato ed espanso da altri sviluppatori. Per comprendere

meglio come è stato pensato e concepito il codice segue una descrizione più dettagliata di alcune scelte implementative.

Gestione dei dispositivi e delle loro informazioni

Per tenere traccia dei dispositivi di cui è già stato realizzato il mapping RML, e di cui è quindi possibile memorizzare i dati come triple RDF sul triplestore, è stato creato un file json.

I dispositivi attualmente inseriti sono Fitbit, Nokia Health, Health Kit e Garmin. Nel file viene indicato il nome del dispositivo selezionabile dall'utente, la logical source di RML, il file di mappatura RML e la classe che permette di modificare il file in input con l'identificatore e lo username. La classe indicata in questo file verrà poi istanziata attraverso l'istruzione `Class.forName(className).newInstance()`. È necessario indicare le logical source in quanto esse sono fisse all'interno dei documenti di mappatura. Di conseguenza, i dati che devono essere mappati, devono per forza essere inseriti all'interno del file indicato come logical source.

Il file json è stato creato anche per consentire in modo semplice l'inserimento di un nuovo dispositivo a quelli già presenti: grazie ad esso infatti, per aggiungere un device basterà inserire un nuovo array json indicando i dati richiesti. L'unica cosa che dovrà essere fatta è creare una classe che estende un'interfaccia (`IFileModify`), e che implementi il metodo richiesto di modifica del file con id e username. Questo è necessario in quanto il modo in cui devono essere aggiornati i file cambia da formato a formato e in base ai nomi dei tag presenti all'interno di essi, di conseguenza non è possibile apportare le modifiche in modo generale. Dallo stesso file json inoltre vengono recuperati i dispositivi selezionabili dall'utente nel menu a tendina presente all'interno della pagina di caricamento del file, così da non dover aggiornare neanche la parte grafica.

Inserimento dello username e dell'identificatore nei dati

È presente un'interfaccia `IFileModify` contenente il metodo `modifyFile`, la quale è implementata dalle classi che hanno come scopo quello di aggiornare il contenuto del file caricato dall'utente. La modifica consiste nell'aggiungere l'identificatore univoco e lo username di chi ha caricato i dati, ed avviene in maniera diversa a seconda del dispositivo di provenienza (Nokia Health, Fitbit, Health Kit, Garmin), in quanto non esistono standard comuni a cui

uniformarsi. L'identificatore univoco è utile per sostituire i blank node delle subjectMap RML, e quindi permette di memorizzare le triple sul triplestore senza gli svantaggi derivanti dal loro utilizzo. Lo username invece permette di associare i dati all'utente che li ha caricati.

La soluzione attualmente adottata funziona, ma non è sicuramente la migliore da un punto di vista logico; inoltre riduce di molto la modularità del codice. Ciò è stato però necessario, in quanto, come spiegato, non esistono standard a cui uniformarsi, di conseguenza i formati dei file e la loro struttura sono tra loro diversi. Non potendo passare dinamicamente valori al documento di mappatura RML per i soggetti delle triple, questa è stata l'unica soluzione attualmente trovata.

Update dei dati

È stato possibile effettuare l'update dei dati (senza caricamento del file da parte dell'utente) grazie alla libreria realizzata dal collega Christian Manfredi. Grazie alla modularità del codice è stata effettuata in modo semplice la sua integrazione.

Per sapere quali dati devono essere scaricati, per ogni utente è stato creato un file json il quale tiene traccia, per ogni dispositivo, di quando è stata l'ultima volta che è stato effettuato l'update. In questo modo, ogni volta che l'utente vuole scaricare i nuovi dati, verranno recuperati solo quelli che sono stati aggiunti successivamente alla data indicata dal file per il dispositivo richiesto, senza scaricare tutti i dati ogni volta.

Caricamento dei file e mapping

Per prima cosa viene verificato il dispositivo di provenienza e il formato del file caricato dall'utente. Se il formato è già RDF il mapping non viene effettuato, altrimenti vengono settati tutti i parametri necessari per una corretta esecuzione del mapping: nome della logical source e del file di mappatura RML. Queste informazioni vengono estratte dal file json di configurazione dei dispositivi.

È presente quindi un'interfaccia `IFileParser`, contenente il metodo `parseFile`, la quale è implementata dalle classi `FileParser` e `FileParserRDF`. `FileParserRDF` è utilizzata nel caso in cui il file caricato dall'utente attraverso la pagina sia già in formato RDF, mentre `FileParser` per tutti gli altri dispositivi.

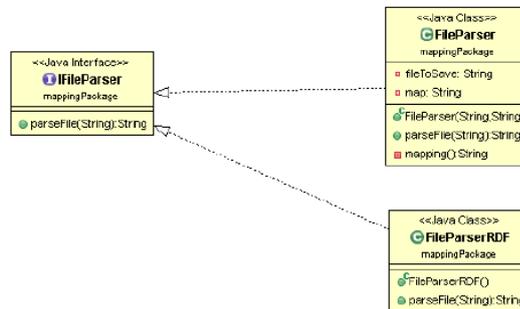


Figura 4.32: Diagramma delle classi per il parsing del file

E' stato quindi implementato un metodo per recuperare le informazioni necessarie al settaggio dei parametri utili al mapping, le quali si trovano nel file json precedentemente descritto. A questo punto:

- si procede al salvataggio del file nella directory dell'utente sul repository: il nome del file salvato nella cartella dell'utente sarà dato dalla concatenazione di `username + timeStampDiCaricamento + nomeDelDispositivo`;
- esecuzione del mapping RML;
- salvataggio delle triple ottenute sul triplestore.

In particolare, il mapping è caratterizzato dalle seguenti operazioni:

- salvataggio del contenuto del file caricato dall'utente su un file con nome uguale alla logical source in modo tale da poter effettuare il mapping RML;
- esecuzione del comando di mapping RML;
- pulizia delle date sui dati ottenuti dal mapping.

Il mapping viene realizzato eseguendo il comando `java -jar RML-Mapper.jar -m file.rml.ttl -o outputMapping.ttl`, dove il parametro che segue `-m` indica il nome del file di mappatura, mentre quello che segue `-o` il nome del file di output. Se il valore di ritorno è 0 allora vuol dire che il mapping è stato terminato con successo, se invece è 1 significa che ci sono stati dei problemi durante l'esecuzione del comando.

Pulizia dei dati

Come spiegato precedentemente, RML non fornisce dei mezzi per la pulizia dei dati, di conseguenza, è necessaria un'operazione di data cleaning prima del salvataggio delle triple sul triplestore. In particolare è stato implementato il data cleaning sulle date.

La modifica delle date è stata effettuata sul file di output ottenuto dal mapping, attraverso l'utilizzo delle regular expressions, in modo tale da ottenere il corretto formato `xsd:dateTime` ([1]).

Gestione delle query

Per la gestione generale dell'esecuzione delle query è presente un'interfaccia `IQueryManager`. L'interfaccia viene implementata da una classe astratta `FusekiQueryManager`, la quale gestisce l'esecuzione delle query su Fuseki: grazie alla classe astratta le parti comuni di tutte le query, ovvero esecuzione e chiusura, sono da essa implementate. La parte che può cambiare è il come estrarre i dati dai risultati di una query. Il risultato di una query SPARQL infatti può essere convertito in JSON, XML, CSV, testo, TSV o altro in base alle esigenze. Per questo motivo è stato creato un metodo astratto (`takeResultQuery()`) nella classe astratta: le classi che estendono `FusekiQueryManager` implementeranno questo metodo, ovvero indicheranno come il risultato dovrà essere convertito e quindi come dovranno essere estratti i dati di interesse dal risultato. Questo è stato possibile utilizzando i tipi generici, in modo tale da potere restituire indipendentemente interi, stringhe, liste... Nel caso in cui si decidesse di non utilizzare più Fuseki, ma un altro web server SPARQL, sarebbe sufficiente creare una nuova classe che implementa l'interfaccia `IQueryManager`.

Tutte le classi che estendono la classe astratta `FusekiQueryManager`, e implementano il metodo `takeQueryResult()`, permettono quindi di recuperare dal risultato della query SPARQL i dati nel formato desiderato. Queste classi sono:

- `QueryDashboard`: per ottenere i valori necessari alla realizzazione dei grafici. Il formato restituito è json;
- `QueryEndpointJson.java`: per ottenere il risultato delle query inserite attraverso l'endpoint dall'utente che richiede il risultato in formato json;

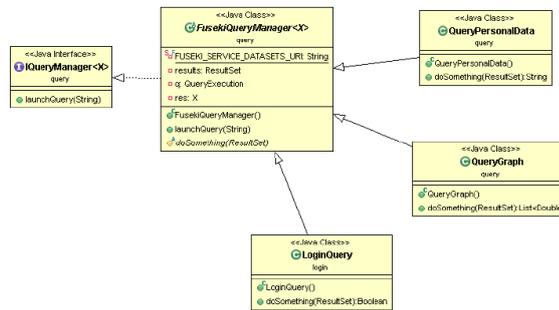


Figura 4.33: Diagramma delle classi la gestione delle query

- QueryEndpointText.java: per ottenere il risultato delle query inserite attraverso l'endpoint dall'utente che richiede il risultato in formato testuale;
- QueryGetQueryUser: per ottenere l'insieme delle query memorizzate da un certo utente. L'insieme delle query viene restituito in una lista di stringhe;
- QueryId: per recuperare il valore dell'identificatore univoco, ritornando quindi un intero;
- QueryLogin: per verificare che un utente sia già registrato alla piattaforma, controllando l'esistenza dello username e della relativa password. Viene restituito un boolean;
- QueryPersonalData: per ottenere i dati di registrazione dell'utente, restituisce una stringa;
- QueryUsernameExist: per verificare al momento della registrazione se esiste già un utente con lo stesso username. Viene restituito un boolean.

Dato che esistono due dataset, uno per gli utenti e uno per i dati dell'utente, per ogni query deve essere indicato il dataset corretto sul quale eseguirla.

Gestione degli update del triplestore

È presente una classe FusekiUpdateManager la quale permette di effettuare gli aggiornamenti sul triplestore presente su Fuseki.

Validazione delle query

Per effettuare la validazione sintattica delle query SPARQL è stato utilizzato sparql.org, integrato anche in Fuseki per il controllo delle query. Per realizzare il validatore viene effettuata una richiesta HTTP a sparql.org e recuperato il risultato: se non sono presenti errori sintattici allora la query viene eseguita sul triplestore, in caso contrario verranno mostrati all'utente gli errori riscontrati. La validazione sintattica viene effettuata ogni volta che l'utente inserisce una query, quindi attraverso l'endpoint o per la creazione dei grafici. Per quanto riguarda la realizzazione di nuovi grafici è stato necessario controllare anche che la query inserita dall'utente rispettasse delle restrizioni, le quali sono indicate anche sulla pagina del sito. Nella clausola SELECT:

- devono essere inserite le variabili di data inizio e fine nominate rispettivamente start0 e end0;
- deve essere presente una variabile per ogni episodio;
- le variabili relative agli episodi devono essere denominate come ?ep+nome episodio, per esempio ?epHeartRate;
- i valori degli episodi devono essere denominati come ?val+intero, per esempio ?val0, ?val1...;
- le variabili rappresentanti i valori degli episodi devono essere tante quanti sono gli episodi;
- le variabili devono essere inserite nel seguente ordine: ?start0 ?end0 ?val.. ?ep..

Questo controllo è necessario in quanto risulterebbe molto complesso provvedere alla creazione di un grafico partendo da una qualsiasi query. Per semplificare il lavoro dell'utente, sul sito sono presenti esempi di query che possono già essere utilizzate, indicando l'intervallo di tempo che si preferisce.

Visualizzazione dei grafici sulla dashboard

Quando l'utente accede alla dashboard, esso dovrà poter visualizzare tutti i grafici che ha precedentemente creato attraverso le query SPARQL memorizzate. Al primo login la dashboard dell'utente sarà caratterizzata da due grafici vuoti (ottenuti dalle query che vengono memorizzate in automatico al momento della registrazione), in quanto sicuramente non avrà ancora caricato dati. La visualizzazione avviene nel seguente modo:

- ogni grafico creato dall'utente corrisponde a una query da esso memorizzata sul triplestore;
- vengono recuperate ed eseguite tutte le query memorizzate dall'utente;
- il risultato di tutte le query viene passato alla pagina in formato json;
- tramite javascript vengono recuperati i dati per i singoli grafici e visualizzati sulla dashboard.

Query e update fissi

Per rendere il codice più pulito e modulare, le query e gli update "standard" sono gestiti all'interno di due classi, rispettivamente una per gli update e una per le query, in modo da esternalizzarli dal contesto tecnologico, in questo caso dall'uso di Fuseki e delle servlet. Per update e query "fissi/standard" si intendono tutti quelli che devono essere utilizzati per il funzionamento del sito. Di conseguenza le query e gli update presenti in queste classi potranno essere riutilizzati anche nel caso in cui venissero cambiate tali tecnologie. Le query fisse sono:

- query di login;
- query per il recupero dei dati personali;
- query per ottenere la lista delle query memorizzate per un utente;
- query per recupero dell'identificare univoco;
- query per verificare se un certo username è già stato scelto da un altro utente.

Per quanto riguarda invece gli update:

- salvataggio delle query per un utente;
- aggiornamento dell'id;
- eliminazione di una query di un certo utente;
- registrazione.

4.5.3 Avvio

Le librerie utilizzate sono:

- Jena;
- RML-Mapper;
- Javax.servlet;
- Commons-fileupload-1.3.3;
- Org.json, libreria per la manipolazione dei dati json;
- Chart.js, libreria JavaScript utilizzata per la creazione dei grafici. Per il disegno del grafico si avvale del Canvas di HTML5, supportato dalla maggior parte dei browser moderni. Permette inoltre di rispondere alle interazioni dell'utente, adattandosi in modo "responsive" al device su cui viene prodotto.

La prima cosa che deve essere fatta è procedere con l'installazione di Tomcat e Fuseki. Una volta attivato il server Fuseki attraverso l'apposito comando `./fuseki-server -update -mem /ds`, è possibile collegarsi a `localhost:3030`, e creare due nuovi dataset, uno per l'ontologia IFO e l'altro per quella di dominio. Andare quindi nella sezione manage dataset e creare:

- un data set con nome dsIFO: caricare il file del dataset relativo all'ontologia IFO;
- un dataset con nome dsUser: caricare il file del dataset per la gestione degli utenti relativo all'ontologia applicativa.

Avendo utilizzato Eclipse come ambiente di sviluppo, è stata creata una configurazione di Tomcat direttamente su di esso. Quindi:

- aprire la view relativa ai server su Eclipse andando sul tab Windows, Show view, Servers, e procedere con la creazione del web server Tomcat.
- andare sul tab Windows, Preferences, Server, Runtime Environments e aggiungere il web server corretto.
- cambiare tab: Run, Run Configuration, Apache Tomcat, andare sul Tab arguments e come working directory selezionare Other e inserire come percorso la cartella FileForMapping presente all'interno del progetto.

Il progetto è stato caricato su BitBucket: per importarlo aprire la view del Git di Eclipse, e andare su Windows, Show view, Git, Git Repositories. Una volta aperta la scheda cliccare su Clone a Git Repository. Ora per importare il progetto nel project explorer, tasto destro sul progetto nel git, Import project. Tasto destro sul progetto importato, properties, Java Build Path: importare le librerie richieste e aggiustare i riferimenti per Tomcat e la versione di java. Copiare la cartella config all'interno della directory home del proprio profilo (diversa a seconda del sistema operativo). Il path dovrà essere user-home/RMLDataRetriever/config. Verificare che la cartella config contenga fitbit.xml e nokia-health.xml.

A questo punto spostarsi su Deployement Assembly, add, e aggiungere le java build path entries e selezionarle tutte. Infine, tasto destro sulla pagina welcomePage.html, Run, Run on Server per avviare il progetto.

Capitolo 5

Conclusioni

Oggi, i device IoT, grazie ai sensori su essi installati, permettono di ottenere molte informazioni sugli allenamenti e in generale sulla salute dell'utente, rilevando informazioni quali battito cardiaco, calorie bruciate, distanza percorsa e tanto altro.

I dati aperti, che per definizione consentono l'accesso a chiunque abbattendo le restrizioni tecnologiche, permettono di rendere pubblici i dati raccolti dai sensori presenti sui dispositivi di un utente. Attraverso un portale Linked Open Data inoltre qualsiasi utente può caricare nuovi dati, aumentando sempre di più la dimensione del dataset, e, attraverso un endpoint SPARQL, eseguire query per ottenere informazioni.

Trasformare i dati raccolti dai sensori dei dispositivi IoT in open data permetterebbe di aiutare vari settori, tra cui quello sanitario. L'integrazione dei dati proveniente da sorgenti diverse consentirebbe di ottenere nuove informazioni e conoscenza: ciò abiliterebbe la possibilità di condurre studi scientifici sui dati raccolti, permettendo addirittura la prevenzione di alcune patologie. Attraverso i dispositivi è possibile infatti osservare i comportamenti degli utenti e il conseguirsi dello sviluppo di certe malattie. In sostanza, i dati raccolti dai dispositivi indossabili potranno essere utilizzati per ricerche mediche e andare quindi a vantaggio di tutta la popolazione che, automonitorandosi, involontariamente aiuta ricercatori di tutto il pianeta a combattere malattie che da sempre affliggono le persone di tutto il mondo, come ad esempio i disturbi cardiovascolari.

I dispositivi IoT utilizzati nel mondo del fitness sono però caratterizzati da un'elevata eterogeneità di formati di rappresentazione dei dati e dall'as-

```
rml:logicalSource [  
    rml:source "garmin.xml";  
    rml:referenceFormulation ql:XPath;  
    rml:iterator "//Lap";  
];
```

Figura 5.1: Logical source RML

senza di standard comuni. Questo porta a una mancanza di interoperabilità in quanto i dati raccolti dai dispositivi rimangono isolati all'interno dei sistemi, impedendo di avere una visione integrata delle informazioni.

In questo progetto di tesi è stato adottato un approccio basato sulle tecnologie del web semantico, al fine di consentire l'interoperabilità dei dati dei dispositivi IoT, facilitando l'integrazione e la condivisione di informazioni, e dando la possibilità di effettuare analisi avanzate. Il web semantico permette di gestire al meglio questi dati e di sfruttare al massimo la loro potenza, fornendo la possibilità di ottenere nuove informazioni e conoscenza ad esempio attraverso operazioni di reasoning.

Attualmente la piattaforma permette agli utenti di caricare i dati relativi al fitness e alla salute rilevati dai sensori dei dispositivi di cui si l'utente è in possesso, di interrogarli e, attraverso query SPARQL, di creare grafici. Il grande vantaggio di questa piattaforma deriva dalla possibilità di convertire tutti i dati di cui l'utente è in possesso grazie ai dispositivi di cui dispone, in un unico formato. L'utente è così in grado di confrontare i valori rilevati e ottenere informazioni sul proprio stato di salute e, più in generale, sulla sua forma fisica. Nel paragrafo che segue vengono elencati alcuni miglioramenti che è possibile effettuare sulla piattaforma sviluppata.

5.0.1 Miglioramenti

Logical source

Un problema derivante dall'uso di RML è la logical source. Un esempio di come specificare la logical source in un documento di mappatura RML è rappresentato nella figura 5.1.

Di conseguenza, i dati che devono essere convertiti in RDF e memorizzati sul triplestore, affinché possano essere sottoposti alla mappatura RML,

devono essere memorizzati in un file con nome uguale a quello indicato per la logical source. Un possibile miglioramento potrebbe essere la modifica della libreria RML-Mapper, cambiando la modalità di lettura del contenuto della logical source.

Inserimento id e username

Attualmente l'identificatore univoco, utilizzato per sostituire i blank node, e lo username, utilizzato per associare i dati al proprietario, vengono aggiunti alle informazioni caricate dall'utente aggiornando il contenuto del file che RML prende in input come logical source.

La soluzione attualmente adottata funziona, ma non è sicuramente la migliore da un punto di vista logico; inoltre riduce di molto la modularità del codice, in quanto, per ogni dispositivo per cui verrà aggiunta la relativa mappatura RML, dovrà essere implementato l'inserimento dell'identificativo univoco e dello username. La causa di ciò è che, come spiegato, non esistono standard a cui uniformarsi, di conseguenza i formati dei file sono diversi, e anche la loro struttura. Non potendo passare dinamicamente valori al documento di mappatura RML per i soggetti delle triple questa è stata l'unica soluzione trovata.

5.0.2 Sviluppi futuri

Aggiunta di specifiche di mappatura

Le specifiche di mappatura sono state scritte solo per quattro sistemi IoT. È necessario quindi, per aumentare il numero di dati memorizzabili in formato RDF sul triple store, implementare nuove mappature, così da includere altri dispositivi che forniscono i dati sull'attività fisica dell'utente.

Pulizia dei dati

La pulizia dei dati è attualmente implementata solo per le date, è quindi necessario un procedimento più completo di rilevamento e correzione dei valori corrotti o errati dei dati grezzi IoT.

Reasoning

Una volta effettuata l'integrazione dei dati, il valore aggiunto è rappresentato dalla loro interpretazione, la quale permette di fruire di nuova conoscenza. A tal fine i sistemi di reasoning permettono di sfruttare appieno il potenziale dei dati relativi al fitness e alla salute. La piattaforma realizzata supporta operazioni di reasoning: jena infatti fornisce delle API che permettono di effettuare reasoning sui dati che si hanno a disposizione [6].

Ringraziamenti

A mia mamma, che si è sempre fatta in mille per me. A mio fratello, perchè nonostante le mille litigate rimane un punto fermo. Alla Stella, senza la quale non saprei immaginare neanche più come si farebbe a vivere senza. A tutte le mie amiche, che quasi da 26 anni mi supportano e mi hanno sostenuto in questo lungo percorso e in questa ultima faticosa tappa. E al mio papà, che manca tanto e vorrei fosse qui, e che spero di rendere orgoglioso da lassù.

Bibliografia

- [1] <http://books.xmlschema.org/relaxng/ch19-77049.html>.
- [2] <http://milicivuk.com/blog/2011/07/14/problems-of-the-rdf-model-blank-nodes/>.
- [3] <https://blog.neongoldfish.com/social-media/the-advantages-and-disadvantages-of-wearable-tech-3>. Vantaggi wereables.
- [4] <https://compassunibo.wordpress.com/2018/01/26/wearable-device-i-dispositivi-indossabili-che-stanno-cambiando-la-nostra-vita-quotidiana/>.
- [5] <http://scuoladelsociale.capitalelavoro.it>.
- [6] <https://jena.apache.org/documentation/inference/reasonerapi>.
- [7] <https://lod-cloud.net/>.
- [8] <https://opendefinition.org/>. Definizione di open.
- [9] <https://www.computerworld.com/article/2474554/emerging-technology/141686-7-hidden-dangers-of-wearable-computers.htmlslide8>. Svantaggi wereables.
- [10] <https://www.ericsson.com/en/trends-and-insights/consumerlab/consumer-insights/reports/wearable-technology-and-the-internet-of-things>.
- [11] <https://www.internet4things.it/iot-library/internet-of-things-gli-ambiti-applicativi-in-italia/>.

-
- [12] <https://www.livescience.com/50184-wearable-tech-knows-you-are-sick-before-you-do.html>.
- [13] <https://www.mouser.it>.
- [14] <https://www.mouser.it/applications/article-iot-wearable-devices/>.
- [15] <https://www.ncta.com/>.
- [16] <https://www.techradar.com/news/wearables/htc-s-wearable-is-a-fitness-tracker-built-with-the-help-of-under-armour-1281453>.
- [17] <https://www.w3.org/designissues/linkedata.html>.
- [18] <https://www.w3.org/wiki/hclsig/lodd>.
- [19] <http://www.centronous.com>.
- [20] <http://www.linkiesta.it/it/article/2014/10/27/i-vantaggi-per-gli-utenti-dei-dispositivi-indossabili/23302/>.
- [21] <http://www.websemantico.org/articoli/approcciwebsemantico.php>.
Web semantico.
- [22] H. Alemdar and C. Ersoy. Wireless sensor networks for healthcare: A survey.
- [23] bio2RDF. <http://bio2rdf.org/>.
- [24] H. T. A. D. R. Y. Bizer, C. The semantic web. scientific american.
- [25] D. Brickley and R. V. Guha. Rdf vocabulary description language 1.0: Rdf schema.
- [26] centro nous. <http://www.centronous.com/centro/biofeedback-della-conduttanza-cutanea/>. IoT nel campo medico.
- [27] T. B.-L. Christian Bizer, Tom Heath. Linked data - the story so far.
- [28] dbpedia. <https://dbpedia.org/sparql>.
- [29] F. Fumelli. Internet, evoluzione e programmazione.

-
- [30] geo data. <http://linkedgeodata.org>.
- [31] J. C. D. M. Giuseppe Rizzo, Federico Morando. Open data: la piattaforma di dati aperti per il linked data.
- [32] humanitas. <http://www.humanitas.it/news/14252-google-glass-parlano-di-noifoto1>.
- [33] ISTAT. <http://datiopen.istat.it/>.
- [34] jena. <http://jena.apache.org/documentation/rdf/index.html>.
- [35] jena. <https://jena.apache.org/documentation/fuseki2/>.
- [36] jena. https://jena.apache.org/documentation/query/app_api.html.
- [37] jena. <https://jena.apache.org/documentation/query/update.html>.
- [38] jena. <https://jena.apache.org/documentation/sdb/>.
- [39] jena. <https://jena.apache.org/documentation/tdb/index.html>.
- [40] M. R. Kamdar and M. J. W. Prism. A data-driven platform for monitoring mental health.
- [41] T. B.-L. L. Masinter and R. T. Fielding. Uniform resource identifier (uri): Generic syntax.
- [42] G. M. Luigi Atzori, Antonio Iera. The internet of things: A survey.
- [43] E. Miller. An introduction to the resource description framework. bulletin of the association for information science and technology.
- [44] music brainz. <https://wiki.musicbrainz.org>.
- [45] ontotext. <https://ontotext.com/knowledgehub/fundamentals/what-is-rdf-triplestore/>.
- [46] open gv. <http://open.gov.it/>.
- [47] open gv. <http://open.gov.it/open-government-partnership/come-funziona-ogp/>.

- [48] open street map. <https://blog.openstreetmap.org>.
- [49] oracle. <http://www.oracle.com/technetwork/articles/java/servlets-jsp-140445.html>.
- [50] O. V. Peter Friess. Internet of things: converging technologies for smart environments and integrated ecosystems.
- [51] R. Reda. A semantic web approach to ontology-based system: Integrating, sharing and analysing iot health and fitness data.
- [52] rml. <http://rml.io/spec.html>.
- [53] A. Sternstein. Kundra's ideas shape book.
- [54] M. Swan. <http://www.mdpi.com/2224-2708/1/3/217/htm>.
- [55] O. L. T. Berners-Lee, J. Hendler. Interlinking open data on the web, in the 4th european semantic web conference.
- [56] C. M. S.-M. E. M. T. Bray, J. Paoli and F. Yergeau. Extensible markup language (xml). world wide web journal.
- [57] C. B. Tom Heath. <http://linkeddatabook.com/editions/1.0/>.
- [58] tomcat. <http://tomcat.apache.org/>.
- [59] visit korea. <http://api.visitkorea.or.kr/about/useguide05.do>.
- [60] w3c. https://www.w3.org/2001/sw/wiki/apache_jena.
- [61] w3c. <https://www.w3.org/2009/talks/0615-qbe/>.
- [62] W3C. <https://www.w3.org/designissues/fragment.html>.
- [63] W3C. <https://www.w3.org/standards/semanticweb/data>.
- [64] w3c. <https://www.w3.org/tr/2013/rec-sparql11-update-20130321/>.
- [65] W3C. <https://www.w3.org/tr/owl-features/>.
- [66] W3C. <https://www.w3.org/tr/rdf-concepts/section-triples>.

- [67] W3C. <https://www.w3.org/tr/rdf11-concepts/dfn-blank-node>.
- [68] W3C. <http://www.w3c.it/papers/wsb08.pdf>.
- [69] wikimedia. <https://commons.wikimedia.org>.
- [70] wikisource. <https://it.wikisource.org>.