Sviluppo di un'applicazione Java ontology-based per la configurazione di oggetti tecnologici

ST1414 Modellazione e Gestione della Conoscenza

Matteo Morelli

Corso di laurea triennale in Informatica per la comunicazione digitale

Università degli Studi di Camerino

Anno Accademico 2023/24

1 DOMINIO E AMBITO APPLICATIVO

Nell'era della digitalizzazione e della diffusione capillare della tecnologia, la personalizzazione dei dispositivi tecnologici sta diventando sempre più importante per gli utenti. Questo progetto si propone di sviluppare un configuratore di oggetti tecnologici (come computer, mouse, tastiere, monitor, ecc.) con l'obiettivo di offrire agli utenti un'esperienza di acquisto altamente personalizzata. L'interfaccia intuitiva del configuratore consentirà di personalizzare in modo dettagliato le caratteristiche tecniche e fisiche dei dispositivi, migliorando l'adattabilità dei prodotti alle esigenze specifiche di ciascun utente.

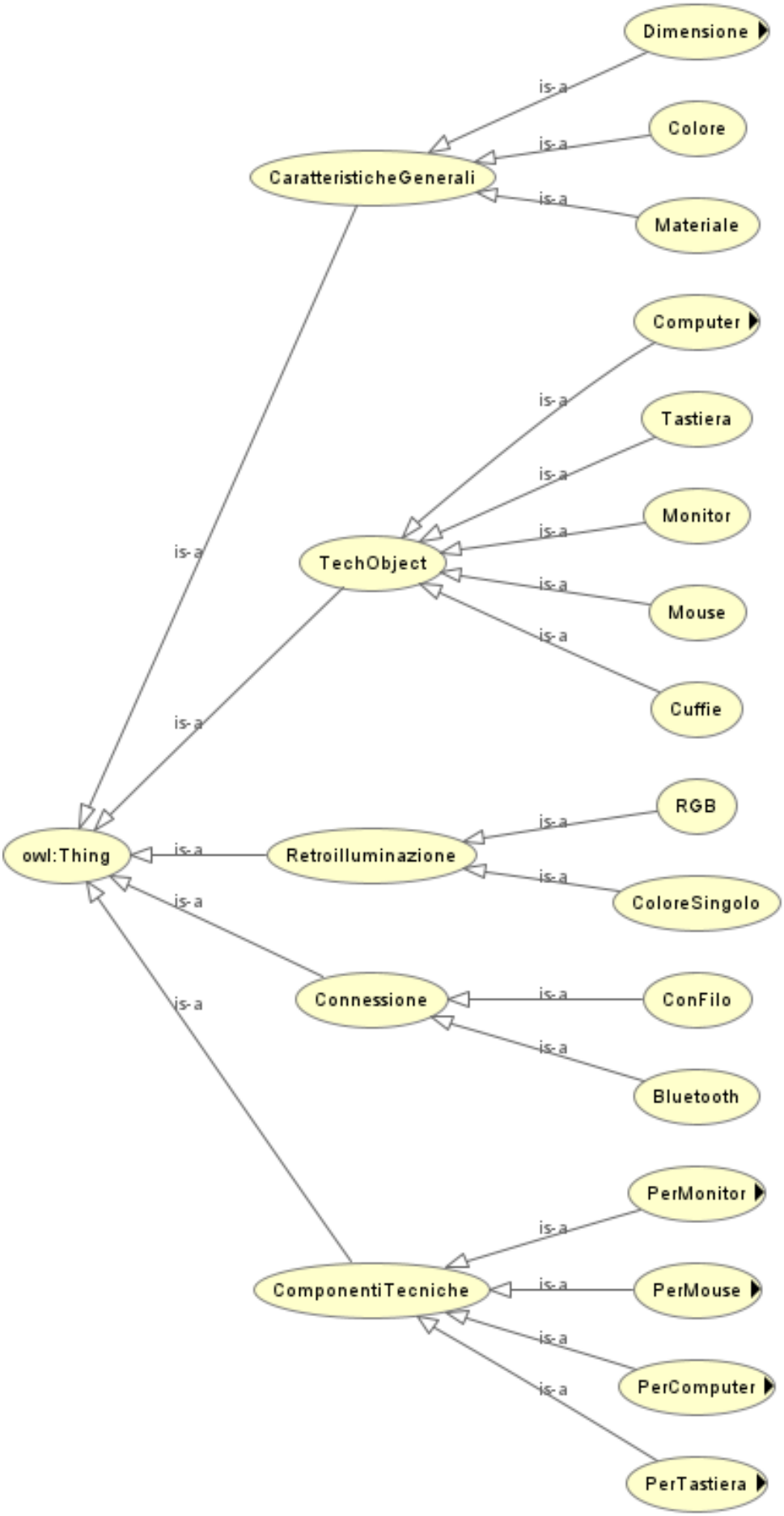
Il configuratore permette di selezionare e personalizzare diverse caratteristiche tecniche e fisiche degli oggetti tecnologici, come dimensioni, materiali, colori e specifiche tecniche.

I principali vantaggi del configuratore includono:

- **Esperienza di Acquisto Personalizzata:** Gli utenti possono selezionare componenti e caratteristiche specifiche, creando un prodotto che rispecchia le loro necessità personali e professionali.
- **Riduzione degli Sprechi e Miglior Efficienza:** La configurazione su misura consente di ridurre gli sprechi di materiali e risorse, offrendo un'alternativa più sostenibile alla produzione in serie.

Per realizzare il configuratore, è stata creata un'ontologia che descrive tutte le caratteristiche rilevanti degli oggetti tecnologici, come le specifiche tecniche, le dimensioni e i materiali. L'ontologia costituisce la base per la struttura dell'applicazione. Le classi principali nell'ontologia coprono vari domini, tra cui le caratteristiche generali degli oggetti, i componenti tecnici specifici per ciascun tipo di dispositivo e le diverse opzioni di connessione e retroilluminazione.

Di seguito allego la struttura dell’ontologia realizzata tramite protege


owl:Thing
CaratteristicheGenerali
Dimensione
Colore
Materiale
TechObject
Computer
Tastiera
Monitor
Mouse
Cuffie
Retroilluminazione
RGB
ColoreSingolo
Connessione
ConFilo
Bluetooth
ComponentiTecniche
PerMonitor
PerMouse
PerComputer
PerTastiera
is-a

2 ONTOLOGIA ED INFERENZA

L’utilizzo di un’ontologia in questo progetto è fondamentale perché fornisce una struttura organizzata delle informazioni. Un altro aspetto molto importante è l'inferenza, che permette di trarre conclusioni logiche a partire da informazioni esistenti, utilizzando regole, assiomi e relazioni prestabilite per ottenere nuove informazioni.

2.1 Obiettivi dell’Ontologia

L'utilizzo dell'ontologia nel progetto ha come obiettivo la corretta composizione di un oggetto tecnologico, sfruttando le proprietà e le relazioni descritte. Gli obiettivi principali includono:

- Classificare i Tech Object in cinque categorie: Computer, Mouse, Tastiera, Monitor e Cuffie.
- Classificare le diverse caratteristiche che possono essere assegnate ai vari Tech Object, distinguendo tra caratteristiche generali, ovvero quelle condivise da tutti i Tech Object, e caratteristiche tecniche, classificate in base ai vari Tech Object.
- Gestire le diverse sottoclassi della classe Dimensioni, suddivise in classi che specificano le dimensioni dei vari Tech Object.

2.2 Strumenti Utilizzati

Nel progetto, per la gestione dell’ontologia e dell’inferenza, sono state impiegate le seguenti API:

- **Apache Jena:** Un framework open source per lo sviluppo di applicazioni che gestiscono linked data. Fornisce varie funzionalità, tra cui creazione e lettura di grafi RDF, esecuzione di query SPARQL, interazione con ontologie RDFS e OWL, e motori di inferenza.
- **Pellet (Openllet):** Un reasoner OWL-DL open source, compatibile con Apache Jena, che fornisce funzionalità di verifica della coerenza delle ontologie, computazione della gerarchia delle classi, spiegazione delle inferenze e risposte alle query SPARQL.

3 Utilizzo dell’ontologia nel progetto

Le query SPARQL presenti nel progetto vengono utilizzate per interrogare l’ontologia ed estrarre le informazioni necessarie all’utente per impostare la propria configurazione, queste informazioni saranno poi mostrate a schermo.

I prefissi utilizzati nel progetto per effettuare correttamente le query SPARQL sono:

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX tc: <https://www.unicam.it/cs/matteomorelli/tech-configurator#>
```

Di seguito mostrerò alcune query presenti nel progetto e ne spiegherò funzionamento ed utilità.

3.1 Lista dei Tech Object

All'avvio dell'applicazione, questa è la prima query richiamata, necessaria per mostrare all'utente i Tech Object. La scelta dell'utente modifica poi le successive scelte e le successive query selezionate. La query restituisce le sottoclassi della classe Tech Object, ignorando però le sottoclassi della classe Computer, che saranno mostrate in un'altra query.

```
SELECT  ?label  ?value
WHERE {
?techobject rdfs:subClassOf tc:TechObject .
FILTER NOT EXISTS {?techobject rdfs:subClassOf ?superclass .?superclass rdfs:subClassOf tc:Computer .}
?techobject rdfs:label ?value .
BIND(STR(?techobject) AS ?label)
}
```

3.2 Lista delle categorie computer

Questa query permette la visualizzazione delle categorie dei computer, recuperando le sottoclassi della classe Computer. Viene richiamata solo se, inizialmente, è stato scelto come Tech Object il computer.

```
SELECT ?label ?value
WHERE {
?TechObjectCategory rdfs:subClassOf tc:Computer.
BIND((?TechObjectCategory) AS ?label).
?TechObjectCategory rdfs:label ?value .
}
```

3.3 Lista colori

Questa query mostra a schermo la lista dei colori che l'utente può selezionare per il suo Tech Object. Il funzionamento è semplice: la query ottiene le istanze della classe Colore.

```
SELECT ?label ?value
WHERE {
?individual rdf:type tc:Colore.
BIND((?individual) AS ?label) .
?individual rdfs:label ?value.
}
```

Questa tipologia di query viene poi riutilizzata anche per tutte le altre query che devono ottenere le istanze delle varie caratteristiche dei Tech Object; ogni query verrà richiamata in base al Tech Object selezionato all'inizio dall'utente.

3.4 Lista dimensioni

Questa query ha una particolarità rispetto alle altre: in base al Tech Object selezionato, vengono mostrate opzioni differenti. Per fare questo, è stato inserito un valore variabile che viene sostituito dal nome del Tech Object.

```
SELECT ?label ?value
WHERE {
?dimtc rdf:type tc:Dimensione%s.
BIND((?dimtc) AS ?label) .
?dimtc rdfs:label ?value .
}
```

Ad esempio, se il Tech Object selezionato è Mouse, l'elemento %s verrà sostituito da Mouse, formando la classe DimensioneMouse, che contiene tutte le istanze per le dimensioni del mouse. Per il Tech Object Computer, questa query viene applicata dopo la selezione della categoria, ovvero dopo la scelta tra Case e Portatile.

4 Descrizione delle Responsabilità

Durante la progettazione sono state individuate le seguenti responsabilità:

- **Costruire il modello dell'ontologia:** Inizialmente viene costruito il modello RDF per rappresentare classi, proprietà e relazioni dell'ontologia. Successivamente, avviene il caricamento dell'ontologia realizzata tramite Protégé.
- **Memorizzare le query SPARQL utili:** Vengono memorizzate le query SPARQL che dovranno essere utilizzate nei vari scenari.
- **Eseguire le query SPARQL sul modello:** Vengono configurate ed eseguite le query sul modello dell'ontologia.
- **Parsing dei risultati delle query:** I dati ottenuti dalle query vengono codificati in un formato accessibile.
- **Strutturare ed esporre i dati interpretati:** I dati interpretati vengono rappresentati e le operazioni di accesso alle singole informazioni sono definite.
- **Gestione del modello dell'ontologia:** Vengono eseguite tutte le operazioni sull'ontologia.
- **Gestione dell'applicazione:** Gestione delle interazioni tra l'interfaccia utente e il modello.
- **Mostrare i dati tramite l'interfaccia:** I dati vengono stampati nei rispettivi elementi dell'interfaccia utente.
- **Gestire l'interfaccia utente:** Gestione degli elementi dell'interfaccia e dei relativi eventi.

5 Come vengono implementate le responsabilità

5.1 Costruire il Modello dell'Ontologia

L'interfaccia OntologyBuilder definisce il contratto comune per la costruzione di modelli RDF. Essa viene implementata dalla classe BasicOntologyBuilder, che consente di creare e caricare modelli RDF standard. A sua volta, BasicOntologyBuilder è estesa dalla classe InferredOntologyBuilder, la quale aggiunge funzionalità specifiche per la costruzione di modelli inferiti, utili nel contesto di questo progetto.

5.2 Memorizzare le Query SPARQL

Per memorizzare le query SPARQL, il progetto utilizza un metodo basato su enumerazioni. L'interfaccia SparqlQuery definisce un contratto comune per la definizione, gestione e parametrizzazione delle query SPARQL. Questa interfaccia è implementata dall'enumerazione SPARQLSelectionQueries, che memorizza tutte le query necessarie.

5.3 Eseguire le Query SPARQL

L'interfaccia QueryProcessor definisce il contratto comune per l'esecuzione delle query SPARQL utilizzando le API di Jena. Questa interfaccia è implementata dalla classe OntologyQueryProcessor, che gestisce l'esecuzione delle query sul modello RDF.

5.4 Parsing dei Risultati delle Query

Il contratto per il parsing dei risultati delle query è definito dall'interfaccia DataParser. La classe JSONParser, che implementa DataParser, si occupa di convertire i dati ottenuti in un formato JSON, rendendoli facilmente accessibili e utilizzabili all'interno dell'applicazione

5.5 Strutturare ed Esporre i Dati Interpretati

L'interfaccia ParsedData consente l'accesso ai risultati delle query dopo che sono stati elaborati e modificati. La classe JSONData, che implementa ParsedData, presenta i dati in un formato JSON-like, facilitando la manipolazione e la visualizzazione delle informazioni.

5.6 Gestione del Modello dell'Ontologia

La classe OntologyController è responsabile della gestione del modello dell'ontologia e del suo collegamento con il controller dell'applicazione. Questa classe mantiene il modello RDF istanziato, il costruttore del modello e l'esecutore delle query SPARQL, coordinando le operazioni necessarie per la gestione dell'ontologia

5.7 Gestione dell'Applicazione

Il progetto adotta il pattern MVC (Model-View-Controller). La classe Controller agisce da mediatore tra l'interfaccia utente (View) e il modello dell'ontologia (Model), gestito dal OntologyController. In questo modo, garantisce una comunicazione efficiente tra i componenti dell'applicazione, facilitando la gestione delle interazioni e dei dati.

5.8 Mostrare i Dati Tramite l'Interfaccia

L'interfaccia grafica dell'applicazione è definita tramite file FXML, che stabiliscono la struttura visiva delle varie scene, e controller Java che gestiscono le azioni dell'utente. Sono presenti tre controller principali:

- **PaginaInizialeController:** Gestisce la pagina iniziale dell'applicazione, inclusa la funzionalità del pulsante "Start" che avvia il processo di configurazione e naviga alla scena successiva.

- **ScenaConfiguratoreController:** Responsabile della gestione della scena principale dove l'utente effettua le scelte di configurazione. Questo controller lavora insieme alla classe di supporto GestoreConfiguratore, che fornisce metodi per gestire le opzioni disponibili e le selezioni dell'utente.
- **ScenaFinaleController:** Raccoglie le scelte dell'utente e genera un riepilogo finale della configurazione selezionata.

5.9 Gestire l'Interfaccia Utente

La classe App funge da punto di ingresso dell'applicazione e si occupa dell'avvio dell'interfaccia utente. All'avvio, carica e visualizza la scena iniziale definita nel file ScenaIniziale.fxml. Da questa scena, l'utente può navigare attraverso le diverse fasi dell'applicazione, passando dalla configurazione iniziale al riepilogo finale delle proprie scelte

6 Conclusione

Il progetto del configuratore di oggetti tecnologici rappresenta un passo significativo verso una maggiore personalizzazione e sostenibilità nell'acquisto di dispositivi tecnologici. L'adozione di un'ontologia ben strutturata e l'utilizzo di strumenti come Apache Jena e Pellet hanno permesso di creare un sistema robusto e flessibile, capace di adattarsi alle esigenze specifiche di ciascun utente attraverso un'interfaccia intuitiva e interattiva.

L'impiego delle query SPARQL per interrogare l'ontologia e visualizzare le opzioni di configurazione in tempo reale ha reso l'interazione utente più efficiente, migliorando l'esperienza d'acquisto e riducendo gli sprechi. L'architettura del progetto, basata sul pattern MVC, garantisce una chiara separazione tra la logica di business e la gestione dell'interfaccia, facilitando la manutenzione e l'estensibilità del sistema.

In futuro, ulteriori sviluppi potrebbero includere l'espansione dell'ontologia per coprire nuovi tipi di dispositivi e caratteristiche, oltre a miglioramenti nell'interfaccia utente per rendere l'esperienza di configurazione ancora più coinvolgente e user-friendly.

In sintesi, il configuratore realizzato non solo risponde alle necessità degli utenti moderni, ma apre la strada a nuove possibilità nel campo della tecnologia personalizzata, combinando innovazione, efficienza e sostenibilità.