UNIVERSITÀ DI PISA

Dipartimento di Filologia, Letteratura e Linguistica

Corso di Laurea Magistrale in Informatica Umanistica

UNA TASSONOMIA PER L'INDUSTRIA 4.0 - PROGETTAZIONE E IMPLEMENTAZIONE DELL'INTERFACCIA UTENTE DI UN MOTORE DI RICERCA PER BISOGNI AZIENDALI E TECNOLOGIE ABILITANTI NEL CONTESTO INDUSTRIA 4.0

Relatore:

Daniele Mazzei

Candidato:

Roberto Figliè

Anno Accademico 2020/2021

Indice

Introduzione

L'Industria 4.0 rappresenta la forza trainante di quella che è stata definita la Quarta Rivoluzione Industriale. Tuttavia, il percorso di digitalizzazione è frenato da una comprensione non metodica dei bisogni e problematiche delle aziende e le relative tecnologie abilitanti che possono affrontarli. Con lo scopo di colmare questo vuoto è stata dunque elaborata una tassonomia inserita all'interno del progetto europeo "Planet4". L'obiettivo finale del progetto Planet4 è quello di fornire degli strumenti adeguati alla formazione di lavoratori capaci di operare nei contesti 4.0 oppure all'aggiornamento dei lavoratori già impiegati. In particolare, la tassonomia permette da una parte alle aziende di individuare nuove soluzioni per i nuovi bisogni emersi nel processo di trasformazione digitale 4.0, dall'altra alle università di definire i campi di applicazione delle nuove tecnologie.

Al fine di rendere disponibile la tassonomia per l'Industria 4.0 all'utilizzo di utenti provenienti dal mondo delle aziende e da quello accademico sono state poste due problematiche: "come si rappresenta una tassonomia?" e, conseguentemente, "come si può interagire con una tassonomia?". Il lavoro trattato in questa tesi cerca di rispondere a queste domande cercando di comprendere in che contesto si situano le tassonomie e quali sono le migliori pratiche di rappresentazione e, infine, proponendo un'interfaccia utente di una piattaforma di ricerca che permetta di navigare ed esplorare la tassonomia per l'Industria 4.0 in questione.

Il progetto di lavoro è stato strutturato in due fasi principali che riguardano la progettazione e l'implementazione del sito per la piattaforma di ricerca. Le metodologie con cui è stata progettata l'interfaccia utente sono derivate dalle tecniche di progettazione per l'esperienza utente (User Experience Design) ed in particolare l'Information Architecture, l'analisi degli utenti e l'elaborazione di *wireframe* per la strutturazione del *layout*. L'implementazione è stata invece realizzata avvalendosi del *framework* React e della libreria di elementi grafici Material UI.

CAPITOLO 1

L'Industria 4.0: il contesto e la sua tassonomia

Il lavoro effettuato e trattato in questa tesi riguarda la costruzione di una piattaforma di ricerca che opera all'interno di una tassonomia già definita. La piattaforma e la tassonomia sono inserite nell'ambito del progetto europeo denominato Planet4[1] volto alla facilitazione del trasferimento di conoscenze e competenze tra accademia e mondo del lavoro. In particolare, il settore specifico per cui è stato pensato Planet4 è quello dell'Industria 4.0, principale fautrice di quella che si definisce come una quarta rivoluzione industriale.

1.1 L'industria 4.0 e le sue tecnologie

La prima definizione della nozione di Industria 4.0 è attribuita a Henning Kagermann, Wolf-Dieter Lukas e Wolfgang Wahlster, esposta in una relazione presentata alla fiera di Hannover nel 2011 (Kagermann H., Lukas W. e Wahlstler W. 2011). Lo sviluppo e la possibilità di integrazione di nuovi sistemi denominati ciberfisici (o IoT, *Internet of Things*) permisero di intravedere nuovi orizzonti di sviluppo nello scenario industriale, in primis tedesco. È proprio dalla Germania che infatti fu tracciata negli anni seguenti al 2011 questa nuova direzione al fine di rendere più competitiva a livello internazionale l'industria tedesca.

Fulcro di tale processo sono i già menzionati sistemi ciberfisici, ovvero particolari sistemi informatici dotati di "corpo" che gli permetta un'interazione continua con il mondo fisico. Ciò si realizza tramite la rappresentazione come modello digitale del

[1] Progetto co-finanziato dal programma Erasmus+ dell'Unione Europea.

dispositivo controllato (il gemello digitale o *digital-twin* appunto) che rende possibile la sincronizzazione tra mondo reale e mondo digitale. Un sistema ciberfisico è quindi un sistema che comprende un insieme di componenti fisiche e digitali (centralizzate o distribuite) che interagiscono tra loro. L'interconnessione tra dispositivi tramite internet è garantita dalla tecnologia IoT (*Internet of Things*)[2].

Si può notare come negli ultimi dieci anni i dispositivi IoT abbiano cominciato a popolare la quotidianità, dalla casa al lavoro. Se questo ha portato dal lato del consumatore alle nuove integrazioni di sistemi intelligenti per la casa (luci, elettrodomestici, sistemi di sicurezza e così via) o a diversi dispositivi indossabili (*wearable* come smart watch/band o sensori per scarpe) che hanno sicuramente suscitato un discreto interesse e semplificato diverse attività, per quanto concerne il lato industriale può condurre ad un miglioramento che riguarda efficienza di produzione e vantaggi economici e anche ambientali.

L'Internet of Things è un sistema di dispositivi collegati tra loro che hanno l'abilità di trasferire dati attraverso una rete senza bisogno di un intervento umano. Al termine "dispositivo", intendibile qui non solo come dispositivo tecnologico, è possibile sostituire una grande varietà di realtà fisiche che vanno dal digitale al meccanico e si estendono a qualunque oggetto o anche persone e animali. Ognuno di tali elementi è provvisto di un identificativo unico che permette di riconoscerlo nella rete in cui è connesso. Il *digital twin* a cui prima si faceva riferimento può essere quindi una replica

[2] I termini "sistemi ciberfisici" e "IoT" sono in letteratura utilizzati a seconda dei casi, per indicare due diversi paradigmi o come sinonimi. Nel primo caso si pone l'IoT come infrastruttura di base sopra la quale si forma quella ciberfisica, o meglio, come le costituenti che formano i sistemi cirberfisici. Nel secondo invece si tratta di paradigmi convergenti o comunque utilizzati in maniera intercambiabile (anche a seconda della provenienza del parlante).

digitale non solo di dispositivi tecnologici ma di qualunque entità appartenente al mondo fisico[3].

Calato nel paradigma dell'Industria 4.0, l'IoT diventa IIoT, ovvero *Industrial Internet of Things*. Questo comporta l'organizzazione dei processi produttivi attraverso dispositivi che comunicano autonomamente tra loro attraverso tutta la catena del valore (*value chain*[4]). L'IoT è però solo una delle tecnologie abilitanti dell'Industria 4.0. Con le Data Sciences, l'Intelligenza Artificiale (e il Machine Learning) e il Cloud Computing (anche *on the edge*[5]), l'IoT forma un insieme olistico di tecnologie che sono alla base del paradigma 4.0.

La Commissione Europea a questo proposito ha individuato nove tecnologie chiave per l'Industria 4.0 (European Commission 2015). Tra queste troviamo:

- Soluzioni Avanzate per la Produzione (*Advanced Manufacturing Solutions*), cioè la cooperazione autonoma tra robot industriali, l'utilizzo di sensori integrati e interfacce standardizzate.
- La Produzione Additiva (*Additive Manufacturing*), che implica l'utilizzo di stampa 3D per realizzare prototipi o sopperire a componenti mancanti.
- La Realtà Aumentata (AR), indirizzata alla manutenzione e logistica, ad esempio mostrando informazioni utili attraverso occhiali.

[3] Luciano Floridi approfondendo infatti questo concetto nel campo della filosofia lo analizza nella sfera umana in cui ogni individuo diviene in questo modo dotato non soltanto di una natura fisica ma anche virtuale che le corrisponde e si integra. Floridi conia così il neologismo "On-life" che elimina la distanza ontologica tra realtà fisica e digitale (Floridi 2009).

[4] Per catena del valore, o *value chain*, si intende un modello che descrive l'insieme di processi e attività che portano al rilascio di un prodotto o servizio. Questi vanno dagli aspetti logistici di ingresso e di uscita alle attività di produzione fino al marketing e al servizio post-vendita.

[5] Mentre il Cloud centralizza il processamento, l'Edge Computing lo esegue localmente nel luogo di raccolta dei dati. In ogni caso non sono approcci in contrapposizione tra loro, ma complementari.

- La Simulazione (*Simulation*), per l'ottimizzazione in tempo reale basata su sistemi intelligenti.
- L'Integrazione Orizzontale e/o Verticale (*Horizontal/Vertical Integration*), l'integrazione di dati tra aziende come precondizione di una *value chain* completamente automatizzata.
- L'Internet Industriale (*Industrial Internet*), un network a comunicazione multidirezionale di macchine e prodotti.
- Cloud, per la gestione di grandi quantità di dati in sistemi aperti e la comunicazione in tempo reale tra i sistemi produttivi.
- Cyber-security, per i sistemi aperti e il collegamento tra macchine intelligenti, prodotti e sistemi.
- Big Data e *Analitycs*, per il supporto e l'ottimizzazione del processo di *decision-making* in tempo reale.

Tali tecnologie rendono possibile l'introduzione di nuove industrie intelligenti (*smart factories*) in cui è possibile raccogliere e analizzare dati, e in base a questi prendere migliori decisioni.

Il valore aggiunto apportato dalle tecnologie 4.0 ai sistemi industriali di automazione e controllo riguarda miglioramenti nella produttività e nell'efficienza permettendo il monitoraggio, la raccolta, lo scambio e l'analisi di informazioni tra i dispositivi collegati (sensori, assets industriali, ecc.) senza l'intervento umano. Il paradigma Industria 4.0 e le sue tecnologie si indirizzano alla soluzione di svariate problematiche manifatturiere. Ad esempio, le strategie predittive di manutenzione (*predictive maintenance*) permettono di individuare problemi e difetti in un dato macchinario nella maniera più efficiente e veloce possibile: il tempo tra il sorgere della problematica e la sua rilevazione (e quindi risoluzione) viene ridotto enormemente.

1.2 Il progetto Planet4

Il progetto Planet4 parte dalla considerazione di una nuova esigenza nel campo aziendale: il passaggio a sistemi AIoT, ovvero *AI on the edge*. Il paradigma dei *digital*

twin prevede che i nodi IoT facciano da ponte tra l'oggetto fisico e il *cloud* dove risiede l'Intelligenza Artificiale (Mazzei 2020). È stato notato però come ciò comporti diverse problematiche di scalabilità legate all'elevato consumo energetico, la necessità di una costante connessione a banda elevata, e le infrastrutture di sicurezza. Per questo è stato ritenuto necessario spostare parte delle attività dell'AI sui nodi decentralizzandola e distribuendola e lasciando al *cloud* l'orchestrazione del tutto. Questo approccio innovativo ha così generato nel mondo del lavoro un bisogno di lavoratori capaci di operare in questi nuovi contesti AIoT che, come sostiene Mazzei, "it is Universities' responsability to train" (Mazzei 2020). Il progetto Planet4 si propone quindi di costruire un *framework* che aiuti la formazione di nuovi esperti nei settori produttivi in grado di lavorare in sintonia con le complessità dell'Industria 4.0. All'interno del paradigma portante della Quarta Rivoluzione Industriale, infatti, è divenuto difficile per le aziende affrontare le nuove problematiche o bisogni che si ritrovano a fronteggiare con le tecnologie che hanno già disponibili. Per questo diviene necessario il trasferimento di tecnologie, conoscenze e competenze tra accademia e mondo del lavoro. Negli obiettivi del progetto Planet4 ciò sarà reso possibile tramite la creazione di un nuovo corso extra-curriculare erogato su di una piattaforma di b-learning (cioè di insegnamento misto) disponibile sia agli studenti universitari sia ai lavoratori già impiegati nei settori di interesse.

Il processo il cui compimento porterà al corso extra-curriculare è diviso in tre fasi che si occupano rispettivamente di:

1. Sviluppo di una tassonomia dedicata che formalizzi i bisogni e le tecnologie legate al paradigma 4.0.
2. Raccolta dei materiali del corso e di *best practice* sulle diverse applicazioni dell'AIoT.
3. Implementazione effettiva del corso composto di parte teorica (in e-learning) e workshop pratici.

Le fasi di interesse per il progetto di cui tratta questa tesi sono la prima e la seconda.

1.3 Una tassonomia per l'industria 4.0

A partire dalle necessità di lavoro sul progetto Planet4 si è delineata una possibile identificazione in una piattaforma a sé stante che riunisse parte delle prime due fasi. La prima parte del lavoro è consistita nella costruzione di una tassonomia (app.) che è quindi da considerare come la base del lavoro successivamente realizzato. Il lavoro realizzato da Riccardo Amadio, Anastasiya Isgandarova e Daniele Mazzei (2021) cerca di rispondere alle seguenti domande:

- Quali problemi dell'Industria 4.0 e quali tecnologie abilitanti sono stati affrontati e implementati in casi d'uso industriali?
- È possibile creare una tassonomia che categorizzi i problemi e le tecnologie di supporto dell'Industria 4.0?
- Qual è la relazione tra i problemi e le tecnologie abilitanti in questione?

Attraverso lo studio e l'analisi di articoli accademici, d'industria, casi d'uso del settore e report delle aziende di consulenza sono state realizzate due tassonomie riguardanti, rispettivamente, l'una i problemi e i bisogni legati all'Industria 4.0, l'altra le tecnologie abilitanti. La mappatura tra bisogni e tecnologie è realizzata tramite proprio i casi d'uso e gli articoli che permettono di ricollegare uno o più bisogni ad una o più tecnologie.

1.3.1 Struttura della tassonomia

Ognuna delle due tassonomie è strutturata su tre livelli in cui sono organizzati gli argomenti dal più generale al più specifico.

Le due principali categorie in cui i problemi e i bisogni dell'Industria 4.0 sono stati divisi sono "Product Innovation" e "Process Optimization". Incluse in "Product Innovation" vi sono problematiche che mirano alla migrazione verso soluzioni basate su servizi legate all'utilizzo di prodotti IoT connessi oltre che al miglioramento dell'usabilità dei prodotti. I problemi contenuti in "Process Optimization" riguardano invece il miglioramento dell'efficienza e dei costi nei processi di produzione, legati cioè a macchinari, forza lavoro e catena del valore.

La tassonomia riguardante le tecnologie abilitanti è divisa in cinque categorie. Sono state scelte solo cinque tra le nove tendenze tecnologiche chiave dell'Industria 4.0 evidenziate dalla Commissione Europea (2015) perché nella letteratura esaminata dimostravano "di giocare un ruolo cruciale nella soluzione dei problemi dell'Industria 4.0" (Amadio, Isgandarova e Mazzei 2021, p. 8). Le tecnologie, quindi, sono: "Big data and analytics", "Industrial internet", "Cloud", "Horizontal/Vertical Integration" e "Advanced Manufacturing solutions".

Nonostante queste tecnologie abilitanti siano già parzialmente utilizzate all'interno dei processi produttivi, l'ulteriore implementazione in un contesto industriale 4.0 è in grado di trasformarli ottimizzando i processi e trasformando le relazioni tra fornitori, produttori e clienti.

I collegamenti tra le due tassonomie sono stati individuati in 41 tra articoli di ricerca e casi d'uso del settore. In questo modo è stato possibile rendere interoperabili le due tassonomie.

1.3.3 Il problema della rappresentazione

Dopo il lavoro di ricerca e di formalizzazione della tassonomia si è aperta la questione che riguarda la sua rappresentabilità attraverso forme anche grafiche che ne rendessero più accessibile la comprensione e l'utilizzo. Il campo specifico che si occupa dello studio e la progettazione di tassonomie è denominato Information Architecture (o IA, da non confondere con "intelligenza artificiale" in italiano). Il contributo di cui questa tesi ne è l'esposizione prende le mosse a partire proprio da un'analisi di stampo IA della tassonomia.

Seconda questione, collegata alla prima, è quella della progettazione della piattaforma. Aspetto complementare all'IA (e anzi in cui l'IA recentemente è confluita) per una buona progettazione è lo User Experience Design (UX). La UX sta solitamente alla base del processo di progettazione e riguarda la messa in pratica di un insieme di metodologie e tecniche di progettazione per prodotti e servizi (non solo digitali) che pongono al centro dello sviluppo l'utente che effettivamente ne farà uso. Nello

sviluppo software è buona pratica tenere a mente chi saranno le persone per cui il prodotto è pensato, dall'inizio alla fine del processo (e oltre nell'assistenza e nel supporto dopo l'uscita), affinché venga realizzato il giusto prodotto. In un processo iterativo che porta al vicendevole raffinamento e miglioramento, UX e IA permettono di giungere insieme alla terza e ultima fase affrontata, ovvero quella legata all'implementazione lato *front-end* e quindi allo sviluppo del sito effettivo. Cosa, per chi, come si interagisce e quale aspetto avrà, confluiscono insieme nella realizzazione pratica della piattaforma.

CAPITOLO 2

Tassonomie e Information Architecture

2.1 L'Information Architecture e le sue definizioni

Nella fase di definizione del problema posto dall'affrontare la rappresentazione di una tassonomia è stato necessario prima di tutto comprendere cosa si intende per questo termine, quali tematiche sottendono ad esso e in generale come sono stati risolti problemi simili. In particolare, è stato importante definire l'ambito in cui le tassonomie sono studiate, ovvero l'Information Architecture.

Sono presenti diverse definizioni di Information Architecture (o IA) tra cui ne saranno riportate quattro: due che si completano tra loro, una legata all'ecologia dell'informazione ed una che ne definisce i termini.

Innanzitutto, una prima definizione degli elementi che compongono l'IA (Covert 2014):

- Ontologia: la definizione di ciò di cui si sta parlando, il senso di quello che si fa. Quello che si intende con ciò che si dice[6].
- Tassonomia: la struttura logica che dà senso a ciò che si sta presentando.
- Coreografia: mette insieme significato (ontologia) e struttura (tassonomia) nel flow di uno specifico contesto.

[6] Quindi da intendere con una sfumatura leggermente diversa e ristretta rispetto al suo significato solitamente filosofico.

Le ontologie riguardano problematiche e aspetti soggettivi, culturali, cognitivi e di traduzione. Tassonomie e coreografie riportano le stesse problematiche con crescente complessità aggiungendo la variabile contestuale.

È possibile una seconda definizione di più alto livello rispetto alla prima, e che è piuttosto di tipo relazionale (Soranzo 2013) e permette quindi di dare una contestualizzazione. Si prendano in considerazione:

- Tassonomia, in quanto informazione sull'informazione (metadati).
- Navigazione, come l'utente si muove e si orienta.
- IA, progettazione della struttura dell'informazione, formata dalla tassonomia (principalmente IA) per permettere la navigazione (principalmente UI).

Le due definizioni, seppur riguardanti aspetti diversi, si completano e integrano in una visione di più ampio spettro. Dove la prima è più specifica ma concettuale, la seconda è più generale ma pratica.

La terza definizione (Rosenfeld, Morville e Arango 2015) riguarda l'ecologia dell'informazione, cioè l'oggetto da strutturare. In questo concetto sono evidenziati i rapporti tra contesto (*business goals*, cultura, tecnologia, risorse, limiti ecc.), utente (bisogni, tasks, comportamenti, esperienza) e contenuto (tipo di dato, strutture esistenti, elementi contenuti, ordine di grandezza).

Tenendo conto delle definizioni esposte, è possibile considerare la quarta e ultima definizione (Hedden 2016, pp. 1-15) che permette di stabilire i termini e i tipi fondamentali in cui si identifica una tassonomia: una tassonomia è un sistema o struttura di organizzazione della conoscenza, che comprende sia strutture propriamente gerarchiche (la tassonomia in senso stretto) sia relazionali (tesauri e ontologie). Quindi, si può considerare il termine "tassonomia" nel modo più ampio possibile, tenendo in considerazione al suo interno tesauri e ontologie.

Infine, un altro aspetto riguarda la tipologia della tassonomia, di cui se ne possono citare quattro:

1. *Flat Taxonomy* (tassonomia piatta): la più semplice, con pochi elementi non sovrapponibili e un solo livello di categorizzazione.
2. *Hierarchical Taxonomy* (tassonomia gerarchiche): espansione di una tassonomia piatta per mezzo di sottolivelli. Ogni elemento appartenente a un sottolivello ha solo un altro elemento come genitore.
3. *Network Taxonomy* (tassonomia a rete): forma una rete tra le categorie o sottocategorie. Può, per esempio, rappresentare i modi con cui l'utente pensa e collega tra loro gli elementi.
4. *Facet Taxonomy* (tassonomia a "faccette"): raffigura un elemento con tutti i collegamenti ad aspetti (o più propriamente *facets*) che delineano i suoi attributi (un esempio classico sono i filtri di ricerca di cui le *facets* sono, per esempio, il prezzo, la marca, la data, il luogo e così via).

Le diverse tipologie non si escludono mutualmente e possono infatti essere utilizzate in combinazione. Inoltre, nel caso in cui un elemento sia ambiguo è possibile assegnargli in tassonomia più genitori, permettendo di giungere in più modi alla stessa categoria. In questo caso particolare si tratta di una tassonomia poligerarchica.

2.2 Tassonomia, tesauro e ontologia

Tassonomia, tesauro e ontologia sono tutte e forme di vocabolari controllati (insiemi di termini con i loro sinonimi ecc.). La differenza risiede nella complessità: dove le tassonomie hanno semplicemente relazioni gerarchiche, i tesauri hanno relazioni più complesse, e le ontologie esprimono anche assiomi e restrizioni. In particolare, le ontologie hanno come obiettivo la rappresentazione della conoscenza in maniera *machine-readable* tramite relazioni semantiche tra termini e attributi. I tesauri sono rivolti più generalmente a catalogazione e recupero di informazioni.

2.3 Concetto e termine

Nella definizione degli elementi di una tassonomia ci sono due nozioni da tenere distinte:

- Concetto è l'idea o la cosa che si vuole individuare
- Termine è l'etichetta che descrive il concetto

Mentre per ogni nodo della tassonomia è presente un solo concetto, per ogni concetto possono esserci più termini (per esempio dei sinonimi). Il concetto è indicato con il termine preferito che lo designa. Al contempo possono però essere indicati più termini non preferiti che reindirizzano a quel concetto. Quindi il concetto è composto da un termine preferito (uno solo) e da dei termini non preferiti (uno o più).

2.3.1 Termini preferiti

La scelta dei concetti e dei loro termini preferiti è da accordarsi ai fini della tassonomia e a chi è rivolta. In generale, nella scelta dei termini preferiti devono essere considerate le seguenti linee guida:

1. Ci si dovrebbe rivolgere agli utenti target.
2. Considerare l'ambito e la materia trattata.
3. Utilizzare, se già esistente, un vocabolario controllato appropriato.
4. Conformarsi all'utilizzo standard, accademico, culturale o aziendale.
5. Consistenza nello stile stesso della tassonomia.
6. Consistenza stilistica con i contenuti che la tassonomia tratta.

Il termine preferito è quello che designa principalmente un dato concetto ed è il più rilevante soprattutto nel caso di una tassonomia rappresentata come albero con cui l'utente può interagire.

2.3.2 Termini non preferiti

Per quanto riguarda un'interfaccia basata sul campo di ricerca è necessario stilare un insieme di sinonimi per ogni concetto. È bene in questo caso tenere conto non soltanto del reale sinonimo del termine preferito del concetto, ma anche di termini non preferiti e ambigui che potrebbero comunque rappresentarlo o essergli correlati.

2.3.3 Termini "precoordinati" e "postcoordinati"

Precoordinare un termine significa utilizzare delle combinazioni di due o più concetti per specificarne uno più complesso. I termini postcoordinati riguardano invece la combinazione di due o più concetti in fase di ricerca da parte dell'utente. La postcoordinazione non è affidabile quanto la precoordinata perché porta con sé più ambiguità e difficilmente si giunge al risultato cercato immediatamente. Tuttavia, l'utilizzo di sistemi intelligenti (per esempio un motore di ricerca) e la navigazione dell'interfaccia (da parte dell'utente) possono aiutare e anzi produrre risultati estremamente affidabili.

2.3.4 Note e attributi

Il concetto è quindi l'insieme di termine preferito, termini non preferiti e, possibilmente, una definizione e altre informazioni. In queste informazioni possono essere inclusi:

- come utilizzare il concetto
- come si traduce il suo termine preferito in altre lingue (in caso di tassonomia multilingua)
- tipo o categoria di concetto

Queste informazioni aggiuntive sono conservate con il concetto e può essere scelto se renderle trovabili/visualizzabili in una ricerca o meno.

2.4 Relazioni tra termini

Le relazioni tra i termini di una tassonomia, secondo gli standard ANSI/NISO Z39.19-2005 (2010, p. 42), possono essere di 3 tipi:

- Di equivalenza: USE/UF (usa o è utilizzato per)
- Gerarchica: BT/NT (*broader term*/*narrower term*)
- Associativa: RT (*related term*)

Le relazioni sono tutte reciproche tra due termini ma possono essere asimmetriche (di equivalenza e gerarchica).

Anche i termini non preferiti hanno le loro relazioni con il termine preferito e sono di equivalenza (nei tesauri). Nelle guidelines W3C per i sistemi di organizzazione della conoscenza, o SKOS (World Wide Web Consortium 2009), sono designate come "Alternate label" e non sono trattate come una forma di relazione, bensì come concetti diversi nel vocabolario. Nelle ontologie non esistono quindi la relazione e la differenza tra termini preferiti e non preferiti.

2.4.1 Relazioni di equivalenza (termini non preferiti)

Negli standard ANSI/NISO Z39.19-2005 (2010, p. 43) è riportata una lista di tipologie di termini non preferiti:

1. Sinonimi.
2. Varianti lessicali (es. online e on-line).
3. Quasi sinonimi (es. meteora, meteorite e meteoroidi).
4. Rinvii generici (nomi di una classe e i suoi membri sono utilizzati come equivalenti, es. arredamento e letto, sedia, tavolo ecc.).
5. Riferimenti incrociati a elementi di termini composti (nei casi in cui la postcordinazione tra i singoli termini che compongono un concetto sono usati per indicarlo).

L’utilizzo di termini non preferiti è legato al tipo di interfaccia che si usa per esplorare la tassonomia: se è navigabile tramite *browsing* (per esempio per mezzo di una gerarchia ad albero) non è necessario avere termini non preferiti; se la ricerca è effettuata dall’utente tramite un’apposita barra sono invece indispensabili. Inoltre, se la ricerca è supportata da algoritmi che se ne occupano (come nell’indicizzazione automatica) non è necessario stilare una lista completa di termini non preferiti.

Hedden (2016, p. 139) fa notare che nel caso in cui sia presente una barra di ricerca la navigazione *browsable* sarà ignorata da parte degli utenti. Nonostante ciò, è bene ricordare come utenti diversi possono avere necessità, interessi e competenze diverse. Quindi la presenza di una gerarchia sfogliabile può fungere da supporto alla navigazione in alcuni casi.

2.4.2 Relazioni gerarchiche

Nel concetto di gerarchia sono implicati due aspetti:

1. Più un livello è di ordine superiore più il concetto è allargato e generico.
2. Più il livello è inferiore più il concetto è ristretto e specifico

In una prospettiva relazionale gli elementi di una gerarchia si designano come BT (*Broader Term*) e NT (*Narrower Term*) oppure come *parent* (padre) e *child* (figlio). I concetti figli possono ereditare caratteristiche (attributi, categorie) dai genitori.

I tipi di relazioni gerarchiche nello standard ANSI/NISO (2010, p. 47) sono i seguenti:

- Generico-specifico, se i concetti figli sono versioni che specificano il genitore. (es. il laptop è un tipo di computer)
- Istanze, se l’entità ha un nome proprio ed è una specificazione del genitore. (es. il Grand Canyon è un’istanza di parco nazionale)
- Tutto-parte, se il figlio non indica un concetto più specifico ma è una parte del genitore. (es. lo stomaco è parte del sistema digerente)

Se un termine ha più di un genitore allora si tratta di una relazione poligerarchica (es. l'Egitto appartiene sia all'Africa sia al Medio Oriente). Le poligerarchie possono essere basate anche su due differenti gerarchie ma un termine non può essere una versione più specifica di due termini che sono già in una relazione padre-figlio tra loro. Per esempio, ingegneria genetica non può essere una specificazione di biotecnologia e tecnologia perché biotecnologia è già una specificazione di tecnologia.

2.4.3 Relazioni associative

In una relazione associativa tra due termini uno si relaziona all'altro in maniera simmetrica. È una relazione bidirezionale e dipende dal contesto. Il caso di applicazione più frequente è rappresentato dal collegare termini appartenenti a differenti gerarchie. Hedden (2016, p. 151) riporta alcuni dei casi più comuni in cui è possibile utilizzare relazioni associative:

Process and agent	**Research** RT **Researchers** **Researchers** RT **Research**
Process and counter-agent	**Infections** RT **Antibiotics** **Antibiotics** RT **Infections**
Action and property	**Environmental cleanup** RT **Pollution** **Pollution** RT **Environmental cleanup**
Action and product	**Programming** RT **Software** **Software** RT **Programming**
Action and target/patient	**Auto repair** RT **Automobiles** **Automobiles** RT **Auto repair**
Cause and effect	**Hurricanes** RT **Coastal flooding** **Coastal flooding** RT **Hurricanes**
Object and property	**Plastics** RT **Elasticity** **Elasticity** RT **Plastics**
Object and origins	**Petroleum** RT **Oil wells** **Oil wells** RT **Petroleum**
Raw material and product	**Timber** RT **Wood products** **Wood products** RT **Timber**
Discipline and practitioner	**Physics** RT **Physicists** **Physicists** RT **Physics**
Discipline and object/phenomenon	**Meteorology** RT **Weather** **Weather** RT **Meteorology**
Part and whole (which are not systems, geographic places, etc.)	**Office furniture** RT **Offices** **Offices** RT **Office furniture**

Tab. 1 Tipologie di relazioni associative (Hedden 2016, tab. 4.2)

2.4.4 Relazioni semantiche

Le relazioni semantiche sono forme di relazioni che riguardano il significato. Più complesse rispetto alle altre forme di relazione, le relazioni semantiche sono la caratteristica principale che distingue le ontologie. I tipi di queste relazioni sono gli stessi di quelle già viste (equivalenti, gerarchiche, associative). La differenza principale è che il tipo di relazione indicata è non-standard, quindi non si indicano con BT/NT o RT ma si devono definire a seconda del loro scopo (es. Luke Skywalker È IL FRATELLO DI Leia Organa).

2.5 Realizzazione e standard

Una semplice tassonomia gerarchica può essere realizzata in un documento o in un foglio di calcolo. Se si tiene però conto di tutte le componenti e relazioni insieme (termini non preferiti, attributi, relazioni di vario tipo), allora potrebbe essere necessario il ricorso a software più o meno specializzati. La maggior parte delle tassonomie sono in ogni caso create per mezzo di Microsoft Excel o applicativi simili, soprattutto per motivi di praticità e budget. Nonostante ciò, sono disponibili anche software specifici indirizzati soprattutto alla creazione di ontologie per il web semantico[7].

Per costruire una tassonomia in un foglio di calcolo la procedura standard è la seguente: i termini più generici sono inseriti nella prima colonna e i più specifici nelle colonne successive. In questo modo si avrà la seguente regola: per ogni termine, il più generico è nella colonna alla sua sinistra, il più specifico nella colonna alla sua destra.

[7] Per esempio, la piattaforma di sviluppo di tesauri e ontologie *web-based* e collaborativa VocBench, http://vocbench.uniroma2.it/

2.5.1 Standard SKOS, OWL e formato RDF/XML

Nel caso si utilizzi un software specifico sono presenti degli standard di scrittura, tra cui il più comune è denominato SKOS (Simple Knowledge Organization System). Lo standard SKOS è stato definito dal W3C (World Wide Web Consortium 2009) ed è rivolto al lavoro con i knowledge organization systems (KOS), come tassonomie e tesauri, nell'ambito del web semantico. SKOS è basato sulle specifiche RDF (sempre definite dal W3C), che lo rendono un sistema *machine-readable* e compatibile con diversi software e per la pubblicazione web. Similmente allo SKOS (indirizzato principalmente ai tesauri) ma più recente, è stato definito lo standard OWL (rivolto alle ontologie).

Le basi dello standard SKOS prevedono:

- *Concepts* (concetti), che si individuano con un "URI" (Uniform Resource Identifier).
- *Labels* (termini), con i loro "prefLabel" (preferiti), "altLabel" (non preferiti), "hiddenLabel" (altre varianti nascoste all'end-user).
- *Relationships* (relazioni), gerarchiche o associative, ma non di equivalenza (perché già incluse nelle *labels*).
- *Documentary Notes*, cioè note e definizioni informali per definire ulteriormente il concetto.

2.6 Mantenimento e aggiornamenti

Un importante aspetto da considerare è il mantenimento della tassonomia e la possibilità di ricevere aggiornamenti, ovvero l'aumento di concetti e termini o di contenuti indicizzati (Hedden 2016, pp. 362-365). In particolare, una tassonomia deve essere in grado di supportare il lavoro del tassonomista (che si occupa della struttura

dei concetti in sé) e dell'indicizzatore (che si occupa dell'aggiunta di nuovi documenti e file nella tassonomia).

I miglioramenti della tassonomia possono essere utili o necessari in uno dei seguenti quattro casi:

- Se vi sono termini non utilizzati o se servono più termini non preferiti o ancora più relazioni con altri termini.
- Se un termine è troppo utilizzato è possibile dividerlo in due o più concetti.
- Se due termini sono troppo spesso usati in combinazione, e quindi la possibilità di fonderli in un unico concetto.
- Se dei termini sono utilizzati in maniera scorretta, si provvede a ridefinire meglio i termini preferiti, non preferiti o aggiungere dettagli aggiuntivi.

Per quanto riguarda l'indicizzazione, questa può essere effettuata da un essere umano oppure automatica. I vantaggi dell'indicizzazione umana sono legati alla qualità nel rilevamento di concetti e termini (soprattutto ma non solo in presenza di file multimediali), mentre quella automatica può garantire una velocità senz'altro superiore (soprattutto quando è presente un'enorme mole di dati o quando i dati cambiano velocemente, oppure ancora quando i dati sono già strutturati e/o i termini utilizzati sono consistenti e standardizzati). In sintesi, un indicizzatore umano garantisce l'affidabilità del risultato a scapito della velocità di elaborazione, mentre l'indicizzazione automatica è veloce ma richiede continuo controllo e training sorvegliato da umani.

2.6.1 Indicizzazione umana

Il tassonomista può fornire supporto al mantenimento e agli aggiornamenti messi in pratica dagli indicizzatori fissando delle linee guida. Nel caso di indicizzazione umana ad esempio potrebbe stabilire:

- Un limite per il numero di termini fratelli a cui un documento indicizzato possa fare riferimento al posto del termine più generico.

- Criteri per determinare se un documento è abbastanza rilevante per essere indicizzato.
- Se vi siano collegamenti proibiti tra concetti.
- Se si possono utilizzare contemporaneamente termini specifici e generici riguardo lo stesso argomento.
- Come sono pesati per la rilevanza, se lo sono.

2.6.2 Indicizzazione automatica

L'indicizzazione automatica è utilizzata principalmente nell'ambito dei motori di ricerca e attua auto-categorizzazioni, cioè associa in maniera automatica i documenti ai corretti termini della tassonomia. Un approccio in questa direzione è quello che si realizza tramite Machine Learning.

Un altro tipo di approccio si basa invece su regole fissate da chi ha composto la tassonomia che permetteranno al computer di indicizzare automaticamente solo sulla base dei pattern testuali (condizioni booleane, prossimità con altre parole, espressioni regolari ecc.) e senza l'analisi statistica del Machine Learning. In ogni caso, i due tipi di procedimento possono essere utilizzati anche in modalità mista.

Per quanto riguarda la tassonomia, se rivolta all'indicizzazione automatica, si dovrebbero evitare il più possibile le piccole sfumature o ambiguità dei termini. La pre-coordinazione è molto più utile in questo caso rispetto a quello umano, perché riduce l'ambiguità della post-coordinazione attuata altrimenti dall'utente nella barra di ricerca. Inoltre, nell'indicizzazione automatica (rispetto a quella umana) è più pratico utilizzare un sistema di ranking o di peso per dare rilevanze diverse.

In ogni caso non è necessario scegliere tra una modalità e l'altra di indicizzazione dato che possono essere utilizzate insieme, per contenuti diversi o in maniera ibrida cioè con una revisione umana del lavoro automatico.

2.7 Rappresentare e navigare una tassonomia

Dal punto di vista della rappresentazione effettiva il risultato sarà diverso a seconda di chi sarà l'utente della tassonomia, cioè un indicizzatore o un utente finale. In fase di progettazione occorre quindi rispondere alle seguenti domande:

1. L'interfaccia sarà rivolta a indicizzatori o utenti finali?
2. Se per indicizzatore, è umano o automatico?
3. Se per utenti finali, chi sono? Come la utilizzeranno?

Un'interfaccia per indicizzatori può essere necessaria perché anche se inizialmente la figura dell'indicizzatore coincidesse con quella del tassonomista, non è detto che lo sia per sempre. L'interfaccia per indicizzatori può o no essere integrata con quella per gli utenti finali. Al contrario, se l'indicizzatore è automatico il problema della visualizzazione non si pone.

In generale, con un'interfaccia dedicata, l'indicizzatore umano può inserire nuovi contenuti in maniera coerente con le linee guida stabilite dal tassonomista incorrendo in meno errori. Gli elementi disponibili in questo tipo di interfaccia possono essere:

1. Lista alfabetica dei termini (con opzione per visualizzare i non preferiti).
2. Gerarchia navigabile tramite visualizzazione ad albero.
3. Collegamenti per rimandare dai termini non preferiti ai preferiti.
4. Dettagli del termine selezionato (i suoi non preferiti, relazioni, note ecc.).

Queste modalità ed elementi di visualizzazione possono essere usati insieme e sono condivisi con la visualizzazione per l'utente finale. La differenza principale risiede nella quantità di informazioni mostrate, più utili agli scopi di indicizzazione che di recupero dell'informazione cercata.

Infine, l'interfaccia per utenti finali può includere sia gli elementi riferiti per gli indicizzatori umani (sebbene rappresentati in maniera meno "tecnica"), sia la barra di ricerca.

2.7.1 La navigazione della tassonomia

Lo studio dell'IA informa il processo di progettazione di navigazione. La navigazione riguarda l'orientamento dell'utente nell'interfaccia ed il suo obiettivo principale è permettere all'utente di trovare le informazioni e le funzioni che cerca e, non solo, incoraggiarlo in una direzione che potrebbe essergli desiderabile. I principi base per una buona navigazione sono la *findability* e la *discoverabililty* (Cardello 2014).

La findability indica il poter arrivare alle informazioni cercate o più precisamente, come l'utente riesce a localizzare l'informazione che ritiene rilevante. In una biblioteca, per esempio, indica la facilità con cui è possibile reperire un libro che si sta cercando. Posizionare gli elementi in maniera coerente all'interno dell'interfaccia e con gli standard solitamente utilizzati favorisce la *findability*.

La *discoverabililty* si riferisce alla possibilità che l'utente ha di scoprire nuove informazioni o nuove funzionalità di cui non era a conoscenza e che non stava cercando specificamente. Nell'esempio della biblioteca, riguarda la possibilità di scoprire un nuovo libro che cattura l'attenzione mentre si osserva uno scaffale oppure mentre si cerca altro. L'interfaccia fornisce spesso indirettamente gli strumenti che consentono questa attività di scoperta serendipica perché valutabile solo a posteriori. Tuttavia, è possibile progettare per la *discoverability* posizionando la nuova caratteristica, ad esempio, vicino agli elementi più visti o più rilevanti.

Effettuare una ricerca è quindi molto di più che arrivare ad informazioni varie: "We search to learn, understand, share, and act" (Morville e Callender 2010, p. 11)

2.7.2 Pattern di comportamento e di progettazione

La struttura dell'ambiente in cui sono disponibili informazioni, in questo caso un'interfaccia utente, non influenza soltanto la sua navigazione ma cambia anche il modo in cui comprendiamo (Rosenfeld, Morville e Arango 2015, pp. 56-58).

Quando un utente effettua una ricerca le sue azioni saranno influenzate dall'informazione disponibile e dall'interfaccia in cui sono presenti. Il *flow* del susseguirsi di eventi tra utente e sistema può essere quindi identificato come un *pattern* di comportamento unico. In particolare, in riguardo alla ricerca di informazioni, sono stati individuati *pattern* di comportamento diversi (Morville e Callender 2010, pp. 52-61) (Hearst 2009) tra cui:

1. *Quit*: l'utente effettua una ricerca, vedi i risultati ed esce.
2. *Narrow*: l'utente effettua una ricerca, vede i risultati e li raffina usando filtri, strumenti di *sorting* o la ricerca avanzata.
3. *Expand*: l'utente effettua una ricerca, vede i risultati e ne espande il campo (visualizzando risultati correlati ecc.).
4. *Pearl growing*: l'utente effettua una ricerca, apre uno dei risultati e poi apre o utilizza i link all'interno del risultato (*pattern* di navigazione tipico in Wikipedia).
5. *Pogosticking*: l'utente effettua una ricerca, e poi ripete l'azione di aprire un risultato e tornare alla pagina dei risultati.
6. *Berry picking*: l'utente effettua una ricerca, apre un risultato e dall'informazione ricevuta effettua una nuova ricerca affinando la *query* di volta in volta.
7. *Trashing*: l'utente effettua una ricerca, vede i risultati e poi torna a effettuare la ricerca aggiungendo dettagli alla *query*.

Il *pattern* indicato dal *berry picking* è una versione "leggera" di *pogosticking* dove l'utente campiona i risultati. Qualora però sia un comportamento reiterato all'interno dell'interfaccia allora deve essere considerato un sintomo di *antipattern*, cioè un *pattern* di ricerca prodotto da una cattiva progettazione (Morville e Callender 2010, p. 59). Un altro esempio di *antipattern* è rappresentato dal *trashing*, in questo caso, però, sistemi di suggerimento all'interno della barra di ricerca possono aiutare ad evitare questo comportamento senza necessità di riprogettazione.

In parallelo e strettamente correlati ai *pattern* di comportamento è possibile individuare dieci possibili ulteriori *pattern* indicati però direttamente alla

progettazione del sistema (Morville e Callender 2010, pp. 82-130) (Russell-Rose e Tate 2013, pp. 99-122):

1. *Autocomplete*: completa la *query* di ricerca nella fase in cui l'utente la formula, cercando così di restituirgli il più direttamente possibile la *query* ricercata.
2. *Autosuggest*: simile all'*autocomplete* ma cerca di aiutare l'utente fornendogli idee anche distanti (ma correlate) dalla *query*.
3. *Instant Results*: fornisce dei risultati durante la fase di digitazione della *query*.
4. *Did you mean*: dopo l'invio della *query*, fornisce un suggerimento sul risultato più appropriato (utile nel caso di errori di ortografia per esempio).
5. *Autocorrect*: invece di suggerire una correzione nella *query* di ricerca, la applica automaticamente e ne mostra i risultati.
6. *Best First*: mostra come primi risultati i migliori scelti da un algoritmo (per rilevanza, popolarità, data, formato, in maniera personalizzata per l'utente specifico ecc.).
7. *Partial matches*: mostra i risultati che più si avvicinano alla *query* permettendo di non restituire una pagina con zero risultati.
8. *Related Searches*: mostra ricerche correlate con la *query* effettuata che possono fornire ispirazione all'utente per ulteriori ricerche oppure aiutarlo a chiarire una *query* ambigua.
9. *Federated Search*: permette di effettuare una ricerca in diversi database contemporaneamente. Di conseguenza si avranno risultati di varia natura difficili da raffinare.
10. *Faceted Navigation*: fornisce all'utente opzioni per raffinare i risultati attraverso campi riguardanti diversi tipi di metadati (prezzo, colore, tag ecc.).
11. *Advanced Search*: permette all'utente di raffinare la ricerca prima di effettuarla tramite più o meno elaborate opzioni di postcoordinazione (v. 2.3.3).
12. *Scoped Search*: se i contenuti sono organizzati in categorie è possibile fornire un menu a *dropdown* che permetta di specificare l'ambito della ricerca.
13. *Personalization*: riguarda l'adattamento dei risultati mostrati all'utente specifico che utilizza il sistema (per esempio, i risultati raccomandati su Amazon oppure i risultati personalizzati di Google Maps).

14. *Pagination*: mostra un massimo di risultati per ogni pagina. Lo standard stabilito da Google è di dieci risultati per pagina. Fondamentali in questo caso sono gli *snippet* forniti per ogni risultato (cosa riguarda, link disponibili ecc.).
15. *Structured Results*: ogni singolo risultato è visualizzato nella maniera più congrua al suo contenuto (un indirizzo sarà rappresentato in una mappa, un indice di borsa in un grafico ecc.).
16. *Actionable Results*: a seconda del suo contenuto, ogni risultato fornirà la possibilità di interagire con esso (un numero di telefono che permette di effettuare direttamente una chiamata, un video che permette la sua riproduzione).
17. *Comparing Results*: permette all'utente di comparare i risultati tra loro (per esempio, le caratteristiche e il prezzo di prodotti differenti).
18. *Unified Discovery*: le modalità di ricerca e raffinazione sono riunite insieme. Nella stessa pagina è possibile trovare una barra di ricerca, l'esplorazione della tassonomia, la navigazione *faceted* e così via.

2.7.3 *Best practice* per la progettazione della ricerca

Le *best practice* in riguardo alla progettazione per l'esperienza di ricerca riguardano posizioni e funzioni degli elementi di navigazione in relazione ai *pattern* di design precedentemente individuati.

Considerando la barra di ricerca, questa dovrebbe essere posizionata dove l'utente si aspetta di trovarla (Chaparro, Shaikh e Lenz 2006), quindi in una posizione alta della pagina. Dovrebbe inoltre essere abbastanza larga da poter supportare la ricerca della maggior parte delle *query*, ovvero almeno 18 caratteri, ma più idealmente 27 caratteri in quanto basterebbe per il 90% delle ricerche (Nielsen e Loranger, Prioritizing Web Usability 2006). In riguardo al funzionamento della barra di ricerca, sono indicate le seguenti *best practice* (Russell-Rose e Tate 2013, pp. 123-125):

- Fornire una guida all'utente nella forma di *placeholder* per aiutarlo a formulare la *query*.

- Mostrare la *query* cercata nella barra anche dopo aver effettuato la ricerca.
- Di default la ricerca dovrebbe essere indirizzata a tutti i contenuti del sito, anche se è possibile indirizzare lo scopo.
- Usare l'*autocomplete* nel caso la ricerca sia effettuata entro un vocabolario controllato.
- Usare l'*autosuggest* se i termini di ricerca possibili sono vasti e non devono corrispondere ad una stringa specifica.
- Usare la funzione "*did you mean"* se una query migliore può produrre migliori risultati.
- Usare l'*autocorrect* per evitare di mostrare una pagina con zero risultati.
- Usare le ricerche correlate (*related search*) per presentare concetti associati in una tassonomia.

La pagina dei risultati, spesso denominata SERP (*Search Engine Results Page*), presenta gli elementi che corrispondono alla *query* effettuata dall'utente. Le *best practice* individuate riguardanti la SERP sono le seguenti (Russell-Rose e Tate 2013, pp. 161-163):

- Fornire in giusta misura le informazioni trovate: troppi o troppo pochi dettagli faranno perdere di vista le informazioni necessarie all'utente.
- Cercare di evitare il *pogosticking* dell'utente.
- Considerare i bisogni di ricerca degli utenti e i contesti d'uso. Quindi adottare il miglior approccio di visualizzazione.
- Utilizzare anteprime dei risultati per fornire maggiori dettagli.
- I risultati non sono in competizione tra loro ma rappresentano tutti una possibile informazione necessaria.
- Presentare il contesto mostrando:
    - La ricerca effettuata.
    - Lo stato di altri elementi di navigazione utilizzati.
    - Il numero dei risultati.
- Utilizzare il *layout* che corrisponde meglio ai risultati da mostrare.
- Permettere all'utente di cambiare il *layout* tra quelli disponibili.

- Provvedere un messaggio specifico quando non stati trovati risultati e aiutare l’utente a riformulare la *query*.
- Fornire più opzioni di navigazione.
- Se presente *pagination*, mostrare la corrente pagina dei risultati e la possibilità di navigare tra le pagine. Quindi, dare la possibilità di mostrare quanti risultati preferisca l’utente.
- Fornire possibilità di ordinamento o filtraggio dei risultati.
- In un *layout* di tipo *comparing* mantenere le liste separate per categorie di risultati diverse.

CAPITOLO 3

Metodologie e strumenti utilizzati

Per la realizzazione del sito sono state impiegate varie tecniche, metodologie e strumenti che possono essere classificati in due aspetti, a seconda della loro pertinenza alla fase o alla questione affrontata. Vi sono quindi tecniche e strumenti per il design e per l'implementazione del *front-end*.

3.1 Tecniche e strumenti per il design

Per design si può intendere la pratica del progettare la costruzione di un prodotto o servizio, oppure è il risultante prodotto di quel processo di progettazione. A seconda del tipo di processo utilizzato o del fine previsto si può parlare di diversi tipi di design. In particolare, le tecniche utilizzate nel lavoro trattato in questa tesi sono tratte da quelle dello UX Design. Lo UX Design si occupa della progettazione per la UX, cioè l'esperienza utente. Termine coniato da Don Norman (Norman 2013), esperienza utente fa riferimento alle interazioni e le sensazioni che appunto l'utente ha, o avrà, in rapporto al prodotto. Per questo motivo la UX è *user-centric*, centrata sull'utente.

Il designer che si occupa dell'esperienza utente pensa al prodotto come ad un qualcosa che effettivamente sarà utilizzato da una persona. Per fare ciò ha a disposizione diversi strumenti e metodologie che vanno dall'architettura dell'informazione, alla ricerca (interviste, sondaggi ecc.), all'analisi dei requisiti (funzionali, che riguardano il prodotto in sé, e non funzionali, che riguardano ciò che sta attorno il processo di produzione), fino al *wireframing* e la prototipazione. Al termine del processo è possibile quindi passare al design dell'interfaccia (di cui si occupa lo UI designer) e all'implementazione effettiva (realizzata da un programmatore *front-end*). Il processo non è però lineare ed implica al contrario la continua iterazione delle fasi di lavoro al

fine di raffinare il risultato che si produce. L'obiettivo è quello di avvicinare il più possibile il prodotto ai bisogni che l'utente realmente ha e agli utilizzi che ne farà.

3.1.1 Information Architecture (IA)

L'Architettura dell'Informazione (IA) è il campo di studi che si occupa della struttura, organizzazione e classificazione dei contenuti tale che l'utente possa raggiungere informazioni e servizi di cui ha bisogno in maniera efficiente.

Sebbene all'inizio l'IA nasca come campo a sé, derivato dall'archivistica bibliotecaria, è recentemente confluita nel più grande contenitore dello UX design, in special modo per quanto riguarda la creazione di *site maps*. In particolare, l'Information Architecture riferita al campo dello UX design si occupa di identificare quale informazione è necessaria per l'utente in un dato contesto (*Content Strategy*) e aiuta il processo di *wireframing*.

3.1.2 User personas

L'elaborazione delle *user personas* è una pratica nata all'interno del movimento *user-centered* design (Cooper 1999) che permette di indurre empatia verso l'utente. Le *personas* sono un pratico modo per capire quali sono le motivazioni e i bisogni dell'utente, e quindi di prendere decisioni di design in accordanza con questi. Sicuramente non è uno strumento scientifico ed esaustivo che permetta di identificare categorie di persone (come si fa negli studi di psicologia ad esempio), ma permettono di avere a disposizione una tecnica che renda conto delle centinaia, migliaia o milioni di utenti che andranno ad utilizzare il prodotto che produca risultati il più memorizzabili ed efficaci possibile. A seconda dei dati di ricerca di cui si dispone possono essere elaborati tre tipi di *personas* (Laubheimer 2020):

- *Proto-personas*: si basano sulle assunzioni di chi le elabora. Si ricorre a questa tipologia nel caso in cui si abbiano pochi o nessun dato a disposizione.

- *Personas* qualitative: si basano su un piccolo campione di ricerca di tipo qualitativo (interviste, studi sul campo e così via).
- *Personas* statistiche: emergono da studi statistici con un grande campione (come questionari e sondaggi).

Il processo che porta alla formazione di *personas* può senz'altro essere fondato su un livello più basilare e semplice come le *proto-personas*, per poi essere affinato con successiva ricerca. In particolare, per i fini di questo progetto si sono utilizzate prima delle *proto-personas* per poi rifinirle in delle *personas* qualitative.

3.1.3 Affinity diagramming

L'*affinity diagramming*, insieme al *journaling* sono tecniche di analisi utilizzate per elaborare *personas* (Pernice 2018). Fatto da parte il caso delle *proto-personas* (in cui ci si basa su delle assunzioni), qualunque sia la natura del dato è necessario e utile servirsi di una tecnica che permetta di identificare, categorizzare e analizzare le informazioni rilevanti di cui si è in possesso. Nello specifico, le due tecniche in questione sono indicate per l'analisi di dati qualitativi.

La prima parte dell'analisi delle interviste è consistita nella pratica del *journaling*, cioè l'annotazione e l'evidenziazione di elementi rilevanti per il ricercatore nel testo che esamina a cui segue un'annotazione delle sue idee o note (Rosala 2019).

Per *affinity diagramming* si intende un'organizzazione a grappoli di elementi correlati. Si fa uso degli elementi evidenziati, delle intuizioni e note che sono risultate dalla pratica del *journaling* e si scrivono su foglietti incollabili. Nel caso di lavori di gruppo si procederebbe poi all'assegnazione di un punteggio per determinare le note o intuizioni più rilevanti, ma in questo caso non è stato necessario. Conseguentemente si organizzano i foglietti raggruppandoli in categorie su una lavagna o un muro. Il risultato aiuta a creare una visione il più completa possibile (dati disponibili permettendo) per un'ulteriore elaborazione delle *personas*.

3.1.4 *Wireframing* e *prototyping*

Un *wireframe* è una rappresentazione semplificata, generalmente a bassa fedeltà, di un sito o un'applicazione. In particolare, si tratta di una "illustrazione a due dimensioni dell'interfaccia di una pagina che si concentra specificamente sull'allocazione dello spazio" (Usability.gov, Wireframing), nonché la gerarchia dei contenuti e delle funzioni disponibili. Dato l'interesse specifico verso la sostanza di ciò che si vuole rappresentare e la modalità in cui realizzarlo, i *wireframes* non presentano alcun tipo di stilizzazione grafica, tipografia ricercata, colori o immagini. A seconda della quantità di dettagli aggiunti si può parlare di *wireframe* a bassa o alta fedeltà.

Nel primo caso ci si concentra più sull'aspetto comunicativo all'interno del gruppo di sviluppo. Il *wireframe* permette di veicolare informazioni riguardanti l'interfaccia nella maniera più intuitiva ed efficace, quella visiva, nel modo più veloce possibile. Per questo si tende maggiormente all'astrazione tramite l'utilizzo di testi fittizi (per esempio i cosiddetti "lorem ipsum" o dei rettangoli che segnalano la presenza di un testo), oppure ancora etichette provvisorie per identificare titoli. A ciò si aggiunga il mezzo attraverso cui il *wireframe* è prodotto. Data l'immediatezza che richiede l'intera operazione, un *wireframe* a bassa fedeltà è disegnato generalmente su carta.

Con il passaggio ad un disegno digitale si parla di *wireframe* ad alta fedeltà. In questo caso si aumenta il livello di dettaglio al fine di provvedere una migliore documentazione. Si provvede quindi all'incorporazione di qualche elemento grafico o testo e in generale a una gerarchia visiva in grado di veicolare meglio le scelte di design.

L'aggiunta di un qualche tipo di interazione con gli elementi presentati dal *wireframe* comporta il passaggio alla fase successiva, quella della prototipazione. In particolare, un prototipo è un'ipotesi, ovvero un candidato per una soluzione di design proposta per risolvere uno specifico problema (Pernice 2016). È una maniera veloce ed economica per individuare problemi di usabilità nelle prime fasi di sviluppo. Un prototipo può, come quanto detto per i *wireframes*, essere a bassa o alta fedeltà.

Un prototipo a bassa fedeltà (Usability.gov, Prototyping) è spesso realizzato su carta (quindi più veloce da realizzare) e prevede quindi un livello di interazioni possibili relativamente basilare: la visualizzazione di una pagina in conseguenza di un click su un bottone, per esempio, deve essere manualmente realizzata sostituendo il foglio.

Un prototipo ad alta fedeltà è invece realizzato già su computer e permette quindi delle interazioni più realistiche e vicine al prodotto finale. L'interfaccia presentata quindi da questo tipo di prototipo è più vicina a quella che sarà presente nel prodotto.

3.1.5 Figma

Figma è un'applicazione web per la grafica vettoriale che permette di progettare wireframe e prototipi ad alta fedeltà. Il primo aspetto, cioè il fatto che Figma è una web app, comporta tre fattori significativi: la non necessità di installare alcunché localmente; la compatibilità con diversi dispositivi (a patto che i browser presenti siano compatibili ovviamente); e il salvataggio online in automatico di ciò che è stato elaborato. Quest'ultimo punto non impedisce comunque il salvataggio in locale (in formato *FIG*) o l'esportazione raster o vettoriale nei formati immagine supportati (*PNG*, *JPG*, *SVG* e *PDF*).

Gli strumenti base forniti da Figma per il disegno vettoriale sono coerenti con gli altri software di grafica vettoriale (per esempio Adobe Illustrator). Si basano infatti sulla manipolazione di linee e curve tramite punti e maniglie, o semplici forme come rettangoli, cerchi e frecce. L'elemento di diversità è rappresentato invece dal paradigma di gestione e utilizzo di *layer*, tavole da disegno e simili. Il tutto è realizzato al livello degli elementi concretamente disegnati. Un rettangolo, per esempio, è un livello a sé. Nel momento in cui alla scena vengono aggiunti altri elementi, questi possono essere altri livelli da sé oppure formare un unico livello se raggruppati o uniti in un frame.

Il frame è in Figma l'equivalente della tavola da disegno per altri editor grafici ma presenta alcune differenze, prima fra tutte la possibilità di annidare i frame e quindi di poterli disporre uno sopra l'altro. Un esempio pratico potrebbe essere la

rappresentazione di una schermata in cui un elemento (come un menu di icone o una lista) presenta la possibilità di effettuare un'azione di *scrolling* a prescindere dagli elementi circostanti (le applicazioni rubrica del cellulare, per esempio, hanno i contatti che scorrono mentre gli altri elementi rimangono fissi). Un ulteriore peculiarità è la capacità del frame di poter agire da maschera tramite il clip dei contenuti. Per tornare all'esempio di prima, il clip dei contenuti permette di nascondere gli elementi non presenti alla vista prima dello *scrolling*.

La creazione di componenti è una caratteristica di Figma in grado di poter avvicinare la fase di progettazione dell'interfaccia a quella d'implementazione. Un componente è un elemento (solitamente un frame con elementi grafici all'interno) dotato di stati definiti dall'utente. Lo stato definito può essere una stringa, un booleano e così via. Uno stato di una *checkbox*, ad esempio, può essere "checked" e "unchecked" oppure più precisamente rappresentato come una *checkbox* nelle proprietà dell'elemento. All'interno di ogni componente e per ogni stato, il designer provvederà a creare una o più varianti. Qualora sia usato un componente in fase di progettazione allora verrà creata una sua istanza di cui il designer indicherà gli stati. L'utilizzo di componenti permette quindi di creare un ambiente di progettazione che faccia ricorso agli stessi elementi in condizioni diverse rendendo quindi coerente lo stile grafico e più efficiente la progettazione stessa.

Figma consente inoltre di utilizzare plugin esterni sviluppati appositamente per apportare aggiunte (come la possibilità di esportare direttamente in HTML e CSS) e modifiche al funzionamento dell'applicazione. Altra possibilità sta nell'utilizzo di file elaborati dalla *Community* di Figma, in cui sono presenti *design systems* completi ed elementi grafici vari. Nel progetto realizzato in questa tesi si è fatto uso di Wireframe Kit (di Reony Tonneyck) e Simple Wireframe Kit (di Tom Reinert), due progetti Figma che raggruppano elementi base per la costruzione di wireframe.

Il passaggio da wireframe a prototipo in Figma si realizza attraverso l'utilizzo di animazioni attivate tramite interazione che cambiano la visuale da un frame all'altro. Il risultato per il designer consiste in un *wireflow* che connette tutti i frame attraverso le interazioni che generano il passaggio. Sono presenti diversi tipi di interazioni

possibili (click, drag, hover, mouse enter/leave/down/up), azioni da svolgere (navigate to, go back, scroll to, open overlay) e animazioni (instant, dissolve, smart animate, move in ecc.).

Particolarità che indirizza maggiormente Figma verso lo sviluppo di siti web, web app e sviluppo per smartphone è la funzione "inspect". Questa permette di visualizzare in codice CSS, XML (per Android) o Swift (per iOS) le proprietà di un elemento grafico selezionato, dalla posizione alla tipografia, fino alle animazioni.

3.1.6 Considerazioni sulle scelte

La scelta delle tecniche e strumenti legati al processo di design è stata elaborata in base a considerazioni sulle necessità teoriche e pratiche oppure sulle disponibilità di tempo e risorse. Soprattutto nel primo caso rientra l'utilizzo che si è fatto degli strumenti di pensiero forniti dall'IA. Le tematiche legate all'Architettura dell'Informazione sono state necessarie per comprendere gli aspetti prima di tutto concettuali (cos'è una tassonomia) e poi rappresentazionali (come visualizzarla). Per questo, l'applicazione dell'IA è servita da un lato a creare una comprensione condivisa e contestualizzare al meglio il lavoro precedentemente effettuato nella creazione della tassonomia, dall'altro a creare una gerarchia visuale che permettesse la navigazione e quindi come supporto alla generazione dei *wireframe*. Lo studio dell'IA ha permesso, inoltre, di evidenziare pratiche di organizzazione dell'informazione andate consolidandosi nel tempo e che hanno definito gli elementi base disponibili per l'elaborazione di *wireframe*.

Nelle considerazioni sulle disponibilità di tempo e risorse rientrano invece le scelte che riguardano lo studio delle *user personas* e la creazione di prototipi. Le *user personas* rappresentano uno degli strumenti fondamentali per una progettazione che sia Human-centered (ISO 9241-210 2010). In questa modalità di progettazione l'intero processo coinvolge direttamente ed esplicitamente l'utente al fine di rendere più utile e utilizzabili i sistemi interattivi. Le *user personas* permettono prima di tutto di rispondere alle domande "chi sono gli utenti?" e "quali sono i loro bisogni?". Sono

possibili differenti approcci multidisciplinari per assolvere a questa prima fase, tra cui alcuni derivati dall'antropologia culturale o etnografia (Travis e Hodgson 2019, 96-106), dalla psicologia e dalla statistica. La scelta dell'approccio da utilizzare deriva dalle capacità del ricercatore e dai limiti di risorse a sua disposizione. Nell'elaborazione delle *user personas*, dunque, sono diverse le possibili tecniche e metodologie di ricerca e analisi dei dati provenienti dalla ricerca sugli utenti. In questo caso si è optato per le tecniche dell'*affinity diagramming* e del *journaling* perché le interviste condotte da Riccardo Amadio e Daniele Atzeni hanno prodotto di per sé risultati di tipo qualitativo e non statistico-quantitavo (come sondaggi e questionari). Data la natura iterativa della progettazione Human-centered, non può essere esclusa la possibilità di ulteriori ricerche future (che tengano conto, per esempio, del contesto d'uso) per una raffinazione delle *user personas* e quindi per una migliore comprensione degli utenti.

Nel passaggio alla fase di design concreto, che permette cioè di traslare l'esperienza di ricerca (astratta) in elementi concreti, i *wireframe* hanno permesso di evidenziare il *layout* e la gerarchia dell'interfaccia definiti nei requisiti individuati a partire dall'analisi degli utenti. Inoltre, l'approfondimento dell'IA e in particolare dei pattern di ricerca degli utenti (le modalità in cui un utente cerca informazioni online) e la loro progettazione, ha consentito di individuare specifici elementi d'interfaccia con le loro caratteristiche, standard e *best practice*. Nella interconnessa fase di prototipazione bisogna considerare che il prototipo, nonostante sia utilizzato solitamente per verificare l'usabilità dell'interfaccia con l'utente, in questo caso è stato impiegato come strumento di comunicazione interna. In questo modo, prototipo e *wireframe* insieme hanno permesso di confrontarsi ed allinearsi ad una visione di design condivisa all'interno del gruppo di lavoro nelle differenti fasi di sviluppo. Il prototipo effettivo del sito che consente l'interazione, navigazione e quindi l'effettuazione di test futuri, è stato definito nella fase di implementazione.

Infine, per quanto riguarda le valutazioni pratiche bisogna rendere conto dei limiti imposti dagli strumenti utilizzati. Il processo di prototipazione su Figma, infatti, non è esente da limitazioni. Nonostante Figma consenta di eseguire le due fasi di *wireframing* e *prototyping* in continuità l'una con l'altra, realizzare le interazioni o

anche le schermate previste nel *wireframe* non è operazione semplice, o a volte nemmeno possibile. Ad esempio, non si può effettuare l'interazione su più elementi diversi che generino a loro volta animazioni e chiamate di altri frame nella stessa schermata. Per aggirare questa problematica si deve ricorrere all'utilizzo di un *overlay* (che emuli un *drop-down* per esempio) ma limitandosi ad un solo suo utilizzo poiché è proibito usarne due contemporaneamente. Altrimenti, si è costretti a ricreare più volte la stessa schermata (in altri frame) collegando in serie il tutto per ogni interazione, rendendo più disordinato l'ambiente di lavoro[8].

3.2 Strumenti per lo sviluppo *front-end*

Per l'implementazione dell'interfaccia lato *front-end* è stato utilizzato il *framework* React sviluppato da Facebook. All'interno dell'ambiente React è consentito, e a volte necessario, l'uso di componenti e moduli esterni che ne permettono di estendere le funzionalità. Quindi, per la navigazione tra le pagine è stato impiegato React-Router mentre per quanto riguarda il design grafico Material UI.

3.2.1 React

React è una libreria open-source JavaScript, creata e mantenuta principalmente da Facebook (oltre che una *community* di sviluppatori indipendenti), destinata allo sviluppo di interfacce grafiche e indirizzata alla creazione di siti e web app (un'altra sua versione denominata React Native è invece indirizzata allo sviluppo di app mobile).

La sintassi base di React è un'estensione della sintassi JavaScript ed è chiamata JSX. Questo tipo di sintassi consente l'utilizzo di tag HTML e codice JavaScript insieme

[8] Nel momento di stesura di questa tesi è disponibile nella versione beta di Figma una nuova funzione che consente la creazione di componenti già dotati di animazione nel passaggio tra uno stato e l'altro.

permettendo quindi di non separare nel codice le parti di markup e quelle logiche. Data la natura permissiva di React, l'utilizzo di elementi JSX non è obbligatorio ma può rivelarsi molto utile poiché rende più facile la comprensione del codice e consente di visualizzare messaggi di errori più efficacemente.

La possibilità di incorporare la logica al suo interno rende molto potente questo tipo di sintassi. Per esempio, all'interno del titolo `<h1>` è possibile inserire variabili o espressioni. Al contempo è possibile utilizzare gli attributi HTML passando i valori tra virgolette oppure espressioni JavaScript tra parentesi graffe. In questo caso, poiché JSX è considerata una sintassi più simile al JavaScript che all'HTML, si utilizza il *camelCasing*, cioè la scrittura della prima lettera della prima parola in minuscolo e la prima lettera della seconda parola in maiuscolo e così via.

Altre caratteristiche principali della sintassi JSX sono la possibilità, come in HTML, di avere figli e la chiamata diretta come oggetti. Il secondo caso, in particolare, è reso possibile dal fatto che un elemento JSX è una rappresentazione di un oggetto.

La gestione del DOM viene effettuata da React in maniera dinamica. Ciò significa che la renderizzazione di un elemento nel DOM sarà aggiornata solo nel caso in cui quello stesso elemento abbia aggiornato il suo stato (per esempio è stato modificato il testo di un elemento `<p>` a seguito di un evento `onClick` che ne modifica il contenuto). È bene notare però che un elemento React nel DOM è di per sé immutabile: quando viene modificato un suo stato, il vecchio elemento sarà confrontato con quello nuovo e quindi sostituito. Questa è una delle caratteristiche più interessanti di React poiché permette la creazione di applicazioni a singola pagina (*single page applications*) in cui i singoli elementi vengono modificati a seguito di interazioni ecc. (proprio come accade nel sito di Facebook).

React rende possibile suddividere il codice in base agli elementi della sua interfaccia utente. Queste parti indipendenti e riutilizzabili sono denominate "componenti". Un componente non è altro che una funzione JavaScript che riceve input di dati e ritorna elementi JSX da renderizzare. Fondamentali per i componenti sono le *prop*, cioè le proprietà (da "properties*"*), oggetti contenente dati che vengono passati al componente come parametri.

Una volta definito, un componente può essere integrato in altri componenti padri, di cui i principali già definiti da React sono App.js e index.js che si occupano rispettivamente dell'organizzazione dell'app e della sua renderizzazione. Il componente può essere richiamato proprio come se fosse un elemento HTML e le *prop* saranno ricevute dal componente proprio come se si trattasse di loro attributi.

La versione utilizzata di React è la 17.0.2. A partire dalla 16.8 sono stati introdotti gli *hook* di cui si è fatto largo uso nella fase di implementazione del progetto. Un *hook* permette di utilizzare gli stati di un componente senza doverlo scrivere come classe. Un *hook* è una particolare funzione che "àncora" gli stati di React dai componenti. Tra gli *hook* più basilari si trovano useState e useEffect.

Il primo si occupa di gestire una o più eventuali variabili di stato dichiarate all'interno di un componente funzione. Utilizzando sempre l'esempio del nome è possibile definire uno useState hook che permetta al componente, per esempio, di modificare un suo stato che riguardi un nome: se si vuole modificare o aggiornare la variabile `nome` si richiama `setNome` e gli si passa un valore (in questo caso una stringa) tra parentesi.

L'*hook* useEffect consente di eseguire una funzione dopo che i componenti che la riguardano siano già stati modificati nel DOM. In pratica, permette di aggiornare per esempio una variabile o uno stato in conseguenza della modifica di un'altra variabile o stato.

Per esempio, allo stato `nome` può essere inviato un nuovo valore contenuto nella variabile `nuovoNome` al variare del suo valore. UseEffect in questo caso è richiamato in base alla dipendenza specificata al termine come `[nuovoNome]`.

Oltre a questi due *hook* e altri già definiti da React è possibile crearne dei propri (*custom hooks*) che assolvano a funzioni specifiche definite dal programmatore.

3.2.2 React Router

React nella sua versione base non consente la navigazione tra pagine diverse. Questa impasse può essere superata tramite l'utilizzo di collezioni di componenti aggiuntive come React Router. Diviene così possibile costruire una web app in maniera composita che permetta la specificazione di URL diversi. In questo modo l'utente è in grado di poter salvare gli indirizzi nei preferiti del suo browser e in generale di poter accedere a sezioni differenti del sito tramite URL.

3.2.3 Material UI

Material UI (o più recentemente MUI) è un'altra collezione di componenti, in questo caso d'interfaccia, che permettono di definire uno stile grafico in linea con le *guidelines* stabilite da Google nella prospettiva Material Design (Google, Material Design Guidelines). MUI però, non cerca soltanto di rendere implementabili i principi delineati dal Material Design, ma fornisce una libreria di componenti grafiche (o combinazioni tra queste) non esplicitate nelle linee guida ufficiali.

Inserire una componente MUI in React significa in essenza importare un modulo. Per esempio, la componente `TextField` importerà un elemento contenitore per l'inserimento di input testuale che include proprietà riguardanti sia l'aspetto grafico (grandezza, stile e colori), sia il funzionamento (etichetta, valore di default, stati di errore).

A livello di *layout*, in rispetto delle linee guida del Material Design, MUI è strutturato in griglie suddivise in 12 colonne. Un elemento della griglia può essere di tipo *container*, l'elemento che contiene tutti gli altri, oppure *item*, il singolo elemento da disporre nella griglia. Ogni *item* ha come proprietà la sua larghezza espressa in numeri interi tra 1 e 12 che andrà a determinare lo spazio che occupa nella griglia. Questa proprietà è indicata come *breakpoint* (xs, sm, md, lg, xl), il che permette di mostrare la griglia in maniera *responsive* su dispositivi diversi.

3.2.4 Considerazioni sulle scelte

Per la scelta degli strumenti da utilizzare nell'implementazione dell'interfaccia utente sono stati considerati diverse soluzioni tra *framework*, librerie e approcci. Oltre a React sono state considerate altre opzioni possibili: Angular (Google), Vue.js (You) oppure l'approccio HTML con CSS e JavaScript. A tal fine sono state prodotte le seguenti considerazioni in riguardo ai primi due *framework*:

- Angular è un popolare *framework* sviluppato da Google. Le sue caratteristiche considerate sono:
    a. La sua maggiore maturità e completezza;
    b. È facile da mantenere;
    c. Indicato per lo sviluppo di web app complesse;
    d. Indirizzato maggiormente a coloro che già conoscono linguaggi come Java e C#;
    e. Ha una curva di apprendimento ripida.
- Vue.js è una libreria open-source le cui peculiarità sono:
    a. Facilità di apprendimento;
    b. Ha prerequisiti basilari come HTML, CSS e JavaScript;
    c. Utilizza una scrittura HTML standard (e non JSX come React);
    d. È il più permissivo di tutti, il che può implicare però difficoltà nel debugging.

HTML, CSS e JavaScript insieme sono gli strumenti più classici e utilizzati per il web design e in quanto tali rappresentavano ad inizio lavori l'unica soluzione conosciuta e già utilizzata precedentemente tra quelle analizzate. Quindi la questione della scelta si poneva inizialmente tra questi strumenti e uno qualunque dei *framework*.

Da questa considerazione inziale è stato possibile elaborare un'analisi costi/benefici che ha permesso di evidenziare un più alto costo (in termini di tempo), ma anche un più alto beneficio (in termini di risultato) nell'imparare uno dei *framework*. La motivazione principale risiede nelle possibilità fornite da queste librerie moderne, continuamente aggiornate e supportate, affinché si potesse ottenere una *single page application* dinamica.

La scelta è ricaduta su React poiché rappresenta un *framework* abbastanza maturo con molti contributi da parte di una folta *community*, il cui apprendimento è facilitato dall'utilizzo di molteplici guide disponibili online a partire dal sito ufficiale. Inoltre, la presenza di un abbondante numero di librerie esterne permette una grande libertà implementative e grafiche. La curva di apprendimento si è infine rivelata essere poco ripida (soprattutto se confrontata a quella per l'apprendimento di Angular) in virtù dell'uso di codice JavaScript e HTML, anche se con le opportune modifiche.

Nell'utilizzo dello stesso React si è scelto sin dall'inizio di utilizzare i paradigmi più recenti indicati dalla documentazione ufficiale o comunque seguiti all'interno della sua *community*, cioè l'impiego di elementi JSX, della sintassi JavaScript ES6 e l'utilizzo di componenti-funzioni piuttosto che classi.

CAPITOLO 4

Contributo

Il contributo di cui tratta questa tesi è stato strutturato in quattro fasi principali:

1. Analisi del problema generale posto dalla tassonomia e sua contestualizzazione e comprensione per mezzo dell'Information Architecture.
2. Elaborazione delle *user personas* e individuazione dei requisiti iniziali.
3. Elaborazione dei *wireframe* e primi prototipi da loro derivati.
4. Implementazione *front-end*.

Il processo è stato realizzato senza soluzione di continuità tra una fase e l'altra affinché fosse possibile procedere iterando tra le diverse fasi. Il ritorno periodico alle considerazioni di ogni fase ha permesso di aggiornare le valutazioni precedentemente fatte e di far evolvere il design in maniera sempre più coerente alla nuova conoscenza disponibile. Per esempio, ad ogni nuovo approfondimento delle tematiche dell'Information Architecture e della conoscenza delle *personas* è stato possibile progettare *wireframe* e prototipi migliori e di conseguenza riconsiderare alcuni aspetti implementativi.

Il risultato finale del processo è rappresentato da una prima versione completamente funzionante del sito in grado di comunicare con il *back-end*.

4.1 Analisi e comprensione della tassonomia

La prima fase di lavoro ha comportato la comprensione dell'oggetto principale in esame: la tassonomia elaborata da Amadio, Isgandarova e Mazzei (2021). Innanzitutto, la sua natura logica: una tassonomia (nella sua accezione più stretta, v. 2.1) è una struttura gerarchica che categorizza elementi di uno stesso dominio in maniera

sussuntiva, cioè dove il più specifico è compreso nel più generale. Data questa prima considerazione, osservando da una parte i problemi e i bisogni dell'Industria e dall'altra le tecnologie abilitanti che li risolvono, si avranno due domini separati con le loro rispettive sottocategorie.

Questo suo aspetto duplice potrebbe essere tale da considerare la tassonomia come due tassonomie separate (come in effetti sarebbero). Tuttavia, può essere assunta una prospettiva che le ponga come integrate tra loro e in cui i casi d'uso analizzati nel lavoro di elaborazione della tassonomia e indicizzati in essa permettono di collegare i due domini e renderli interoperabili ai fini di una ricerca. Pertanto, è necessario considerare sia l'aspetto logico (la natura dei concetti, la loro categorizzazione e l'interconnessione e interoperabilità tra le tassonomie) sia l'aspetto pratico (l'utilizzo di cui realmente se ne farà) al fine di produrre una comprensione dell'oggetto "tassonomia" manifestabile in un'interfaccia. A questo fine sono state vagliate le modalità di combinazione tra tassonomie.

Quando si tratta di combinare due tassonomie sono possibili quattro operazioni (Hedden 2016, pp. 366-379) da cui risulteranno:

1. Tassonomie integrate (*Integrating taxonomies*): differenti tassonomie che trattano diversi aspetti si combinano in un'unica tassonomia "master". Le tassonomie rimangono intatte dopo l'integrazione.
2. Tassonomie fuse (*Merging taxonomies*): si fondono due o più tassonomie che trattano la stessa materia o una molto simile. Le categorie risultanti saranno l'incontro tra le due tassonomie (ovvero scompare la struttura originale).
3. Tassonomie mappate (*Mapping taxonomies*): si collegano due tassonomie che trattano la stessa materia, lasciando però intatta le loro singole strutture, al fine di rendere disponibili due modalità di utilizzo diverse.
4. Tassonomie con dati collegati (*Linked data taxonomies*): nascono dalle specifiche W3C per il Semantic Web. I nodi delle tassonomie sono collegati semanticamente come concetti utilizzando link.

Eccettuata la fusione tra le tassonomie, dove i concetti delle due gerarchie sarebbero dovuti confluire in concetti che li accorpassero insieme, sono state prese in esame le operazioni 1, 3 e 4.

Dato l'accento sull'aspetto del collegamento tra le due tassonomie, è stata prima di tutto considerata la terza opzione. La mappatura tra due tassonomie (*Mapping taxonomies*) riguarda il collegamento tra due gerarchie tramite la corrispondenza dei loro concetti. Una corretta operazione di mappatura richiede quindi che tra i termini collegati ci sia una relazione di equivalenza. Per questo motivo un tipico utilizzo di questa modalità di collegamento tra tassonomie consiste nel mappane una indicata per l'utente finale ed una per l'indicizzazione, in cui quindi i termini specialistici sono corrisposti a termini più d'uso comune. Questo però non si applica alla tassonomia per l'Industria 4.0 perché concetti come "Smart warehouse" (che appartiene al dominio dei bisogni) e "Cloud Data Storage" (che appartiene al dominio delle tecnologie), sono collegati tra loro attraverso alcuni casi d'uso ma non si corrispondono in maniera equivalente.

La quarta modalità pone l'accento sull'utilizzo dei concetti in quanto essi stessi contenuti e fini della ricerca. Qualora i concetti di una tassonomia siano dotati di un indirizzo HTTP che li renda identificabili si rende possibile il collegamento tra concetti di due tassonomie raggiungibili online. Anche in questo caso non sembra trattarsi della problematica della tassonomia per l'Industria 4.0, almeno non nelle sue fattezze inziali[9]. In particolare, il collegamento dei concetti è stato realizzato tramite l'individuazione di casi d'uso indicizzati all'interno di tutte e due le tassonomie senza il ricorso a collegamenti di tipo semantico tra i concetti stessi o reperimenti online.

L'ultima possibilità è quella indicata dall'integrazione di tassonomie. In questa modalità due tassonomie (e particolarmente quelle gerarchiche o quelle *faceted*) utilizzano dei contenuti condivisi permettendo il loro uso combinato nella stessa interfaccia. Nella tassonomia per l'Industria 4.0 i casi d'uso sono appunto dei

[9] La situazione potrebbe essere riconsiderata nel momento in cui la tassonomia sarà aggiornata.

documenti condivisi dalle sue due gerarchie (bisogni e tecnologie) e le relazioni che sussistono tra i concetti di queste sono di tipo associativo (sono cioè correlati tra loro, v. 2.4.3). Inoltre, il fine di questo progetto risiede proprio nella realizzazione di un'interfaccia che renda disponibili contemporaneamente e in maniera interoperabile le due tassonomie. La tassonomia per l'Industria 4.0 può essere quindi precisamente definita come il risultato dell'integrazione della tassonomia dei bisogni d'industria e della tassonomia delle tecnologie abilitanti attuata per mezzo dell'associazione tra i loro concetti individuata a partire dai casi d'uso indicizzati.

Nella sua organizzazione interna la tassonomia per l'Industria 4.0 è strutturata su tre livelli per ognuna delle sue gerarchie[10] (app. 1). Il collegamento tra bisogni e tecnologie però non si verifica solo al livello delle "foglie" ma ad ognuno di questi livelli. Questo perché una tecnologia può essere indicata per la risoluzione di una problematica d'industria che viene identificata in un concetto più generico rispetto ad altri possibili. A loro volta, le tecnologie abilitanti possono sia essere identificate in maniera generica (per esempio "Embedded Computing") sia in una sua specifica soluzione ("Arduino").

Per costruire un'interfaccia che espliciti il modo in cui le due gerarchie si collegano e ne permetta la navigazione è necessario identificare i fini per cui devono essere rappresentate. Se il fine è la ricerca e quindi il reperimento dei contenuti indicizzati nelle due gerarchie (i casi d'uso), allora possono essere adottate strategie di design convenzionali e già identificate nelle *best practice*. Se, invece, il fine è il collegamento e il trasferimento di saperi e conoscenze tra il mondo accademico e quello aziendale (come prospetta il progetto Planet4), allora il fine della rappresentazione sta nel collegamento stesso tra le gerarchie. In questa seconda prospettiva, i casi d'uso indicizzati all'interno della tassonomia svolgono un ruolo decisivo nel collegamento tra le due gerarchie interne ma non nell'essere considerati solamente come dei nodi di termine (cioè dove l'utente termina la navigazione): ci si focalizza sulla visualizzazione della tassonomia piuttosto che sul suo utilizzo per la ricerca di

[10] Con l'unica eccezione della tecnologia "Industrial IoT" che si sviluppa su quattro livelli.

documenti indicizzati in essa. La comprensione della tassonomia fornita dall'IA ha permesso quindi di evidenziare il ruolo atipico che svolge: la tassonomia non è solo il mezzo tramite cui si ottiene il risultato ma è l'oggetto stesso della ricerca.

4.2 Elaborazione delle personas

Parallelamente al lavoro sulla comprensione della tassonomia per l'Industria 4.0 si è cercato di comprendere quali sono gli utenti finali a cui il sito è indirizzato. È infatti impossibile costruire il giusto prodotto senza la cognizione di chi saranno le persone che effettivamente lo utilizzeranno e quali sono le loro problematiche. Le *personas* sono uno degli strumenti principali a disposizione dello UX designer per tale scopo. Consentono di catturare i tratti salienti e i bisogni degli utenti nei rispetti del problema di design che si vuole risolvere.

Una *persona* è un personaggio di finzione archetipico che viene rappresentato usualmente con alcuni dei seguenti elementi o caratteristiche:

- Un nome e una foto.
- Caratteristiche personali rilevanti: genere, età (o range di età), istruzione e formazione, ruolo e luogo di lavoro.
- Tratti personali: personalità, attitudini e comportamenti.
- Bisogni ed obiettivi.

La presenza di questi dettagli aiuta i progettisti e gli sviluppatori nel processo di empatizzazione con gli utenti *target*. Il fine è quello di pensare a cosa vorrebbe e farebbe l'utente piuttosto che a quello che farebbe il designer o il programmatore.

Dati questi elementi una *persona* non può tenere insieme tipologie di utenti troppo diversi tra loro. Andrà invece a descrivere e rappresentare classi di utenti individuate a seconda del dettaglio e dello scopo necessari. Lo *scope*, ovvero la portata delle *personas*, identifica la qualità del dettaglio di ogni singola *persona*: determina quindi quante tipologie di utenti sono identificate all'interno di una *persona*. Sono possibili

portate ampie e ristrette, a seconda delle necessità e della quantità di dati a disposizione.

La prima fase per l'elaborazione delle personas è stata la preparazione di *proto-personas*, cioè di *personas* create senza l'accesso a dati (qualitativi o quantitativi) e quindi basate su ipotesi. Innanzitutto, si è cercato di individuare lo *scope* domandandosi "per cosa sono destinate le *personas*?". Per rispondere a questa domanda si sono esaminati gli obiettivi più generali del progetto Planet4 che includono fondamentalmente due aspetti: il collegamento tra mondo del lavoro e accademia e la formazione di nuovi lavoratori adatti a lavorare in un contesto di Industria 4.0. D'altra parte, l'elaborazione della tassonomia e la sua rappresentazione e navigazione sono all'interno della prima fase del progetto Planet4 e raffigurano uno dei punti d'entrata per gli utenti principali. Le *personas* possono quindi includere sia dettagli di ampia portata che rispondono alla domanda "quali sono i suoi obiettivi di lavoro?", sia dettagli di portata ristretta che rispondono alla domanda "come utilizzerebbe il motore di ricerca?".

A partire da queste considerazioni è stato individuato il numero e i tipi di *personas* necessari. L'obiettivo stabilito dal progetto Planet4 di connettere il mondo del lavoro e accademia permette di identificare in questi due ambienti la prima divisione tra *personas*. Nel mondo del lavoro, più specificamente in quello delle aziende che hanno bisogno di operare in un contesto 4.0, coloro che si occupano maggiormente dell'innovazione sono i responsabili di ricerca e sviluppo (R&D Manager) e, per l'appunto, i responsabili per l'innovazione (Innovation Manager). I primi lavorano all'interno del contesto aziendale mentre i secondi agiscono come consulenti d'azienda. A livello accademico, invece, i principali utenti target sono stati individuati nei professori e negli studenti che si occupano o studiano materie informatiche (ICT professor e ICT student).

Per cercare di tenere lontani possibili pregiudizi personali il genere delle *personas* è stato selezionato in maniera casuale per mezzo di un generatore di scelte[11]. Per quanto

[11] https://www.textfixer.com/tools/random-choice.php

riguarda l'età, invece, sono stati esaminati profili diversi di lavoratori nei rispettivi ruoli per avvicinarsi ad una stima di cui il risultato mostrato è la media.

Stabiliti gli utenti target sono state stilate le rispettive *proto-personas*:

- Sylvia Weber
    - Lavoro: R&D Manager.
    - Informazioni generali: 44 anni, non spostata, formazione scientifico/ingegneristica, parla inglese.
    - Tratti caratteriali: razionale, pianificatrice, "hard-skilled".
    - Responsabilità e comportamenti: assume e introduce nuovi impiegati; deve tenere traccia di più progetti al contempo; lavora insieme ad un team.
    - Obiettivi e bisogni: rimanere aggiornata sugli sviluppi tecnologici; coordinare il team; costruire relazioni.
- Paolo Rinaldi
    - Lavoro: Innovation Manager.
    - Informazioni generali: 48 anni, spostato con due figli, formazione economica, parla inglese e tedesco.
    - Tratti caratteriali: ottimista, curioso, socievole, "soft-skilled".
    - Responsabilità e comportamenti: partecipa a molte riunioni durante la giornata; consiglia scelte strategiche.
    - Obiettivi e bisogni: cerca di trovare modi per usare nuove tecnologie in nuove soluzioni di mercato; ha bisogno di collaboratori con una mentalità flessibile.
- Filippo Carrera
    - Lavoro: professore di Informatica.
    - Informazioni generali: 50 anni, spostato con un figlio, formazione ingegneristica, parla inglese.
    - Tratti caratteriali: disponibile, analitico.
    - Responsabilità e comportamenti: insegna nei corsi e fa ricerca; valuta gli esami; supervisiona il lavoro degli studenti.

    - Obiettivi e bisogni: vuole fare da guida agli studenti; cerca di supportare il collegamento con il mondo del lavoro.
- Alessandro Triani
    - Lavoro: studente di Informatica
    - Informazioni generali: 26 anni, studente di Intelligenze Artificiali o Data Science, parla un po' di inglese, ha poca esperienza lavorativa.
    - Tratti caratteriali: appassionato di tecnologia,
    - Responsabilità e comportamenti: si prepara per gli esami; è interessato nelle nuove tecnologie; segue i corsi universitari.
    - Obiettivi e bisogni: vuole trovare un lavoro che sia in linea con la propria preparazione; ha bisogno di collegamenti con le aziende.

Le *proto-personas* sono basate su ipotesi del progettista o del gruppo in cui sono elaborate e quindi parte delle valutazioni possono essere generate a partire da *bias* e pregiudizi. Oltre a ciò, sono comunque limitate alle conoscenze pregresse del designer e in quanto tali utili solo come strumento di contestualizzazione iniziale e non per uno sviluppo completamente coerente con i principi dello Human-Centered Design. Dove e quando possibile si deve quindi provvedere ad una osservazione o studio degli utenti reali. Nei confronti delle figure più lontane dal contesto in cui il progetto è stato sviluppato, ovvero quelle provenienti dal mondo del lavoro, è stato possibile fare affidamento all'esame di interviste semi-strutturate (con una media di otto domande per intervistato) effettuate da Riccardo Amadio e Daniele Atzeni.

L'obiettivo principale delle interviste è stato quello di identificare bisogni, aspettative e conoscenze delle tematiche legate al paradigma Industria 4.0 e le sue tecnologie. Particolare attenzione è stata rivolta anche alle aspettative di competenze che le aziende hanno nei confronti dei loro impiegati. Le persone intervistate sono state cinque e nelle loro rispettive aziende sono impiegati nei dipartimenti di ricerca e sviluppo, manifattura, manutenzione, IT, tecnologia e Data Science, mentre i loro ruoli vanno dal R&D Manager e Innovation Manager alla direzione commerciale e all'amministrazione.

Le domande e le risposte delle interviste sono state riportate in un documento di cui, tramite *journaling*, si sono esaminati i contenuti. Durante la fase di lettura sono state dunque evidenziate prima le frasi salienti dei rispondenti, poi annotati i processi di pensiero e le idee generate a partire da quelle risposte. La tecnica del *journaling* ha permesso così di incoraggiare la riflessione sulle interviste per poi arrivare all'individuazione dei temi più ricorrenti trattati. Le tematiche hanno permesso di individuare bisogni non ipotizzati nelle *proto-personas* come, ad esempio, le necessità di poter utilizzare vecchi macchinari integrandoli nei nuovi sistemi e di avere a disposizione personale che possieda buone capacità comunicative anche a costo di rinunciare alle capacità tecniche. I risultati evidenziati nel *journaling* sono stati riportati su carta in modo da poterli categorizzare in temi.

In questa maniera è stato poi possibile raffinare le *proto-personas* già preparate. I temi identificati sono stati suddivisi a seconda della congruenza dei ruoli e delle caratteristiche tecniche e di pensiero degli intervistati con quelle delle *proto-personas*. In questo modo è stato possibile osservare, confrontare e, in caso, eliminare o aggiornare gli elementi precedentemente ipotizzati (fig. 1).

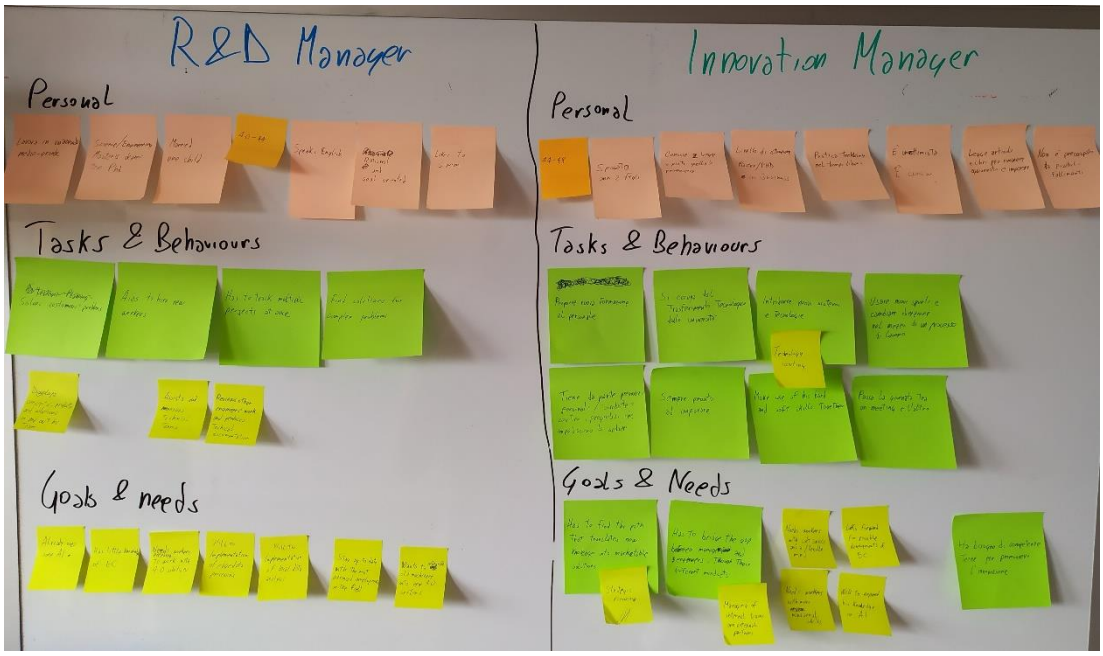

Figura 1. Raffinazione delle personas (R&D Manager e Innovation Manager) per mezzo dei nuovi temi individuati.

Dall'analisi delle interviste, inoltre, è stato possibile evidenziare una nuova categoria da aggiungere ad ogni singola persona: la sua esperienza e abilità con le tecnologie abilitanti dell'Industria 4.0 (fig. 2 e 3). Inoltre, per veicolare nella maniera più efficiente possibile i tipi di utenti le quattro *personas* sono state rappresentate scegliendo per ognuna di loro un volto che le rendesse meglio memorizzabili e identificative delle loro caratteristiche. La scelta della foto rappresentativa di una persona è stata effettuata tenendo conto delle informazioni generali e dei tratti caratteriali.

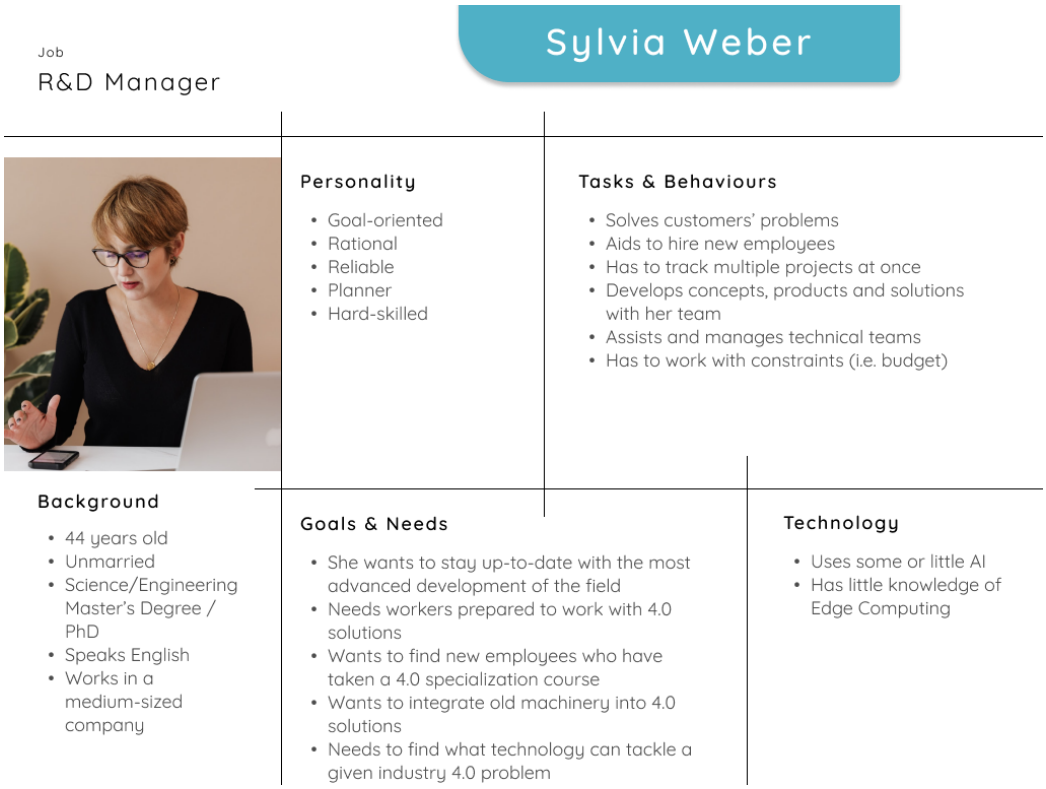

Figura 2. User persona R&D Manager

Figura 3. User persona Innovation Manager

Per le *proto-personas* del professore e dello studente (fig. 4 e 5) si è invece proceduto ad una loro raffinazione tramite il confronto interno nel gruppo di lavoro, in quanto esso stesso rappresentante di queste categorie. Inoltre, in questo caso si è potuto fare affidamento a ricerche che fornissero dati psicometrici (Raza, Zaka-ul-Mustafa e Capretz 2011). Il raffinamento della personalità delle *personas* consente di comprendere in che modo sia più opportuno direzionare le scelte di design affinché siano in linea con i tratti caratteriali degli utenti finali. Questo può rilevarsi utile perché la personalità è l'insieme dei modi in cui una persona interagisce con il mondo, e quindi anche con i prodotti e servizi che risultano da un processo di design. In particolare, i tratti caratteriali implementati nelle *personas* dalle ricerche psicologiche sono stati definiti per mezzo dell'indicatore di tipi Myer-Briggs (o MBTI) (Myers e Myers 1980).

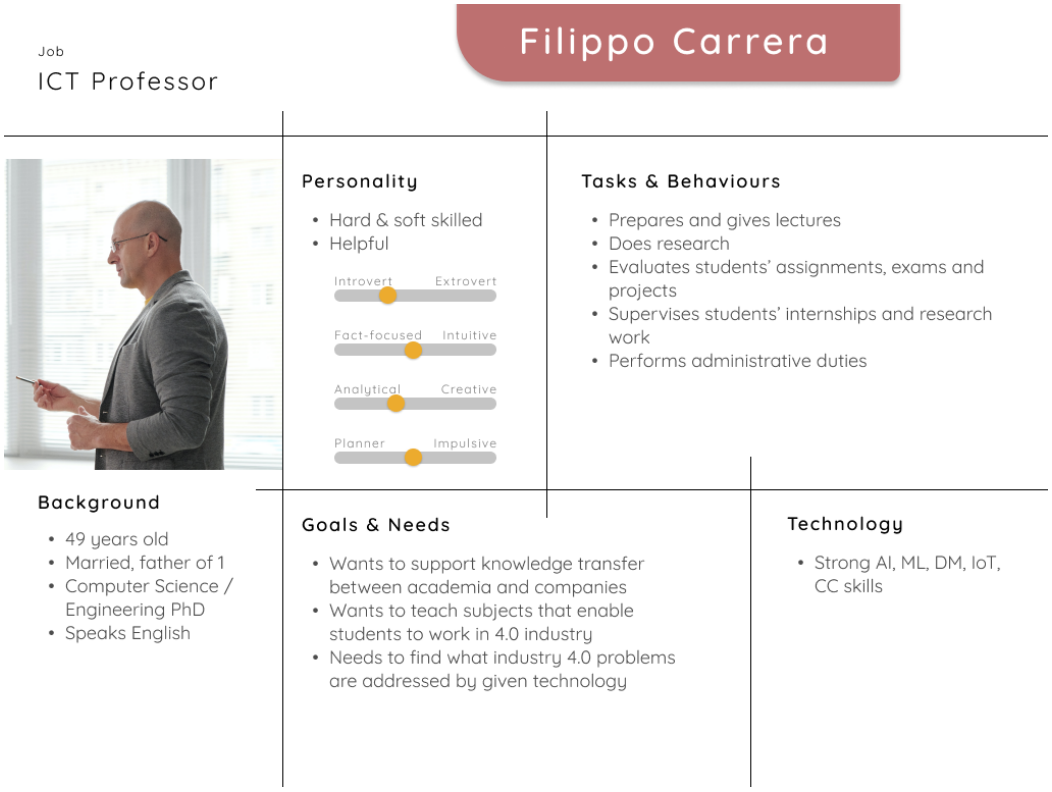

Figura 4. User persona professore di informatica

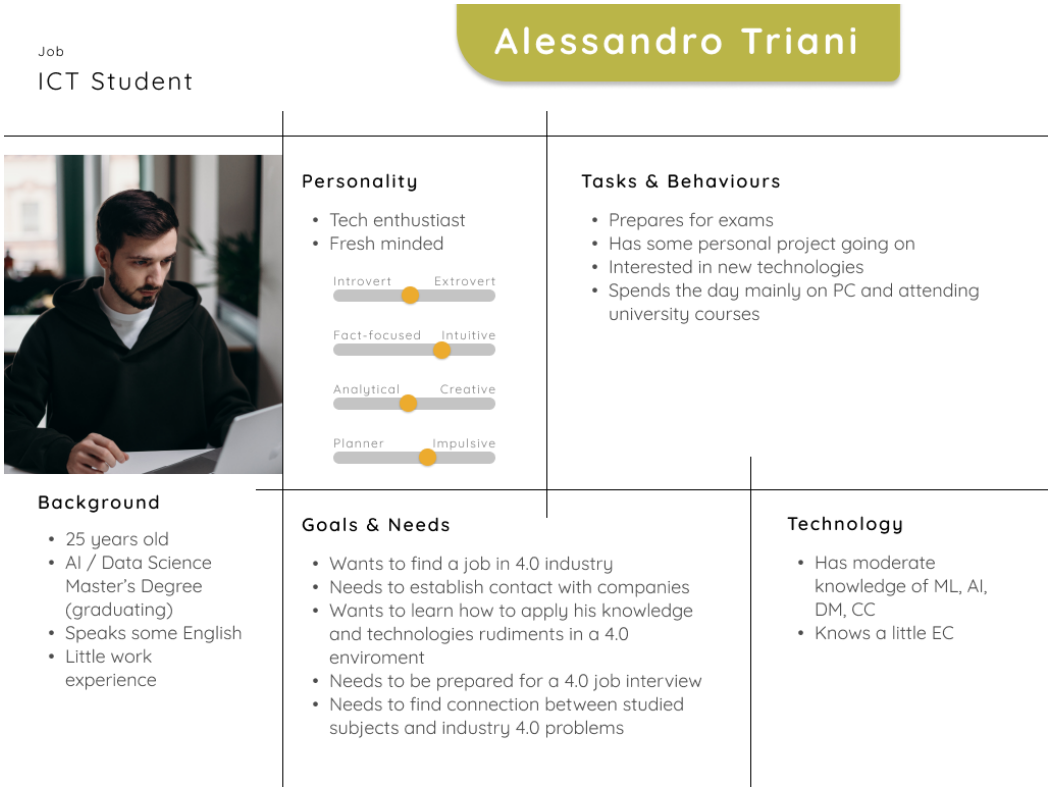

Figura 5. User persona studente di informatica

4.3 Requisiti identificati

A seguito dell'analisi della tassonomia e dell'elaborazione delle *personas* sono stati individuati dei requisiti per lo sviluppo dell'interfaccia. I requisiti sono le caratteristiche e gli elementi di cui l'utente ha bisogno. Dato un prodotto, un servizio o in generale una soluzione, i requisiti funzionali sono le caratteristiche necessarie per il suo funzionamento, mentre i requisiti non funzionali indicano i suoi criteri o limiti di sviluppo (tempistiche di rilascio previste ecc.) oppure i suoi dettagli performativi (performance, usabilità, adattabilità ecc.).

I requisiti funzionali individuati sono:

1. L'utente dovrebbe poter esplorare la tassonomia nella sua interezza.
2. Il sito dovrebbe avere consentire di essere esplorato a seconda dell'esperienza dell'utente.
3. La barra di ricerca potrebbe dare dei suggerimenti quando l'utente effettua una ricerca.
4. Il sito dovrebbe restituire i risultati della tassonomia più appropriati.
5. I risultati devono aiutare l'utente a collegare tecnologie e bisogni.
6. I termini utilizzati dovrebbero essere comprensibili agli utenti.
7. I casi d'uso devono essere mostrati.
8. L'utente deve poter aggiungere nuovi casi d'uso e classificarli nella tassonomia.
9. L'utente dovrebbe poter effettuare una ricerca utilizzando i termini che ritiene più appropriati.

I requisiti non funzionali invece sono:

1. Il sito deve essere *responsive.*
2. Il sito deve poter essere facilmente aggiornabile.
3. Le risposte fornite dal server devono essere visualizzate nel minor tempo possibile.
4. Il sito potrebbe garantire la sua adattabilità in base all'utente che lo naviga.
5. Deve essere possibile rilasciare una prima versione del sito in pochi mesi.

I requisiti riportati hanno influenzato le scelte sia di design nella fase di *wireframing* sia di implementazione.

4.4 Elaborazione dei *wireframe*

Il procedimento seguito per la preparazione dei *wireframe* può essere definito "*top-down*": è stata individuata prima la struttura generale del sito, poi gli elementi principali delle schermate di ricerca e dei risultati, ed infine si è scesi nel dettaglio di ogni elemento. Quindi, si è cercato di dare prima di tutto una coerenza generale al sistema di visualizzazione e poi occuparsi in modo granulare di ogni sua caratteristica.

Figura 6. Sitemap del sito

Il sito ha una struttura semplice che prevede tre pagine (fig. 6):

- "Home" è la pagina di entrata che contiene i principali elementi di navigazione.
- "SERP" (*Search Engine Results Page*) è la pagina in cui vengono mostrati i risultati.
- "Add Article" è la pagina in cui l'utente può aggiungere nuovi casi d'uso o articoli e indicizzarli nella tassonomia.

4.4.1 Gli elementi di navigazione

In questa prima iterazione sono stati progettati dei *wireframe* che cercassero di risolvere le problematiche di rappresentazione della tassonomia e che assolvessero prima di tutto ai requisiti funzionali. Attraverso un'analisi del task flow indirizzato alla sola prima attività di ricerca e/o navigazione, e in accordanza con le *best practice* per la navigazione delle tassonomie (v. 2.7), sono stati individuati due elementi principali per supportare questa attività e per rispondere al primo e al secondo requisito funzionale. Considerando un utente esperto nelle tematiche trattate dalla tassonomia ed uno meno esperto si avranno:

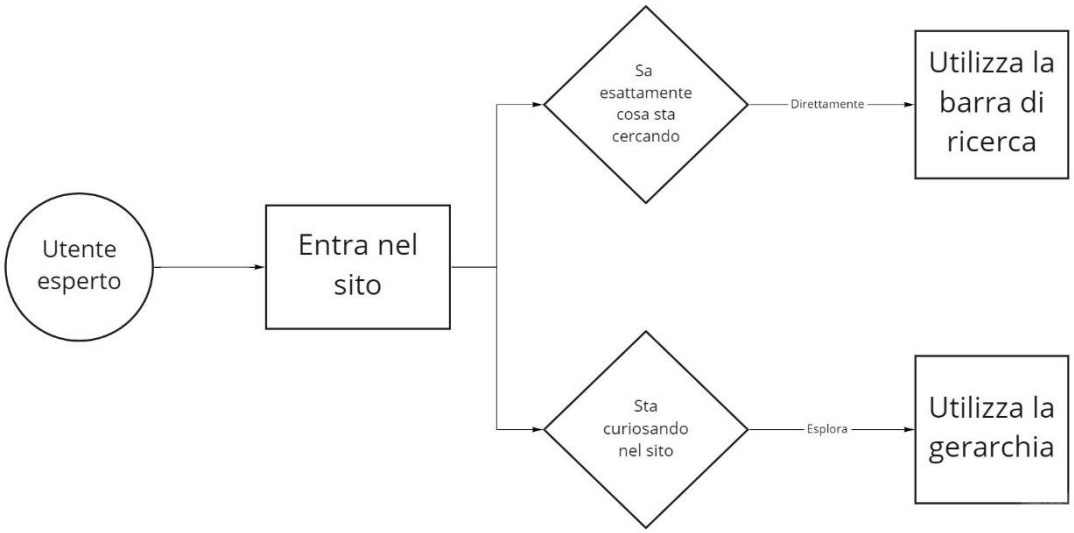

Figura 7. Task flow ricerca utente esperto

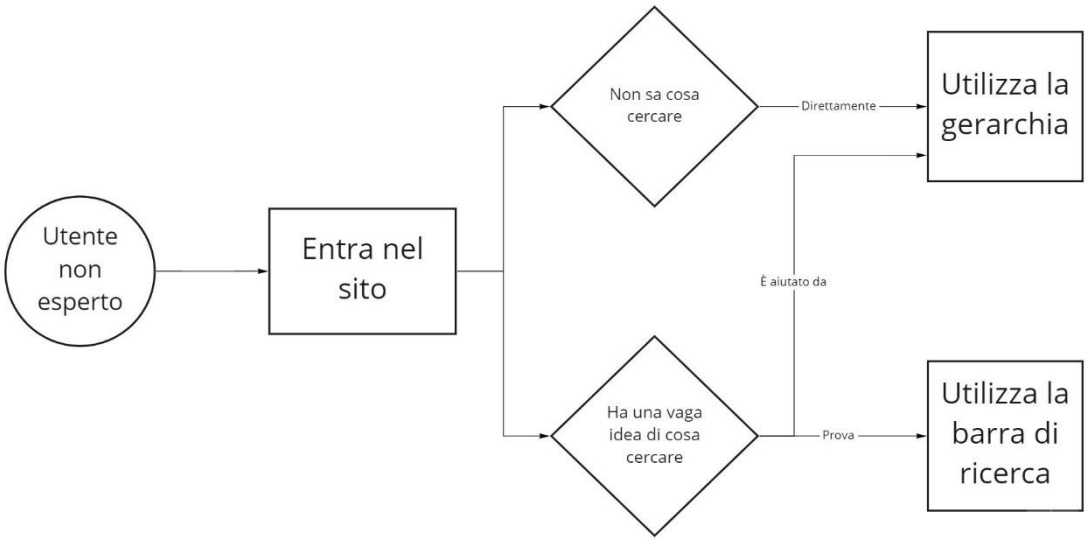

Figura 8. Task flow ricerca utente non esperto

Gli elementi principali di navigazione sono stati individuati precocemente in base ad una considerazione sul livello di esperienza dell'utente. Nel caso in cui l'utente abbia già una discreta esperienza nelle tematiche del paradigma Industria 4.0, allora utilizzerà direttamente una barra di ricerca. Al contrario, in caso l'utente abbia meno esperienza si avvarrà dell'esplorazione dei contenuti disponibili e quindi utilizzerà una gerarchia sfogliabile (*browsable*). Un utente con poca esperienza può per esempio essere uno studente che abbia cominciato da poco ad utilizzare la terminologia corretta. Inoltre, l'analisi di questo singolo primo task permette di individuare situazioni in cui anche l'utente esperto possa utilizzare la gerarchia sfogliabile oppure dove l'utente meno esperto provi ad effettuare una ricerca tramite l'apposita barra. Infatti, anche un utente esperto che sia un professore o un manager d'azienda, per esempio, potrebbe ritrovarsi a vagare nel sito senza un obiettivo prefissato ma solo allo scopo di navigarne i contenuti. Supportare una tale attività "serendipica" può essere di qualche rilievo per una piattaforma che consente di trovare soluzioni a problematiche di varia natura e viceversa. D'altra parte, l'utente meno esperto potrebbe comunque avere una vaga idea di cosa sta cercando e quindi tentare una ricerca tramite l'apposita barra. Quindi gli elementi principali di navigazione della tassonomia individuati sono la barra di ricerca e una gerarchia sfogliabile. Nella fattispecie della gerarchia sfogliabile, la modalità di rappresentazione adottata è quella della visualizzazione dell'albero (*treeview*) in quanto consente di comunicare visualmente in maniera chiara la struttura gerarchica

della tassonomia. A questi elementi si aggiunge la n*avbar* che consente la navigazione tra le pagine del sito fornendo la possibilità di tornare alla Home oppure di raggiungere la pagina per l'aggiunta di articoli.

4.4.2 La pagina principale di ricerca

Con i primi *wireframe* a bassa fedeltà su carta si è cercato di elaborare in maniera visuale una proposta di design che includa gli elementi di navigazione individuati.

In questa fase sono stati adottati i principi che indicano il miglior posizionamento di ogni elemento, cioè barra di ricerca in alto e gerarchia sfogliabile (in quanto *navigation bar*) a sinistra, prevedendo la successiva visualizzazione dei risultati nello spazio a destra.

Al fine di tenere separata la gestione dei risultati in un'apposita pagina (quindi rispettare la divisione in pagine elaborata nella *sitemap*) e di rendere il CTA (*Call To Action*) alla barra di ricerca o alla gerarchia più immediato, sono stati elaborati (fig. 9), nel passaggio all'alta fedeltà, dei *wireframe* che contenessero solo questi elementi senza considerare lo spazio per la visualizzazione dei risultati.

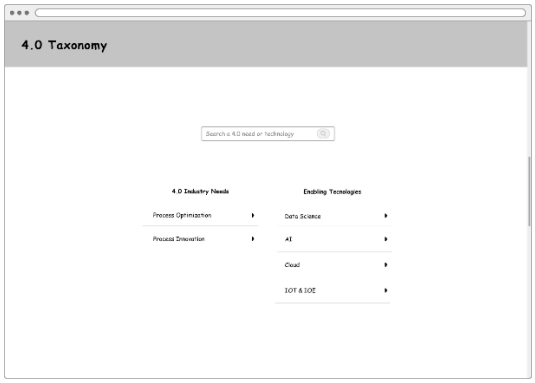

Figura 9a. Primo wireframe ad alta fedeltà

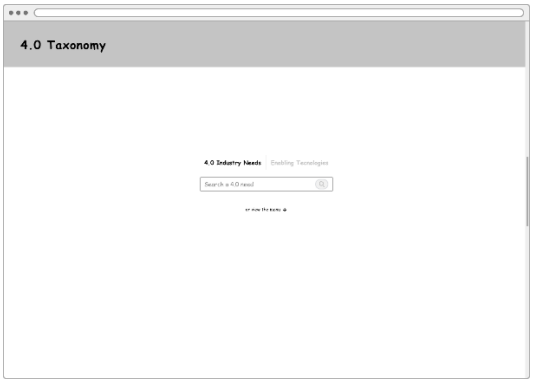

Figura 9b. Seconda elaborazione wireframe

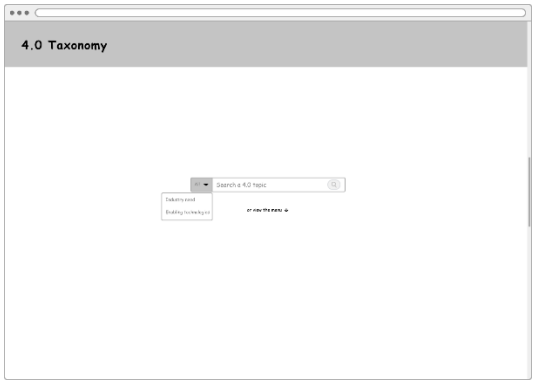

Figura 9c. Rielaborazione del pre-filtering

In questo wireframe si è cercato di dare eguale peso alla presenza della barra di ricerca e alla gerarchia. Tuttavia, date le *personas* tra cui l'unico meno esperto è rappresentato dallo studente, è stata riconsiderata la presenza della gerarchia come una possibile opzione piuttosto che appesantire la schermata con troppi dettagli. Per questo, la gerarchia è stata successivamente (fig. 9b) sostituita con un bottone che permette di visualizzarla solo nel caso in cui l'utente lo decida. Questo è anche un modo per cercare di indirizzare l'utilizzo più verso la barra di ricerca e quindi invogliare l'utente a provare a effettuare una ricerca (cosa che può condurre a più risultati di per sé rispetto ad una ricerca tramite la gerarchia).

Cercando di utilizzare un linguaggio il più possibile comprensibile e vicino a quello dell'utente – in accordanza con la seconda euristica di Nielsen, "Match between system and the real word" (Nielsen 1994-2020) – sono stati evitati termini più tecnici

come “taxonomy” o “hierarchy” a favore del più immediato “menu” in questa fase. È stata inoltre esplorata l’ulteriore possibilità di fornire all’utente un modo per raffinare la ricerca prima ancora di effettuarla attraverso due bottoni che permettono di selezionare il ramo della ricerca (*pre-filtering*): cliccando su “4.0 Industry Needs” per esempio l’utente indirizzerebbe la sua ricerca primariamente nella tassonomia riguardante i bisogni d’industria, ricevendo come risultati i collegamenti alle tecnologie. In queste prime fasi di design, infatti, si sono considerati i risultati della ricerca da mostrare seguendo questo schema: se l’utente ha un bisogno vuole trovare le sue soluzioni e viceversa.

In questa direzione è stato rielaborato il *wireframe* (fig. 9c) cercando di migliorare la visualizzazione del *pre-filtering* e permettere anche una ricerca all’interno di tutta la tassonomia.

Questa modalità di ricerca è stata abbandonata però in favore di qualcosa di più immediato, quindi alla semplice barra di ricerca priva di *pre-filtering*. La motivazione di questa scelta risiede nel fatto che vuole essere data all’utente la maggiore flessibilità di ricerca possibile utilizzando il linguaggio che ritiene più appropriato (sarà il motore di ricerca ad aiutarlo poi). Inoltre, gli aspetti in cui è divisa la tassonomia sono solo due: bisogni industriali e tecnologie abilitanti. Un motore di ricerca che debba tenere conto di molte più categorie diverse (come, per esempio, quello di Amazon) permette un *pre-filtering* in grado di discernere prima i possibili risultati ambigui di categorie non pertinenti agli interessi dell’utente. In questo caso invece i risultati saranno sempre sia bisogni sia tecnologie, proprio perché l’obiettivo è quello di creare una conoscenza condivisa che permetta di connettere questi due aspetti.

Il risultato finale (fig. 10) mostra il *wireframe* utilizzato come riferimento per l’implementazione della Home, dotato cioè di barra di ricerca, gerarchia sfogliabile (*treeview*) opzionale e n*avbar*.

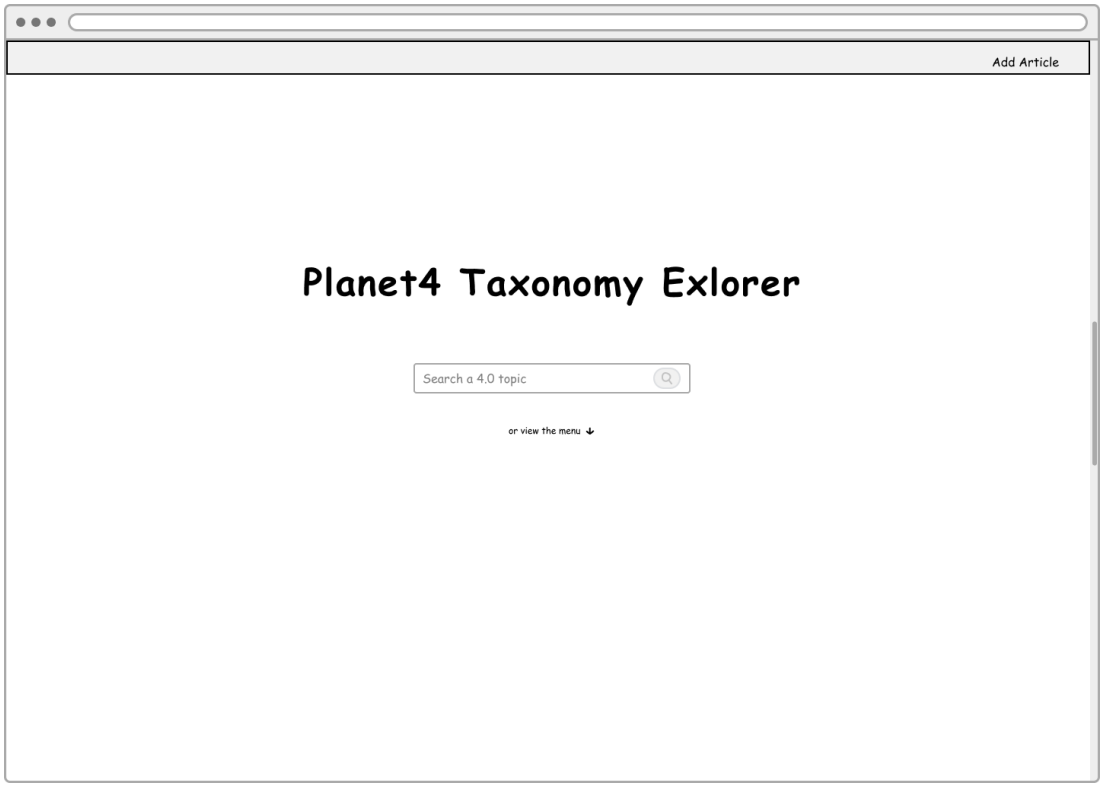

Figura 10. Wireframe della Homepage finale

4.4.3 La pagina dei risultati della ricerca (SERP)

I risultati della ricerca sono mostrati in una pagina apposita che li presenti e permetta di interagire con essi. Le prime elaborazioni sono state guidate dalle iniziali informazioni rese disponibili dallo studio dell'Information Architecture, cioè dalla considerazione che i risultati da mostrare fossero i casi d'uso. Infatti, le tassonomie sono solitamente utilizzate come strutture per categorizzare i contenuti disponibili e permetterne la navigazione e l'accesso con più efficienza. Ponendo il caso che l'utente sia un R&D Manager, qualora ricerchi una problematica d'industria che si ritrova ad affrontare vorrebbe trovare i modi in cui la possa risolvere. Se l'accento è posto sul modo in cui il problema deve essere affrontato allora i casi d'uso rappresentano direttamente come quella problematica sia risolta da una o più tecnologie.

Figura 11a. SERP con accento sui casi d'uso

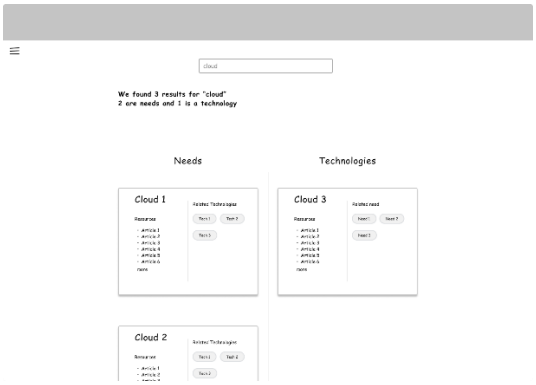

Figura 11b. SERP focalizzata sulla tassonomia (doppia colonna)

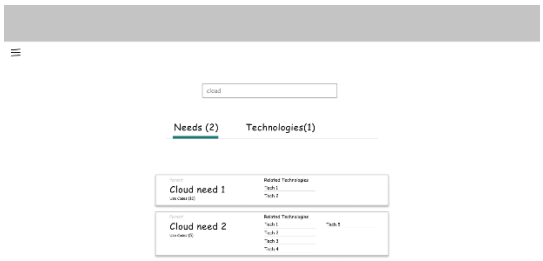

Figura 11c. "Doppia visualizzazione" in tab

Figura 11d. Layout di un singolo risultato

Questo design prevedeva quindi che dopo la ricerca fosse restituita una pagina che presentasse come titolo l'elemento della tassonomia più vicino alla parola cercata e come risultati i suoi casi d'uso. Ad ogni caso d'uso sarebbe stata associata una sua descrizione generale estrapolata dall'articolo e i suoi collegamenti con altri elementi presenti nella tassonomia (sotto forma di tag). Una tale elaborazione comporta però delle problematiche legate all'utilizzo stesso: la restituzione di una pagina riguardante l'elemento della tassonomia più vicino al termine della ricerca è fuorviante e può portare a risultati completamente errati. L'utente potrebbe cercare qualcosa che presenti diverse correlazioni con gli elementi della tassonomia di cui non per forza il primo risultato, cioè quello calcolato come più affine, è quello che lo interessa. Un professore universitario, per esempio, potrebbe cercare semplicemente la parola "database" per visualizzare quali sono gli ambiti aziendali in cui sono utilizzate tecnologie legate alla gestione di database. I risultati mostrati in questa maniera non

permetterebbero di visualizzare ciò in modo chiaro e, oltretutto, mostrerebbero soltanto un solo ambito tecnologico in cui i database sono trattati. Quindi, se il fine dell'interfaccia del motore di ricerca è quello di connettere bisogni e tecnologie, problemi e soluzioni, in questa maniera non potrebbe essere chiarito come una data problematica individuata sia risolta da alcune tecnologie, se non costringendo l'utente a cercarsi da sé i correlati in ogni caso d'uso.

In base a questo tipo di riconsiderazioni generate dallo studio dell'IA e delle *personas* è stata quindi valutata un'interfaccia che rispondesse prima di tutto ai seguenti due requisiti funzionali:

4. Il sito dovrebbe restituire i risultati della tassonomia più appropriati.
5. I risultati devono aiutare l'utente a collegare tecnologie e bisogni.

Da ciò ne è conseguito che la soluzione da adottare deve prevedere una schermata con doppia di visualizzazione dei risultati formata da una colonna per gli elementi strettamente correlati alla ricerca (se si cerca un bisogno il sistema dovrà comprendere che è un bisogno e non una tecnologia) ed un'altra per tutti gli altri elementi che con i primi hanno una relazione (ogni bisogno è collegato ad una o più tecnologie e viceversa). Nessuna delle *best practice* (v. 2.7.2 e 2.7.3) individuate indica un tale *layout* per la pagina dei risultati. Si è proceduto quindi nell'incorporare prima tutti gli elementi necessari già previsti: risultati per i bisogni e per le tecnologie, casi d'uso per ogni risultato e suoi correlati nella tassonomia.

Una prima questione posta da questo tipo di interfaccia ha riguardato la coerenza nella disposizione dei risultati. Ciò che è più pertinente alla ricerca dell'utente dovrà essere mostrato sempre nella colonna di sinistra? In questa modalità qualora l'utente ricerchi un bisogno d'industria allora i relativi bisogni correlati alla ricerca verranno mostrati nella colonna di sinistra mentre le tecnologie correlate a loro volta ai bisogni trovati saranno nella colonna di destra. Se l'utente cerca un qualcosa che si correla meglio con una tecnologia allora le colonne saranno invertite. Questo potrebbe portare ad un aumento del carico cognitivo per l'utente: durante l'attività di ricerca, cliccando su altri elementi o effettuando ulteriori ricerche, oltre ad essere concentrato sul proprio interesse (i risultati) si troverebbe a dover controllare ogni volta come sono posizionati

gli elementi. Il tutto si tradurrebbe in una diminuzione della coerenza interna del sito che, come evidenziano le euristiche di Nielsen (Nielsen 1994-2020) e in particolare la quarta ("Consistency and standards"), crea un problema di usabilità. Per affrontare questo problema di coerenza e di aumento di carico cognitivo si è optato quindi per un'interfaccia statica: i bisogni industriali sono visualizzati sempre nella colonna di sinistra e le tecnologie sempre nella colonna di destra. Per aiutare il processo di individuazione le colonne sono segnalate come appunto "Needs" e "Technologies".

Il *layout* elaborato per la SERP presenta delle somiglianze con il *layout* di tipo *comparing* (v. 2.7.2) mantenendo tuttavia una differenza sostanziale: le informazioni presenti nelle due colonne non sono destinate ad essere comparate tra loro, bensì rappresentano un ordinamento nei due diversi domini dei risultati ottenuti dalla ricerca.

Un altro possibile approccio sarebbe consistito nel ricorso a delle *tab* (fig. 11c) in cui i risultati sarebbero stati mostrati nelle apposite sezioni. Nonostante l'approccio sarebbe stato più vicino agli standard di *layout* utilizzati, avrebbe tradito il fine preposto della tassonomia, e quindi dell'interfaccia: in questa maniera sarebbe venuta meno l'idea di tenere insieme bisogni e tecnologie.

Definita la modalità di visualizzazione sono stati studiati gli elementi interni al singolo risultato e la gerarchia di visualizzazione interna ad ognuno di questi. In questo caso per ogni risultato sono stati inizialmente individuati quattro elementi da visualizzare:

- Titolo del risultato: il bisogno o la tecnologia.
- *Parent* del risultato: il concetto genitore del risultato nella tassonomia.
- Le risorse disponibili: i casi d'uso, gli articoli ecc.
- I concetti correlati nella tassonomia, bisogni per le tecnologie e viceversa.

L'ultimo elemento, residuo dei primi *wireframe* in cui il risultato era il caso d'uso, ha creato però una problematica legata alla ricorsività dell'interfaccia: se le tecnologie correlate ad un dato bisogno sono già nella colonna di destra, perché dovrebbero ritrovarsi anche nel singolo risultato? Per evitare di nuovo un inutile aumento di carico cognitivo sull'utente dovuto questa volta alla presenza di informazioni aggiuntive

irrilevanti (ottava euristica di Nielsen) oltretutto già presenti, questo elemento è stato eliminato nelle successive rielaborazioni.

La gerarchia di visualizzazione all'interno del singolo risultato è stata gestita in questa maniera (fig. 11d): il titolo, cioè il primo elemento da individuare, ha un peso maggiore; la posizione nella tassonomia indicata dal *parent* è posizionata al di sopra del titolo con un peso minore; le risorse disponibili sono indicate tramite un'apposita etichetta ("Resources"), sono disposti in una lista. Cliccando sul titolo del risultato oppure sull'intestazione del genitore sarà effettuata una nuova ricerca riguardante l'argomento selezionato, consentendo un *pattern* del tipo *pearl growing* (v. 2.7.2). Ognuno dei casi d'uso e articoli presenti tra le risorse sarà rappresentato dal suo titolo e tramite click su di esso permetterà di raggiungere il sito dov'è possibile consultarlo. Inoltre, è stato aggiunto un elemento che potesse fornire una spiegazione per ogni risultato: un breve estratto estrapolato da Wikipedia. Questo elemento è posizionato sotto il titolo e fornisce la possibilità di approfondimento diretto tramite link ("Read more").

Il *layout* finale della SERP (fig. 12) aggiunge infine gli elementi di navigazione:

- un'icona "hamburger" per per visualizzare la gerarchia sfogliabile che in questo caso sarà inserita in un *drawer*;
- un bottone per tornare alla homepage;
- la barra di ricerca che tolta dal corpo permette di avere più spazio per mostrare i risultati;
- e un bottone per raggiungere la pagina di aggiunta di un articolo.

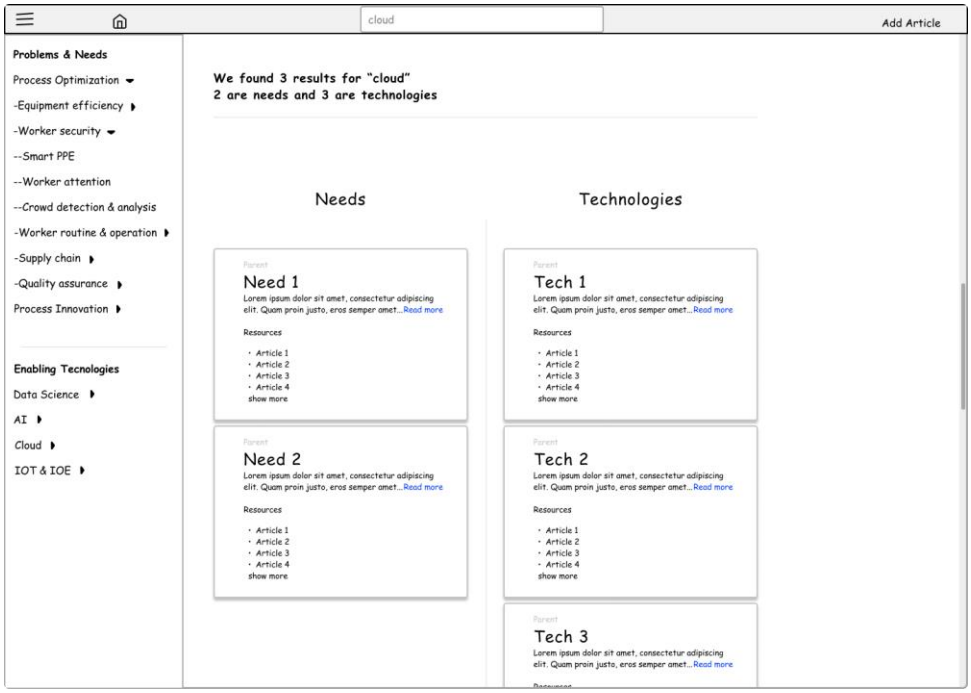

Figura 12. Wireframe finale della SERP con drawer della gerarchia sfogliabile aperto

4.4.4 La pagina di aggiunta di un articolo

All'utente viene fornita la possibilità di inserire un articolo e quindi di indicizzarlo all'interno della tassonomia.

Figura 13. Form di inserimento dell'articolo

Gli elementi di navigazione previsti nella *navbar* per questa pagina si riducono al solo bottone che consente di tornare alla homepage. L'inserimento dell'articolo (fig. 13) viene effettuato tramite un apposito *form* i cui campi sono: il titolo dell'articolo, il suo abstract e il suo corpo del testo. Quando l'utente invia l'articolo viene aperta una finestra di dialogo che consente all'utente di indicizzare l'articolo inviato (fig. 14). Un primo possibile *layout* (fig. 14a) elaborato consisteva in una *transfer list* che mettesse insieme *topic* corrispondenti trovati all'interno della tassonomia e possibili *topic* individuati nel testo dal sistema da poter aggiungere come tag all'articolo. Questo tipo di *layout* non consentiva però di poter fornire nuovi concetti possibilmente rilevanti al tassonomista in vista di ulteriori aggiornamenti della tassonomia. I *topic* individuati sarebbero rimasti solamente come etichette all'articolo, inibendo così le potenzialità di aggiornamento della tassonomia che un'identificazione da parte dell'utente delle corrispondenze con la preesistente tassonomia potrebbe suggerire. Per questo, è stato studiato un sistema di *progressive disclosure* (Nielsen 2006) in differenti passaggi. In questa maniera vengono passo-passo presentate all'utente le operazioni da eseguire per indicizzare l'articolo e al contempo gli viene data la possibilità di aiutare il sistema fornendo ulteriori dettagli sull'indicizzazione.

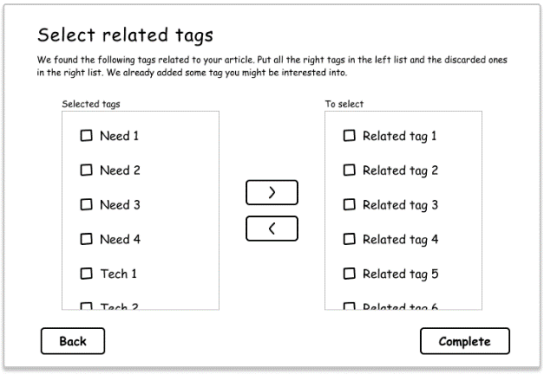

Figura 14a.

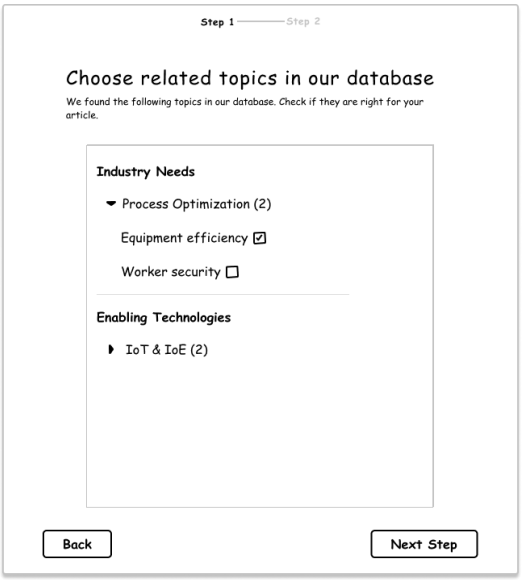

Figura 14b.

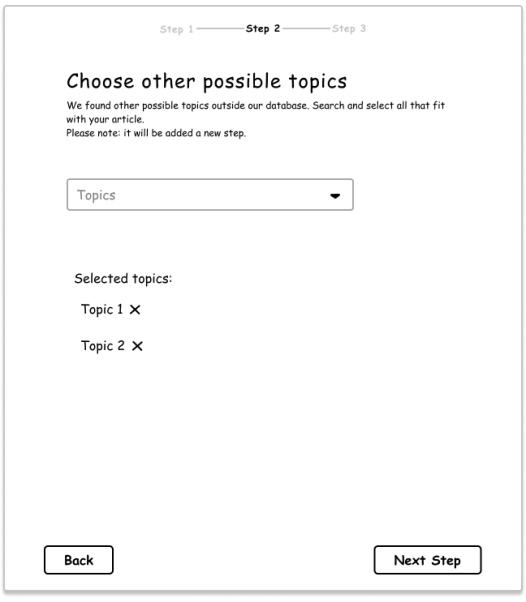

Figura 14c.

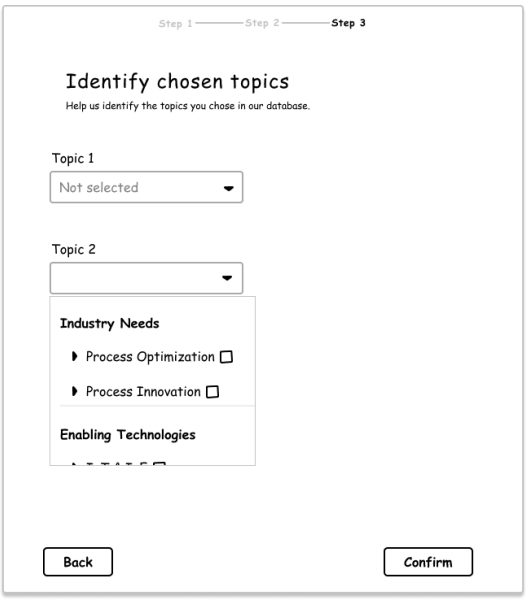

Figura 14d.

Le fasi previste per tale operazione sono due principali, una opzionale ed una aggiuntiva in caso venga effettuata quella opzionale:

1. L'utente inserisce l'articolo.

2. L’articolo sarà esaminato dal sistema che quindi chiederà all’utente come indicizzarlo nella tassonomia (attraverso la gerarchia sfogliabile dotata di *checkbox*), mostrando solo gli elementi che il sistema ha individuato come pertinenti all’articolo.
3. (Opzionale) Il sistema chiederà all’utente se ci sono altri possibili *topic* che possano individuare meglio l’articolo selezionandoli da una lista. Qualora l’utente ne selezioni uno o più, sarà aggiunto un nuovo *step* per identificare i *topic* selezionati.
4. Se l’utente ha effettuato delle scelte nella fase precedente gli sarà chiesto se e dove ogni termine selezionato dovrebbe posizionarsi nella tassonomia a suo giudizio. In quest’ultimo caso per ogni termine sarà mostrata tutta la gerarchia sfogliabile dotata di *checkbox*.

4.5 Architettura e implementazione del *front-end*

Le idee elaborate nei *wireframe* in fase di design sono state implementate lato *front-end* per mezzo dell’utilizzo del *framework* React e delle librerie Material UI e React Router. L’utilizzo di React ha permesso di concepire ogni elemento definito nei design in forma di componente riutilizzabile (fig. 15). Inoltre, parte del codice si occupa della comunicazione con il *back-end* per l’invio di richieste (per esempio, gli input di ricerca) e la ricezione di risposte (i risultati della ricerca). Il *back-end* realizzato da Riccado Amadio si serve delle API Tagme per poter identificare l’input dell’utente. Tagme (Ferragina e Scaiella 2010) permette di identificare “significative sequenze di termini che sono annotati con concetti tratti da un catalogo o un’ontologia”. In particolare, il catalogo utilizzato è fornito dall’intero database strutturato di Wikipedia. In questa maniera Tagme ricerca una “mention” (anche definita “spot”) e ciò che restituisce sarà confrontato con il database della tassonomia al fine di trovare corrispondenze. Infine, il *back-end* comunica al *front-end* i risultati.

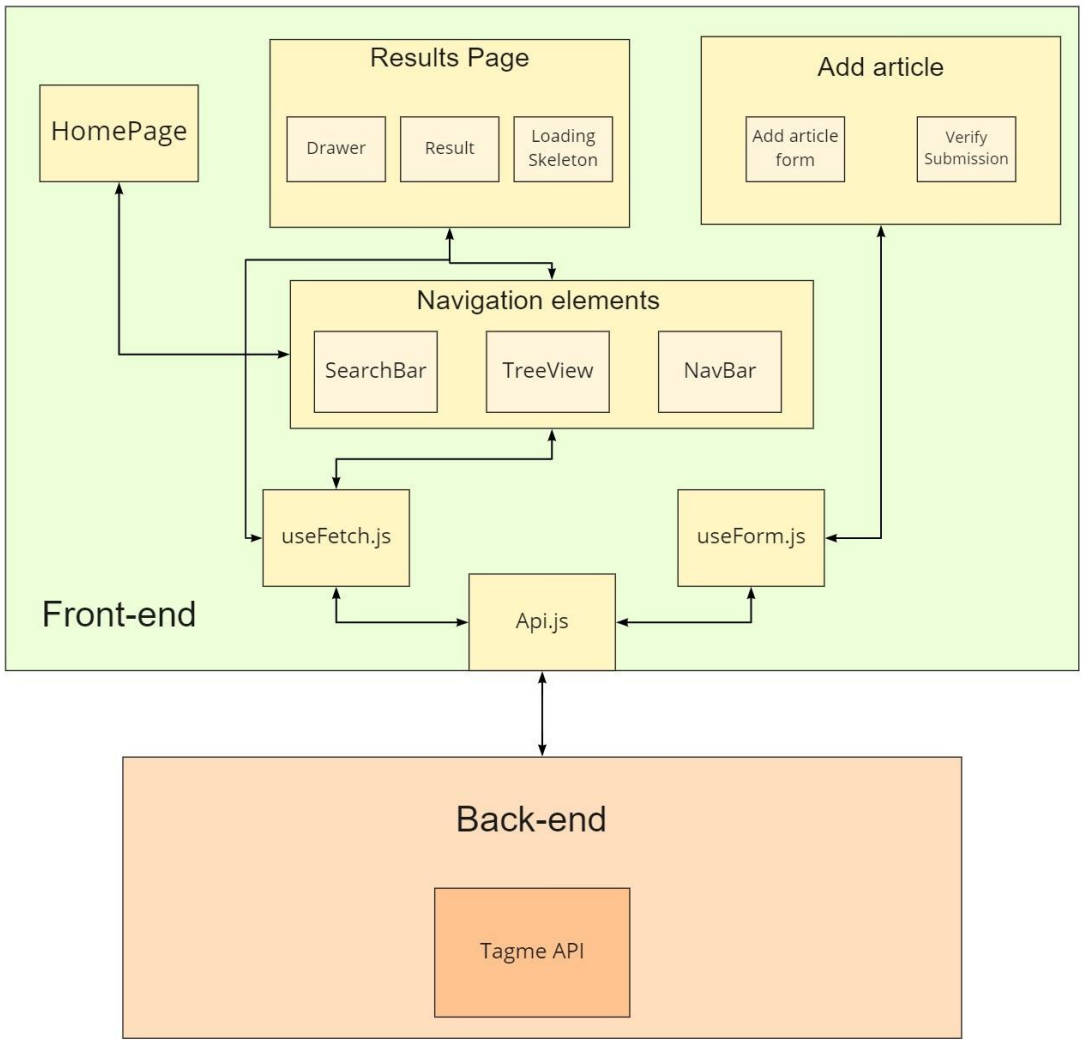

Figura 15. Architettura front-end

4.5.1 Comunicazione con il *back-end*

La comunicazione con il *back-end* è gestita da uno script denominato "API.js". Al suo interno è definita una funzione `apiSettings` con tre proprietà `fetchResults`, `fetchAutocomplete` e `submitArticle` che si occupano rispettivamente di:

- inviare una richiesta per effettuare una ricerca e riceverne i risultati;
- ricevere una lista di corrispondenze con l'input dell'utente per l'*autocomplete*;
- e inviare un articolo inserito dall'utente.

L'unica variabile presente è una costante denominata `API_URL` che definisce l'indirizzo del server[12]. In tutte le proprietà si fa uso delle parole chiave `await` e `async` che permettono la generazione di una "promessa" e quindi l'attesa di una sua risposta. Questo vuol dire che l'esecuzione della funzione è sospesa fino a che la promessa non sia mantenuta oppure respinta[13]. In questa maniera è possibile avere un comportamento asincrono per la modifica delle componenti del sito solo quando arrivi effettivamente una risposta e senza interrompere l'esecuzione. Inoltre, le chiamate al server saranno di tipo "`GET`" per l'invio di parametri tramite URL, e di tipo "`POST`" nel caso in cui saranno inviati dei dati strutturati come `form`.

```
    fetchResults: async (queryId) => {
        if (queryId === "" || queryId === undefined) return {};
        const formData = new FormData()
        formData.append('search-input', queryId)
        const results = await fetch(`${API_URL}`, {
          method: "POST",
          body: formData,
        });
        const data = await results.json()
        JSON.parse(JSON.stringify(data))
        return data;
      }
```

La prima proprietà è individuata in una funzione asincrona che come parametro riceve la *query* della ricerca. Innanzitutto, viene controllato se la *query* è stata inserita e in caso contrario si ritorna un oggetto vuoto. Questo è necessario perché, dato che il fetch dei risultati è richiamato in dipendenza al cambiamento di `query`, qualora l'utente cancelli ciò che ha scritto si effettuerebbe una chiamata al server vuota. La *query* è quindi resa disponibile alla lettura delle API nel *back-end* che necessitano una richiesta di un *form*. Viene quindi definito un *form* costituito dal campo `'search-input'` che come valore ha appunto la *query* della ricerca. Della richiesta si occupa il metodo

[12] Nel periodo di stesura della tesi il sito non è ancora pubblicato online quindi l'indirizzo del server è individuato in una porta locale.

[13] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function

globale `fetch()` in cui sono passati i parametri che indicano la posizione da raggiungere (`API_URL`) e un oggetto che contiene delle impostazioni da applicare alla richiesta. Nelle impostazioni sono specificati il metodo della richiesta `POST` e il corpo della richiesta (il `form` preparato). La risposta ricevuta in formato JSON è quindi elaborata per costruire un oggetto JavaScript da poter utilizzare successivamente.

La risposta che si riceve dal *back-end* è un oggetto JSON formato da tre parametri definiti come:

- `topics`: gli argomenti relativi alla *query* inviata trovati su Wikipedia da Tagme e che possono corrispondere a concetti della tassonomia;
- `related_elements`: gli elementi (i concetti) strettamente correlati nella tassonomia;
- `unrelated_elements`: gli elementi correlati ai primi elementi.

I risultati da mostrare nel *layout* a doppia colonna elaborato nei *wireframe* (v. 4.4.3) sono quindi quelli indicati all'interno dei parametri `related_elements` e `unrelated_elements`. Se, per esempio, viene ricercato qualcosa che è identificato come bisogno, saranno restituiti i bisogni che lo riguardano e tutte le tecnologie correlate.

Per ogni argomento trovato su Wikipedia è indicato il termine, il dominio di cui fa parte (bisogni o tecnologie) ed il suo link di Wikipedia che ne fornisce la spiegazione. Per ogni concetto correlato (sia `related_elements`, sia `unrelated_elements`) è indicato il termine preferito, il suo genitore nella gerarchia, il suo dominio e i casi d'uso che lo riguardano. A sua volta ogni caso d'uso sarà indicato da un titolo e un link.

Un'altra proprietà disponibile in `apiSettings` è `fetchAutocomplete` e consente di recuperare le parole dal server per supportare la funzione di *autocomplete* della barra di ricerca. Anche in questo caso la funzione è asincrona ma riceve l'input effettuato dall'utente nella barra di ricerca nel momento in cui lo scrive. I parametri che `fetch()` riceve riguardano una concatenazione di indirizzo del server e `query` e il metodo della richiesta ("`GET`"). Se la promessa è mantenuta allora saranno restituite le parole che il server ha individuato come corrispondenti all'input dell'utente.

4.5.2 Gestione degli stati nei *custom hook*

La gestione degli stati e la comunicazione tra API.js e le componenti si realizza al livello di due *custom hook*: `useFetch` e `useForm`. Questi due *hook* sono stati definiti al fine di evitare di utilizzare gli *hook* predefiniti di React all'interno di ogni classe consentendo inoltre la condivisione di stati e quindi la comunicazione tra componenti. All'interno dei *custom hook* sono infatti utilizzati gli *hook* di React `useState` e `useEffect`.

Il primo *custom hook* `useFetch` si occupa di ricevere gli input di ricerca, recuperare i risultati dal server e quindi gestirli. Per fare questo sono utilizzati quattro *hook* `useState` che rispettivamente sono:

- `query`: il valore della *query* da cercare (inizializzato come stringa vuota);
- `results`: l'oggetto che conterrà i risultati restituiti dal server (inizializzato con proprietà ma senza valori);
- `loading`: stato booleano che indica se l'operazione è in fase di caricamento;
- `error`: stato booleano che indica se l'operazione ha condotto ad errori;
- `found`: stato booleano che indica se sono stati trovati risultati o meno.
- `options`: oggetto che contiene tutte le opzioni possibili per l'*autocomplete* nella barra di ricerca.
- `queryMatch`: oggetto che contiene tutte le corrispondenze trovate dal server per l'input nella barra di ricerca.

La logica che utilizza gli stati e si collega allo script `API.js` è inserita nello `useEffect` *hook* che permette di eseguire la funzione di recupero dati in dipendenza del cambio di valore della variabile `queryId`. Quindi la chiamata al server è effettuata solo quando il termine da cercare cambia.

Per la gestione dello stato di errore la logica della funzione di recupero `fetch()` è espressa nei blocchi di `try e catch`, dove `setError` imposterà a `true` il valore di `error` nel caso, per esempio, ci sia un problema di connessione con il server.

Lo stato di caricamento ha valore `true` all'inizio dell'esecuzione del blocco e false al suo termine. Ciò consente di visualizzare l'animazione di caricamento nell'attesa che la promessa generata da `fetchResults()` in `API.js` riceva una risposta, cioè nell'attesa che il recupero sia completato.

Lo stato `found` consente di indicare ai componenti che si occupano della visualizzazione dell'interfaccia se sono stati ritrovati dei risultati oppure se non è stato trovato nulla. Infatti, al contrario dello stato di errore, qualora una risposta sia data ma non abbia portato a risultati l'oggetto JSON restituito sarà vuoto. Per evitare che i componenti di interfaccia restituiscano errori perché non trovano nessun oggetto da mostrare, viene utilizzata la renderizzazione condizionale concessa da React.

```
const fetchResults = await API.fetchResults(queryId);
if (fetchResults.related_elements == null) {
  setFound(false)
} else {
  setResults(() => ({
    related_elements: [...fetchResults.related_elements],
    topics: [...fetchResults.topics],
    unrelated_elements: [...fetchResults.unrelated_elements]
  }));
  setFound(true)
}
```

Perciò, data una variabile `fetchResults` che contiene i risultati restituiti dalle API, se l'array di elementi correlati e quello dei *topics* sono vuoti `found` sarà falso. Se invece sono presenti elementi allora questi saranno assegnati alle proprietà dello stato `results` (ogni proprietà è assegnata con lo *spread operator* "`...`") e `found` sarà vero.

In riguardo al funzionamento dell'*autocomplete* nella barra di ricerca, `useFetch` attende un cambiamento nello stato della *query* per poter andare a ricercare le possibili corrispondenze nel server:

```
const fetchAutocomplete = await API.fetchAutocomplete(query);
if (fetchAutocomplete == null) {
  console.log("not found")
  setQueryMatch(initialQueries)
} else if (fetchAutocomplete.word != null) {
  setQueryMatch(() => ({
    ...fetchAutocomplete
  }))
}
```

Se l'operazione conduce a risultati, tutte le parole trovate sostituiscono i valori già presenti in `queryMatch` (nessuno di *default*). In dipendenza al cambiamento di questo stato saranno assegnate le parole ad `options` che sarà a sua volta esportato per l'utilizzo nella barra di ricerca.

4.5.3 Gestione routing

Il sito agisce come una *single page application*, cioè come un'unica pagina dinamica, ma al suo interno è formata da diverse pagine (in realtà componenti React) a cui è possibile indirizzarsi. Questo particolare funzionamento è reso grazie a React Router e consente, per esempio, di raggiungere una pagina specifica del sito per mezzo del suo indirizzo. I principali componenti di React Router utilizzati sono importati nello script principale di gestione del sito, `APP.js`, e ne formano la struttura.

Gli instradamenti possibili sono quindi tre e riguardano la Homepage, la pagina dei risultati e la pagina per l'aggiunta di articoli. Ogni *route* ha quindi un indirizzo (`path`) e un elemento (`element`) che indica quale componente React sarà richiamato per quell'indirizzo specifico. La `path` per la pagina dei risultati richiama una funzione particolare di React Router che richiama lo stato della *query* (v. 4.5.2 e 4.5.5.1).

4.5.4 Gestione grafica

La gestione degli elementi dell'interfaccia utente del sito è stata in gran parte realizzata grazie alla libreria Material UI (MUI). Molti degli elementi definiti nella fase di

wireframe sono stati implementati quindi facendo ricorso ai componenti presenti in MUI e riadattandoli ai diversi scopi.

La base dello stile grafico di tutto il sito è data dal componente MUI `CssBaseline` che, inserito nello script `APP.js`, fornisce una base coerente con gli standard del Material Design. MUI inoltre consente di definire temi personalizzati. Questa possibilità è stata utilizzata solamente per dichiarare i colori base utilizzati in tutto il sito mentre lo stile basilare di ogni altro componente utilizzato è stato invece definito all'interno degli stessi componenti.

4.5.5 Elementi di navigazione nell'interfaccia utente

Nella gestione del *routing* tra le pagine (v. 4.5.3) sono stati definiti tre elementi principali: rispettivamente, la Homepage, una pagina dei risultati (SERP) e una pagina per l'aggiunta degli articoli. Alcuni componenti sono riutilizzati in alcune delle pagine e quindi sono stati definiti una sola volta. In particolare, tre metodi di navigazione possibile compaiono sia nella Homepage (costituita per l'appunto solo da questi elementi) sia nella pagina dei risultati.

4.5.5.1 La barra di ricerca

Figura 16. Barra di ricerca

La barra di ricerca (fig. 16) è il primo e più importante elemento di navigazione disponibile nel sito. Per rispettare i requisiti funzionale e più specificamente quello indicato dal requisito n. 9 (v. 4.3) è stata implementata una funzione di *autocomplete*. Il campo di testo `SearchBox` è stato ridefinito come componente con stile a partire da `Autocomplete` definito in MUI. In questa maniera è stato possibile utilizzare le funzioni garantite dal componente MUI e uno stile personalizzato alla barra di ricerca. I

parametri ricevuti dall'hook `useFetch` sono `query`, `setQuery` e `options`, e rispettivamente consentono di visualizzare lo stato della *query*, modificarlo, e ricevere le opzioni disponibili per la lista di suggerimenti. Dunque, qualora l'utente inserisca un input che trova riscontro nella tassonomia, l'*autocomplete* fornisce dei suggerimenti nella lista (fig. 17). Inoltre, lo stato di *focus* sulla barra di ricerca viene segnalato da un'ombreggiatura che aumenta l'elevazione della barra ponendola così in risalto.

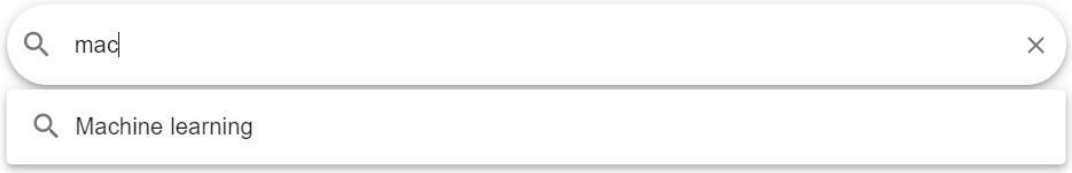

Figura 17. Suggerimento autocomplete

Le proprietà di `SearchBox` sono quindi:

- `freeSolo` consente di inserire input nel campo di testo anche se non sono presenti nella lista.
- `options`, in cui sono contenute tutte le opzioni rese disponibili dall'apposito stato in useFetch.
- `inputValue` definisce il valore dell'input nel campo di testo e permette di visualizzare la *query* dopo aver effettuato una ricerca.
- `getOptionLabel`, ovvero tutte le etichette da mostrare per ogni opzione.
- `onInputChange` cambia il valore dello stato della *query* ad ogni input dell'utente nel campo di testo.
- `onChange` invia il valore selezionato dalla lista delle opzioni per effettuare una ricerca.
- `renderOption` regola lo stile della lista di opzioni. In questo caso viene aggiunta un'icona di una lente di ingrandimento prima di ogni opzione suggerita.
- `PaperComponent` regola lo stile del contenitore della lista di opzioni. In questo caso vengono fornite `elevation` (cioè più ombreggiatura) e un `marginTop` che distanzi di poco la lista dalla barra di ricerca.

- `renderInput` permette di modificare proprietà e stile del campo di testo. In particolare, sono definite la grandezza (che cambia in dipendenza di un parametro che riceve il componente) e un *placeholder* da mostrare all'utente.

Lo stato `query` è inizializzato come stringa vuota ma, nel caso in cui la pagina dei risultati sia raggiunta tramite link diretto e non tramite ricerca, lo stato sarà definito in dipendenza di questo.

```
const { queryId } = useParams();
useEffect(() => {
  setQuery(queryId)
}, [queryId])
```

L'*hook* `useParams`() è definito nella libreria React Router e permette l'accesso dinamico al parametro dell'URL definito come `queryId` (v. 4.5.3).

Quando l'utente cambia un valore questo cambiamento attiva l'onChange che a sua volta richiama la funzione handleSubmit che consente lo spostamento alla pagina dei risultati:

```
const handleSubmit = (e, value) => {
  if (value === "" || value === undefined) return;
  navigate("/" + value);
};
```

4.5.5.2 La gerarchia sfogliabile

Figura 18. TreeView della gerarchia

La gerarchia sfogliabile è l'elemento di navigazione che consente di esplorare le gerarchie dei due domini della tassonomia. È stata implementata a partire dal componente `TreeView` di MUI ma, trattandosi di due gerarchie da visualizzare in maniera *responsive*, ogni albero è stato inserito in una colonna (fig. 18). È stato così possibile implentare una sola volta il codice sia nel caso di utilizzo nella Home sia all'interno del *drawer* nella pagina dei risultati (fig. 19).

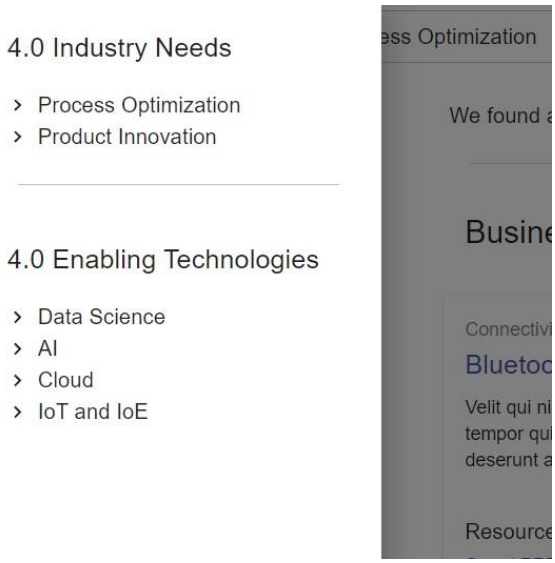

Figura 19. Gerarchia all'interno del drawer

Ogni elemento di ogni gerarchia viene generato da una funzione ricorsiva che esplora un file JSON strutturato in livelli che contiene l'intera tassonomia:

```
const renderItem = (nodes) => (
    <CustomTreeItem
        key={nodes.label}
        nodeId={nodes.label}
        label={nodes.label} >
        {Array.isArray(nodes.subLevels)
            ? nodes.subLevels.map((node) => renderItem(node))
            : null}
    </CustomTreeItem>
)
```

4.5.5.3 La navbar

Un ulteriore elemento di navigazione presente in tutte le pagine è la *navbar* del sito denominata in questo contesto `TopNavBar` per la sua posizione. Le direzioni rese possibili in questo elemento sono due: tornare alla Homepage e raggiungere la pagina per l'aggiunta degli articoli. I componenti utilizzati nella *navbar* (da sinistra a destra) sono:

- Un bottone con icona *hamburger* per aprire il *drawer* con la gerarchia sfogliabile.
- Un bottone per tornare alla Home.
- La barra di ricerca in una versione di dimensioni minori.
- Un bottone per raggiungere la pagina per l'aggiunta degli articoli.

Figura 20. Navbar della pagina dei risultati

Figura 21. Navbar durante lo scorrimento

Il componente MUI `AppBar` (le cui proprietà definiscono l'aspetto e il comportamento generale della *navbar*) è utilizzato principalmente come contenitore e per la sua proprietà di posizione (statica), perciò a livello grafico la *navbar* è resa trasparente e senza bordi o ombreggiature (fig. 20). Tuttavia, il comportamento denominato *sticky* consente di "attaccare" la *navbar* al bordo superiore della finestra in caso di scorrimento della pagina. In questo caso comparirà il bordo inferiore della *navbar* ed un'ombreggiatura per renderla elevata rispetto al contenuto della pagina (fig. 21).

Nel comportamento *responsive* per dispositivi con larghezza dello schermo inferiore a 600 pixel, il bottone per raggiungere la pagina per l'aggiunta di articoli scompare per fornire maggiore spazio alla barra di ricerca.

4.5.6 La pagina dei risultati (SERP)

La pagina dei risultati elaborata in fase di *wireframing* è stata implementata utilizzando i seguenti componenti:

- La barra di ricerca (`SearchBar`).
- La gerarchia sfogliabile (`BrowsableTree`).
- La *navbar* (`TopNavBar`).
- La lista dei risultati (ResultsList).
- Il singolo risultato (`Result`).
- La pagina per gestire la situazione in cui si hanno zero risultati (`NotFound`)

Il funzionamento generale della SERP prevede che questa sia inizializzata richiamando gli stati di `useFetch()` in base alla `queryId` ricevuta come parametro da React Router. Quindi, la renderizzazione sarà di tipo condizionale: si mostrano i risultati solo se sono stati trovati dalle API, se sono stati caricati e se non ci sono stati errori catturati in fase di recupero.

Se è stato trovato qualcosa, allora verrà renderizzato il componente che gestisce la list dei risultati passandogli come *props* le liste dei bisogni d'industria (`problems`) e delle tecnologie (`technologies`) già filtrate. Infatti, l'oggetto generato a partire dal JSON

restituito dal server è separato all'interno di un *hook* `useEffect()` in due liste diverse a seconda che si tratti di bisogni o tecnologie.

```
    useEffect(() => {
      if (results.related_elements.length !== 0 ||
results.unrelated_elements.length !== 0) {
        if (results.related_elements[0].at(3) === "Problems") {
          setProblems(results.related_elements)
          setTechnologies(results.unrelated_elements)
        }
        if (results.related_elements[0].at(3) === "Technology") {
          setProblems(results.unrelated_elements)
          setTechnologies(results.related_elements)
        }
      }
    }, [results])
```

Nel JSON sono restituite due liste di risultati: una contenente gli elementi correlati alla *query*, l'altra contente gli elementi correlati a quelli della prima lista (v. 4.5.1). Per far sì che il sistema divida e poi riconosca le liste come una contenente solo bisogni e un'altra solo tecnologie è stato perciò necessario individuare a cosa corrispondesse la lista dei `related_elements`. Quindi, se la lista contiene elementi ed il primo elemento ha come proprietà "`Problems`" o "`Technologies`" (quarto indice del singolo elemento), la lista in questione sarà assegnata allo stato corrispondente. Non è necessario usare metodi per filtrare le liste come `filter()` perché i risultati provengono sempre e solo da un vocabolario controllato (la tassonomia).

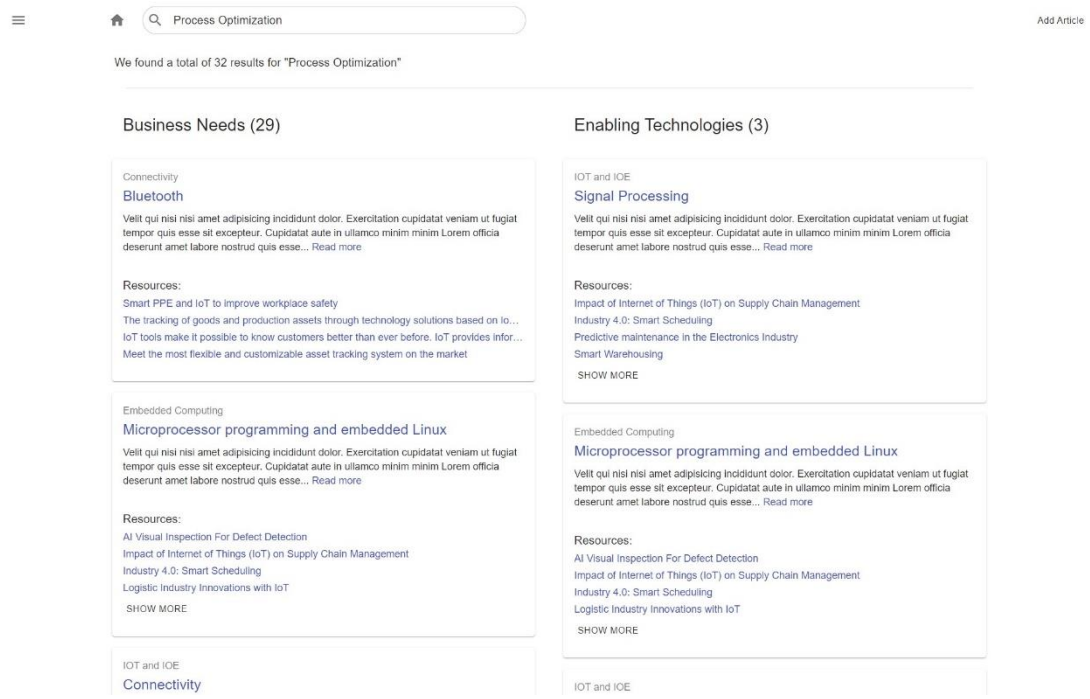

Figura 22. Pagina dei risultati

4.5.6.1 La lista dei risultati

I risultati della ricerca, contenuti negli stati `problems` e `technologies` e passati come *props* a `ResultsList`, sono quindi pronti ad essere disposti in colonne e visualizzati. La disposizione in colonne è realizzata tramite il *layout* a griglia di Material UI in cui lo spazio è diviso in 12 colonne. Il comportamento *responsive* per ognuna delle due liste è regolato nella seguente maniera:

```
    <Grid item xs={12} sm={6}>
```

dove sugli schermi più piccoli (larghezza inferiore a 600 pixel) la lista occupa 12 colonne e per tutti gli altri schermi sei colonne per lista. Nel primo caso, quindi, saranno visualizzate una sopra l'altra, nel secondo una accanto all'altra (fig. 22).

La visualizzazione di ogni risultato è ottenuta iterando le rispettive liste e assegnando ogni singola proprietà al componente che se ne occupa (`Result`). Nel caso della lista dei bisogni si avrà:

```
{problems.map((el, i) => {
    return (
      <Result
        key={el[0] + i}
        name={el[0]}
        parent={el[1]}
        category={el[3]}
        articles={el[2]}
      />
    );
  })
}
```

A seconda della posizione di ogni proprietà da riportare (`name`, cioè il termine preferito; `parent`, il concetto padre; `category`, il dominio; `articles`, la lista di casi d'uso) sono riportati gli indici relativi. La proprietà `key` serve invece a React per determinare quale elemento viene modificato nel DOM (in questo caso il termine preferito in quanto unico per ogni risultato).

Qualora non siano restituiti risultati non saranno generate le rispettive liste e sarà renderizzato il componente `NotFound` che consente di mostrare in maniera condizionale due possibili messaggi a seconda del caso in cui non siano stati trovati risultati oppure sia stato catturato un errore.

4.5.6.2 Il singolo risultato

La rappresentazione del singolo risultato elaborata nei *wireframe* è stata riportata in fase di implementazione tenendo conto prima di tutto dei seguenti elementi:

- Termine preferito del concetto.
- Genitore del concetto.
- Spiegazione del concetto tratta da Wikipedia[14].

[14] Per quanto riguarda il testo estratto da Wikipedia al suo posto è stato inserito un "Lorem Ipsum" poiché al momento della stesura di questa tesi non è stata ancora implementata la funzionalità lato server.

- Lista di casi d'uso correlati.

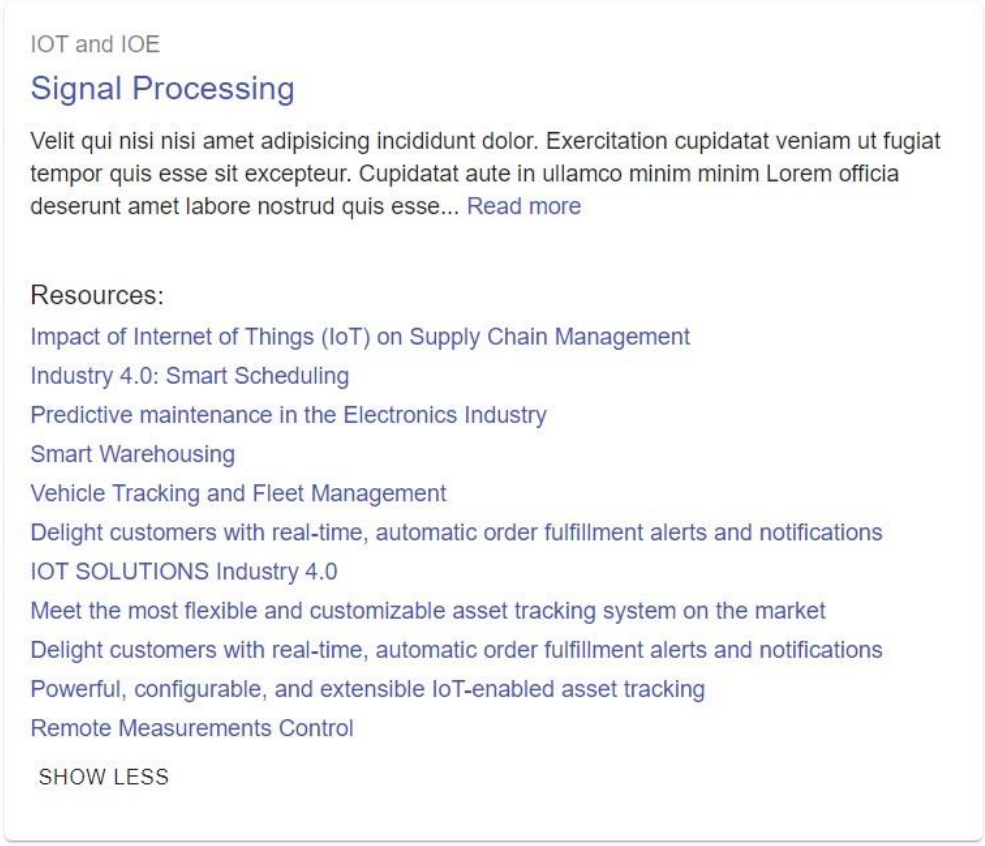

Figura 23. Singolo risultato con lista delle risorse aperta

I primi due sono visualizzati come testo e sono interagibili come link che conduce ad una nuova *query* specifica per quel concetto:

```
<Link
  component={RouterLink}
  to={"/" + name}>
  <Typography variant="h6" gutterBottom >
    {name}
  </Typography>
</Link>
```

In questo esempio sarà visualizzato il nome del concetto, cioè il termine preferito.

I casi d'uso (indicati come "Resources", risorse) sono contenuti in un array e possono essere presenti o meno, perciò la renderizzazione è condizionale. La gestione delle risorse è separata in uno specifico componente `Resources` (definito però sempre

all'interno di `Result`) che, come *props*, riceve `articles` e utilizza tre stati (`isLong`, `showMore`, `buttonText`) che si occupano rispettivamente di:

- Controllare se la lunghezza dell'array dei casi d'uso è superiore a quattro elementi.
- Indicare lo stato del bottone per la renderizzazione condizionale di ulteriori casi d'uso.
- Indicare il testo del bottone che permette di mostrare o meno altri casi d'uso.

4.5.6.3 Visualizzazione stato di caricamento

Nel momento in cui l'utente effettua una *query* avviene un ritardo tra l'apertura della pagina dei risultati e la comparsa dei risultati stessi. Questo perché la funzione di recupero dei risultati (`fetch`) viene eseguita in modalità asincrona permettendo di non interrompere il *flow* di utilizzo. Per far sì che nell'attesa l'utente sia avvisato visivamente dello stato del sistema è stato implementato uno *skeleton*, ovvero una versione ridotta ai minimi termini (uno scheletro appunto) del *layout* che la pagina dei risultati assumerà a caricamento avvenuto.

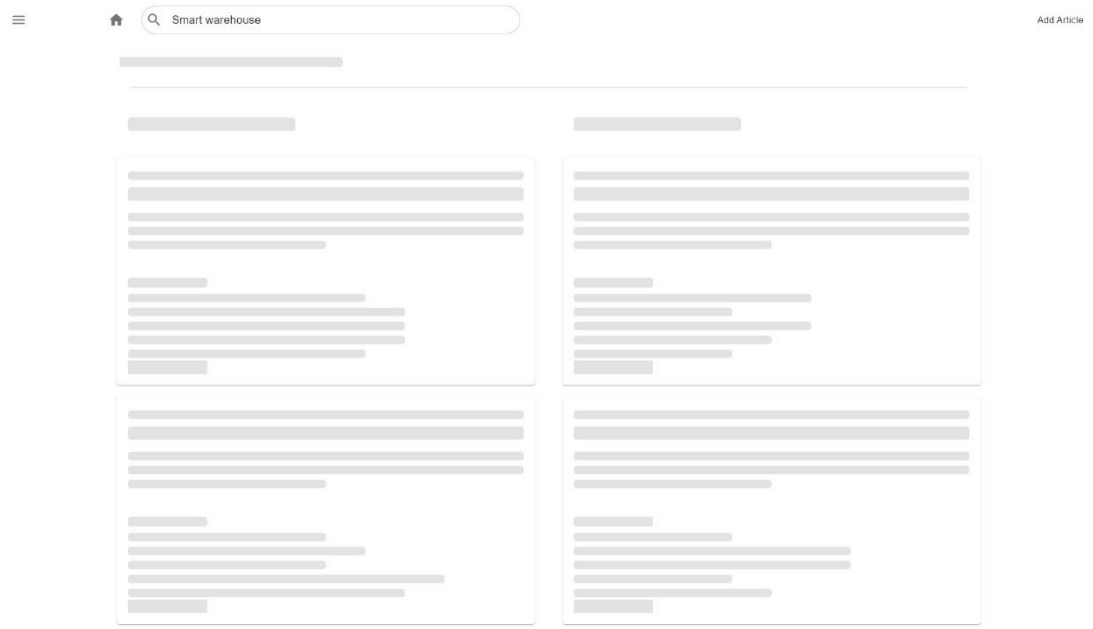

Figura 24. Skeleton per il caricamento della SERP

Per ricreare la struttura dello *skeleton* (fig. 24) è stata simulata una pagina sostituendo i suoi elementi di testo e bottoni con dei segnaposto il cui colore grigio è animato per segnalare all'utente che il sistema non è bloccato. Al termine del caricamento, quindi, lo *skeleton* sarà sostituito dai risultati caricati (se ve ne sono).

Prospettive future

Al momento di stesura di questa tesi l'implementazione della pagina per l'aggiunta degli articoli non è stata terminata. Per questo motivo si è scelto di presentarne la fase di progettazione omettendo però i dettagli implementativi. Questo ed altri elementi rientrano perciò nelle prospettive di lavoro futuro, ovvero:

1. Implementazione della pagina per l'aggiunta degli articoli.
2. Rifinimento dello stile e miglioramento generale della UI e della sua usabilità.
3. Elaborazione di un nome del prodotto più efficace.
4. Valutazione dell'interfaccia tramite test sugli utenti.

In riferimento al primo punto, le fasi di progettazione indicate nei *wireframe* della pagina per l'aggiunta degli articoli (v. 4.4.4) dovranno essere implementate al fine di consentire ai gestori della tassonomia di poterla aggiornare sia in conseguenza di possibili rifinimenti rispetto ai riscontri forniti dagli utenti in fase di verifica dell'invio, sia in vista della comparsa di bisogni aziendali o tecnologie abilitanti nuove.

Il sito è stato progettato cercando di concentrarsi il più possibile sull'Information Architecture e quindi sulle modalità di rappresentazione e interazione con la tassonomia per l'Industria 4.0. L'usabilità e l'aspetto estetico della UI necessitano però di un'ulteriore attenzione affinché l'interfaccia produca un migliore riscontro con l'utente (punto 2). Inoltre, dovrà anche essere elaborato un nome che catturi al meglio le funzionalità del prodotto e lo renda più identificabile (punto 3).

In ultima istanza, ma subordinato al punto 2 riguardo il miglioramento dell'usabilità, è necessario provvedere all'esecuzione di test sugli utenti (punto 4). I test sugli utenti *target*, ovvero quelli appartenenti alle categorie indicate dalle *user personas*, saranno necessari per poter identificare ed effettuare, se necessario, cambiamenti nella progettazione del sito.

Conclusioni

Nel lavoro trattato in questa tesi si è cercato di rispondere alle domande “come si rappresenta una tassonomia?” e “come si può interagire con una tassonomia?” in conseguenza dell’elaborazione di una tassonomia per l’Industria 4.0 che permetta di coadiuvare il processo di trasformazione digitale.

In riguardo alla prima domanda e grazie all’utilizzo di approcci teorici e pratici provenienti dall’Information Architecture, si è cercato di comprendere come è strutturata una tassonomia e quali siano le migliori pratiche di rappresentazione disponibili. Per rispondere alla seconda domanda, e quindi al fine di rendere possibile l’esplorazione e in generale l’utilizzo della tassonomia agli utenti, sono state esaminati i pattern di ricerca eseguiti dagli utenti e le pratiche di progettazione per siti che ospitano motori di ricerca.

D’altra parte, grazie alle tecniche derivate dallo User Experience Design, sono stati analizzati gli utenti finali di tale motore di ricerca per la tassonomia. In questo modo è stato possibile evidenziare quattro tipologie principali di utenti e di elaborare le loro rispettive *user personas* affinché si producesse una comprensione più approfondita e il più possibile vicina alla realtà dei loro bisogni e obiettivi in riguardo al contesto di utilizzo della tassonomia per l’Industria 4.0.

Le informazioni prodotte dall’esame dell’Information Architecture per la tassonomia e il lavoro di analisi degli utenti insieme hanno consentito di studiare un *layout* delle pagine del sito attraverso l’elaborazione di *wireframe*. Ciò ha permesso di definire gli elementi basilari per la navigazione della tassonomia e della sua interfaccia in rispetto alle esigenze degli utenti previste nelle *user personas* e alle *best practice* di design analizzate.

Infine, l’ultima fase ha comportato l’implementazione degli elementi e del *layout* evidenziati nei *wireframe* conducendo quindi all’elaborazione di un primo prototipo

del sito che si interfacci con il server che gestisce il motore di ricerca per la tassonomia. Questa prima versione del sito rappresenta un primo passo che pone le basi per futuri sviluppi successivi, tanto a livello progettuale e valutativo (attraverso test con gli utenti), quanto in quello implementativo.

Bibliografia

Amadio, Riccardo, Anastasiya Isgandarova, e Daniele Mazzei. «Building a Taxonomony of Industry 4.0 Needs and Enabling Technologies.» *EasyChair Preprint no. 5621*, 2021.

ANSI/NISO Z39.19-2005. «Guidelines for the Construction, Format, and Management of Monolingual Controlled Vocabularies.» (National Information Standards Organization) 2010.

Chaparro, Barbara S., A. D. Shaikh, e Kelsi Lenz. «Where's the Search? Re-examining User Expectations of Web Objects.» 2006.

Cooper, Alan. *The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity.* Indianapolis: Sams, 1999.

Covert, Abby. *How to Make Sense of Any Mess: Information Architecture for Everybody.* CreateSpace Independent Publishing Platform, 2014.

Ferragina, Paolo, e Ugo Scaiella. «TAGME: on-the-fly annotation of short text fragments (by wikipedia entities).» *Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM '10).* New York: Association for Computing Machinery, 2010. 1625–1628.

Floridi, Luciano. *Infosfera, Etica e filosofia nell'era dell'informazione.* Torino: Giappichelli Editore, 2009.

Hearst, Marti. *Search User Interfaces.* Cambridge University Press, 2009.

Hedden, Heather. *The Accidental Taxonomist.* Information Today, 2016.

Mazzei, Daniele. «Filling the Gap Between Scientific Research on Artificial Intelligence and Industry 4.0 Needs.» *Proceedings of the 1st Workshop on*

Flexible Resource and Application Management on the Edge (FRAME '21). New York: Association for Computing Machinery, 2020. 1-2.

Morville, Peter, e Jeffery Callender. *Search Patterns. Design for Discovery.* O'Really Media, 2010.

Myers, Isabel B., e Peter B. Myers. *Gifts Differing: Understanding Personality Type.* Mountain View: Davies-Black Publishing, 1980.

Norman, Donald A. *The Design of Everyday Things (2nd Edition).* New York: Basic Books, 2013.

Nielsen, Jakob, e Hoa Loranger. «Prioritizing Web Usability.» 2006.

Raza, Arif, Zaka-ul-Mustafa, e Luiz Fernando Capretz. «Personality Dimensions and Temperaments of Engineering Professors and Students – A Survey.» *Journal of Computing, 3(12),* 2011: 13-20.

Rosenfeld, Louis, Peter Morville, e Jorge Arango. *Information Architecture: For the Web and Beyond (4th Edition).* Sebastopol: O'Reilly Media, 2015.

Russell-Rose, Tony, e Tyler Tate. *Designing the Search Experience. The Information Architecture of Discovery.* Waltham: Morgan Kaufmann-Elsevier, 2013.

Travis, David, e Philip Hodgson. *Think Like a UX Researcher.* Boca Raton: Taylor & Francis Group, 2019.

Sitografia

Cardello, Jen. *The Difference Between Information Architecture (IA) and Navigation.* 2014. https://www.nngroup.com/articles/ia-vs-navigation/.

European Commission. *Industry 4.0, Digitalisation for productivity and growth.* 2015. https://www.europarl.europa.eu/RegData/etudes/BRIE/2015/568337/EPRS_BRI(2015)568337_EN.pdf.

Facebook. *React.* https://it.reactjs.org/ (consultato il giorno Ottobre 28, 2021).

Figma. https://www.figma.com/ (consultato il giorno Ottobre 28, 2021).

Google. *Angular.* https://angular.io/ (consultato il giorno Ottobre 28, 2021).

Google. *Material Design Guidelines.* https://material.io/design/guidelines-overview (consultato il giorno Ottobre 28, 2021).

ISO 9241-210. *Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems.* 2010. https://www.iso.org/obp/ui/#iso:std:iso:9241:-210:ed-1:v1:en.

Kagermann H., Lukas W., e Wahlstler W. *Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution.* 2011. https://www.ingenieur.de/technik/fachbereiche/produktion/industrie-40-mit-internet-dinge-weg-4-industriellen-revolution/.

Laubheimer, Page. *3 Persona Types: Lightweight, Qualitative, and Statistical.* 2020. https://www.nngroup.com/articles/persona-types/.

MUI. *Material UI.* https://mui.com/ (consultato il giorno Ottobre 28, 2021).

Nielsen, Jakob. *10 Usability Heuristics for User Interface Design.* 1994-2020. https://www.nngroup.com/articles/ten-usability-heuristics/.

Nielsen, Jakob. *Progressive Disclosure.* 2006. https://www.nngroup.com/articles/progressive-disclosure/.

Pernice, Kara. *Affinity Diagramming for Collaboratively Sorting UX Findings and Design Ideas.* 2018. https://www.nngroup.com/articles/affinity-diagram/.

Pernice, Kara. *UX Prototypes: Low Fidelity vs. High Fidelity.* 2016. https://www.nngroup.com/articles/ux-prototype-hi-lo-fidelity/.

Planet4. *Practical Learning of Artificial Intelligence on the Edge for indusTry 4.0.* https://www.planet4project.eu/ (consultato il giorno Ottobre 28, 2021).

React Router. https://reactrouter.com/ (consultato il giorno Ottobre 28, 2021).

Rosala, Maria. *How to Analyze Qualitative Data from UX Research: Thematic Analysis.* 2019. https://www.nngroup.com/articles/thematic-analysis/.

Soranzo, Alberta. *Taming Taxonomy.* 2013. https://www.slideshare.net/atrebla/taming-taxonomy-25969547.

Usability.gov. *Prototyping.* https://www.usability.gov/how-to-and-tools/methods/prototyping.html (consultato il giorno 12 Settembre, 2021).

Usability.gov. *Wireframing.* https://www.usability.gov/how-to-and-tools/methods/wireframing.html (consultato il giorno 12 Settembre, 2021).

World Wide Web Consortium. *SKOS Simple Knowledge Organization System Reference.* 2009. https://www.w3.org/TR/2009/REC-skos-reference-20090818/.

You, Evan. *Vue.js.* s.d. https://vuejs.org/ (consultato il giorno 28 Ottobre, 2021).

Appendice – Tassonomia per l'Industria 4.0

4.0 Industry Needs

1. Process Optimization
    - 1.1. Equipment efficiency improvement
        - 1.1.1. Real-time Production monitoring analysis and supervision.
        - 1.1.2. Predictive maintenance.
        - 1.1.3. Vertical interconnection and integration (between departments in a factory)
    - 1.2. Worker security improvement and accident prevention
        - 1.2.1. Smart PPE (personal protection equipment).
        - 1.2.2. Worker attention and mental state monitoring
        - 1.2.3. People counting, analysis and crowd detection
    - 1.3. Worker routine and operation optimization
        - 1.3.1. Time and Method smart measurement
    - 1.4. Supply Chain
        - 1.4.1. Horizontal interconnection and integration (between different actors of the supply chain)
        - 1.4.2. Production on demand enabling technology
        - 1.4.3. Smart warehouse.
        - 1.4.4. Intralogistics 4.0 - material flow control
        - 1.4.5. Supply chain transparency and reliability improvement
    - 1.5. Quality Assurance
        - 1.5.1. Technological Processes intelligent supervision
        - 1.5.2. Intelligent FMEA
2. Product Innovation
    - 2.1. Product servitization
    - 2.2. Usability improvement
    - 2.3. Smart products - Intelligent self-diagnosing products
    - 2.4. Cost and number of parts/component reduction
    - 2.5. After-Sales support
        - 2.5.1. Automatic consumables reorder
        - 2.5.2. Inventory Management

4.0 Enabling Technologies

1. Data science
    a. Data Visualization and Dashboarding
        i. Grafana
        ii. Kibana
        iii. Metabase
    b. Data Analytics
        i. Data lake and Data Warehouse design
        ii. Data Mining
            1. Python (Numpy,Pandas,sklearn,Scipy)
            2. R
            3. Julia
        iii. Process Mining
    c. Advanced reporting and self-service business intelligence tools
        i. Tableau
        ii. PowerBI
        iii. Elastic Stack
        iv. SAP Business Intelligence
        v. QlikView/Qlik Sense
    d. DataBases
        i. SQL DB
        ii. Non SQL DB
        iii. Time series DB
            1. influxDB
            2. Prometheus
            3. Graphite
        iv. Data engines
            1. Apache Hadoop
            2. Apache Kafka
            3. Apache Spark
2. AI
    a. Machine learning
    b. Deep Learning
    c. Reinforcement Learning
    d. Continuous Learning
    e. Computer Vision
    f. Natural Language Processing
3. Cloud
    a. Container technology
        i. Docker
        ii. Kubernates

iii. Terraform

b. Serverless programming
   i. AWS Lambda functions
   ii. Azure functions
c. Device Management
   i. Zerynth Device Manager
   ii. Aws IoT Device Management
   iii. Azure IOT Hub
   iv. WinCC OA IOT OPA
d. Cloud Data Storage
   i. AWS S3
   ii. Google Cloud Storage
   iii. Microsoft Azure Storage
e. Edge computing
   i. AWS green grass
   ii. Azure Edge IoT
   iii. Custom solutions based on docker swarm
   iv. Multi-access edge computing (MEC)
   v. AWS Wavelength

4. IOT and IOE
   a. Industrial IOT
      i. Industrial communication protocols
         a. Ethernet Protocols (EtherNet/IP,ProfiNET,Modbus,OPC,EtherCAT)
         b. Fieldbus Protocols (Profibus DP,Modbus-RTU)
         c. Wireless Protocols (WLAN,Bluetooth)
      ii. Industrial Gateway and data acquisition device
   b. Embedded Computing
      i. Microcontroller programming and RTOS
         1. Arduino
         2. STM32
         3. ESP32
         4. FPGA
      ii. Microprocessor programming and embedded Linux
         1. RaspberryPi
         2. Other SBC
   c. Sensors (hardware)
   d. Signal Processing
   e. Blockchain
   f. Connectivity

i. GSM/4G/5G
ii. MQTT, Node-Red
iii. REST API and Webhook
iv. RFID/NFC
v. Bluetooth
vi. LPWAN (Low power wide area network)

g. IOE (Internet of Everything)