TREBALL DE FI DE GRAU

Creació d'un sistema expert per al diagnòstic d'intoleràncies alimentàries

Cristina Llort Ramírez

dirigit pel Dr. David Sánchez Ruenes

UNIVERSITAT ROVIRA I VIRGILI

Grau d'Enginyeria Informàtica

Tarragona

2021

Vull expressar el meu més sincer agraïment a les següents persones:

Al meu tutor, Dr. David Sánchez, per tot l'esforç i tot el recolzament realitzat en la direcció d'aquest projecte.

Al meu padrí, per l'assessorament al llarg de tots aquests anys i per transmetre'm la seva passió per la ciència des de petita.

Al meu company i amic Magí, pel suport que m'ha brindat durant tot el grau.

I, finalment, als meus pares. És gràcies a ells que el dia d'avui sóc aquí.

Resum.

Aquest projecte implementa un sistema expert que pretén facilitar el procés a l'equip mèdic i als pacients en la detecció d'intoleràncies alimentàries, proporcionant un entorn que relaciona els diferents àpats del pacient amb la simptomatologia que aquest presenta.

Per fer-ho, es desenvolupen cada una de les parts del que s'anomena un sistema expert o sistema basat en el coneixement: es construeix una ontologia des de zero per representar la base de coneixement específica, es dissenya una estructura on el pacient és capaç d'enregistrar cada àpat realitzat i s'implementa el motor d'inferència que aproxima un diagnòstic del pacient basant-se en les consultes realitzades i les regles designades a la base del sistema.

Per tant, la intenció del projecte es formalitzar el coneixement d'un àmbit molt específic i digitalitzar el procés d'emmagatzematge dels àpats i la simptomatologia pertinent dels pacients per tal de facilitar l'extracció d'un diagnòstic.

Resumen.

Este proyecto implementa un sistema experto que pretende facilitar el proceso al equipo médico y a los pacientes en la detección de intolerancias alimentarias, proporcionando un entorno que relaciona las diferentes comidas del paciente con la sintomatología que este presenta.

Para hacerlo, se desarrollan cada una de las partes de lo que se conoce como un sistema experto o sistema basado en el conocimiento: se construye una ontología desde cero para representar la base de conocimiento específica, se diseña una estructura donde el paciente sea capaz de registrar cada comida realizada y se implementa el motor de inferencia que aproxima un diagnóstico del paciente basándose en las consultas realizadas y las reglas designadas en la base del sistema.

Por lo tanto, la intención del proyecto es formalizar el conocimiento de un ámbito muy específico y digitalizar el proceso de almacenamiento de las comidas y la sintomatología pertinente de los pacientes para facilitar la extracción de un diagnóstico.

Abstract.

This project implements an expert system whose aim is make easier for the medical team and their patients the process of food intolerances detection, providing an environment that relates the different meals of the patient with the symptoms they present.

To achieve this goal, each part of an expert system is developed: an ontology is built from scratch to represent the specific knowledge base, a structure is designed where the patient is able to record each meal taken and the inference engine is implemented to approximate a diagnosis of the patient based on the queries made and the rules designated in the base of the system.

Therefore, the intention of the project is to formalize the knowledge of a very specific area and digitize the process of recording meals and its pertinent symptoms of patients to facilitate the extraction of a diagnosis.

Índex

Índex de taules

Índex de figures

Índex de diagrames

Índex de codis

1 Introducció

1.1 Descripció

Cada dia és més habitual escoltar parlar de casos d'intoleràncies alimentàries entre la gent del nostre entorn. L'estil de vida frenètic de la societat en la que vivim promou una base d'alimentació que inclou molts productes híper processats, on sovint s'abusa dels àpats precuinats, el que provoca un increment en les respostes adverses.

El desenvolupament dels productes anomenats *light* també ha estat còmplice de l'augment dels casos d'intolerància. Avui en dia és força complicat consumir exclusivament productes que assegurin al nostre organisme la qualitat que ens allunyi d'alguna reacció al·lèrgica o intolerant. És molt més fàcil accedir a productes nocius que a productes realment saludables, sovint és fins i tot complicat conèixer la veritable composició dels productes seleccionats, doncs els fabricants no estan obligats a etiquetar específicament tots els additius d'aquests (un exemple és el cas de l'edulcorant E-420, sorbitol).

D'aquesta problemàtica sorgeix aquest treball, el qual consisteix en la creació d'una ontologia i d'un sistema expert que permeti diagnosticar potencials pacients d'intoleràncies alimentàries. Aquest diagnòstic es duu a terme mitjançant la informació proporcionada pels pacients i emmagatzemada en una base de coneixement, utilitzant mètodes propis del camp de la intel·ligència artificial.

Aquesta problemàtica, però, sovint presenta dificultats a l'hora de ser diagnosticada; no existeixen tests mèdics per a tot tipus d'intolerància i sovint és complicat definir els límits d'aquestes, doncs el pacient pot estar encara desenvolupant-la (cas en el que un diagnòstic detectat a temps pot ser molt positiu en la qualitat de vida del pacient) o tractar-se d'una intolerància poc habitual.

Així doncs, el que diferencia aquest projecte de tants altres és que el sistema pretén facilitar el procés a l'equip mèdic i als pacients en la detecció de la malaltia, proporcionat un entorn que relacioni els diferents àpats del pacient amb la simptomatologia que presenta. És per això que el projecte presenta una estructura on el pacient és capaç d'enregistrar cada àpat realitzat seguit de la posterior simptomatologia, en cas de que es presenti.

La intenció del sistema es digitalitzar el procés del que s'anomena un *diari nutricional* [15], tot incloent la simptomatologia dels pacients. Aquest mètode és ja utilitzat en les consultes d'al·lergòlegs però força complicat d'aplicar, doncs l'estil de vida frenètic de la societat constitueix també una gran limitació als pacients per enregistrar-ho a mà.

Per altra banda, a més d'enregistrar el diari nutricional, el sistema pretén aproximar un diagnòstic mitjançant la utilització dels algorismes més adients.

1.2 Motivacions

Aquest projecte sorgeix d'una motivació personal per la temàtica degut a la pròpia experiència com a pacient de diverses intoleràncies. Aquest és el cas també d'altres membres de la meva família i persones properes.

En els darreres anys he pogut apreciar l'increment d'aquesta problemàtica amb els meus ulls, tant en el meu entorn més proper com en la societat. A dia d'avui, comentar sobre la intolerància al sorbitol, per exemple, ja no és un tema de debat exclusiu dins del sector mèdic especialitzat, la majoria de persones pateixen o coneixen algú que presenta alguna patologia semblant.

De fet, la idea base d'aquest projecte neix de la pròpia experiència en la consulta del meu al·lergòleg, el qual em va animar a realitzar l'enregistrament d'un diari nutricional on representés les simptomatologies que anava patint. Va ser en aquest moment quan em vaig adonar de la importància d'aquest registre, ja que sovint la simptomatologia podia variar molt: alguns cops el mateix producte podia afectar-me en major o menor mesura depenent de la composició de la resta de l'àpat o de la diferència temporal entre l'última ocasió en que havia consumit el producte en qüestió i l'àpat actual.

És per això que, facilitar aquest procés tant a metges com a pacients, em va semblar una idea molt interessant.

1.3 Objectius

1.3.1 *Objectius generals*

L'objectiu general d'aquest projecte, tal i com s'introduïa en l'apartat anterior, no és un altre que el de desenvolupar un sistema de software que ajudi a simplificar el procés de diagnòstic d'una intolerància alimentària, així com que faciliti la vida dels pacients fent més còmode el procés d'enregistrament de la informació rellevant per al diagnòstic.

1.3.2 *Objectius específics*

Tanmateix, existeixen diversos motius més específics pels quals la realització d'aquest projecte resulta molt interessant.

1.3.2.1 Objectius acadèmics

- Aprofundiment teòric del concepte d'ontologia i tots els coneixements associats.
- Adquisició i pràctica dels coneixements necessaris per a la construcció d'una ontologia, el que requereix aprofundir en l'ús de noves metodologies i *frameworks*.
- Aprofundiment teòric dels sistemes experts dins de la intel·ligència artificial.
- Aprenentatge pràctic sobre la creació d'un sistema expert, el que requereix assolir nous coneixements utilitzant diverses API's.
- Avaluació i anàlisi de resultats a partir de la creació d'un sistema propi.

1.3.2.2 Objectius personals

- Ampliar la base de coneixement sobre la temàtica escollida.
- Realitzar un projecte propi des de zero.
- Desenvolupar les capacitats de recerca i independència.

1.4 Organització del document

Degut a que l'estructura d'aquest projecte no segueix els estàndards dels projectes d'enginyeria del software, s'ha decidit necessari incloure aquest apartat per detallar l'estructura del document.

Així doncs, en el projecte es diferencien tres grans blocs:

i) ***Background.*** En aquest apartat s'inclou el marc teòric necessari per desenvolupar el projecte posteriorment. Els aspectes a destacar són:

   a. Coneixement sobre sistemes experts.

   b. Coneixement d'un sistema ontològic.

   c. Metodologia per desenvolupar una ontologia: *Ontolgy Engineering.*

ii) **Creació de l'ontologia.** El següent bloc correspon a la creació del sistema ontològic utilitzant els conceptes adquirits i detallats en el *Background*, emprant la metodologia esmentada.

iii) **Implementació del sistema expert.** Finalment, la creació del sistema expert que implementa i enllaça cadascun dels diferents components que el formen, explicats novament en el primer bloc.

Finalment, es realitza una avaluació dels elements creats, tant de l'ontologia com del sistema expert i s'extreuen les conclusions pertinents.

2 *Background*

2.1 Sistemes experts

Per a la realització d'aquest projecte cal assolir una base teòrica sobre els anomenats sistemes basats en el coneixement o sistemes experts, degut que l'objectiu final és implementar-ne un.

Un sistema expert és aquell capaç de resoldre problemes sobre dominis complexos. L'aparició d'aquests data de finals dels anys seixanta i va estar impulsada pels motius que s'exposen a continuació:

- Reducció de costos, ja que disposar de persones qualificades i expertes en cada àmbit de coneixement és molt costós en quant a temps i econòmicament.
- Posar en comú el coneixement de molts experts.
- Preservar el coneixement d'un domini de coneixement específic.
- Facilitar l'aprenentatge del domini a una persona no experta en aquest.
- Resoldre els problemes amb major eficàcia; sovint un sistema computacional és capaç de resoldre problemes més complexos de forma més ràpida i amb una solució de major qualitat.
- Els mètodes bàsics de resolució de problemes implementats fins al moment (basats en la cerca en un espai d'estats) no comprenien coneixement del domini.

2.1.1 *Components d'un sistema expert*

Per altra banda, els components principals que el formen són els següents:

- Base de coneixement (BC): espai on està representada la informació general del domini, ja sigui en forma de fórmules lògiques, regles, ..., que descriuen els mecanismes de raonament que permeten resoldre el problema.
- Base de fets o memòria de treball (MT): espai on s'emmagatzemen les dades inicials del problema a resoldre (predicats) i les que s'obtenen durant el procés de resolució.
- Motor d'inferència (MI): intèrpret que realitza els passos necessaris de deducció/raonament sobre les dades del problema utilitzant la BC per solucionar el problema. La solució d'un problema dependrà del procés seguit per l'intèrpret.

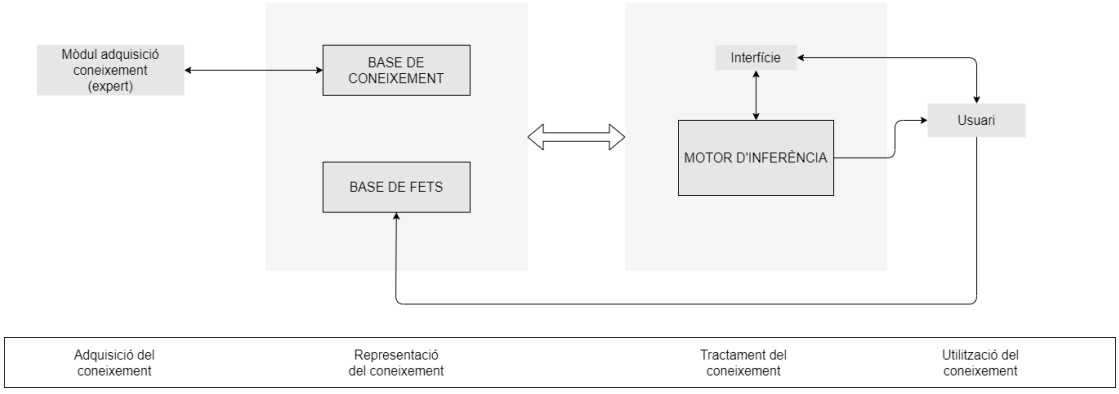


Diagrama 1. Arquitectura d'un sistema expert.

2.1.2 *Sistema basat en regles*

A més, es poden diferenciar diferents tipus de sistemes en base a la naturalesa de la base de coneixement descrita anteriorment. Així doncs, si la BC d'un sistema és un conjunt de regles, el sistema s'anomena sistema de producció o sistema basat en regles.

Una regla té la següent forma:

si [condicions] llavors [accions]

on les condicions de la part esquerra de la regla es refereixen a la presència (condicions positives) o absència (condicions negatives) d'objectes amb certes característiques dins de la MT. Per altra banda, les accions de la part dreta són normalment operacions d'entrada/sortida (escriure per pantalla, demanar una dada a l'usuari) i/o accions sobre els elements de la MT (afegir, modificar o esborrar un element).

El cas d'aquest projecte entra dins dels sistemes basats en regles, ja que s'utilitzen com a base de coneixement les regles descrites per una ontologia, concepte que s'explica en l'apartat 2.2.

2.1.3 *Algorismes d'inferència*

Per altra banda, per tal d'implementar el motor d'inferència existeixen diversos algorismes, anomenats algorismes d'inferència, que pretenen respondre les qüestions que presenta el problema de forma lògica.

Es distingeixen tres grans famílies d'algorismes d'inferència:

- **Encadenament cap endavant o raonament deductiu.** Està dirigit pels fets, parteix d'evidències, símptomes i/o dades per arribar a les conclusions. Es basa en el *modus ponens*[1] $A, A \rightarrow B \mid\!- B$ i s'utilitza en sistemes de producció.

  En aquest tipus d'inferència la base de fets (o MT) s'inicialitza amb els fets ja coneguts i d'aquests s'obtenen les conseqüències derivables de la següent forma:

  1. Es comparen els fets de la MT amb la part esquerra de les regles; es seleccionen les regles aplicables: les que tenen antecedents coneguts (que estan a la base de fets).
  2. S'agreguen a la MT les noves conclusions de les regles aplicades.
  3. S'itera fins trobar una condició de finalització.

- **Encadenament cap enrere o raonament inductiu.** Està dirigit per un objectiu que és la conclusió (o la hipòtesis) que es tracta de validar reconstruint la cadena de raonament en ordre invers. Per tant, el procés de

[1] En lògica, el *modus ponendo ponens* (del llatí, manera que afirmant afirma), també anomenat *modus ponens* i generalment abreujat MPP o MP, és una regla d'inferència.

resolució consisteix en l'exploració d'un arbre, ja que cada pas enrere implica nous sub-objectius (hipòtesis que s'ha de validar). El funcionament d'aquest tipus d'inferència és el següent:

- S'inicialitza la base de fets (o MT) amb un conjunt inicial de fets.
- S'inicialitza el conjunt d'hipòtesis (CH) amb els objectius a verificar.
- Mentre existeixen hipòtesis a validar, s'escull una d'elles i es valida com segueix:
    a. Es compara la MT i la part dreta de les regles amb la hipòtesis.
    b. Es comprova si la hipòtesis està a la MT.
        i. Si la hipòtesis està a la MT s'elimina del CH.
        ii. Si no hi és: es busquen regles que tinguin a com a conclusió la hipòtesis. Es selecciona una d'aquestes i s'afegeix a la llista de premisses no satisfetes al CH com a sub-objectiu.

- **Mixt o encadenament híbrid**. En aquest tipus de sistema, algunes parts de la cadena de raonament es construeixen de forma deductiva i unes altres de forma inductiva; és una exploració bi-direccional.

2.1.4 *Característiques d'un sistema expert*

L'arquitectura que dóna lloc a un sistema expert descrita en els anteriors apartats té una raó de ser, doncs busca assegurar que el sistema expert posseeixi les característiques següents.

- Separació de l'explicitació del coneixement (BC) i dels mecanismes de control (MI), el que permet refinar el coneixement i/o provar diferents mecanismes de raonament sobre ell.
- Permet l'ús en dominis poc estructurats, on no hi ha un model matemàtic clar del problema a resoldre (com per exemple en un diagnòstic mèdic).
- Permet el tractament de coneixement simbòlic, a més del numèric.
- Permet el tractament de coneixement imprecís, incert, inexacte o probabilístic.
- Pot incorporar coneixement heurístic, per exemple escollint el mecanisme d'inferència més adient pel domini a tractar.
- Són sistemes força interactius, habitualment demanen dades i ofereixen respostes utilitzant llenguatge natural.
- Acostumen a tenir una alta capacitat d'explicació/justificació de com s'han obtingut els resultats.
- Han d'operar sota uns requisits temporals específics, dependents de l'aplicació.

2.1.5 *Tipus de sistemes experts*

Per últim, segons l'àmbit d'utilització del sistema i de les seves funcionalitats, aquests es poden classificar en diversos tipus.

- Anàlisi: s'estudien unes dades d'entrada per tal d'arribar a una conclusió. Dins aquest grup es poden classificar entre:
  - Interpretació: anàlisi de les dades disponibles per tal d'obtenir el seu significat, on la informació és imprecisa i/o contradictòria (per exemple PROSPECTOR, utilitzat per a l'anàlisi de dades geològiques).
  - Diagnòstic: determinar l'estat d'un procés en base a les dades disponibles (per exemple MYCIN, utilitzat per al diagnòstic de malalties infeccioses).
  - Monitorització: interpretació continuada en el temps de les senyals d'un sistema (per exemple monitoritzar una xarxa telefònica).
  - Control: monitorització + diagnòstic (per exemple el control d'un procés industrial)
  - Predicció: construir un model per a un sistema a partir de dades passades, de forma que permeti anticipar situacions futures.
- Síntesis: construcció d'un objecte complex a partir de components simples. Es distingeixen dos subcategories:
  - Planificació: construcció de plans o seqüències d'accions a realitzar per tal d'assolir un objectiu concret (per exemple planificar el procés de fabricació d'un vehicle).
  - Disseny: construcció de sistemes complexos (per exemple la configuració del hardware d'ordinadors).

El **sistema implementat** es pot classificar dins dels sistemes de **diagnòstic**, ja que compleix l'objectiu d'aquests.

2.2 Ontologia

Tal i com T. Gruber descriu en el seu llibre [1], una ontologia és *una especificació formal d'una conceptualització,* on la conceptualització es refereix a un model abstracte del domini o del fenomen del món que representa. També Gruber descriu una ontologia com a un conjunt de primitives representatives amb les que es pot modelar un domini de coneixement.

Així doncs, dins de la informàtica, una ontologia és aquella que formula de manera exhaustiva i rigorosa un esquema conceptual dins d'un domini donat, amb la finalitat de facilitar la comunicació i la compartició d'informació entre diferents sistemes, els quals són els principals objectius i avantatges de la utilització d'ontologies.

2.2.1 *Ús d'ontologies*

Tal i com s'introduïa en la descripció general d'una ontologia, l'ús d'aquestes facilita enormement l'estudi de molts camps de coneixement. A continuació s'enumeren algunes de les principals avantatges d'utilitzar ontologies.

- Compartir coneixement comú sobre l'estructura d'algun àmbit.
- Permet reutilitzar el coneixement d'un domini.
- Explicitar suposicions sobre un domini.
- Separar el coneixement del domini del coneixement operacional.
- Possibilitar l'anàlisi del coneixement del domini.

2.2.2 *Tipus d'ontologies*

Basant-se en el coneixement extret de *The Evaluation of Ontologies* [2], es distingeixen tres tipus fonamentals d'ontologia:

- Ontologies d'un domini, aquelles en les que es representa el coneixement especialitzat d'un domini o subdomini.
- Ontologies genèriques, aquelles en les que es representen conceptes generals i nocions bàsiques per a fenòmens tals com l'espai, el temps, l'estat, els esdeveniments. Són vàlides en diversos dominis.
- Ontologies de representacions, aquelles en les que s'especifiquen les conceptualitzacions subjacents als formalismes de representació del coneixement, també conegudes com a meta-ontologies (*meta-level* o *top-level ontologies*).

A aquests tipus es poden afegir les anomenades *task ontologies*, ontologies que han estat creades per descriure una activitat o una tasca específica, tal i com relata Guarino [3].

Un exemple d'aquest darrer tipus d'ontologies són aquelles que han estat creades amb la finalitat de diagnosticar una malaltia o un desordre, ja que compleixen la premissa d'estar dissenyades per a una aplicació específica.

Per tant, l'ontologia que es construeix en aquest projecte es pot englobar dins aquest darrer grup.

2.2.3 *Entitats d'una ontologia*

Per altra banda, per entendre com es pot representar una ontologia caldrà identificar les entitats que la formen. A continuació s'enumeren i expliquen breument.

- Classes. Representen els conceptes del domini.
- *Slots* o propietats o rols. Representen les propietats de cada concepte que descriuen les seves característiques i atributs.
- *Facets* o restriccions de rols. Representen les restriccions sobre les propietats (*slots*).

2.2.4 *Representació d'una ontologia*

Finalment, per tal de representar correctament una ontologia cal conèixer les eines adequades per desenvolupar-la. És per això que s'han de definir l'entorn de treball, els llenguatges i els formats adequats.

- **Entorn de treball.** Per tal d'implementar de forma còmoda una ontologia, existeixen diversos frameworks que ens permeten treballar utilitzant els estàndards d'aquestes. Alguns exemples són Protégé o OBO Edit. En el cas d'aquest projecte s'usa Protégé, tal i com es detalla a l'apartat 2.5.1.
- **Llenguatge de representació.** Per tal de representar el coneixement de forma estandarditzada (el que permet la reutilització de les ontologies, característica fonamental d'aquestes) cal usar algun formalisme de representació. OWL és aquest formalisme, en forma de llenguatge estàndard, que permet representar ontologies i que es detalla en apartats posteriors.

  Per tal de treballar amb un entorn com Protégé utilitzant el llenguatge OWL, com és el cas d'aquest projecte, les dades es tracten en els formats que es mostren a la figura següent.

Figura 1. Formats d'una ontologia.

XML	XML, de l'anglès *eXtensible Markup Language*, és un metallenguatge extensible d'etiquetes, desenvolupat pel World Wide Web Consortium, que permet definir la gramàtica de llenguatges específics.
XML Schema	XML Schema és un llenguatge d'esquema utilitzat per descriure l'estructura i les restriccions dels continguts dels documents XML d'una forma molt precisa, més enllà de les normes sintàctiques imposades pel mateix llenguatge XML.
RDF	El *Resource Description Framework* és un model conceptual de dades estandarditzat pel World Wide Web Consortium i usat per definir dades en el web semàntic
RDF Schema	RDF Schema és un vocabulari i una extensió semàntica d'RDF.

Taula 1. Formats.

2.3 Llenguatge OWL

Com s'ha dit anteriorment, el llenguatge OWL és aquell que estandarditza la representació del coneixement ontològic.

Tal i com s'extreu de. «OWL Web Ontology Language Overview». W3C Recommendation for OWL, the Web Ontology Language [19] :

"OWL, de l'anglès Ontology Web Language, està dissenyat per a ser utilitzat per aplicacions que necessiten processar el contingut de la informació en lloc de presentar la informació als éssers humans.

OWL facilita la interpretació per part de la màquina del contingut web, proporcionant un vocabulari amb semàntica formal addicional a XML, Marc de Descripció de Recursos (RDF, Resource Description Framework) i Esquema de Marc de Descripció de Recursos (RDF Schema), base sobre la que se suporta OWL. Aquestes eines, fan possible el projecte de web semàntica. OWL disposa de tres sub-llenguatges ordenats de menys a més expressivitat: OWL Lite, OWL DL i OWL Full. Aquest llenguatge és aprovat pel World Wide Web Consortium (W3C) i ha despertat l'interès acadèmic, mèdic i comercial."

2.3.1 *Components*

OWL és un llenguatge que ens permet representar una ontologia i, per tant, tal i com s'ha vist en l'apartat 2.2.3, ha de permetre formalitzar cadascun dels components esmentats d'aquesta.

Abans d'enumerar els components específics que formalitza OWL, es defineixen a grans trets els conceptes generals.

Instància (*individual*)	Defineix els objectes específics del domini. Una instància pot formar part de diverses classes.
Classe (*set*)	Conjunt d'instàncies amb unes característiques comunes. Una classe ha de poder contenir instàncies.
Subclasse (*subset*)	Defineix especialitzacions de classes.
Propietat	Defineix una relació entre instàncies o entre una instància i un valor (*data*). Existeixen propietats de tres tipus: • *Object Property* • *Datatype Property* • *Annotation Property*
Subpropietat	Defineix especialitzacions de propietats.
Domini	Indica les instàncies a les quals la propietat pot ser aplicada.
Rang	Indica les instàncies que la propietat pot tenir com a valor.

Taula 2. Conceptes OWL.

Tenint en compte les especificacions de la W3C, els components d'OWL 2 són els següents [21].

i) Noms, prefixos i anotacions

Els noms a OWL 2 són IRI's[2], sovint escrits com a `prefix:localname`, on `prefix:` és un nom de prefix que s'expandeix a un IRI i `localname` és la resta del nom. Els noms de prefix estàndard a OWL 2 són:

Nom del prefix	Expansió
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
òliba:	http://www.w3.org/2002/07/owl#
xsd:	http://www.w3.org/2001/XMLSchema#

Taula 3. Prefixos OWL.

Utilitzem convencions de notació a les taules següents[3]:

Lletres	Significat	Lletres	Significat	Lletres	Significat	Lletres	Significat
C	expressió de classe	CN	nom de la classe	D	rang de dades	DN	nom del tipus de dades
P	expressió de propietat d'objecte	PN	nom de la propietat de l'objecte	R	propietat de dades	A	propietat d'anotació
a	individual	aN	nom individual	_: a	individu anònim (una etiqueta de node en blanc)	v	literal
n	enter no negatiu [4]	f	facet	ON	nom d'ontologia	U	IRI
s	IRI o individu anònim	t	IRI, individu anònim o literal	p	nom del prefix	_: x	node en blanc
(a_1 ... a_n)	Llista RDF						

Taula 4. Notació taules OWL.

[2] El *Internationalized Resource Identifier* va ser definit com un nou estàndard d'Internet per estendre l'esquema existent *Uniform Resource Identifier* (URI).

[3] Tot allò anterior pot tenir subíndexs.

[4] com a abreviatura de "n" ^^ xsd: nonNegativeInteger

ii) Construccions i axiomes

Per a una ontologia OWL 2 DL existeixen algunes restriccions globals sobre els axiomes.

A les taules següents, la primera columna especifica el nom de l'expressió, la segona columna la sintaxi funcional i la tercera proporciona la tripleta semàntica RDF[5] en sintaxi *Turtle*[6].

a. Expressions de classe

Classes predefinides

Language Feature	Functional Syntax	RDF Syntax
named class	CN	CN
universal class	owl:Thing	owl:Thing
empty class	owl:Nothing	owl:Nothing

Taula 5. Classes predefinides OWL.

Connectius booleans i enumeració d'individus

Language Feature	Functional Syntax	RDF Syntax
intersection	ObjectIntersectionOf(C_1 … C_n)	_:x rdf:type owl:Class. _:x owl:intersectionOf (C_1 … C_n).
union	ObjectUnionOf(C_1 … C_n)	_:x rdf:type owl:Class. _:x owl:unionOf (C_1 … C_n).
complement	ObjectComplementOf(C)	_:x rdf:type owl:Class. _:x owl:complementOf C.
enumeration	ObjectOneOf(a_1 … a_n)	_:x rdf:type owl:Class. _:x owl:oneOf (a_1 … a_n).

Taula 6. Connectius booleans i enumeració d'individus OWL.

[5] Una tripleta semàntica o terna semàntica és l'entitat atòmica de dades en el model de dades *Resource Description Framework* (RDF). Com el seu nom indica, una terna és un conjunt de tres entitats que codifica una declaració sobre dades semàntiques en forma d'expressions subjecte-predicat-objecte.

[6] Turtle (també conegut com a Llenguatge Concís de Tripletes RDF o Terse RDF Triple Language en anglès) és un format o una sintaxi textual que permet serialitzar RDF, similar a SPARQL.

Object Property Restrictions

Language Feature	Functional Syntax	RDF Syntax
universal	ObjectAllValuesFrom(P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:allValuesFrom C
existential	ObjectSomeValuesFrom(P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:someValuesFrom C
individual value	ObjectHasValue(P a)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:hasValue a.
local reflexivity	ObjectHasSelf(P)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:hasSelf "true"^^xsd:boolean.
exact cardinality	ObjectExactCardinality(n P)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:cardinality n.
qualified exact cardinality	ObjectExactCardinality(n P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:qualifiedCardinality n. _:x owl:onClass C.
maximum cardinality	ObjectMaxCardinality(n P)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:maxCardinality n.
qualified maximum cardinality	ObjectMaxCardinality(n P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:maxQualifiedCardinality n. _:x owl:onClass C.
minimum cardinality	ObjectMinCardinality(n P)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:minCardinality n.
qualified minimum cardinality	ObjectMinCardinality(n P C)	_:x rdf:type owl:Restriction. _:x owl:onProperty P. _:x owl:minQualifiedCardinality n. _:x owl:onClass C.

Taula 7. *Object Property Restrictions* OWL.

Data Property Restrictions

Language Feature	Functional Syntax	RDF Syntax
universal	DataAllValuesFrom(R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:allValuesFrom D.
existential	DataSomeValuesFrom(R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:someValuesFrom D.
literal value	DataHasValue(R v)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:hasValue v.
exact cardinality	DataExactCardinality(n R)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:cardinality n.
qualified exact cardinality	DataExactCardinality(n R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:qualifiedCardinality n. _:x owl:onDataRange D.
maximum cardinality	DataMaxCardinality(n R)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:maxCardinality n.
qualified maximum cardinality	DataMaxCardinality(n R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:maxQualifiedCardinality n. _:x owl:onDataRange D.
minimum cardinality	DataMinCardinality(n R)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:minCardinality n.
qualified minimum cardinality	DataMinCardinality(n R D)	_:x rdf:type owl:Restriction. _:x owl:onProperty R. _:x owl:minQualifiedCardinality n. _:x owl:onDataRange D.

Taula 8. *Data Property Restrictions* OWL.

b. Propietats

Object Property Expressions

Language Feature	Functional Syntax	RDF Syntax
named object property	PN	PN
universal object property	owl:topObjectProperty	owl:topObjectProperty
empty object property	owl:bottomObjectProperty	owl:bottomObjectProperty
inverse property	ObjectInverseOf(PN)	_:x owl:inverseOf PN

Taula 9. *Object Property Expressions* OWL.

Data Property Expressions

Language Feature	Functional Syntax	RDF Syntax
named data property	R	R
universal data property	owl:topDataProperty	owl:topDataProperty
empty data property	owl:bottomDataProperty	owl:bottomDataProperty

Taula 10. *Data Property Expressions* OWL.

c. Particulars i literals

Language Feature	Functional Syntax	RDF Syntax
named individual	aN	aN
anonymous individual	_:a	_:a
literal (datatype value)	"abc"^^DN	"abc"^^DN

Taula 11. Particulars i literals OWL.

d. Rang de dades

Language Feature	Functional Syntax	RDF Syntax
named datatype	DN	DN
data range complement	DataComplementOf(D)	_:x rdf:type rdfs:Datatype. _:x owl:datatypeComplementOf D.
data range intersection	DataIntersectionOf(D_1…D_n)	_:x rdf:type rdfs:Datatype. _:x owl:intersectionOf (D_1…D_n).
data range union	DataUnionOf(D_1…D_n)	_:x rdf:type rdfs:Datatype. _:x owl:unionOf (D_1…D_n).
literal enumeration	DataOneOf(v_1 … v_n)	_:x rdf:type rdfs:Datatype. _:x owl:oneOf (v_1 … v_n).
datatype restriction	DatatypeRestriction(DN f_1 v_1 … f_n v_n)	_:x rdf:type rdfs:Datatype. _:x owl:onDatatype DN. _:x owl:withRestrictions (_:x_1 ... _:x_n). _:x_j f_j v_j. j=1…n

Taula 12. Rang de dades OWL.

e. Axiomes

Class Expression Axioms

Language Feature	Functional Syntax	RDF Syntax
subclass	SubClassOf(C_1 C_2)	C_1 rdfs:subClassOf C_2.
equivalent classes	EquivalentClasses(C_1 … C_n)	C_j owl:equivalentClass C_{j+1}. j=1…n-1
disjoint classes	DisjointClasses(C_1 C_2)	C_1 owl:disjointWith C_2.
pairwise disjoint classes	DisjointClasses(C_1 … C_n)	_:x rdf:type owl:AllDisjointClasses. _:x owl:members (C_1 … C_n).
disjoint union	DisjointUnionOf(CN C_1 … C_n)	CN owl:disjointUnionOf (C_1 … C_n).

Taula 13. *Class Expression Axioms* OWL.

Object Property Axioms

Language Feature	Functional Syntax	RDF Syntax
subproperty	SubObjectPropertyOf(P_1 P_2)	P_1 rdfs:subPropertyOf P_2.
property chain inclusion	SubObjectPropertyOf(ObjectProperty Chain(P_1 … P_n) P)	P owl:propertyChainAxiom (P_1 … P_n).
property domain	ObjectPropertyDomain(P C)	P rdfs:domain C.
property range	ObjectPropertyRange(P C)	P rdfs:range C.
equivalent properties	EquivalentObjectProperties(P_1 … P_n)	P_j owl:equivalentProperty P_{j+1}. j=1…n-1
disjoint properties	DisjointObjectProperties(P_1 P_2)	P_1 owl:propertyDisjointWith P_2.
pairwise disjoint properties	DisjointObjectProperties(P_1 … P_n)	_:x rdf:type owl:AllDisjointProperties. _:x owl:members (P_1 … P_n).
inverse properties	InverseObjectProperties(P_1 P_2)	P_1 owl:inverseOf P_2.
functional property	FunctionalObjectProperty(P)	P rdf:type owl:FunctionalProperty.
inverse functional	InverseFunctionalObjectProperty(P)	P rdf:type owl:InverseFunctionalProperty.

property		
reflexive property	ReflexiveObjectProperty(P)	P rdf:type owl:ReflexiveProperty.
irreflexive property	IrreflexiveObjectProperty(P)	P rdf:type owl:IrreflexiveProperty.
symmetric property	SymmetricObjectProperty(P)	P rdf:type owl:SymmetricProperty.
asymmetric property	AsymmetricObjectProperty(P)	P rdf:type owl:AsymmetricProperty.
transitive property	TransitiveObjectProperty(P)	P rdf:type owl:TransitiveProperty.

Taula 14. *Object Property Axioms* OWL.

Data Property Axioms

Language Feature	Functional Syntax	RDF Syntax
subproperty	SubDataPropertyOf(R_1 R_2)	R_1 rdfs:subPropertyOf R_2.
property domain	DataPropertyDomain(R C)	R rdfs:domain C.
property range	DataPropertyRange(R D)	R rdfs:range D.
equivalent properties	EquivalentDataProperties(R_1 … R_n)	R_j owl:equivalentProperty R_{j+1}. j=1…n-1
disjoint properties	DisjointDataProperties(R_1 R_2)	R_1 owl:propertyDisjointWith R_2.
pairwise disjoint properties	DisjointDataProperties(R_1 … R_n)	_:x rdf:type owl:AllDisjointProperties. _:x owl:members (R_1 … R_n).
functional property	FunctionalDataProperty(R)	R rdf:type owl:FunctionalProperty.

Taula 15. *Data Property Axioms* OWL.

Datatype Definitions

Language Feature	Functional Syntax	RDF Syntax
datatype definition	DatatypeDefinition(DN D)	DN owl:equivalentClass D.

Taula 16. *Datatype Definitions* OWL.

Assertions

Language Feature	Functional Syntax	RDF Syntax
individual equality	SameIndividual(a_1 … a_n)	a_j owl:sameAs a_{j+1}. j=1…n-1
individual inequality	DifferentIndividuals(a_1 a_2)	a_1 owl:differentFrom a_2.
pairwise individual inequality	DifferentIndividuals(a_1 … a_n)	_:x rdf:type owl:AllDifferent. _:x owl:members (a_1 … a_n).
class assertion	ClassAssertion(C a)	a rdf:type C.
positive object property assertion	ObjectPropertyAssertion(PN a_1 a_2)	a_1 PN a_2.
positive data property assertion	DataPropertyAssertion(R a v)	a R v.
negative object property assertion	NegativeObjectPropertyAssertion(P a_1 a_2)	_:x rdf:type owl:NegativePropertyAssertion. _:x owl:sourceIndividual a_1. _:x owl:assertionProperty P. _:x owl:targetIndividual a_2.
negative data property assertion	NegativeDataPropertyAssertion(R a v)	_:x rdf:type owl:NegativePropertyAssertion. _:x owl:sourceIndividual a. _:x owl:assertionProperty R. _:x owl:targetValue v.

Taula 17. *Assertions* OWL.

f. Declaracions

Language Feature	Functional Syntax	RDF Syntax
class	Declaration(Class(CN))	CN rdf:type owl:Class.
datatype	Declaration(Datatype(DN))	DN rdf:type rdfs:Datatype.
object property	Declaration(ObjectProperty(PN))	PN rdf:type owl:ObjectProperty.
data property	Declaration(DataProperty(R))	R rdf:type owl:DatatypeProperty.
annotation property	Declaration(AnnotationProperty(A))	A rdf:type owl:AnnotationProperty.
named individual	Declaration(NamedIndividual(aN))	aN rdf:type owl:NamedIndividual.

Taula 18. Declaracions OWL.

g. Anotacions

Annotations

Language Feature	Functional Syntax	RDF Syntax
annotation assertion	AnnotationAssertion(A s t)	s A t.
annotation of an axiom where the axiom in RDF is one or more triples of the form si U ti, i.e., with the same predicate U.	AXIOM(Annotation(A t) …)	_:x_i A t. s_i U t_i. … _:x_i rdf:type owl:Axiom. _:x_i owl:annotatedSource s_i. _:x_i owl:annotatedProperty U. _:x_i owl:annotatedTarget t_i.
annotation of an axiom where the axiom in RDF is _:x U t1	AXIOM(Annotation(A t) …)	_:x A t. _:x U t_1. …
annotation of another annotation (the other annotation in RDF starts with s1)	Annotation(Annotation(A t) … A_1 t_1)	_:x A t. s_1 A_1 t_1. … _:x rdf:type owl:Annotation. _:x owl:annotatedSource s_1. _:x owl:annotatedProperty A_1. _:x owl:annotatedTarget t_1.

Taula 19. Anotacions OWL.

Annotation Properties

Language Feature	Functional Syntax	RDF Syntax
named annotation property	A	A
human-readable name	rdfs:label	rdfs:label
human-readable comment	rdfs:comment	rdfs:comment
additional information	rdfs:seeAlso	rdfs:seeAlso
defining agent	rdfs:isDefinedBy	rdfs:isDefinedBy
version information	owl:versionInfo	owl:versionInfo
deprecation	owl:deprecated	owl:deprecated
backwards compatibility	owl:backwardCompatibleWith	owl:backwardCompatibleWith
incompatibility	owl:incompatibleWith	owl:incompatibleWith
prior version	owl:priorVersion	owl:priorVersion

Taula 20. *Annotation Properties* OWL.

h. Ontologies

Language Feature	Functional Syntax	RDF Syntax
OWL ontology (importing)[7][8]	Ontology([ON [U]] Import(ON_1)... Annotation(A t) ...)	ON rdf:type owl:Ontology. [ON owl:versionIRI U.] ON owl:imports ON_1. ... ON A t. ...
prefix declaration[9]	Prefix(p=U)	@prefix p U.

Taula 21. Ontologies OWL.

[7] [] representa construccions opcionals

[8] A la sintaxi RDF, s'utilitza _: x en lloc d'ON si no hi ha un nom d'ontologia.

[9] La sintaxi RDF es troba a Turtle, altres serialitzacions RDF poden variar.

2.4 Metodologies

Finalment, en tot projecte és imprescindible determinar els protocols i les pautes adequats per desenvolupar-lo correctament. Tenint en compte els conceptes explicats, s'enumeren i detallen els procediments que s'utilitzaran per a la construcció tant de l'ontologia com del sistema expert.

2.4.1 *Ontology Engineering*

L'Enginyeria ontològica és un camp de les ciències de la computació i ciències de la informació que estudia els mètodes i metodologies per construir esquemes conceptuals (ontologia): aquesta correspon a la representació formal d'un grup de conceptes dins d'un domini i de les relacions entre aquests conceptes. Una representació a gran escala de conceptes abstractes com accions, temps, objectes físics i creences podria ser un exemple d'enginyeria ontològica [7].

Així doncs, té com a meta fer explícit el coneixement contingut dins de les aplicacions de programari i els procediments dins d'empreses i negocis per obtenir un domini particular. L'enginyeria ontològica ofereix les adreces per resoldre problemes d'operació interns que tinguin obstacles semàntics, per exemple, els obstacles relacionats amb la definició dels termes de negoci i les classes de programari. L'enginyeria ontològica és un conjunt de tasques relacionades amb el desenvolupament d'un esquema conceptual d'un domini particular [8].

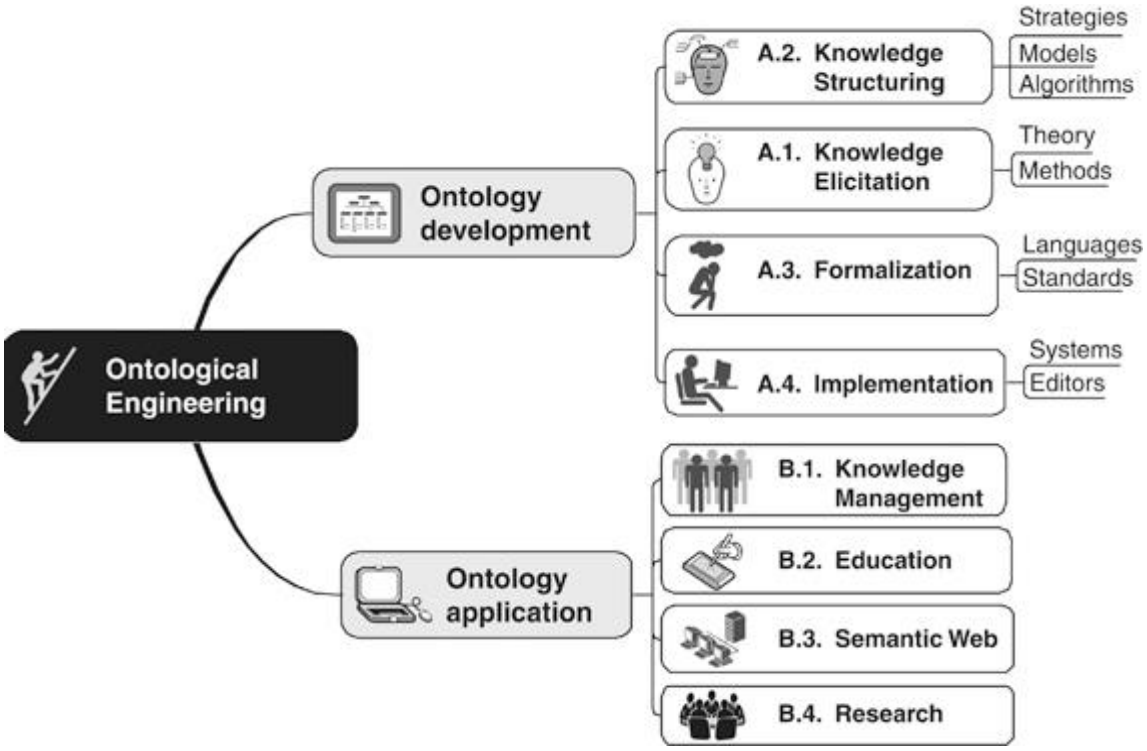


Figura 2. Ontology Engineering.

A més, existeix una àmplia varietat de metodologies per desenvolupar una ontologia.

Una de les més conegudes és *"ONTOLOGY DEVELOPMENT 101"* proposta per la Universitat d'Stanford EEUU, on els principals passos a seguir per a la creació d'una ontologia es defineixen com: (1) determinar el domini i àmbit de l'ontologia, (2) determinar la intenció d'ús de l'ontologia, (3) reutilitzar ontologies o vocabularis controlats existents, (4) enumerar els termes importants del domini, (5) definir la jerarquia de classes i (6) crear les instàncies.

Aquesta última es tracta d'una aproximació a la metodologia que finalment s'usa per desenvolupar el projecte; doncs avui dia són diversos els llibres disponibles sota el nom *Ontology Engineering*, on consultar les bases òptimes per definir una ontologia dins del marc d'un projecte d'enginyeria.

L'elecció d'aquesta es veu recolzada per la familiarització obtinguda durant la realització de l'assignatura d'Intel·ligència Artificial cursada a la URV, així com que es tracta d'una metodologia simple però eficient.

A la Figura 3 s'observen els passos que es realitzaran en el desenvolupament del projecte i que es detallen a continuació.

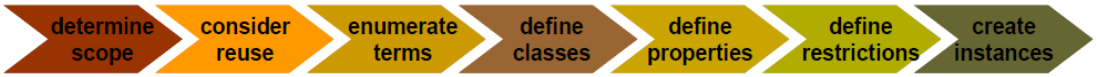


Figura 3. Metodologia *Ontology Engineering*.

Determinació del domini i de l'abast	Determinar resposta a les següents qüestions: • Quin és el domini que cobrirà l'ontologia? • Per a què serà usada? • A quin tipus de preguntes ha de donar resposta (*competency qüestions*)?
Reutilització d'ontologies	Considerar l'ús d'altres ontologies per a: • Estalviar esforç. • Interactuar amb eines que utilitzen altres ontologies. • Utilitzar ontologies que han estat validades.
Enumeració de termes	Determinar resposta a les següents qüestions: • De quins termes s'ha de parlar? • Quines són les propietats d'aquests termes?

	• Que es vol expressar sobre aquests termes?
Definició de classes	Definir cada classe. Una classe és un concepte del domini i està formada per una col·lecció d'elements amb propietats similars. La definició de totes les classes construeix una taxonomia jeràrquica (on intervenen relacions de subclass-superclass).
Definició de propietats	Definir cada slot. Els slots descriuen els atributs d'una instància d'una classe i les relacions amb altres instàncies. Una subclasse hereta tots els slots de la seva superclasse. En el cas de tenir diverses superclasses, hereta els slots de totes elles.
Definició de restriccions	Definir els valors possibles dels slots. Veure en l'apartat 2.3.1 els components del llenguatge ontològic OWL.
Creació d'instàncies	Crear instàncies de les classes definides, assignant valor als slots definits.

Taula 22. Metodologia *Ontology Engineering*.

Tanmateix, la construcció d'una ontologia es caracteritza per ser un procés iteratiu, ja que durant el procés de desenvolupament sovint s'han d'ajustar elements ja definits. Per tant, l'elaboració d'una ontologia no seguirà un procés lineal. Tot i això, la documentació d'aquest procés es basarà en els punts definits en la figura anterior.

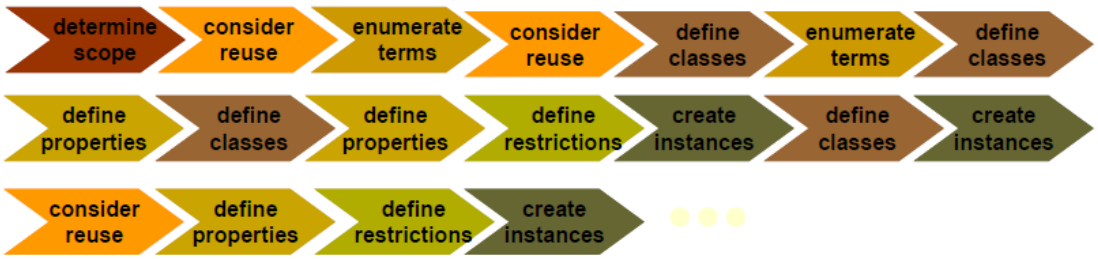


Figura 4. Metodologia *Ontology Engineering* II.

2.4.1.1 Top-down

A més, la construcció de l'ontologia es desenvoluparà utilitzant la metodologia de tipus top-down, la qual defineix el problema des dels aspectes més globals cap als més específics.

Això significa que l'estratègia d'identificació dels conceptes s'inicia en aquells més abstractes i es desenvolupa fins als més concrets, és a dir, primer classes i després subclasses.

2.4.2 *Raonament deductiu*

Per altra banda, tal i com s'ha vist en l'apartat 2.1.3 Algorismes d'inferència, el motor d'inferència d'un sistema expert actua de forma diferent segons la seva implementació.

En aquest cas, el tipus de raonament del sistema implementat és d'encaminament cap endavant, ja que per naturalesa un sistema de diagnòstic analitza uns fets per tal d'extreure unes conclusions, el que descriu un sistema purament deductiu.

En el sistema proposat els fets estan formats per la informació aportada pel pacient (simptomatologia presentada respecte certs aliments) i les conclusions corresponen al diagnòstic extret (detecció, si s'escau, d'una intolerància).

2.5 Eines de desenvolupament

En aquest apartat es descriuen les principals eines de desenvolupament utilitzades en la realització d'aquest projecte. A més, en el **Diagrama 2** es mostra la correspondència entre els components d'un sistema expert il·lustrats en el **Diagrama 1** i les eines que s'esmenten a continuació. Abans però, es justifiquen els motius de l'elecció d'aquestes.

Per una banda, Protégé és l'estat de l'art en eines per a la creació d'ontologies en OWL, que com s'introduïa a l'apartat 2.2.4, és el llenguatge estàndard de representació d'ontologies.

Per altra banda, l'elecció de la utilització de l'API Jena per a explorar i modificar l'ontologia des d'un entorn de treball Java, es deu a la recerca sobre el rendiment de les diverses opcions existents. Tal i com es conclou en *Estudio del manejo de ontologías para la monitorización de pacientes*[20], el rendiment pel que fa al temps d'execució o a l'ús de la memòria RAM és molt superior utilitzant Jena que utilitzant les eines i el raonador que inclou OWLAPI, l'altra principal API per desenvolupar la tasca esmentada.

Per conseqüència, per tal de realitzar consultes complexes sobre les instàncies de l'ontologia s'usa el llenguatge estàndard de consulta sobre ontologies SPARQL, ja que forma part de l'estructura de Jena tal i com s'aprecia en la Figura 6. Apache Jena.

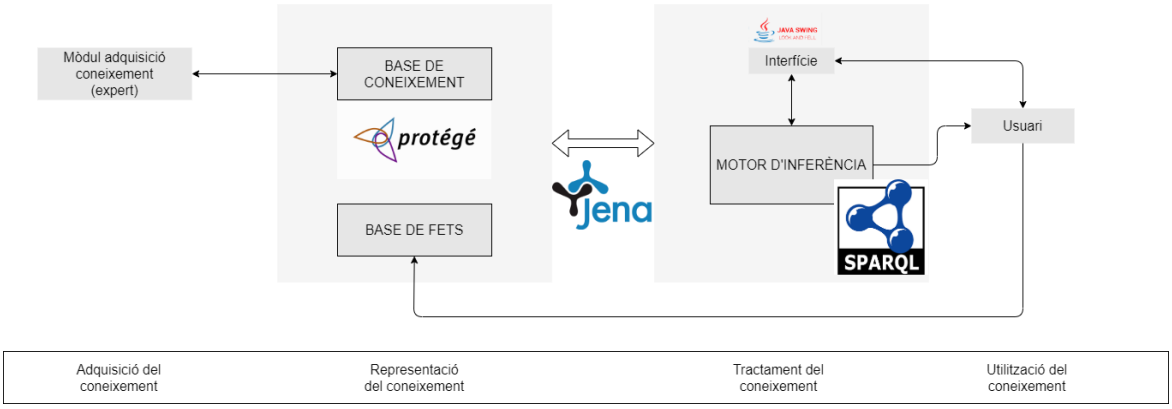


Diagrama 2. Eines de desenvolupament.

2.5.1 *Protégé Framework*

Protégé és un editor d'ontologies i bases de coneixement gratuït i obert. La seva arquitectura es pot adaptar per construir aplicacions basades en ontologies tant simples com complexes. A més, els desenvolupadors poden integrar la sortida de Protégé amb sistemes de regles o altres raonadors per construir una àmplia gamma de sistemes intel·ligents.

El framework admet plenament les últimes especificacions del llenguatge d'ontologia web OWL 2 i les especificacions RDF del World Wide Web Consortium. Es basa en Java, és extensible i proporciona un entorn plug-and-play que el converteix en una base flexible per al desenvolupament ràpid de prototips i aplicacions. A més, suporta els següents formats: RDF / XML, Turtle, OWL / XML, OBO, entre d'altres.

La Figura 5 mostra un exemple de com es visualitza una ontologia de robòtica [27] utilitzant aquest entorn de treball.

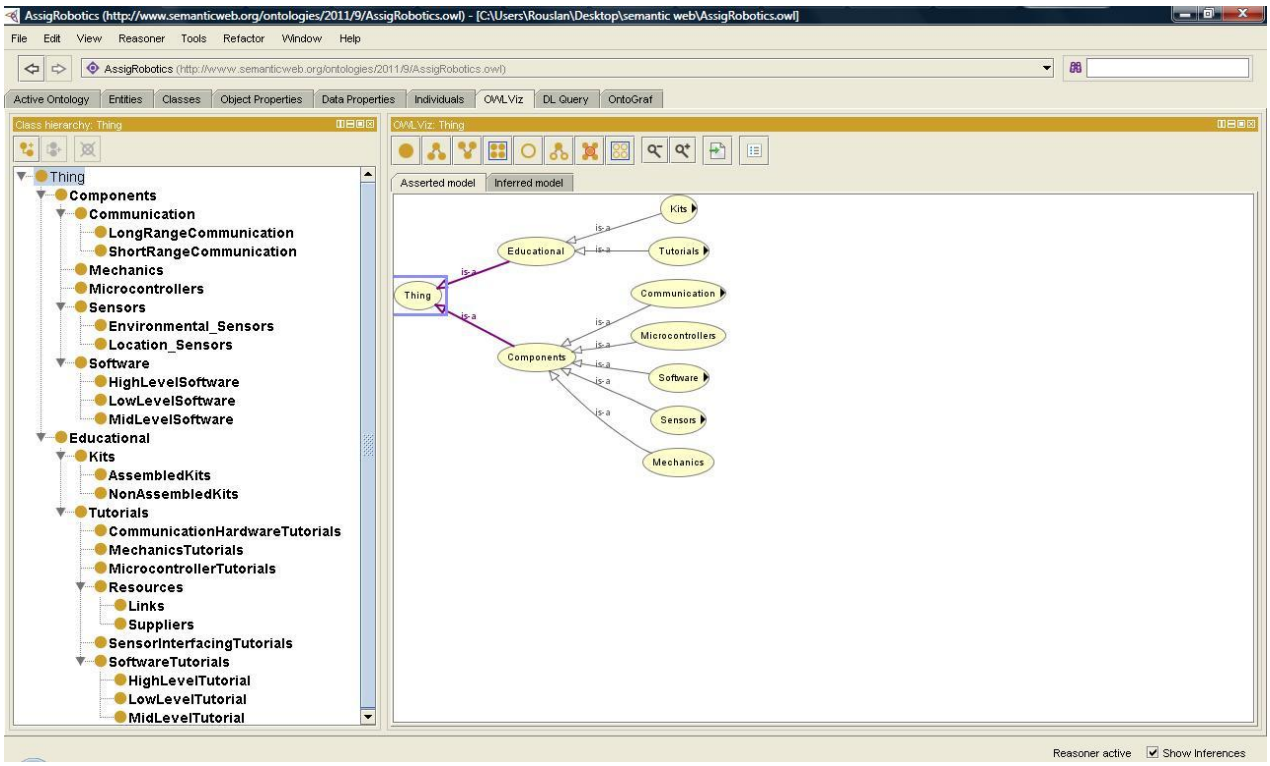


Figura 5. Exemple d'ús de Protégé.

2.5.2 *Apache Jena Semantic Web Framework*

Jena és un framework de software lliure desenvolupat per HP Labs amb la finalitat de ser utilitzat en el desenvolupament d'aplicacions Java relacionades amb les Webs semàntiques i/o ontologies. Entre les característiques del framework cal destacar la capacitat de gestionar, emmagatzemar i realitzar consultes sobre tot tipus d'ontologies. A més, suporta diferents llenguatges, com per exemple RDF, DAML o OWL i presenta un motor de consulta ARQ que suporta SPARQL.

Pel que fa a la seva estructura, Jena està composta de diversos mòduls, tal i com es pot apreciar a la Figura 6.

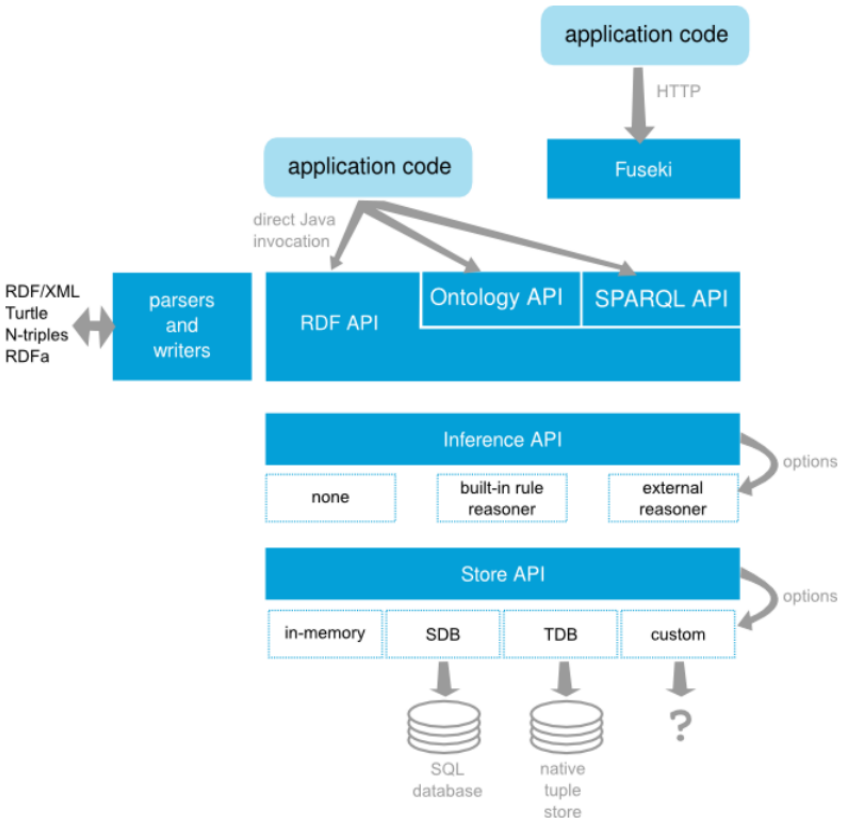


Figura 6. Apache Jena.

- RDF API. S'utilitza per manipular els RDF *graph*.
- Ontology API. S'utilitza per treballar amb les ontologies representades en RDF i OWL.
- SPARQL API. S'utilitza per poder realitzar consultes sobre les ontologies.
- Inference API. S'utilitza per inferir nou coneixement sobre representacions RDF.
- Store API. S'encarrega de l'emmagatzematge de les ontologies.

2.5.3 *SPARQL*

SPARQL (acrònim recursiu de SPARQL Protocol and RDF Query Language) és un llenguatge estandarditzat pel World Wide Web Consortium (W3C) per a la consulta i manipulació de bases de dades o arxius RDF. La versió actual és la 1.1, publicada el 2013.[22]

SPARQL és un llenguatge molt complet, amb una gran capacitat expressiva. L'estructura bàsica, però, és simple, i està basada en dues clàusules: SELECT i WHERE, que tenen una certa semblança amb les del mateix nom en el llenguatge SQL per a bases de dades relacionals. Les consultes s'executen en un processador de consultes SPARQL i aquest accedeix a un conjunt de dades RDF per tal d'obtenir la resposta a la consulta plantejada.[9]

2.5.3.1 Gramàtica SPARQL

Respecte a la gramàtica, una consulta SPARQL és una cadena de caràcters Unicode[25] en el llenguatge definit per les següents paraules clau.

BASE	SELECT	ORDER BY	FROM	GRAPH	STR	isURI
PREFIX	CONSTRUCT	LIMIT	FROM NAMED	OPTIONAL	LANG	isIRI
	DESCRIBE	OFFSET	WHERE	UNION	LANGMATCHES	isLITERAL
	ASK	DISTINCT		FILTER	DATATYPE	REGEX
		REDUCED		a	BOUND	true
					sameTERM	false

Figura 7. *Keywords in* SPARQL.

on es distingeixen els següents conceptes:

2.5.3.1.1 Referències IRI

La regla gramatical ***IRIref :: = IRI_REF | PrefixedName*** designa un conjunt d'IRI [RFC3987], els quals són una generalització dels URI [RFC3986] i són totalment compatibles amb els URI i els URL. A més, les referències IRI poden ser relatives o absolutes i són designades per la producció *IRI_REF* , on els delimitadors '<' i '>' no formen part de la referència IRI.

Sobre les IRI podem aplicar les següents paraules clau:

BASE	Defineix la IRI base utilitzada per resoldre els IRI relatius i aquests poden provenir d'un document encapsulant, com ara un sobre que utilitza el protocol SOAP amb una directiva *xml:base*.
PREFIX	Associa una etiqueta de prefix amb una IRI i permet utilitzar un nom prefixat, el qual està format pel prefix i una part local, separades per dos punts " :". D'aquesta forma, el nom prefixat s'assigna a un nou IRI que està format per la concatenació de l'IRI associat amb el prefix i la part local.

Taula 23. IRI *References in* SPARQL.

2.5.3.1.2 Formes de consulta

SPARQL té quatre formes de consulta. Aquests formularis de consulta utilitzen les solucions de la concordança de patrons per formar conjunts de resultats o gràfics RDF.

Els formularis de consulta són:

SELECT	Retorna totes les variables o un subconjunt de les variables lligades en una concordança de patró de consulta.
CONSTRUCT	Retorna un gràfic RDF construït substituint variables en un conjunt de plantilles triples.
DESCRIBE	Retorna un gràfic RDF que descriu els recursos trobats.
ASK	Retorna un booleà que indica si un patró de consulta coincideix o no.

Taula 24. *Query forms in* SPARQL.

2.5.3.1.3 Modificadors

Els patrons de consulta generen una col·lecció no ordenada de solucions, sent cada solució una funció parcial, des de variables fins a termes RDF. Aquestes solucions es tracten després com a una seqüència solucions, inicialment en cap ordre específic; els modificadors de seqüència s'apliquen per crear una altra seqüència. Finalment, aquesta darrera seqüència s'utilitza per generar un dels resultats d'un formulari de consulta SPARQL.

Alguns dels principals modificadors són els següents:

ORDER BY	Ordena les solucions segons un criteri donat.
LIMIT	Limita el nombre de solucions.
OFFSET	Controla d'on parteixen les solucions en la seqüència general de solucions.
DISTINCT	Assegura que les solucions de la seqüència siguin úniques.
REDUCED	Permet l'eliminació d'algunes solucions no úniques.

Taula 25. *Solution sequence modifiers in* SPARQL.

2.5.3.1.4 Especificació de dades i patrons

Una sentència SPARQL pot especificar el conjunt de dades RDF sobre el que realitzarà la consulta a partir d'una referència i, també, indicar quin tipus de representació vol utilitzar per al resultat.

Per tal d'especificar la procedència de les dades o els patrons que s'aplicaran, s'utilitzen les següents clàusules:

FROM	El conjunt de dades resultant inclou un gràfic per defecte que consisteix en la combinació RDF dels gràfics referenciats a la clàusula.

FROM NAMED	El resultat inclou un conjunt de parells (IRI, gràfic) per cada clàusula.
WHERE	Indica els patrons a aplicar sobre el conjunt de dades seleccionat.

Taula 26. *Specifying RDF Datasets and patterns in* SPARQL.

2.5.3.1.5 Patrons

SPARQL es basa al voltant de la concordança de patrons de gràfics. Es poden formar patrons de gràfics més complexos combinant patrons més petits de diverses maneres:

- Patrons bàsics de gràfics, on han de coincidir un conjunt de patrons triples.
- Patró de gràfics de grup, on un conjunt de patrons de gràfics ha de coincidir.
- Patrons de gràfics opcionals, on els patrons addicionals poden estendre la solució.
- Patró de gràfics alternatius, on s'intenten dos o més patrons possibles.
- Patrons als gràfics amb nom, on els patrons es fan coincidir amb els gràfics amb nom.

Algunes de les clàusules que permeten controlar aspectes sobre els patrons esmentats són les següents:

FILTER	Restringeix les solucions de manera que totes han de complir els patrons que s'inclouen dins de la clàusula.
OPTIONAL	Els patrons bàsics de gràfics permeten a les aplicacions fer consultes on ha de coincidir tot el patró perquè existeixi una solució, i on cada variable de la consulta està lligada a un terme RDF. Tot i això, no es poden assumir estructures completes i regulars en tots els gràfics RDF. Per això, és útil poder realitzar consultes que permetin afegir més informació a la solució, en cas que existeixi, però que no rebutgin la solució perquè alguna part del patró de consulta no coincideix. La concordança opcional proporciona aquesta facilitat: si la part opcional no coincideix, no crea cap vinculació però no elimina la solució.
UNION	Proporciona un mitjà per combinar patrons de gràfics perquè un dels diversos patrons de gràfics alternatius pugui coincidir.

Taula 27. *Patterns in* SPARQL.

2.5.3.1.6 Operadors

La gramàtica SPARQL identifica un conjunt d'operadors que s'utilitzen per construir restriccions. Les taules següents mostren alguns d'aquests operadors i associen cadascuna d'aquestes produccions gramaticals amb els operands adequats i una funció d'operador definida per XQuery 1.0, XPath 2.0[26] o SPARQL. A la pràctica, en seleccionar la definició d'operador per a un conjunt de paràmetres determinat, s'aplica la definició amb els paràmetres més específics.

Operador	Tipus (A)	Funció	Tipus de resultat
! A	xsd: booleà (EBV)	fn: no (A)	xsd: booleà
+ A	numèric	op: numèric-unari-plus (A)	numèric
- A	numèric	op: numèric-unari-menys (A)	numèric
BOUND (A)	variable	lligat (A)	xsd: booleà
isIRI (A) **isURI (A)**	Terme RDF	isIRI (A)	xsd: booleà
isBLANK (A)	Terme RDF	isBlank (A)	xsd: booleà
isLITERAL (A)	Terme RDF	isLiteral (A)	xsd: booleà
STR (A)	literal	str (A)	literal simple
STR (A)	IRI	str (A)	literal simple
LANG (A)	literal	lang (A)	literal simple
DATATYPE (A)	literal escrit	tipus de dades (A)	IRI
DATATYPE (A)	literal simple	tipus de dades (A)	IRI

Taula 28. *Operators in* SPARQL.

Operador	Tipus (A)	Tipus (B)	Funció	Tipus de resultat
A = B	Terme RDF	Terme RDF	RDFterm-igual (A, B)	xsd: booleà
A ! = B	Terme RDF	Terme RDF	fn: no (RDFterm-igual (A, B))	xsd: booleà
sameTERM (A)	Terme RDF	Terme RDF	sameTerm (A, B)	xsd: booleà
langMATCHES (A, B)	literal simple	literal simple	langMatches (A, B)	xsd: booleà
REGEX (CORDA, PATRÓ)	literal simple	literal simple	fn: coincideix (STRING, PATTERN)	xsd: booleà

Taula 29. *Operators in* SPARQL II.

2.5.3.2 Exemples

Finalment, s'aporten algunes consultes realitzades sobre l'ontologia pròpia per tal d'exemplificar alguns dels conceptes bàsics definits anteriorment.

En el següent exemple es mostra una consulta bàsica on es seleccionen els identificadors de tots els pacients registrats en el sistema.

```
  1 PREFIX  dfd:  <http://www.owl-ontologies.com/diagnose_food_diary.owl#>
  2
  3 SELECT  ?hasID
  4 WHERE
  5   { ?x  dfd:hasID  ?hasID }
---------------
| hasID       |
===============
| "33333333C" |
| "22222222B" |
| "11111111A" |
---------------
```

Figura 8. Exemple: Consulta 1.

En el següent exemple es mostra una consulta on es seleccionen l'identificador i la dieta únicament dels pacients que estan realitzant alguna dieta.

```
  1 PREFIX  :     <http://www.owl-ontologies.com/diagnose_food_diary.owl#>
  2
  3 SELECT  ?id ?diet
  4 WHERE
  5   { ?x  :hasID    ?id ;
  6         :hasDiet  ?diet
  7   }
------------------------
| id          | diet   |
========================
| "22222222B" | :Vegan |
------------------------
```

Figura 9. Exemple: Consulta 2.

En el següent exemple, en canvi, es mostra una consulta on es mostren els identificadors de tots els pacients registrats i, en cas que existeixi, també la seva dieta.

```
  1 PREFIX  :     <http://www.owl-ontologies.com/diagnose_food_diary.owl#>
  2
  3 SELECT  ?id ?diet
  4 WHERE
  5   { ?x  :hasID  ?id
  6     OPTIONAL
  7       { ?x  :hasDiet  ?diet }
  8   }
------------------------
| id          | diet   |
========================
| "33333333C" |        |
| "11111111A" |        |
| "22222222B" | :Vegan |
------------------------
```

Figura 10. Exemple: Consulta 3.

En el següent exemple es mostra una consulta on es selecciona la dieta d'un pacient a partir d'un identificador concret.

```
  1 PREFIX  :     <http://www.owl-ontologies.com/diagnose_food_diary.owl#>
  2
  3 SELECT  ?diet
  4 WHERE
  5   { ?x  :hasID    "22222222B" ;
  6         :hasDiet  ?diet
  7   }
----------
| diet   |
==========
| :Vegan |
----------
```

Figura 11. Exemple: Consulta 4.

3 Construcció de l'ontologia

Per desenvolupar el projecte i construir un sistema expert cal implementar cadascuna de les parts que el formen: (1) base de coneixement, (2) memòria de treball i (3) motor d'inferència, tal i com s'ha detallat en l'apartat 2.1.1.

Així doncs, el primer que cal construir és la base de coneixement que, en aquest cas, es tracta d'una ontologia. En aquest apartat es detallen tots els passos del procés.

A més, com s'ha descrit en l'apartat 2.4.1, la construcció d'aquesta ontologia es basa en la metodologia esmentada i està formada pels passos següents: (1) Definició i domini de l'abast i plantejament de las *comepetency questions*, (2) Reutilització d'ontologies, (3) Enumeració de termes, (4) Definició de classes, (5) Definició de propietats (tant les *Object Property* com les *Data Property*), (6) Definició de restriccions i (7) Creació d'instàncies.

3.1 Definició del domini i de l'abast

En primer lloc es planteja el domini, el propòsit i l'abast de l'ontologia. Per altra banda, es crea el projecte mitjançant Protégé [32]. L'arxiu pren el nom de *diagnosi_food_diary.owl* i la seva IRI queda definida per *http://www.owl-ontologies.com/diagnose_food_diary.owl*.

Domini	Representació de la dieta i la simptomatologia de pacients d'intoleràncies alimentàries.
Propòsit	Diagnosticar i/o millorar el diagnòstic dels pacients.
Abast	Determinar si un pacient presenta probabilitats de patir una intolerància.
	Determinar quin tipus de nutrients són els causants d'una possible intolerància.
	Aproximar quina intolerància presenta un alt percentatge de patir el pacient.
	Recomanar alguna directriu sobre la dieta del pacient per millorar la seva qualitat de vida basant-se en el diagnòstic.

Taula 30. Domini i abast.

3.1.1 *Competency questions*

Per tal de definir els requisits i les funcionalitats que el sistema ha d'implementar, la metodologia emprada defineix les anomenades *competency questions*, les quals plantegen preguntes que el resultat dissenyat ha de ser capaç de respondre.

- Quins són els principals tipus d'aliments? Com es classifiquen?
- Quins són els principals tipus de nutrients? Com es classifiquen?
- Quines són les principals simptomatologies? Com es classifiquen?
- Quines són les principals intoleràncies?
- Quines són les principals dietes/tractaments?
- Quins nutrients conté un aliment X?
- Quins aliments/nutrients prohibeix una dieta X?
- Quin tractament s'aplica a la reacció adversa (intolerància) a X tipus d'aliment/nutrient?
- Està el pacient (prèviament a la realització de l'estudi) realitzant algun tipus de dieta específic? Si és així, quina? Quines restriccions té?
- Quins àpats ha realitzat el pacient?
- Quants àpats ha realitzat el pacient?
- Quants aliments ha ingerit el pacient?
- Per quin/s aliment/s i per quina simptomatologia (en cas que es presenti) està composat l'àpat?
- Quina severitat té la simptomatologia experimentada pel pacient (en cas de presentar-se)?
- Quin diagnòstic (intolerància) es conclou (si es pot concloure) de la simptomatologia del pacient?
- Quin tractament (dieta) s'aplica al pacient?

3.2 Reutilització d'ontologies

El següent pas consisteix en considerar la reutilització d'una ontologia ja existent. Tal i com s'havia esmentat en l'apartat 2.2, els objectius principals de l'ús d'ontologies en el desenvolupament de projectes són facilitar la comunicació i compartició d'informació. Conseqüentment, una de les principals característiques i avantatges de les ontologies és la seva reutilització en projectes futurs o la utilització d'ontologies ja existents en els projectes actuals.

En aquest cas, s'ha considerat la reutilització d'una ontologia sobre simptomatologia humana[31], ja que és un camp força complex i s'ha decidit utilitzar el coneixement ja proporcionat per experts.

En la Figura 12 s'aprecia com l'ontologia esmentada ha estat importada al projecte mitjançant la seva direcció IRI. Finalment, l'ontologia queda vinculada permanentment al sistema mitjançant un document RDF/XML.

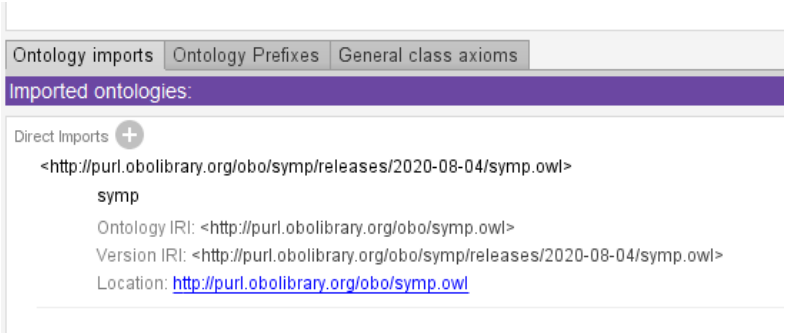


Figura 12. Reutilització d'una ontologia.

Així doncs, un cop importada l'ontologia es poden apreciar totes les classes que es mostren a la Figura 13.

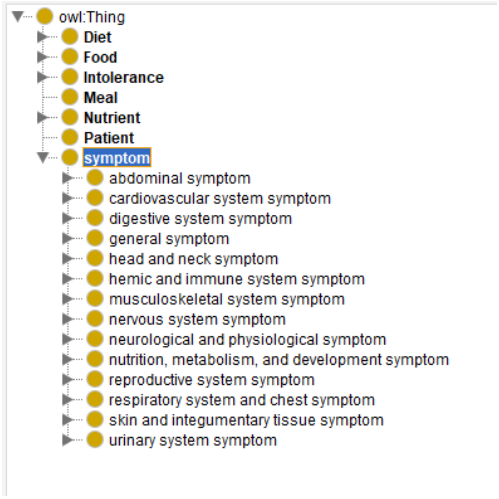


Figura 13. Reutilització d'una ontologia II.

Tanmateix, l'ontologia reutilitzada comprèn un rang molt superior al necessari per desenvolupar el projecte, pel que s'han decidit eliminar alguns camps d'aquesta.

Finalment, després de considerar a partir d'investigació de camp quins són els possibles efectes simptomatològics que deriven d'intoleràncies alimentàries o al·lèrgies lleus, és a dir, després de realitzar un estudi del domini, els camps inclosos en l'ontologia creada són els següents.

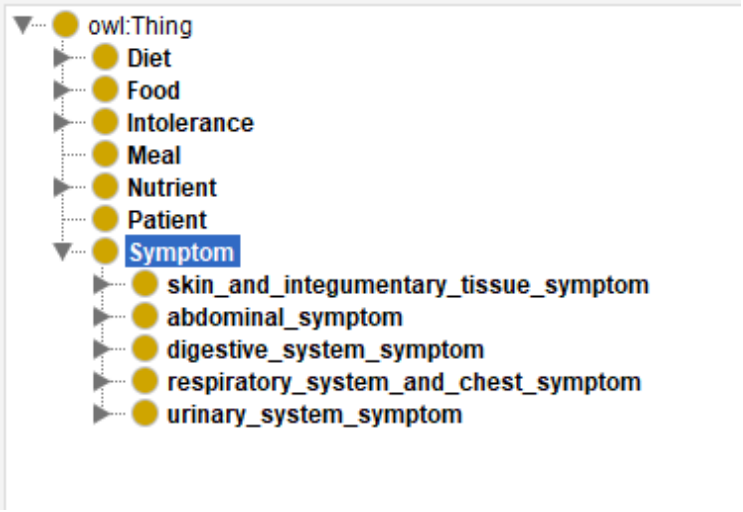


Figura 14. Reutilització d'una ontologia III.

Per tal de poder treballar de forma exclusiva i permanent amb les categories que es desitgen, s'han duplicat les classes seleccionades a l'ontologia pròpia, el que permet no haver d'eliminar totes les classes no desitjades cada vegada que s'executa el *framework* de nou i es llegeix l'ontologia mitjançant la direcció IRI.

3.3 Enumeració de termes

A continuació s'enumeren els aspectes clau que fomenten l'estructura de l'ontologia, a partir de la qual es pot entendre el domini del coneixement a tractar.

Intoleràncía	La intoleràncía alimentària és una reacció adversa de l'organisme davant de la ingesta de determinats aliments, additius i conservants degut a un dèficit digestiu enzimàtic que provoca la formació d'anticossos contra proteïnes de determinades substàncies alimentàries.[16]
Pacient	Cadascun dels individus dels quals es pretén analitzar la simptomatologia per tal de realitzar un diagnòstic i aplicar un tractament.
Símptoma	Un símptoma és la referència subjectiva que dóna un pacient per la percepció o canvi que pot percebre com a anòmal o causat per un estat patològic o malaltia.
Diagnòstic	Identificació dels problemes de salut d'una persona, en aquest cas, d'un pacient.
Tractament	Un tractament és una intervenció mèdica destinada a corregir els símptomes o les causes subjacents causants d'un problema de salut en una persona. Normalment el tractament té lloc després d'un diagnòstic.

Taula 31. Termes de l'ontologia.

3.4 Definició de classes

L'esquema que es mostra en el Diagrama 3. Classes. representa l'estructura principal de l'ontologia.

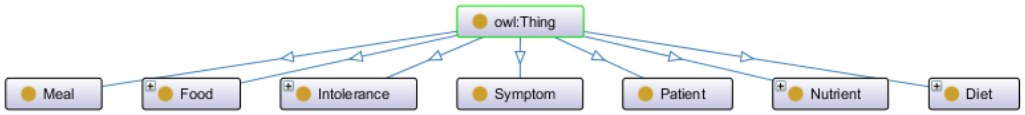


Diagrama 3. Classes.

Aplicant la metodologia *top-down* descrita a l'apartat 2.4.1.1, s'ha construït l'ontologia a partir de les classes principals. A continuació es defineixen cadascuna d'aquestes classes (*sets*).

3.4.1 *Classe Patient*

Classe	Patient
Domini	Informació rellevant del conjunt d'individus sobre els que s'analitzen possibles patologies.

Taula 32. Classe *Patient*.

3.4.2 *Classe Food*

Classe	Food
Domini	Concepte d'aliment, crea una jerarquia de subclasses on cadascuna presenta les característiques bàsiques d'aquesta.

Taula 33. Classe *Food*.

3.4.3 *Classe Nutrient*

Classe	Nutrient
Domini	Concepte de nutrient, crea una jerarquia de subclasses on cadascuna presenta les característiques bàsiques d'aquesta.

Taula 34. Classe *Nutrient*.

3.4.4 *Classe Symptom*

Classe	Symptom
Domini	Concepte de símptoma, crea una jerarquia de subclasses on cadascuna presenta les característiques bàsiques d'aquesta.

Taula 35. Classe *Symptom*.

3.4.5 *Classe Meal*

Classe	Meal
Domini	Informació sobre les característiques d'un àpat concret d'un pacient. Aquest inclou els aliments, la simptomatologia presentada (si és el cas) i la data d'aquest.

Taula 36. Classe *Meal*.

3.4.6 *Classe Intolerance*

Classe	Intolerance
Domini	Concepte d'intoleràncIa, crea una jerarquia de subclasses on cadascuna presenta les característiques bàsiques d'aquesta. Es diagnostiquen una o diverses classes de la jerarquia a cada pacient en funció de l'anàlisi d'àpats realitzat (diagnòstic).

Taula 37. Classe *Intolerance*.

3.4.7 *Classe Diet*

Classe	Diet
Domini	Concepte de dieta, crea una jerarquia de subclasses on cadascuna presenta les característiques bàsiques d'aquesta. S'utilitza com a tractament d'un pacient en funció del diagnòstic realitzat.

Taula 38. Classe *Diet*.

Tal i com s'ha esmentat, les anteriors descriuen les classes principals de l'ontologia. A la Figura 15. Jerarquia de subclasses. es mostren el conjunt de subclasses (*subsets*) que pertanyen a cadascuna de les classes principals descrites.

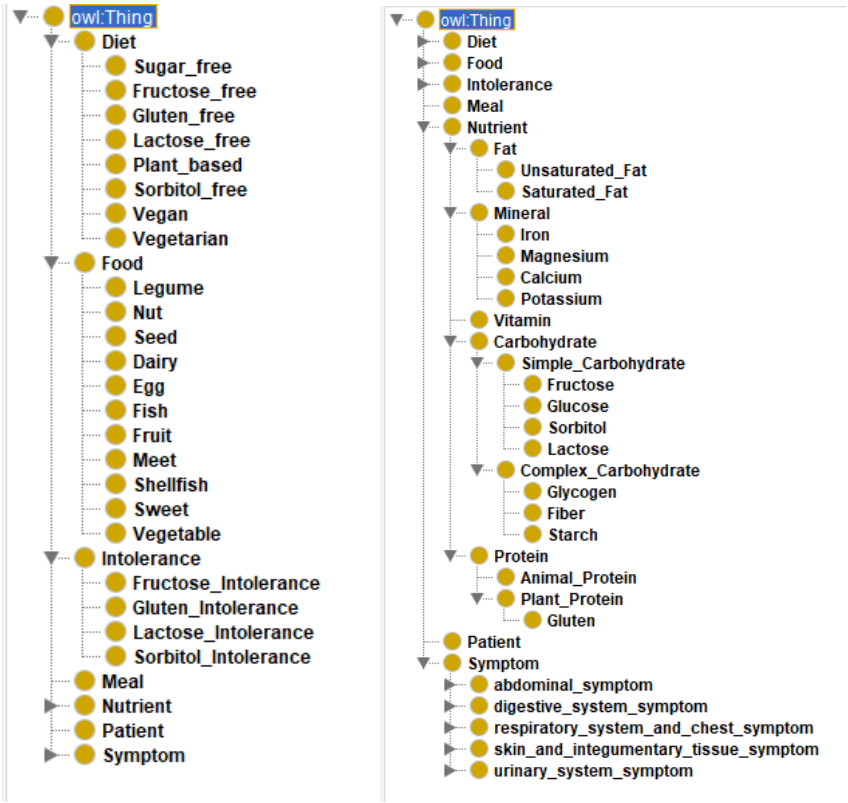


Figura 15. Jerarquia de subclasses.

Finalment, a la Figura 16. Jerarquia de la classe *Symptom*. es mostra la jerarquia de classes de l'ontologia reutilitzada en l'apartat anterior.

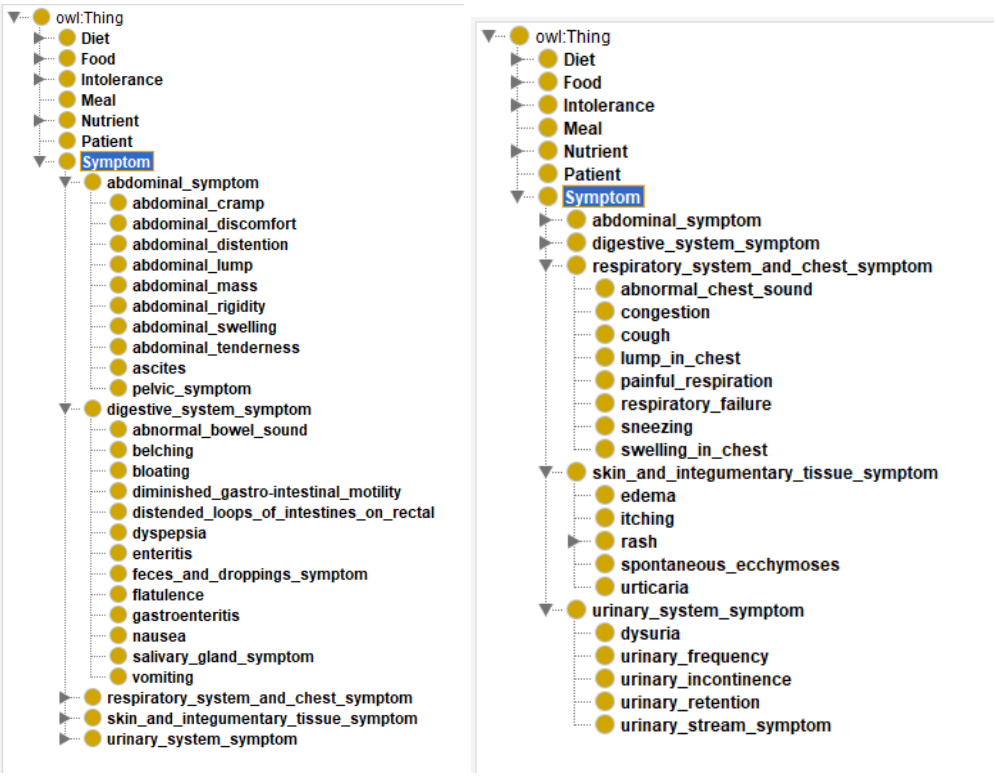


Figura 16. Jerarquia de la classe *Symptom*.

Cal afegir que en la creació de totes les classes principals s'ha utilitzat l'opció de *disjoint*, ja que cap d'aquestes comparteix cap instància.

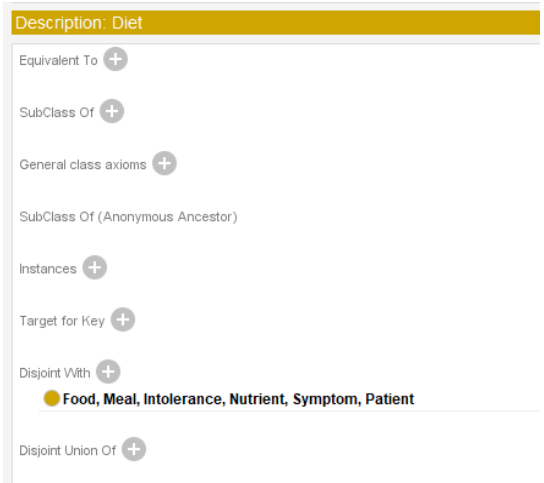


Figura 17. Classes de tipus *disjoint*.

3.5 Definició de propietats

3.5.1 *Object property*

En aquest apartat es defineixen les propietats de tipus *object*, les quals relacionen classes entre sí i responen a algunes de les principals *competency questions*. La Figura 18. *Object properties*. mostra el llistat d'aquestes.

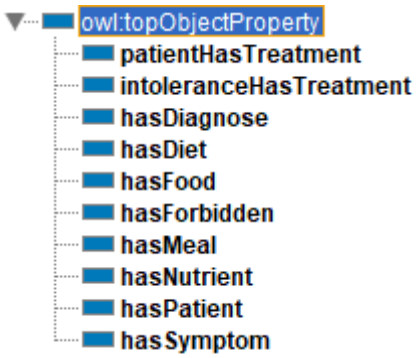


Figura 18. *Object properties*.

A més, cal afegir que la propietat hasPatient i hasMeal s'han definit com inverses entre sí, característica esmentada en l'apartat 2.3.1.

Les propietats definides són les següents:

Domini	**Rang**	**Propietat**	**Descripció**
Patient	Diet	*hasDiet*	La propietat atribueix una dieta a un pacient. Aquesta dieta és prèvia a cap diagnòstic, és a dir, correspon a algun tipus de dieta específica que el pacient realitzi de forma prèvia a l'estudi (per exemple, segueix un estil de vida vegetarià).
	Intolerance	*hasDiagnose*	La propietat atribueix un diagnòstic (intolerància) a un pacient. Aquest valor pot admetre el valor de *null* (en cas de no detectar cap patologia).
	Diet	*patientHasTreatment*	La propietat atribueix una dieta (en funció de tractament) a un pacient.
	Meal	*hasMeal*	La propietat atribueix un àpat a un pacient.
Food	Nutrient	*hasNutrient*	La propietat atribueix un nutrient a un aliment.
Meal	Food	*hasFood*	La propietat atribueix un aliment a un àpat.

	Symptom	*hasSymptom*	La propietat atribueix un símptoma a un àpat.
	Patient	*hasPatient*	La propietat atribueix un pacient a un àpat.
Diet	Food	*hasForbidden*	La propietat atribueix un aliment prohibit a una dieta.
	Nutrient	*hasForbidden*	La propietat atribueix un nutrient prohibit a una dieta.
Intolerance	Diet	*hasTreatment*	La propietat atribueix una dieta a una intolerància (en funció de tractament d'aquesta).

Taula 39. *Object properties*.

3.5.2 *Propietats de dades*

En aquest apartat es defineixen les propietats de tipus *data*, les quals relacionen classes amb valors. La Figura 19. *Data Properties*. mostra el llistat d'aquestes.

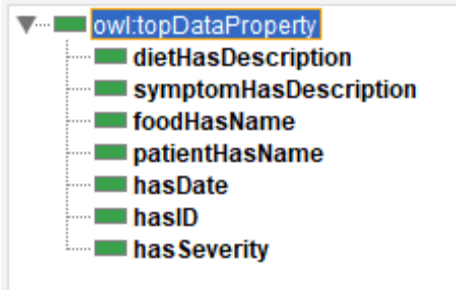


Figura 19. *Data Properties*.

Les propietats definides són les següents:

Domini	**Rang**	**Propietat**	**Descripció**
Patient	String	*hasID*	La propietat atribueix un String com a ID del pacient.
	String	*patientHasName*	La propietat atribueix un String com a nom del pacient.
Food	String	*foodHasName*	La propietat atribueix un String com a nom d'un menjar.
Meal	Date	*hasDate*	La propietat atribueix una data de tipus Date a un àpat.
Symptom	Int	*hasSeverity*	La propietat atribueix un grau de severitat a un símptoma, representat per un enter.

	String	*symptomHasDescription*	La propietat atribueix un String com a descripció d'un símptoma.
Diet	String	*dietHasDescription*	La propietat atribueix una descripció de tipus String a una dieta.

Taula 40. *Data Properties.*

3.6 Definició de restriccions

Per definir les restriccions sobre les classes de l'ontologia s'utilitzen els conceptes explicats en l'apartat 2.3.1.

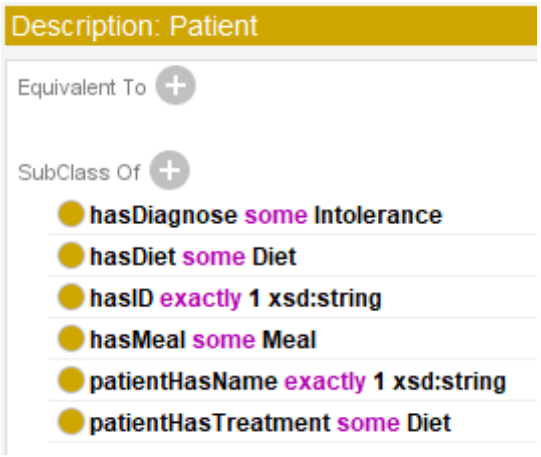


Figura 20. Restriccions *Patient*.

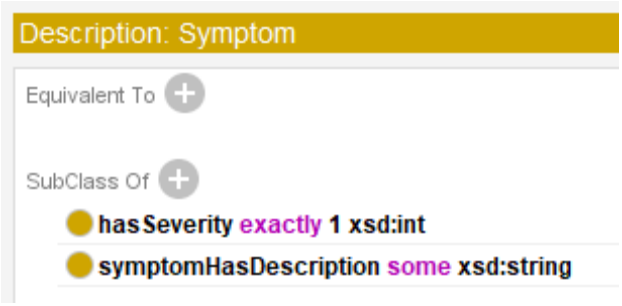


Figura 21. Restriccions *Symptom*.

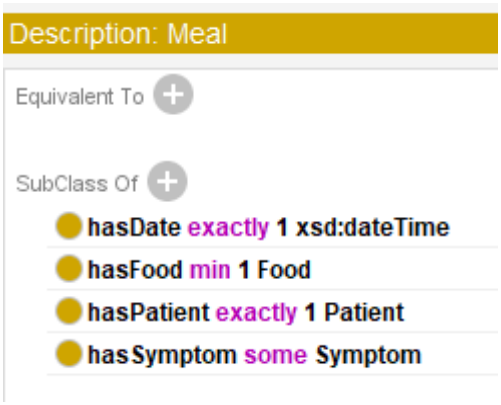


Figura 22. Restriccions *Meal*.

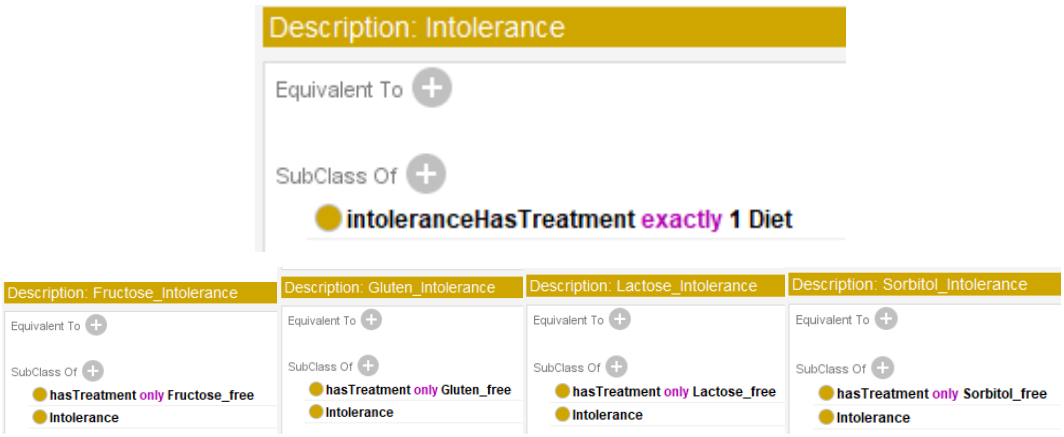
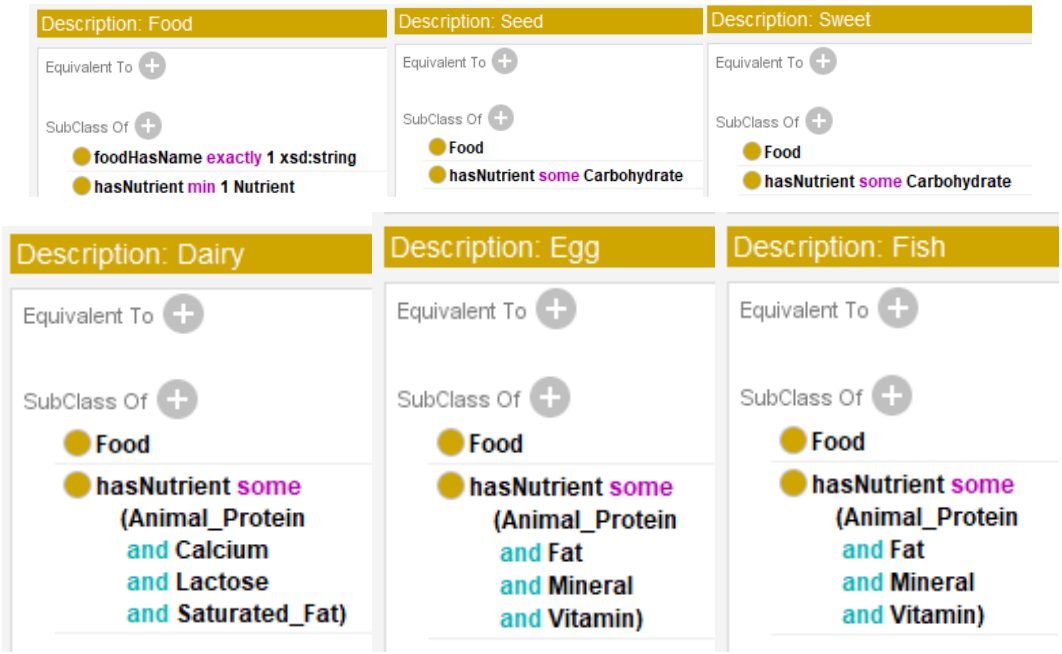


Figura 23. Restriccions *Intolerance*.



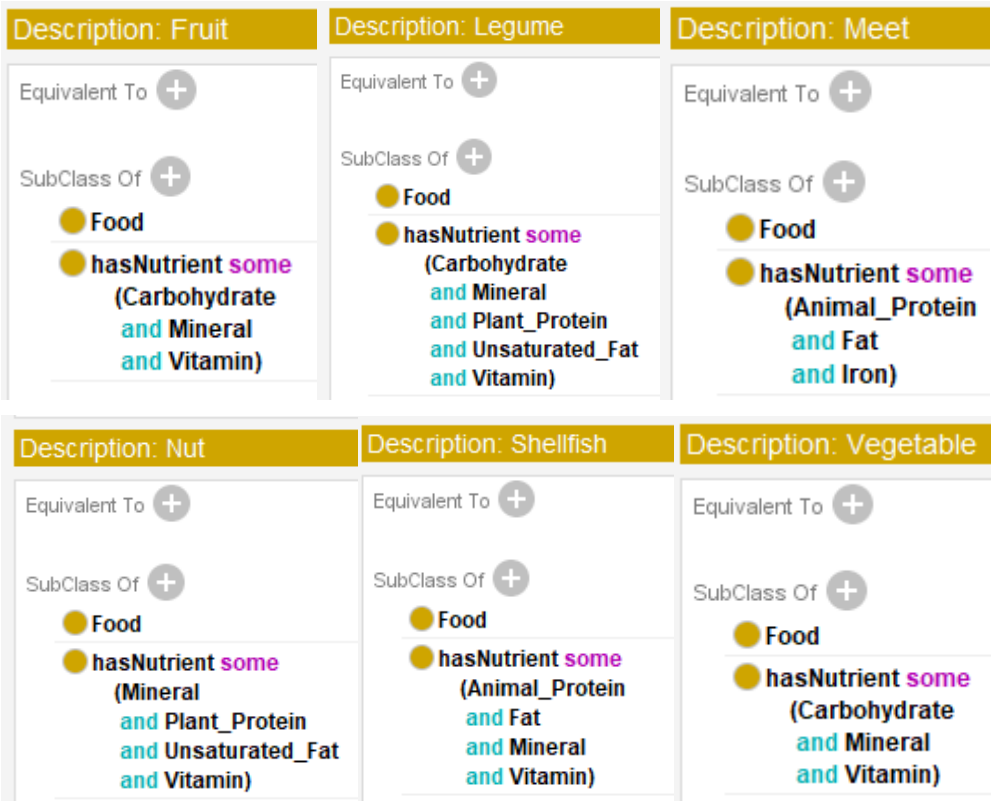


Figura 24. Restriccions *Food*.

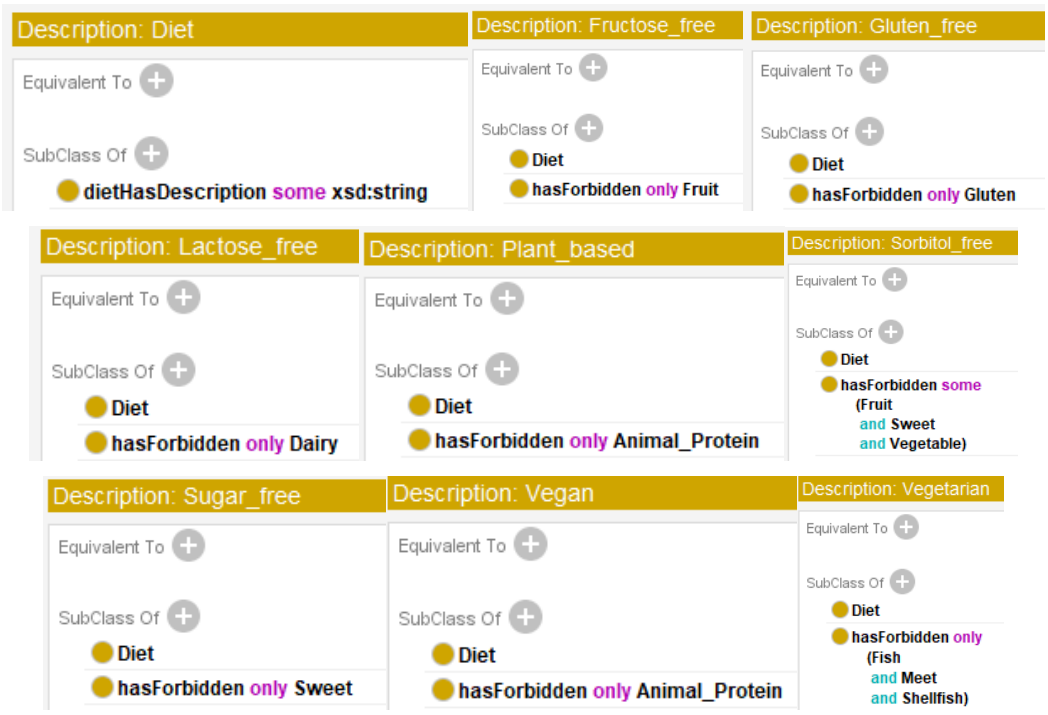


Figura 25. Restriccions *Diet*.

3.7 Creació d'instàncies

Degut a que l'objectiu d'aquesta ontologia és ser utilitzada com a base de coneixement d'un sistema expert, la creació d'instàncies no és un element principal, doncs les instàncies formaran la base de fets i s'obtindran directament a partir de l'usuari mitjançant una interfície.

Tanmateix, s'han creat diverses instàncies de pacients per poder realitzar proves posteriorment amb el sistema expert.

A continuació es mostra com es visualitzen les instàncies en la UI de Protégé.

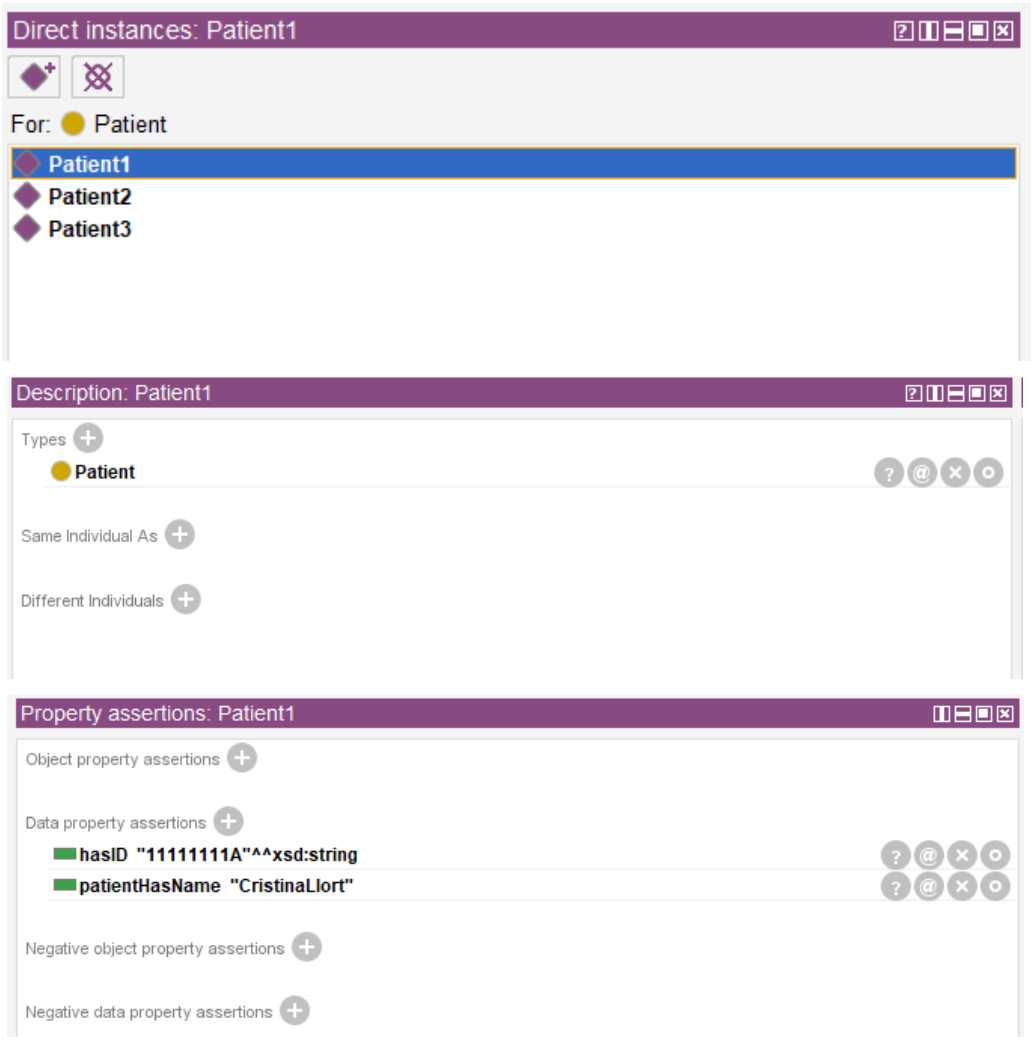


Figura 26. Instàncies amb Protégé.

4 Construcció del sistema expert

En la implementació del sistema expert es diferencien tres apartats que corresponen als components bàsics d'un sistema d'aquestes característiques, tal i com s'ha explicat en l'apartat 2.1.1.

Així doncs, la construcció del sistema es basa en l'anàlisi, el disseny i la implementació de tres parts diferenciades:

- **Vinculació de la base de coneixement.** S'utilitza l'ontologia construïda com a base de coneixement del sistema i, per tant, cal vincular-la a aquest.
- **Creació de la memòria de treball.** La memòria de treball del sistema està formada per la informació que els pacients afegeixen respecte als seus àpats. La introducció d'aquestes dades es realitza a partir d'una interfície gràfica.
- **Creació del motor d'inferència**. El sistema ha de respondre a les qüestions d'interès per facilitar un diagnòstic i proporcionar la informació rellevant de la base de coneixement a l'administrador del sistema.

4.1 Captura de requisits

L'anàlisi de requisits, també conegut com a Enginyeria de requisits, és el procés de definició de les expectatives dels usuaris per a un nou programari que es construeix o modifica. En enginyeria de programari, de vegades es fa referència a aquest com la recopilació de requisits o la captura de requisits. L'anàlisi de requisits comprèn aquelles tasques que serveixen per determinar les necessitats o les condicions a satisfer per a un producte o projecte nou o alterat, tenint en compte aquells requisits possiblement conflictius per a les diverses parts interessades, analitzant, documentant, validant i gestionant els requisits del programari o del sistema [29].

Existeixen dos classes de requisits [30]:

- **Requisits funcionals.** Declaren els serveis que proveirà el sistema i de la manera en què aquest reaccionarà a entrades particulars. En alguns casos, els requisits funcionals dels sistemes també declaren explícitament el que el sistema no ha de fer.
- **Requisits no funcionals.** Són aquells requisits que no es refereixen directament a les funcions específiques que lliura el sistema, sinó a les propietats emergents d'aquest com la fiabilitat, la resposta en el temps i la capacitat d'emmagatzematge. De manera alternativa, defineixen les restriccions de sistema com la capacitat dels dispositius d'entrada / sortida i la representació de dades que s'utilitza en la interfície de sistema.

4.1.1 *Requisits funcionals*

Els requisits funcionals del sistema es basen en les *Competency questions* plantejades durant la construcció de la base de coneixement a l'apartat 3.1.1.

ID	Requisit
RF-T1	Mostrar els tipus principals d'aliments i la seva classificació.
RF-T2	Mostrar els tipus principals de nutrients i la seva classificació.
RF-T3	Mostrar els tipus principals de simptomatologies i la seva classificació.
RF-T4	Mostrar els tipus principals d'intoleràncies i la seva classificació.
RF-T5	Mostrar els tipus principals de dietes i la seva classificació.
RF-X1	Mostrar els nutrients d'un aliment X.
RF-X2	Mostrar els aliments/nutrients prohibits per una dieta X.
RF-X3	Mostrar el tractament que s'aplica a la reacció adversa (intolerància) a un tipus d'aliment/nutrient X.
RF-P1	Mostrar el nom de la dieta prèvia a l'estudi (en cas que existeixi) i restriccions d'aquesta (en cas que existeixin) realitzada pel pacient.
RF-P2	Mostrar els àpats realitzats pel pacient.
RF-P3	Mostrar el nombre total d'àpats realitzats pel pacient.
RF-P4	Mostrar el nombre total d'aliments ingerits pel pacient.
RF-P5	Mostrar el nombre total de símptomes experimentats pel pacient.
RF-P6	Mostrar els aliment/s i la simptomatologia (en cas que es presenti) de cada àpat.
RF-P7	Mostrar la severitat de la simptomatologia experimentada pel pacient (en cas de presentar-se).
RF-P8	Realitzar un diagnòstic (si es pot concloure) de la simptomatologia del pacient.
RF-P8.1	Calcular el percentatge de probabilitats de patir intolerància a la fructosa.
RF-P8.2	Calcular el percentatge de probabilitats de patir intolerància al gluten.
RF-P8.3	Calcular el percentatge de probabilitats de patir intolerància a la lactosa.
RF-P8.4	Calcular el percentatge de probabilitats de patir intolerància al sorbitol.
RF-P9	Definir quin tractament (dieta) s'aplica al pacient.

RF-U1	El sistema permet a l'usuari identificar-se.
RF-U2	El sistema permet a l'usuari registrar un aliment en un àpat.
RF-U3	El sistema permet a l'usuari registrar un símptoma en un àpat.
RF-U4	El sistema permet a l'usuari afegir un àpat.

Taula 41. Requisits funcionals.

4.1.2 *Requisits no funcionals*

En la definició dels requisits no funcionals s'ha volgut assegurar que la interacció persona-ordinador sigui exitosa i s'han tingut en compte aspectes com la jerarquia visual i les lleis de la Gestalt [11] [12].

ID	**Requisit**
RNF-D01	S'aplica el principi de semblança de la Gestalt, el qual estableix que quan les coses semblen ser semblants entre sí, l'individu tendeix a agrupar-les i associa que tenen la mateixa funció.
RNF-D02	S'aplica el principi de proximitat de la Gestalt, el qual estableix que les coses que estan juntes semblen estar més relacionades que les coses que estan més separades.
RNF-D03	S'aplica el principi de regió comuna de la Gestalt, el qual està molt relacionat amb el de la proximitat. Afirma que quan els objectes es troben dins de la mateixa regió tancada, es perceben com agrupats.
RNF-D04	S'aplica el principi de continuïtat de la Gestalt, el qual estableix que els elements que estan disposats sobre una línia o corba es perceben més relacionats que els elements que no es troben sobre la línia o la corba.
RNF-L01	L'anàlisi del projecte i la implementació del codi i de la interfície d'usuari es realitza en anglès (ús internacional) per assegurar l'accessibilitat d'aquest.
RNF-T01	La persistència de les dades es realitza utilitzant el format RDF/XML.
RNF-T02	Es maneja la base de coneixement (ontologia) amb l'ús de l'API Java Jena i es realitzen les consultes sobre aquesta amb el llenguatge SPARQL.

Taula 42. Requisits no funcionals.

4.1.3 *Casos d'ús*

Un cop definits els requisits funcionals que el sistema ha de complir, es representen els casos d'ús que contenen els serveis que l'aplicació proporciona.

4.1.3.1 Actors

En primer lloc, es descriuen breument els actors que participen en els casos d'ús del sistema, als quals se'ls atribueixen aquells que els pertoca en el següent apartat.

- **Administrador.** És aquell que accedeix a les funcionalitats internes i bàsiques del sistema proposat (aquelles que no corresponen a recollida de dades i que no presenten interfície gràfica).
- **Pacient o usuari.** És el principal client de l'aplicació i aquell que proporciona les dades que completen la memòria de treball del sistema descrita anteriorment.

Actualment l'aplicació s'implementa considerant només aquests dos actors però es planteja la possibilitat d'incloure el rol de *Metge o expert* en el cas de seguir desenvolupant el sistema en un futur.

4.1.3.2 Diagrama de casos d'ús

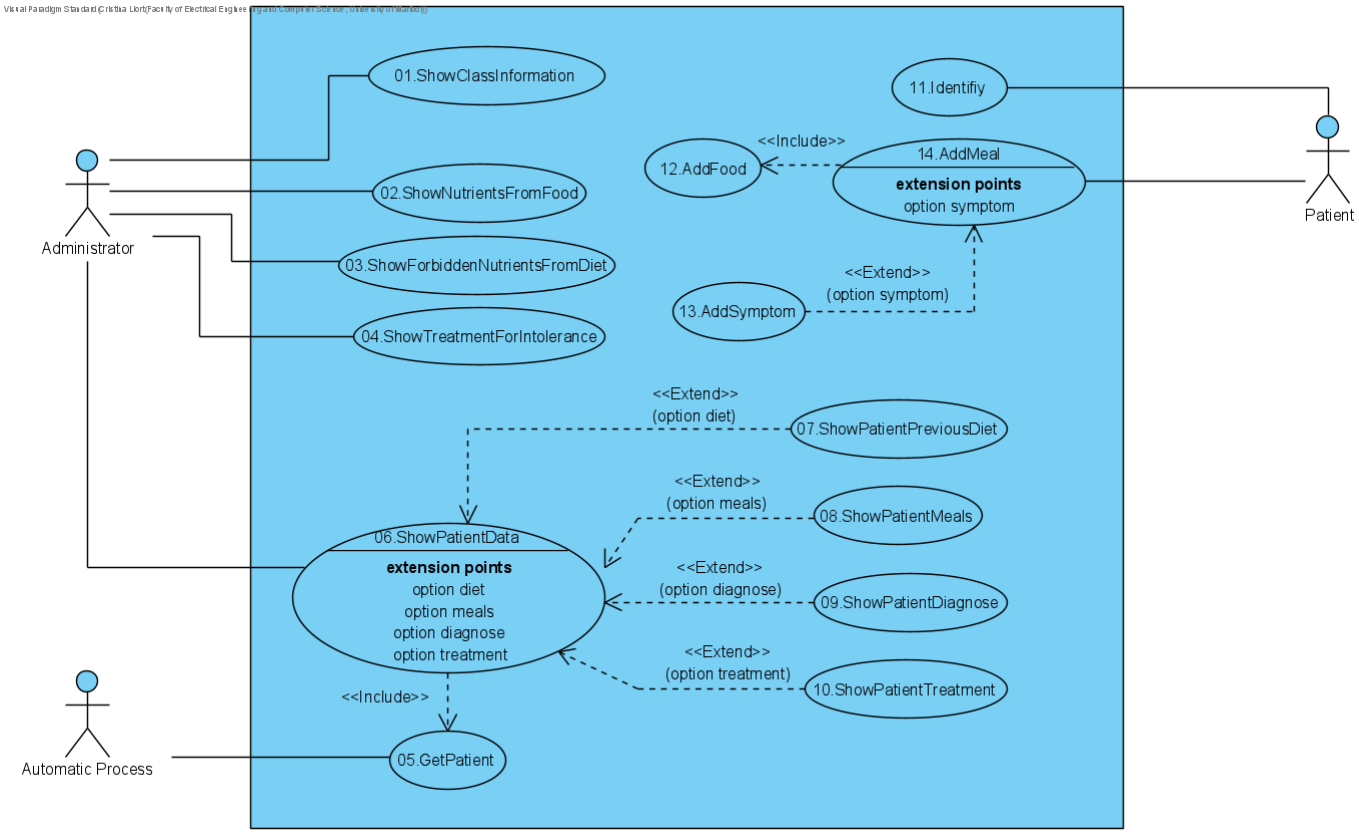


Diagrama 4. Diagrama de casos d'ús.

4.2 Anàlisi

Durant la fase d'anàlisi s'estudien l'estructura del sistema mitjançant un diagrama de classes i el desenvolupament dels diferents casos d'ús mitjançant els diagrames de seqüències corresponents a cadascun d'aquests.

4.2.1 *Diagrama de classes*

Les classes representades en el diagrama corresponen a les classes principals designades pel model ontològic, les quals proporcionen l'estructura de dades bàsica del sistema.

El següent diagrama mostra les relacions establertes entre aquestes, així com la cardinalitat de cadascuna de les relacions.

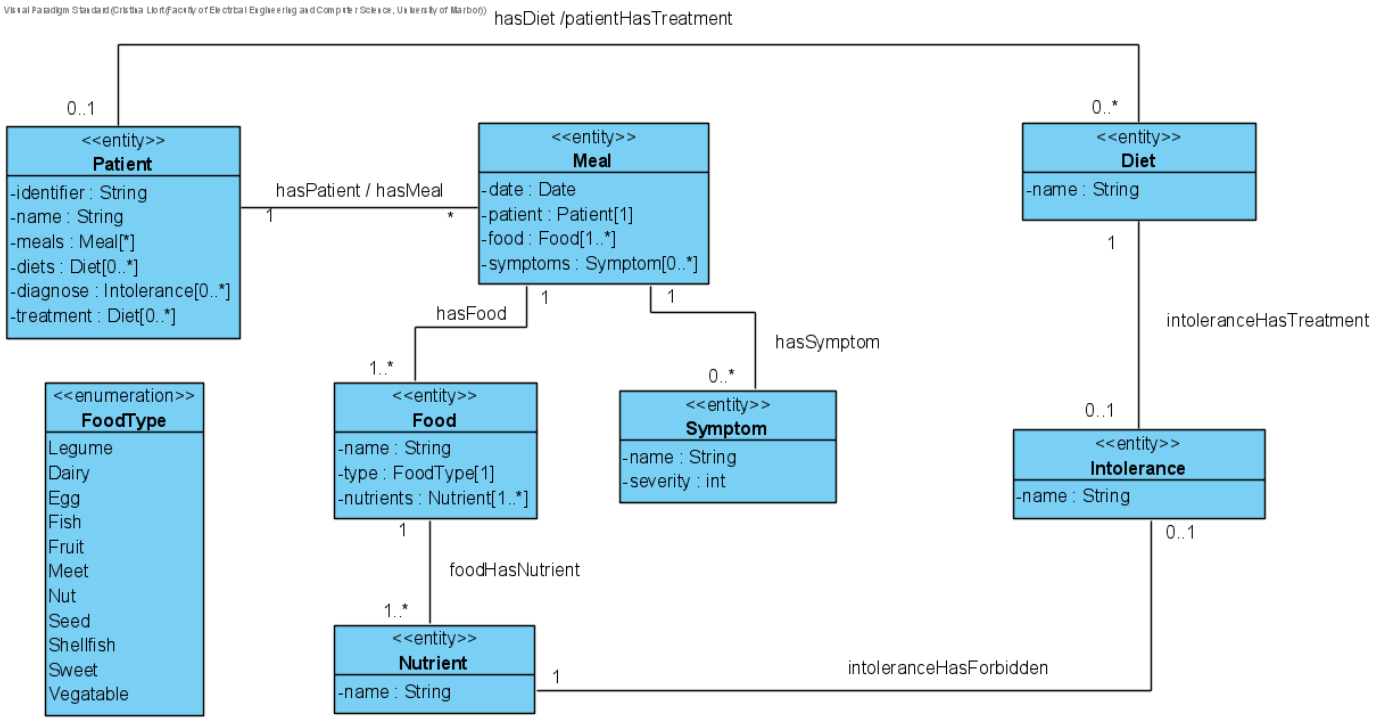


Diagrama 5. Diagrama de classes.

4.2.2 *Diagrames dels casos d'ús*

4.2.2.1 Cd'ú 01.*ShowClassInformation*

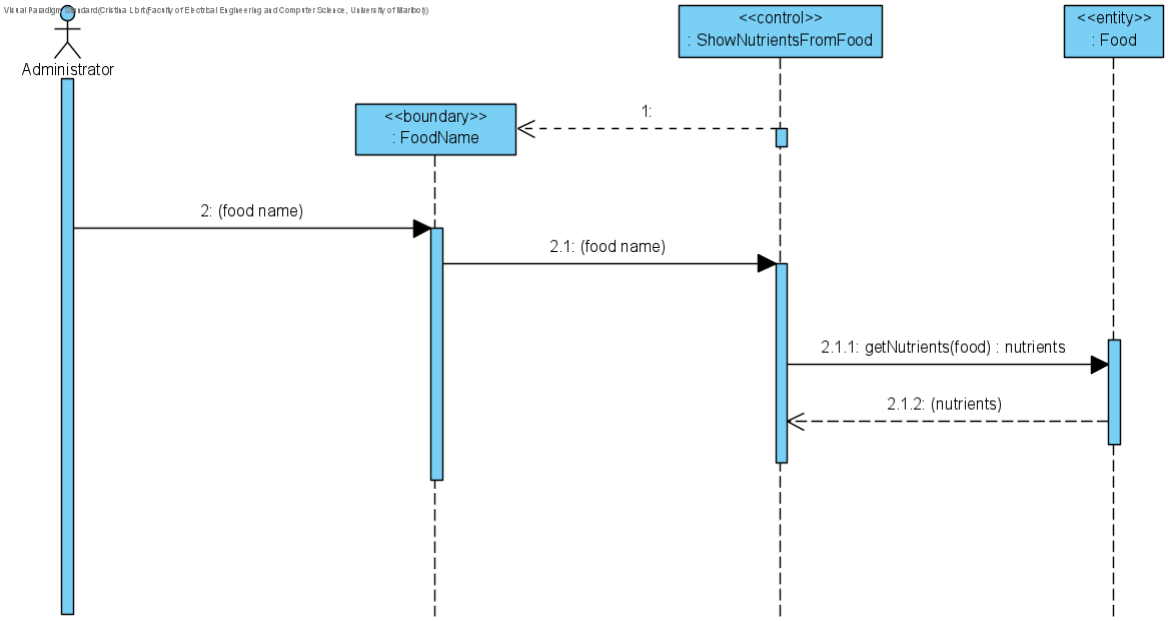


Diagrama 6. Diagrama de seqüències del cd'ú 01.

4.2.2.2 Cd'ú 02. *ShowNutrientsFromFood*

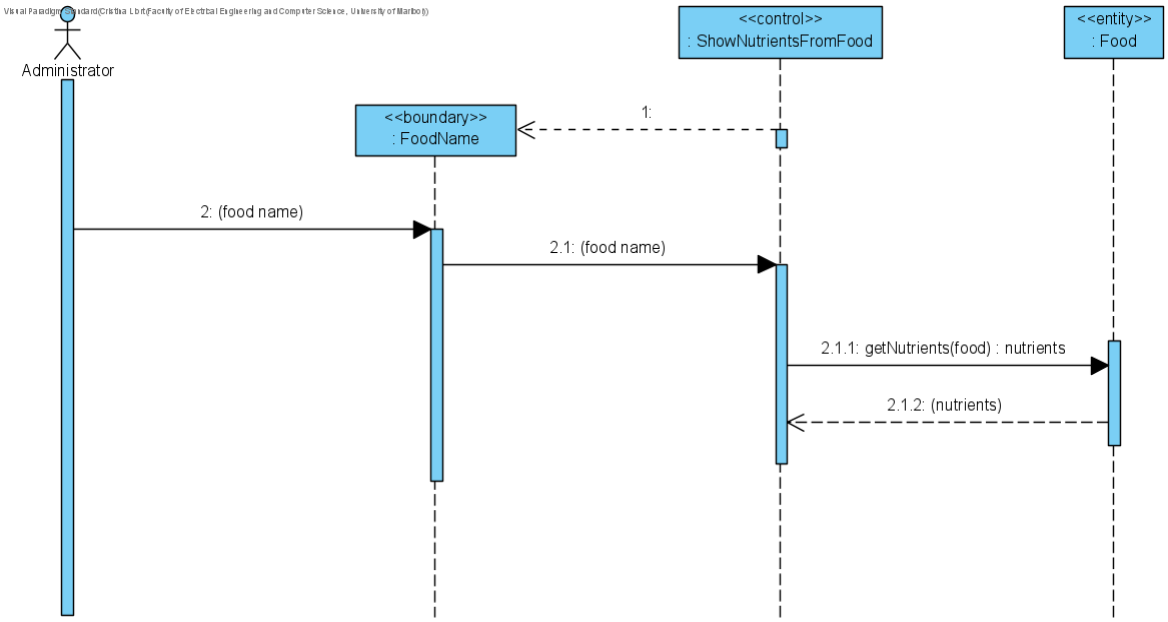


Diagrama 7. Diagrama de seqüències del cd'ú 02.

4.2.2.3 Cd'ú 03. *ShowForbiddenNutrientsFromDiet*

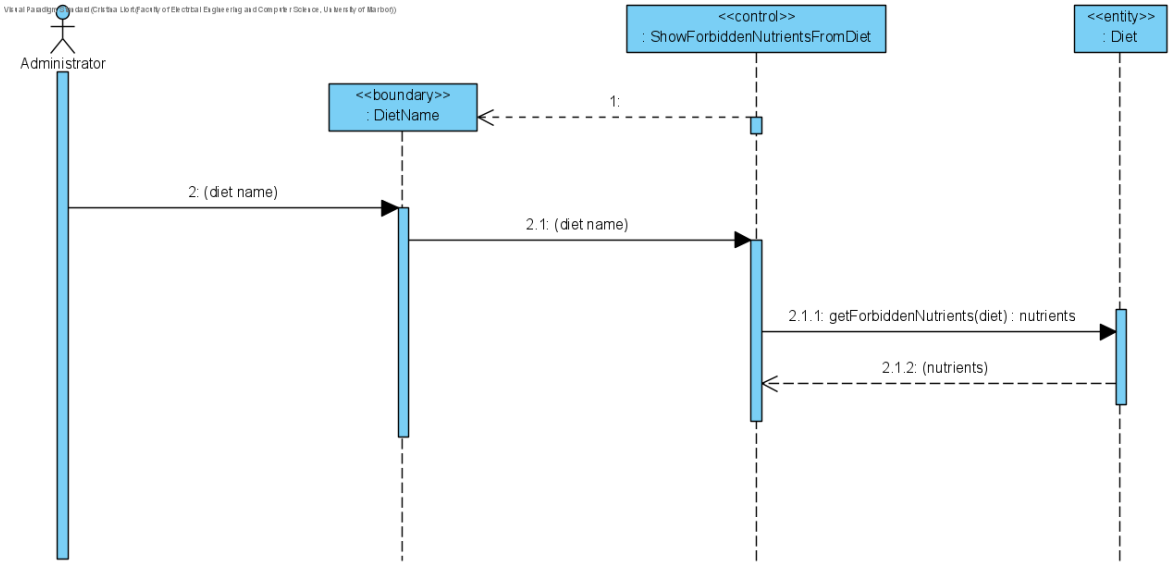


Diagrama 8. Diagrama de seqüències del cd'ú 03.

4.2.2.4 Cd'ú 04. *ShowTreatmentForIntolerance*

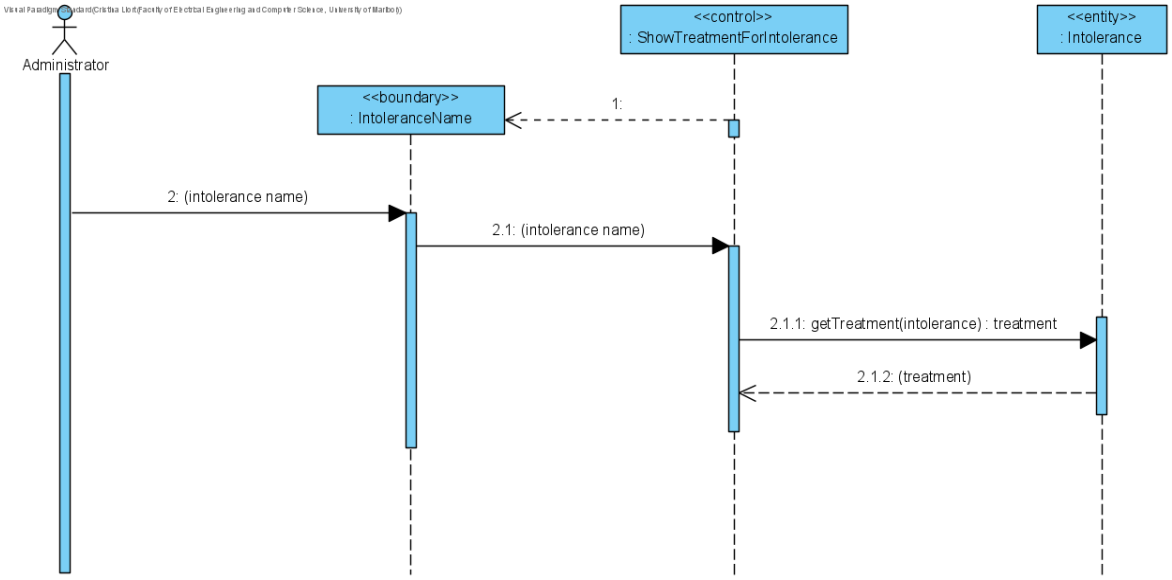


Diagrama 9. Diagrama de seqüències del cd'ú 04.

4.2.2.5 Cd'ú 05. *GetPatient*

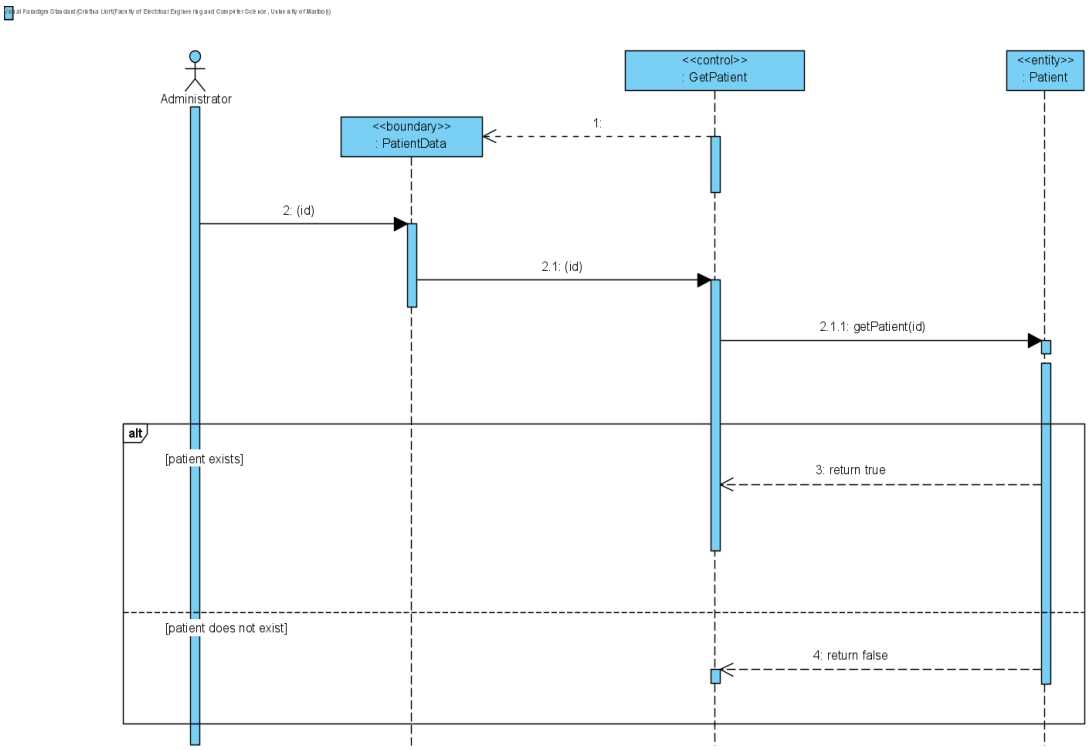


Diagrama 10. Diagrama de seqüències del cd'ú 05.

4.2.2.6 Cd'ú 06. *ShowPatientData*

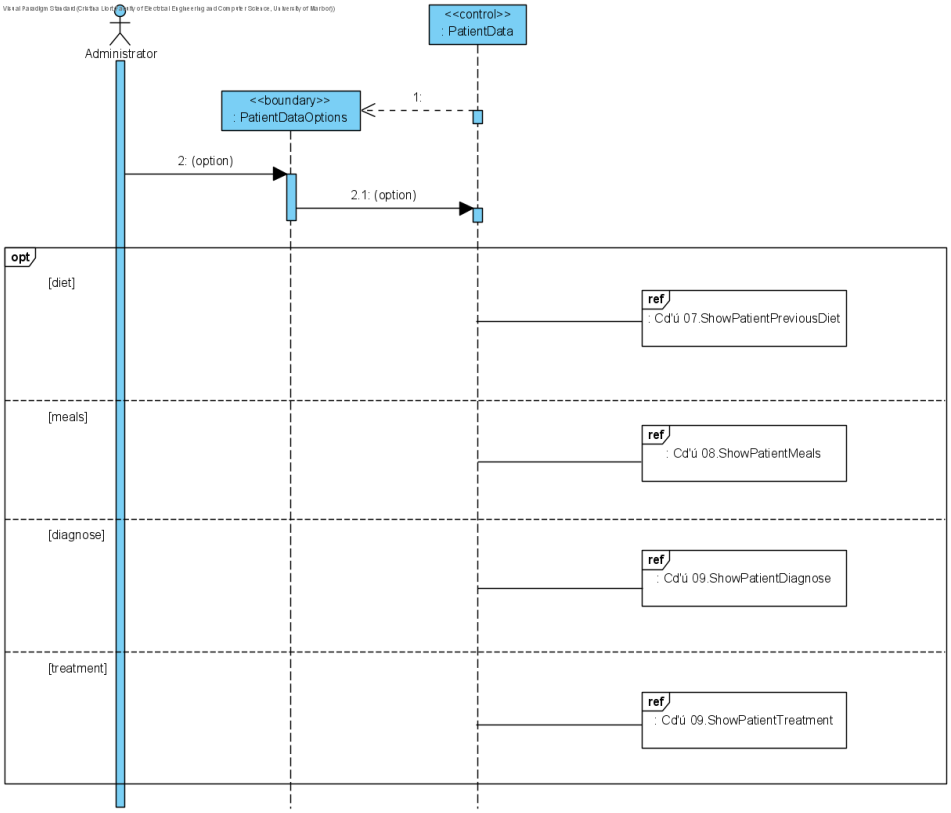


Diagrama 11. Diagrama de seqüències del cd'ú 06.

4.2.2.7 Cd'ú 07. *ShowPatientPreviousDiet*

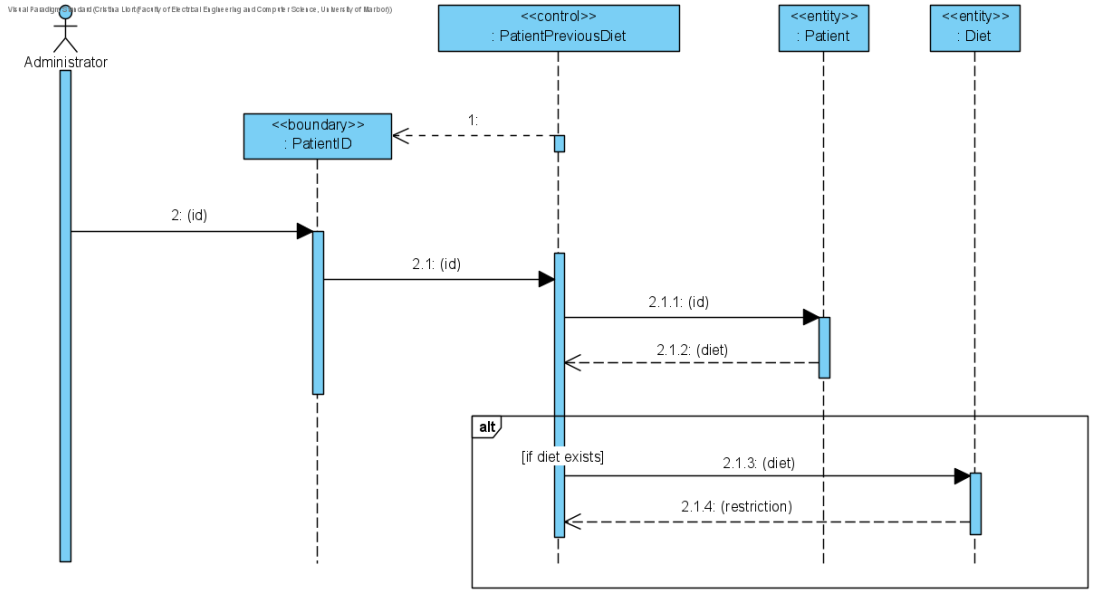


Diagrama 12. Diagrama de seqüències del cd'ú 07.

4.2.2.8 Cd'ú 08. *ShowPatientMeals*

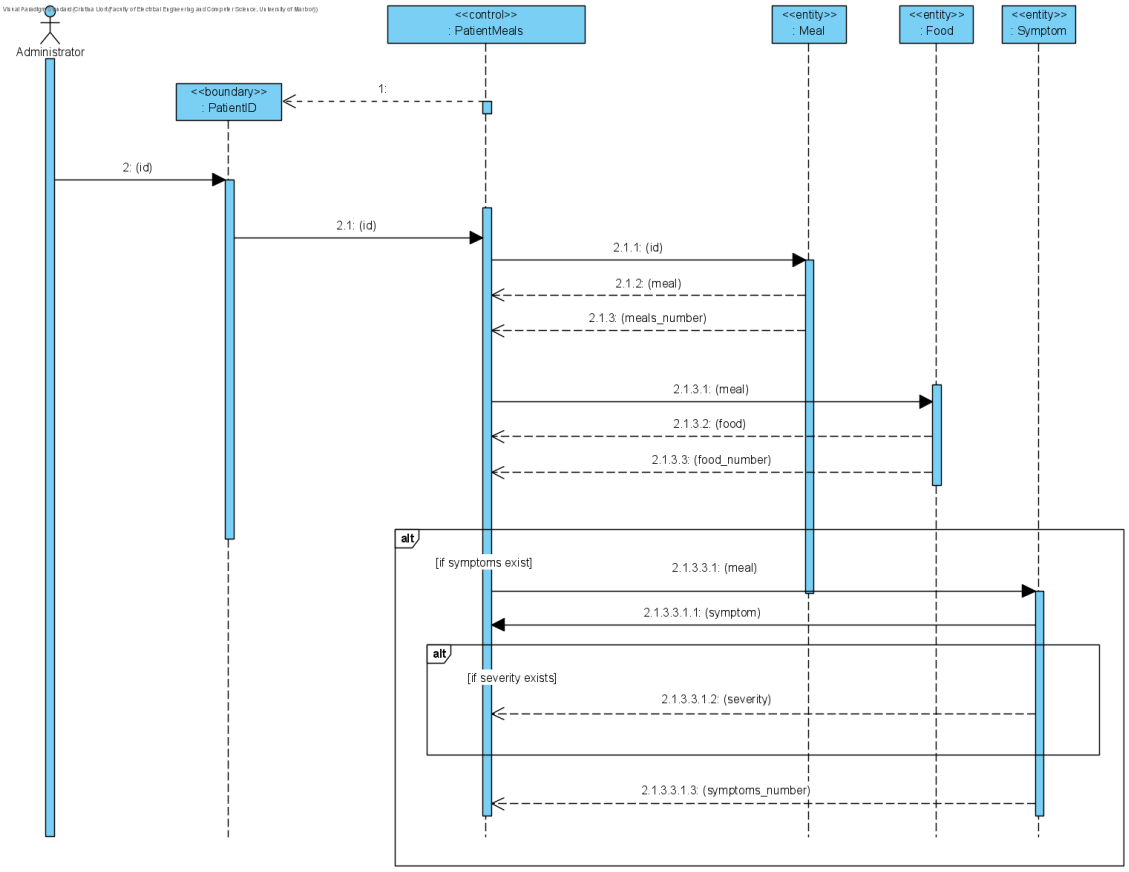


Diagrama 13. Diagrama de seqüències del cd'ú 08.

4.2.2.9 Cd'ú 09. *ShowPatientDiagnose*

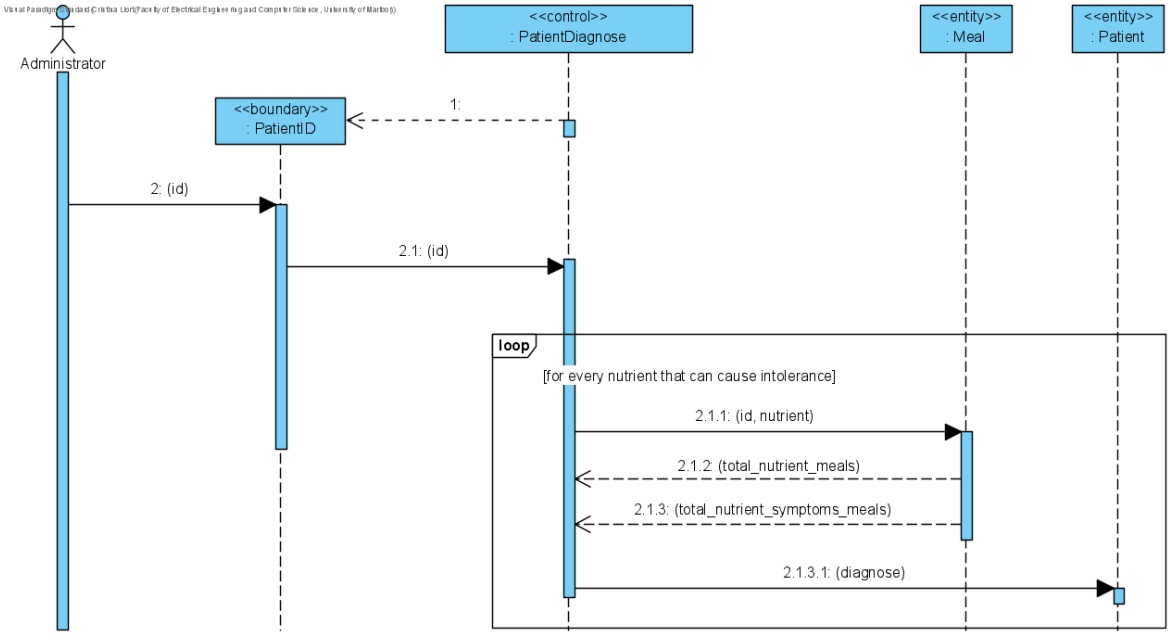


Diagrama 14. Diagrama de seqüències del cd'ú 09.

4.2.2.10 Cd'ú 10. *ShowPatientTreatment*

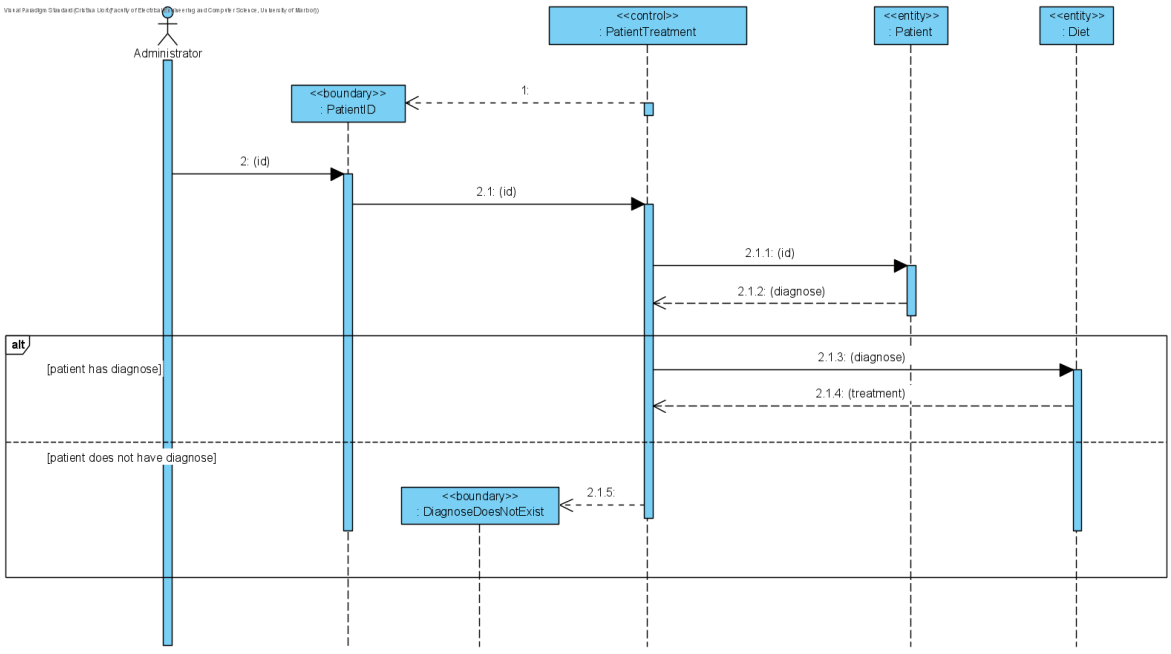


Diagrama 15. Diagrama de seqüències del cd'ú 10.

4.2.2.11 Cd'ú 11. *Identify*

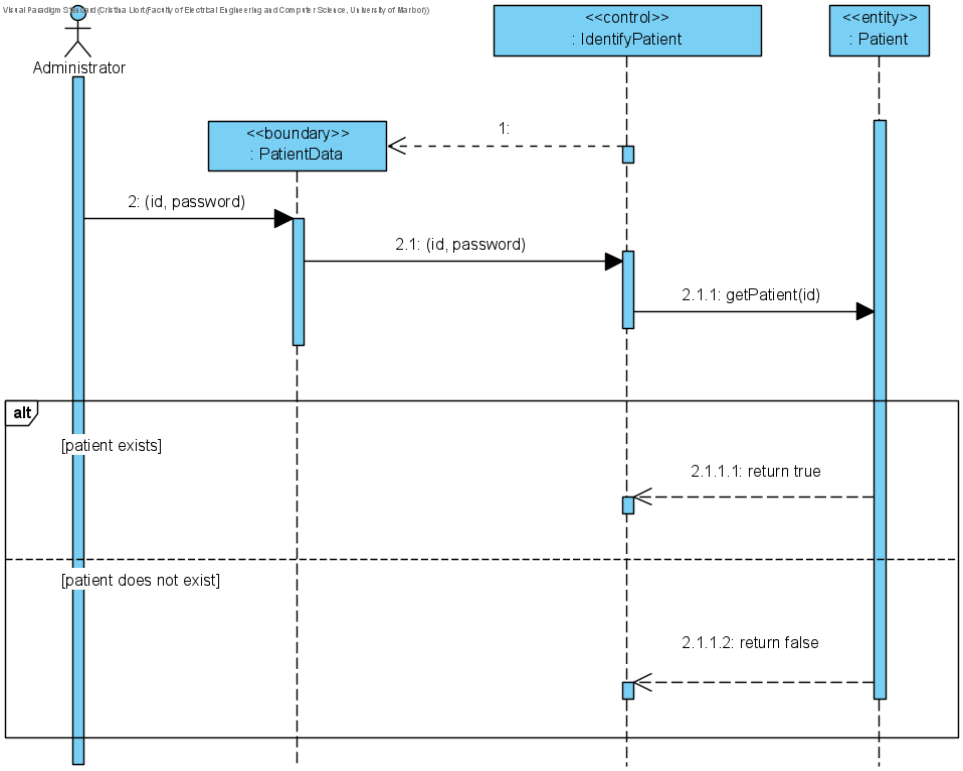


Diagrama 16. Diagrama de seqüències del cd'ú 11.

4.2.2.12 Cd'ú 12. *AddFood*

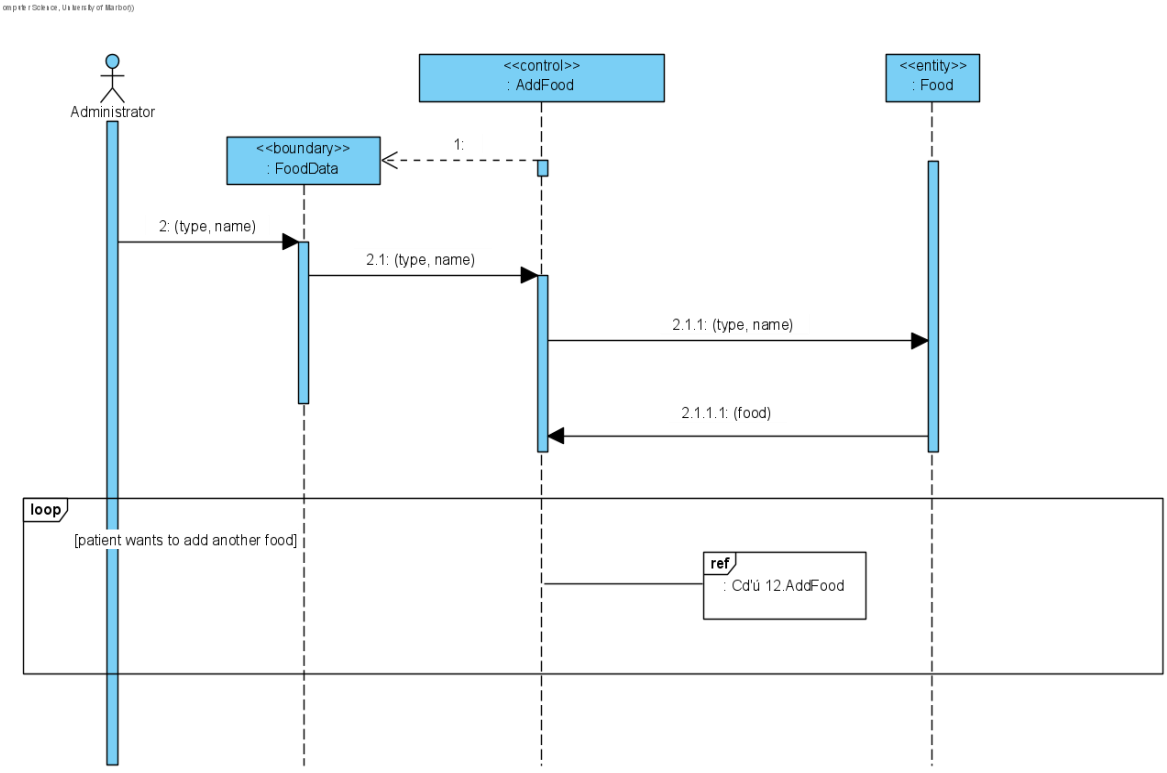


Diagrama 17. Diagrama de seqüències del cd'ú 12.

4.2.2.13 Cd'ú 13. *AddSymptom*

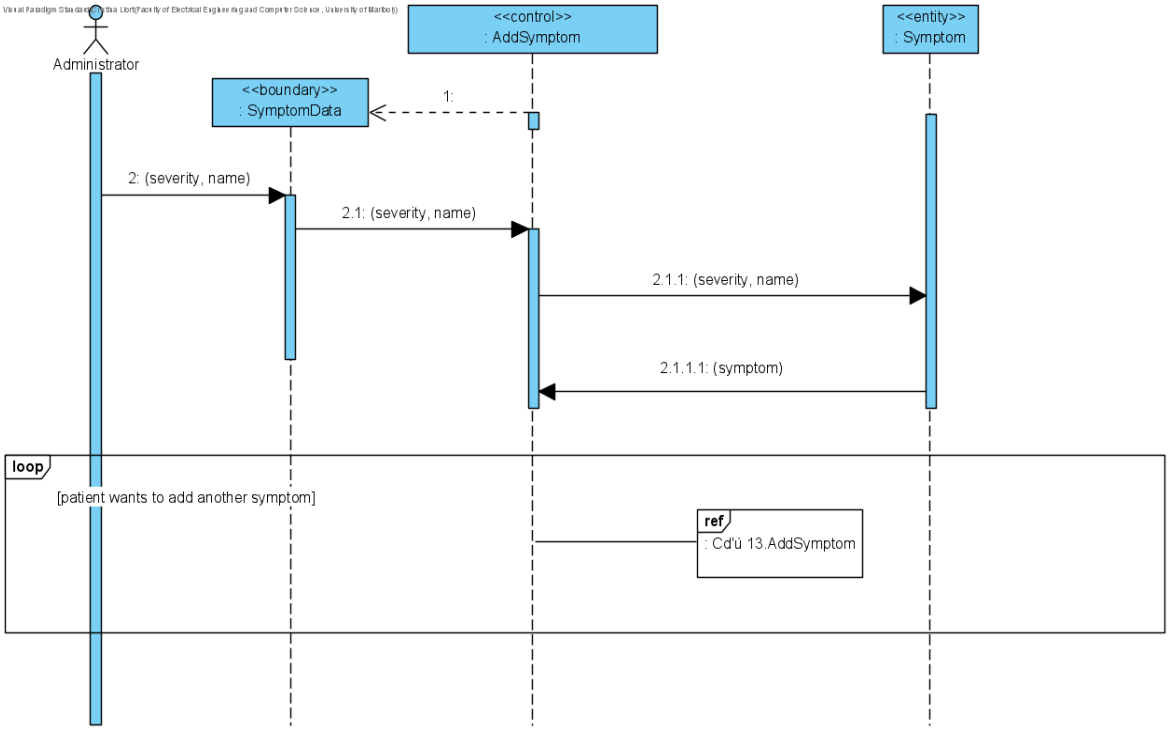


Diagrama 18. Diagrama de seqüències del cd'ú 13.

4.2.2.14 Cd'ú 14. *AddMeal*

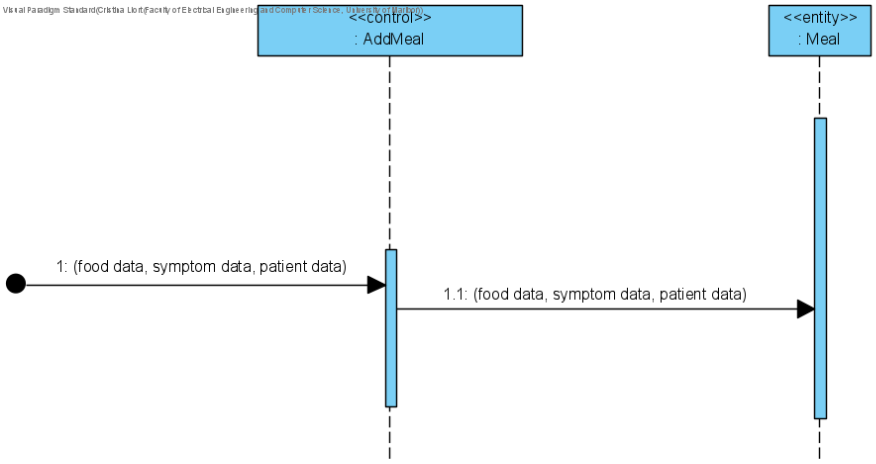


Diagrama 19. Diagrama de seqüències del cd'ú 14.

4.3 Disseny

En aquesta etapa s'han dissenyat les finestres corresponents als casos d'ús que pertanyen a les funcionalitats de l'usuari (pacient), és a dir, aquells que s'executen mitjançant una interfície gràfica.

4.3.1 *Disseny de la interfície d'usuari*

4.3.1.1 Cd'ú 11. *Identify*

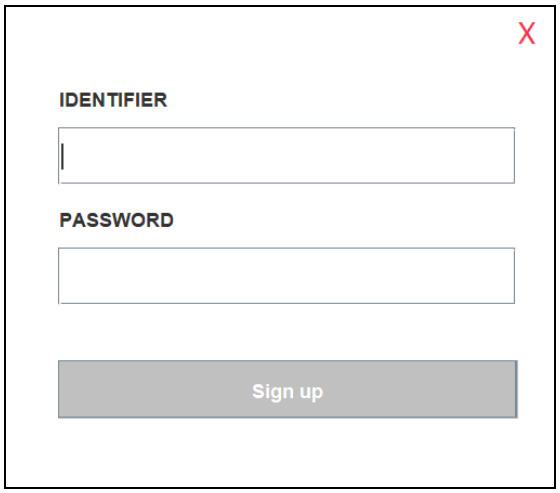


Figura 27. Cd'ú *11.Identify.*

4.3.1.2 Cd'ú 12. *AddFood*

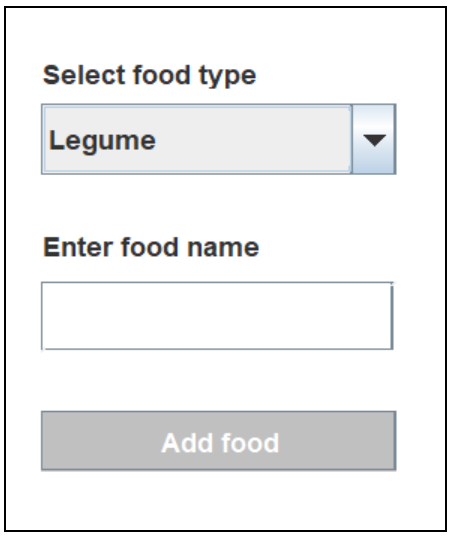


Figura 28. Cd'ú *12.AddFood.*

4.3.1.3 Cd'ú 13. *AddSymptom*

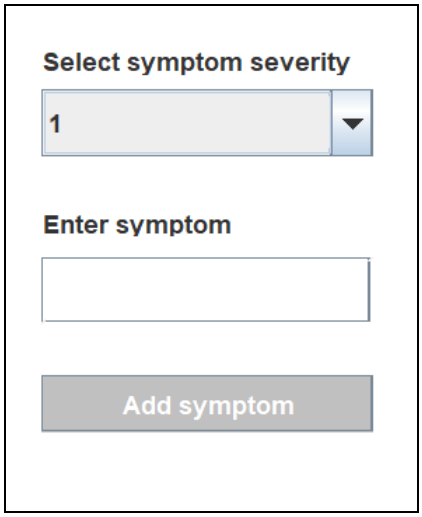


Figura 29. Cd'ú *13.AddSymptom.*

4.3.1.4 Cd'ú 14. *AddMeal*

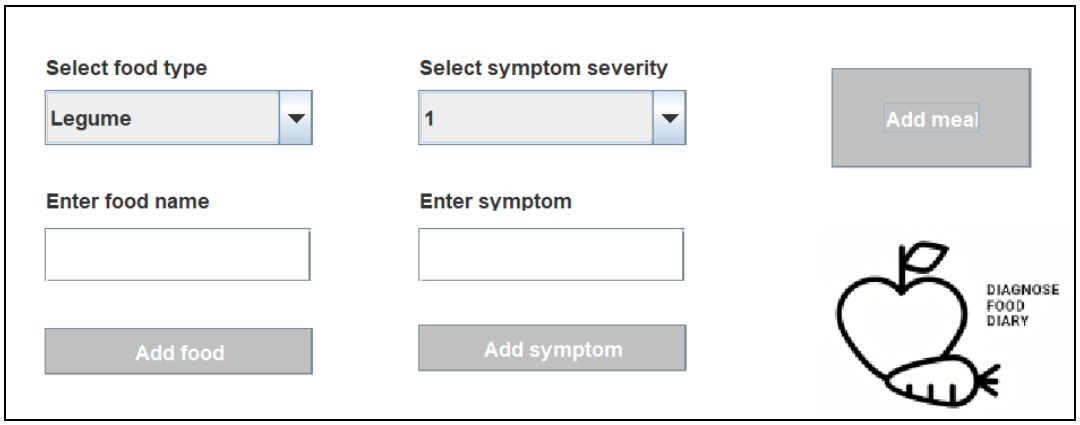


Figura 30. Cd'ú 14.AddMeal.

4.4 Implementació

En aquest apartat s'aporten fragments de codi que il·lustren com s'ha resolt la implementació del sistema basant-se en l'anàlisi i el disseny realitzats.

4.4.1 *Vinculació de la base de coneixement*

El primer pas per poder implementar el sistema és obtenir la base de coneixement construïda prèviament, tal i com ja s'ha comentat. L'API Java Jena ens permet importar la base ontològica i treballar amb ella tal i com es mostra en el fragment de codi següent, el qual pertany a la classe principal del sistema.

```
public class Ontology {

      public static OntModel model;
      public static String ontologyFile = "diagnose_food_diary.owl";
      public static final String URL_SOURCE = "http://www.owl-
ontologies.com/diagnose_food_diary.owl";
    protected static final String FILE_SOURCE =
"./src/diagnose_food_diary.owl";
    public static final String NS = URL_SOURCE + "#";


      public static void main(String[] args) throws
FileNotFoundException {

            model =
ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
            model.read(new FileReader(FILE_SOURCE), null, "RDF/XML");
            model.write(System.out) ;
      }
}
```

Codi 1. Creació i importació del model.

Per altra banda, per tal de gestionar la jerarquia de classes de l'ontologia s'ha implementat la classe *ClassHierarchy*, la qual permet treballar de forma senzilla i àgil amb les entitats ontològiques.

En el següent fragment de codi es mostren algunes de les funcions principals de la classe esmentada.

```
public class ClassHierarchy {
    protected OntModel m_model;

    /** Show the sub-class hierarchy encoded by the given model */
    public void showAllHierarchy(PrintStream out, OntModel m) {
        Iterator<OntClass> i =
m.listHierarchyRootClasses().filterDrop(OntClass::isAnon);

        while (i.hasNext()) {
            showClass(out, i.next(), new ArrayList<OntClass>(), 0);
        }
    }
```

```
    /** Show the sub-class hierarchy encoded by the given class */
    public void showClassHierarchy(PrintStream out, OntClass cls) {
      Iterator<OntClass> i =
cls.listSubClasses().filterDrop(OntClass::isAnon);

        while (i.hasNext()) {
            showClass(out, i.next(), new ArrayList<OntClass>(), 0);
        }
    }
    /** Present a class, then recurse down to the sub-classes */
    protected void showClass(PrintStream out, OntClass cls,
List<OntClass> occurs, int depth) {
        renderClassDescription(out, cls, depth);
        out.println();
        if (cls.canAs(OntClass.class)  &&  !occurs.contains(cls)) {
            for (Iterator<OntClass> i = cls.listSubClasses(true);
i.hasNext();) {
                OntClass sub = i.next();
                occurs.add(cls);
                showClass(out, sub, occurs, depth + 1);
                occurs.remove(cls);
            }
        }
    }
    /** Render a description of the given class to the given output
stream. */
    public void renderClassDescription(PrintStream out, OntClass c, int
depth) {
        indent(out, depth);

        if (c.isRestriction())
            renderRestriction(out, c.as(Restriction.class));
        else {
             if (!c.isAnon()) {
                 out.print("Class: ");
                 out.print( c.getLocalName());
                 out.print(' ');
              }else
                 renderAnonymous(out, c, "class");
    }

    /** Handle the case of rendering a restriction.</p>
    protected void renderRestriction(PrintStream out, Restriction r) {

        if (!r.isAnon()) {
            out.print("Restriction: ");
            out.print(r.getLocalName());
        }else
            renderAnonymous( out, r, "restriction" );
        out.print(" on property");
        out.print(r.getOnProperty().getLocalName());
    }

    /** Generate the indentation */
    protected void indent(PrintStream out, int depth) {
        for (int i = 0;  i < depth; i++) {
            out.print("  ");
        }
    }
```

Codi 2. Classe *ClassHierarchy*.

4.4.2 *Implementació de la memòria de treball*

Per altra banda, la memòria de treball del sistema està formada principalment pels àpats que realitzen els pacients, els quals inclouen els aliments i la simptomatologia d'aquests. Per tal d'emmagatzemar aquesta informació s'ha dissenyat una interfície senzilla que faciliti la introducció de la informació a l'usuari, tal i com s'ha dissenyat en l'apartat 4.3.1 i aplicant els requisits no funcionals descrits a l'apartat 4.1.2.

Finalment, la interfície gràfica implementada està formada per tres classes de tipus *Application Window* utilitzant la tecnologia que proporciona *WindowBuilder* [38].

Les classes implementades són:

- **Home**. Implementa la pantalla que permet a l'usuari identificar-se.
- **User.** Implementa la pantalla principal del perfil de l'usuari.
- **Meal.** Implementa la pantalla que permet registrar un àpat a l'usuari.

4.4.2.1 Classe *Home*

En primer lloc, quan s'executa la classe *Home.java* es mostra un missatge de benvinguda a l'usuari, el logotip del projecte i un sistema d'identificació usual composat per dos camps: un per a la introducció de l'identificador del pacient i un altre per a la contrasenya registrada per l'usuari.

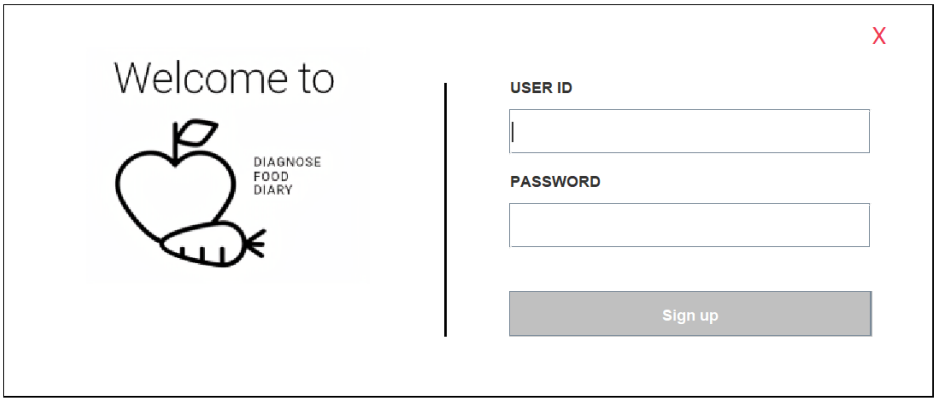


Figura 31. Pantalla *Home* de la interfície.

Es controla que l'usuari estigui prèviament registrat al sistema, és a dir, instanciat a l'ontologia, procés que actualment realitza manualment l'administrador del sistema. En un futur es contempla l'opció d'afegir el procés d'enregistrament per part del propi usuari.

En cas de no ser identificat el pacient al sistema, es mostra un missatge en forma de finestra emergent que informa de l'error.

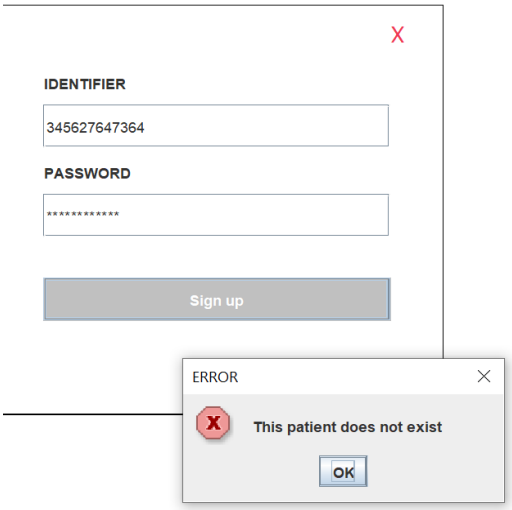


Figura 32. Error: *Patient does not exist*.

4.4.2.2 Classe *User*

Un cop identificat el pacient, la pantalla de Home deixa de visualitzar-se i apareix el perfil de l'usuari i les opcions de les que disposa.

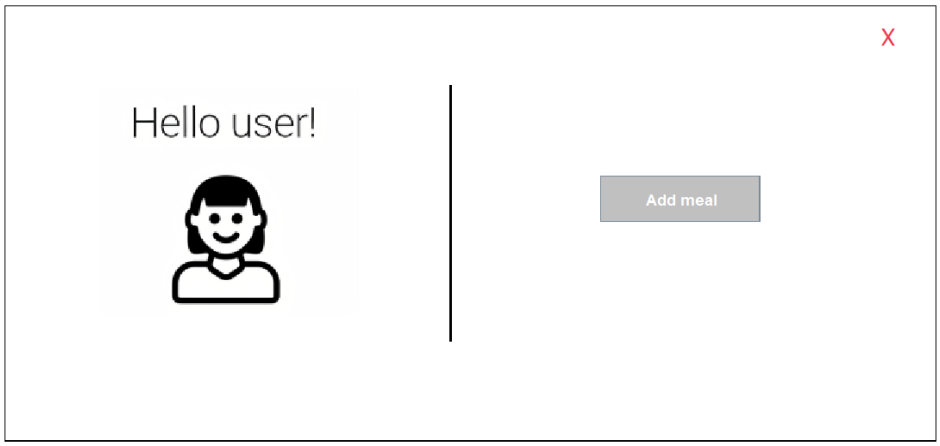


Figura 33. Pantalla *User* de la interfície.

Actualment aquesta pantalla només disposa de l'opció que permet afegir un àpat, però s'ha decidit incloure ja que es contempla la possibilitat d'afegir diverses funcionalitats en el futur com: modificar un àpat, veure dades personals, etc.

4.4.2.3 Classe *Meal*

Finalment, en cas d'escollir l'opció *Add meal* en el perfil d'usuari, la pantalla que es mostra és la següent:

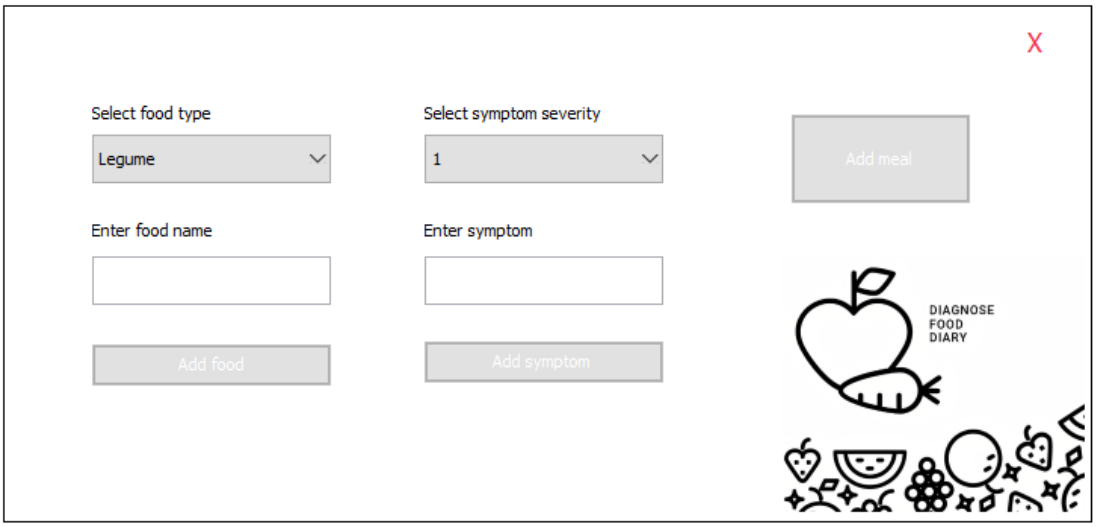


Figura 34. Pantalla *Meal* de la interfície.

La usabilitat de la pantalla s'ha implementat de manera que sigui intuïtiva, tal i com s'ha definit en els requisits. La interfície permet realitzar les funcions següents, les quals s'activen mitjançant un element de tipus *JButton* que conté una funció de tipus *Listener* que activa un *MouseEvent* quan es realitza un clic al botó.

- ***Add food.*** Permet afegir un aliment a l'àpat. Per fer-ho cal: introduir el tipus o la categoria de l'aliment, el nom[10] d'aquest i, finalment, seleccionar l'opció d'afegir l'aliment.

  Aquest procés es pot repetir tantes vegades com es desitgi (per tal d'afegir tants aliments com s'hagin ingerit en l'àpat en qüestió) abans d'afegir l'àpat al sistema definitivament.

- ***Add symptom.*** Permet afegir un símptoma a l'àpat. Per fer-ho cal: introduir la severitat del símptoma experimentat en funció d'un valor comprès en un rang del 1 al 5, el nom del símptoma en llenguatge natural (de la mateixa forma que per un aliment) i, finalment, seleccionar l'opció d'afegir el símptoma.

[10] La introducció del nom de l'aliment o símptoma al sistema s'ha decidit implementar mitjançant l'ús de llenguatge natural per part de l'usuari, el que suposa un avanç significatiu en la usabilitat d'un sistema d'aquestes característiques. A més, s'ha decidit tenint en compte la possibilitat d'incorporar en el futur un *framework*, com *Rasa* o *dialogflow*, que permeti la creació d'assistents o *chatbots* que s'encarreguin d'agafar el text i analitzar-lo per tal de sofisticar i millorar el sistema.

Aquest procés es pot repetir tantes vegades com es desitgi (per tal d'afegir tants símptomes com s'hagin experimentat) abans d'afegir l'àpat al sistema definitivament.

- ***Add meal.*** Afegeix l'àpat al sistema i retorna a l'usuari a la pantalla del seu perfil.

En la següent imatge es mostren els desplegables implementats per a la selecció del tipus d'aliment i de la severitat de la simptomatologia experimentada.

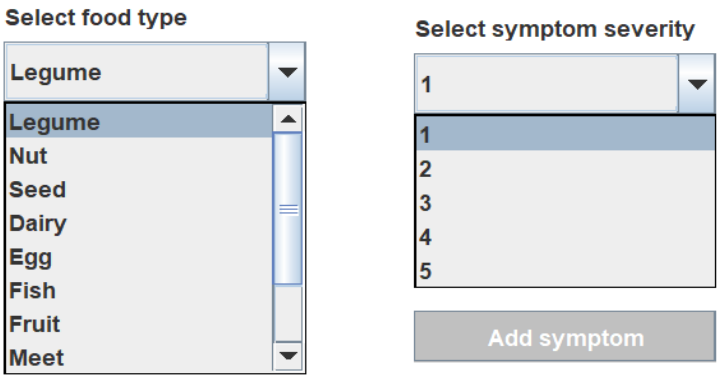


Figura 35. Desplegables.

A més, per tal d'informar a l'usuari que l'aliment o el símptoma que es vol afegir a l'àpat s'ha introduït correctament, s'han implementat els missatges pertinents en forma de finestra emergent.

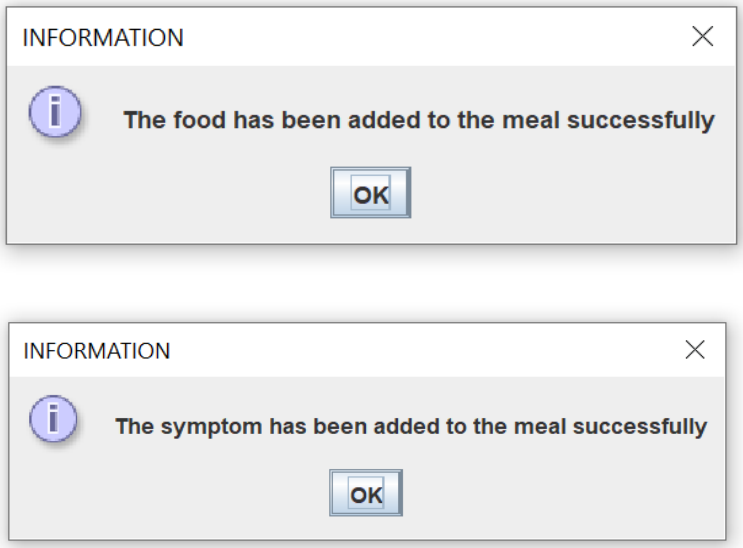


Figura 36. Control d'accions.

4.4.3 *Implementació del motor d'inferència*

Un cop vinculada la base de coneixement i creada la interfície que permet a l'usuari proporcionar la memòria de treball, es procedeix a implementar el codi que forma part del motor d'inferència que permet respondre als requisits plantejats.

Per obtenir resposta als requisits plantejats s'utilitzen majoritàriament **consultes SPARQL** que mostren el resultat per pantalla utilitzant la funció *showQuery*.

```
protected static void showQuery(Model m, String q) {
   Query query = QueryFactory.create( q );
   query.serialize(new IndentedWriter(System.out,true)) ;

   try(QueryExecution qexec = QueryExecutionFactory.create(query,
model)){
       org.apache.jena.query.ResultSet rs = qexec.execSelect();
       ResultSetFormatter.out(System.out, rs, query);
   } catch (Exception e) {
       e.printStackTrace();
   }
}
```

Codi 3. Funció *showQuery*.

En primer lloc, s'implementa un menú que mostra les funcionalitats del sistema.

```
do {
      System.out.println("\n---------MENU---------");
      System.out.println("1. General data types");
      System.out.println("2. Specific data information");
      System.out.println("4. Patient data");
      System.out.println("3. Other functionalities");
      System.out.println("5. Close");
      System.out.println("Choose an option: ");
} while (op != 4);
```

Codi 4. Menú principal.

Com s'aprecia en el fragment de codi anterior, el sistema s'organitza en quatre mòduls diferenciats, estructurats de la mateixa forma que els requisits funcionals.

Aquests mòduls són els següents:

- ***General data types.*** Mostra la informació general que forma la base de coneixement, així com la jerarquia d'aquesta.
- ***Specific data information.*** Proporciona resposta a qüestions específiques sobre instàncies de l'ontologia.
- ***Patient data.*** Inclou totes les consultes que es poden realitzar sobre un pacient, així com la funcionalitat principal del sistema: el diagnòstic.
- ***Other functionalities.*** Permet realitzar operacions bàsiques sobre l'ontologia que no corresponen a l'usuari (tasques d'administrador).

4.4.3.1 *General data types*

En el primer mòdul, s'implementa un menú que permet seleccionar sobre quina entitat (classe) principal de l'ontologia es volen conèixer les seves subclasses. El resultat es mostra sempre de forma jeràrquica, utilitzant la classe *ClassHierarchy* esmentada en l'apartat 4.4.1. Aquest mòdul respon als requisits funcionals del rang comprès entre RF-T1 i RF-T5, ambdós inclosos.

4.4.3.1.1 Cd'ú 01.ShowClassInformation

```
do {
      System.out.println("\n---------GENERAL DATA TYPES---------");
      System.out.println("1. Food types");
      System.out.println("2. Nutrient types");
      System.out.println("3. Symptom types");
      System.out.println("4. Intolerance types");
      System.out.println("5. Diet types");
      System.out.println("6. Close");
      System.out.println("Choose an option: ");

      op = Integer.parseInt(scan.nextLine());

      switch (op) {
      case 1:
            OntClass food = model.getOntClass( NS + "Food" );
            new ClassHierarchy().showClassHierarchy( System.out, food);
            break;
      case 2:
            OntClass nutrient = model.getOntClass( NS + "Nutrient" );
            new ClassHierarchy().showClassHierarchy( System.out,
nutrient);
            break;
      case 3:
            OntClass symptom = model.getOntClass( NS + "Symptom" );
            new ClassHierarchy().showClassHierarchy( System.out,
symptom);
            break;
      case 4:
            OntClass intolerance = model.getOntClass( NS + "Intolerance"
);
            new ClassHierarchy().showClassHierarchy( System.out,
intolerance);
            break;
      case 5:
            OntClass diet = model.getOntClass( NS + "Diet" );
            new ClassHierarchy().showClassHierarchy( System.out, diet);
            break;
      }
} while (op != 6);
```

Codi 5. Cd'ú *01.ShowClassInformation*.

4.4.3.2 *Specific data information*

En aquest mòdul s'inclouen consultes específiques sobre instàncies de l'ontologia que no formen part de la classe *Patient*, que responen als requisits funcionals RF-X1, RF-X2 i RF-X3.

4.4.3.2.1 Cd'ú 02.ShowNutrientsFromFood

```
System.out.println("Enter food's name: ");
String food = scan.nextLine();
StmtIterator i = model.getIndividual(NS + food).listProperties();
while(i.hasNext()) {
  Statement s = (Statement) i.next();
  if (!s.getObject().isLiteral()) {
      if(s.getPredicate().getLocalName().equals("hasNutrient"))
            System.out.println(food+"
"+s.getPredicate().getLocalName()+" "+s.getResource().getLocalName());
  }
}
```

Codi 6. Cd'ú *02.ShowNutrientsFromFood*.

4.4.3.2.2 Cd'ú 03.ShowForbiddenNutrientsFromDiet

```
System.out.println("Enter diet's name: ");
String diet = scan.nextLine();
StmtIterator i2 = model.getIndividual(NS + diet).listProperties();
while(i2.hasNext()) {
  Statement s = (Statement) i2.next();
  if (!s.getObject().isLiteral()) {
      if(s.getPredicate().getLocalName().equals("hasForbidden"))
         System.out.println(diet+" "+s.getPredicate().getLocalName()+"
"+s.getResource().getLocalName());
  }
}
```

Codi 7. Cd'ú *03.ShowForbiddenNutrientsFromDiet.*

4.4.3.2.3 Cd'ú 04.ShowTreatmentForIntolerance

```
System.out.println("Enter intolerance's name: ");
String intolerance = scan.nextLine();
StmtIterator i3 = model.getIndividual(NS +intolerance).listProperties();
while(i3.hasNext()) {
  Statement s = (Statement) i3.next();
  if (!s.getObject().isLiteral()) {
      if(s.getPredicate().getLocalName().equals("intoleranceHasTreatment
         System.out.println(diet+" "+s.getPredicate().getLocalName()+"
diet type "+s.getResource().getLocalName());
}
```

Codi 8. Cd'ú *04.ShowTreatmentForIntolerance.*

4.4.3.3 *Patient data*

Finalment, aquest mòdul implementa totes les consultes que es poden realitzar sobre la instància d'un pacient registrat al sistema, les quals responen als requisits funcionals del rang comprès entre RF-P1 i RF-P9, ambdós inclosos.

4.4.3.3.1 Cd'ú 05.GetPatient

Per fer-ho, s'implementa la funció *getPatient*() que identifica i indica l'existència de la instància d'un pacient corresponent a un ID donat, el qual es pren com a input de totes les consultes del mòdul.

```
boolean p = false;
while(!p) {
      System.out.println("Enter patient's ID: ");
      String id = scan.nextLine();
      p = getPatient(id);
      if(!p)
            System.out.println("ERROR: This patient does not exist");
      else
            System.out.println("OK: This patient exists");
            p = true;
}


public static boolean getPatient(String id) {
      ExtendedIterator  individuals = model.listIndividuals();
      boolean find = false;
       while(individuals.hasNext() && find == false){
              Individual ind =
model.getIndividual(individuals.next().toString());
              StmtIterator it = ind.listProperties();
              while (it.hasNext() && find == false) {
                  Statement s = (Statement) it.next();
                  if (s.getObject().isLiteral() &&
s.getLiteral().getLexicalForm().toString().contains(id)) {
                        find = true;
                        patient = ind;
                  }
              }
        }
      return find;
}
```

Codi 9. Cd'ú *05.GetPatient.*

4.4.3.3.2 Cd'ú 06.ShowPatientData

Un cop s'ha comprovat l'existència del pacient al sistema, es mostra un menú amb totes les opcions de consulta disponibles.

```
do {
      System.out.println("\n---------PATIENT OPTIONS---------");
      System.out.println("1. Patient previous diet");
      System.out.println("2. Patient meals");
      System.out.println("3. Patient diagnose");
      System.out.println("4. Patient treatment");
      System.out.println("5. Close");
      System.out.println("Choose an option: ");
} while (op != 5);
```

Codi 10. Cd'ú *06.ShowPatienData.*

4.4.3.3.3 Cd'ú 07.ShowPatientPreviousDiet

Aquesta opció permet conèixer si el client està o no realitzant alguna dieta específica i, en cas que ho estigui i aquesta presenti alguna restricció, es mostra també per pantalla. Implementa el requisit funcional RF-P1.

```
prefix = "PREFIX :<http://www.owl-
ontologies.com/diagnose_food_diary.owl#>";
queryString =
      prefix + "SELECT ?diet ?restriction WHERE {"
      + "?x :hasID \""+id+"\" . "
      + "?x :hasDiet ?diet . "
      + "OPTIONAL {?diet :hasForbidden ?restriction}"
      + "}";
showQuery( model, queryString);
```

Codi 11. Cd'ú *07.ShowPatientPreviousDiet.*

4.4.3.3.4 Cd'ú 08.ShowPatientMeals

Aquesta funcionalitat permet conèixer tota la informació rellevant dels àpats del pacient, ja que implementa els requisits RF-P2, RF-P3, RF-P4, RF-P5, RF-P6 i RF-P7.

```
prefix = "PREFIX :<http://www.owl-
ontologies.com/diagnose_food_diary.owl#>"
+ "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>";
queryString =
      prefix + "SELECT ?meal ?food ?symptom ?severity WHERE {"
      + "?meal rdf:type :Meal . "
      "?patient :hasID \""+id+"\" ."
      + "?meal :hasPatient ?patient ."
      + "?meal :hasFood ?food ."
      + "OPTIONAL {?meal :hasSymptom ?symptom ."
      + "OPTIONAL {?symptom :hasSeverity ?severity }}"
      + "}ORDER BY ?meal";
showQuery( model, queryString);
```

Codi 12. Cd'ú *08.ShowPatientMeals.*

```
String prefix = "PREFIX :<http://www.owl-
ontologies.com/diagnose_food_diary.owl#>"
+ "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>";
String queryString =
      prefix + "SELECT (COUNT(?meal) AS ?NELEMENTS) WHERE {"
      + "?meal rdf:type :Meal . "
      + "?meal :hasPatient ?patient ."
      + "?patient :hasID \""+id+"\" "
      + "}";
```

Codi 13. Nombre d'àpats.

```
String prefix = "PREFIX :<http://www.owl-
ontologies.com/diagnose_food_diary.owl#>"
+ "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>";
String queryString =
      prefix + "SELECT (COUNT(?food) AS ?NELEMENTS) WHERE {"
      + "?meal rdf:type :Meal . "
      + "?meal :hasPatient ?patient ."
      + "?patient :hasID \""+id+"\" ."
      + "?meal :hasFood ?food "
      + "}";
```

Codi 14. Nombre d'aliments.

```
String prefix = "PREFIX :<http://www.owl-
ontologies.com/diagnose_food_diary.owl#>"
+ "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>";
String queryString =
      prefix + "SELECT (COUNT(?symptom) AS ?NELEMENTS) WHERE {"
      + "SELECT DISTINCT ?meal ?symptom WHERE {"
      + "?meal rdf:type :Meal . "
      + "?meal :hasPatient ?patient ."
      + "?patient :hasID \""+id+"\" ."
      + "?meal :hasFood ?food ."
      + "?meal :hasSymptom ?symptom }"
      + "}";
```

Codi 15. Nombre de símptomes.

4.4.3.3.4.1 Exemple de PatientMeals

```
 1 PREFIX  :     <http://www.owl-ontologies.com/diagnose_food_diary.owl#>
 2 PREFIX  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 3
 4 SELECT  ?meal ?food ?symptom
 5 WHERE
 6   { ?meal     rdf:type     :Meal .
 7     ?patient  :hasID       "22222222B" .
 8     ?meal     :hasPatient  ?patient ;
 9               :hasFood     ?food
10     OPTIONAL
11       { ?meal  :hasSymptom  ?symptom }
12   }
-----------------------------------------
| meal              | food    | symptom   |
=========================================
| :Meal_Patient2_21 | :Banana | :Vomit    |
| :Meal_Patient2_21 | :Apple  | :Vomit    |
| :Meal_Patient2_22 | :Apple  |           |
| :Meal_Patient2_23 | :Banana | :Vomit    |
| :Meal_Patient2_23 | :Banana | :Diarrhea |
-----------------------------------------
Total of meals: 3
Total of food: 4
Total of symptoms: 3
```

Figura 37. Exemple: *Patient meals*.

En aquest darrer exemple s'observa que tant el càlcul del nombre d'aliments total com el del nombre de símptomes total tenen en compte que s'han d'excloure els resultats repetits, doncs un mateix símptoma o aliment que forma part d'un mateix àpat es pot veure duplicat en els resultats de consulta general sobre tota la informació de l'àpat en qüestió.

Així doncs, en l'exemple anterior on tenim els següents àpats:

Meal	**Food**	**Symptom**
Meal_Patient2_21	Banana Apple	Vomit
Meal_Patient2_22	Apple	
Meal_Patient2_23	Banana	Vomit Diarrhea
TOTAL: 3	TOTAL: 4	TOTAL: 3

Taula 43. Exemple: *Patient meals*.

S'aprecia doncs que el càlcul té en compte els aspectes que s'esmentaven, excloent una de les combinacions resultants de concatenar Banana amb Vomit i Apple amb Vomit del primer àpat, però que no exclou la simptomatologia també de tipus Vomit del tercer.

4.4.3.3.5 Cd'ú 09.ShowPatientDiagnose

Aquest mòdul realitza les consultes necessàries per obtenir un diagnòstic basat en els àpats realitzats pel pacient. S'ha decidit considerar que es pateix intolerància **a partir d'un 25% de forma arbitrària.**

Implementa el requisit funcional RF-P8 i tots els seus subrequisits.

```
DecimalFormat df = new DecimalFormat("#.##");



//FRUCTOSE
int total_fructose = totalNutrient("Fructose");
int total_symptom_fructose = totalNutrientSymptom("Fructose");
if (total_fructose == 0)
      System.out.print("The patient have not eaten any food with
fructose.\n");
else if (total_symptom_fructose== 0)
      System.out.print("The patient have not had any symptom related
with fructose.\n");
else {
      float fructose_intolerance = ((float) total_symptom_fructose /
(float) total_fructose)*100;
      System.out.print("Fructose intolerance: "
+df.format(fructose_intolerance) + "%\n");
}
if(fructose_intolerance > 25) {
      OntClass fi = model.getOntClass( NS + "Fructose_Intolerance");
      patient.addProperty(hasDiagnose, fi);
}



//GLUTEN
int total_gluten = totalNutrient("Gluten");
int total_symptom_gluten = totalNutrientSymptom("Gluten");
if (total_gluten == 0.0)
      System.out.print("The patient have not eaten any food with
gluten.\n");
else if (total_symptom_gluten == 0)
      System.out.print("The patient have not had any symptom related
with gluten.\n");
else {
      float gluten_intolerance = ((float) total_symptom_gluten / (float)
total_gluten)*100;
      System.out.print("Gluten intolerance: "
+df.format(gluten_intolerance) + "%\n");
}
if(gluten_intolerance > 25) {
      OntClass gi = model.getOntClass( NS + "Gluten_Intolerance");
      patient.addProperty(hasDiagnose, gi);
}



//LACTOSE
int total_lactose = totalNutrient("Lactose");
int total_symptom_lactose = totalNutrientSymptom("Lactose");
if (total_lactose == 0)
      System.out.print("The patient have not eaten any food with
```

```
lactose.\n");
else if (total_symptom_lactose == 0)
      System.out.print("The patient have not had any symptom related
with lactose.\n");
else {
      float lactose_intolerance = ((float) total_symptom_lactose /
(float) total_lactose)*100;
                              System.out.print("Lactose intolerance: "
+df.format(lactose_intolerance) + "%\n");
}
if(lactose_intolerance > 25) {
      OntClass li = model.getOntClass( NS + "Lactose_Intolerance");
      patient.addProperty(hasDiagnose, li);
}



//SORBITOL
int total_sorbitol = totalNutrient("Sorbitol");
int total_symptom_sorbitol = totalNutrientSymptom("Sorbitol");
if (total_sorbitol == 0)
      System.out.print("The patient have not eaten any food with
sorbitol.\n");
else if (total_symptom_sorbitol == 0)
      System.out.print("The patient have not had any symptom related
with sorbitol.\n");
else {
      float sorbitol_intolerance = ((float) total_symptom_sorbitol /
(float) total_sorbitol)*100;
      System.out.print("Sorbitol intolerance: "
+df.format(sorbitol_intolerance) + "%\n");
}
if(sorbitol_intolerance > 25) {
      OntClass si = model.getOntClass( NS + "Sorbitol_Intolerance");
      patient.addProperty(hasDiagnose, si);
}




StmtIterator i = model.getIndividual(NS +
patient.getLocalName()).listProperties();
while(i.hasNext()) {
  Statement s = (Statement) i.next();
  if (!s.getObject().isLiteral()) {
      if(s.getPredicate().getLocalName().equals("hasDiagnose")) {
                  System.out.println(patient.getLocalName()+"
"+s.getPredicate().getLocalName()+" "+s.getResource().getLocalName());
      }
   }
}
```

Codi 16. Cd'ú *09.ShowPatientDiagnose.*

```
String queryString =
      prefix + "SELECT ?meal ?nutrient WHERE {"
      + "?meal rdf:type :Meal . "
      + "?meal :hasPatient ?patient ."
      + "?patient :hasID \""+id+"\" ."
      + "?meal :hasFood ?food ."
      + "?food :hasNutrient ?nutrient"
      + "}";

org.apache.jena.query.Query query = QueryFactory.create( queryString );
try(QueryExecution qexec = QueryExecutionFactory.create(query, model)){
   org.apache.jena.query.ResultSet rs = qexec.execSelect();
   while(rs.hasNext()) {
if(rs.next().getResource("nutrient").getLocalName().equals(nutrient))
      total++;
}
} catch (Exception e) {
     e.printStackTrace();
}
```

Codi 17. Nombre total d'àpats que contenen un nutrient concret.

```
String queryString =
      prefix + "SELECT ?meal ?nutrient WHERE {"
      + "?meal rdf:type :Meal . "
      + "?meal :hasPatient ?patient ."
      + "?patient :hasID \""+id+"\" ."
      + "?meal :hasFood ?food ."
      + "?food :hasNutrient ?nutrient ."
      + "?meal :hasSymptom ?symptom"
      + "}";

org.apache.jena.query.Query query = QueryFactory.create( queryString );
try(QueryExecution qexec = QueryExecutionFactory.create(query, model)){
   org.apache.jena.query.ResultSet rs = qexec.execSelect();
   while(rs.hasNext()) {
if(rs.next().getResource("nutrient").getLocalName().equals(nutrient))
      total++;
}
} catch (Exception e) {
     e.printStackTrace();
}
```

Codi 18. Nombre total d'àpats que contenen un nutrient concret i han presentat alguna simptomatologia.

4.4.3.3.6 Cd'ú 10.ShowPatientTreatment

Finalment, aquest mòdul estableix un tractament al pacient en cas de que aquest hagi rebut un diagnòstic prèviament. Implementa el requisit funcional RF-P9.

```
boolean fructose = false;gluten = false;lactose = false;sorbitol = false
StmtIterator i2 = model.getIndividual(NS +
patient.getLocalName()).listProperties();
while(i2.hasNext()) {
      Statement s = (Statement) i2.next();
  if (!s.getObject().isLiteral()) {
      if(s.getPredicate().getLocalName().equals("hasDiagnose")) {

      if(s.getResource().getLocalName().equals("Fructose_Intolerance"))
            fructose=true;
      elseif(s.getResource().getLocalName().equals("Gluten_Intolerance"))
            gluten=true;
      elseif(s.getResource().getLocalName().equals("Lactose_Intolerance"))
            lactose=true;
      elseif(s.getResource().getLocalName().equals("Sorbitol_Intolerance"))
            sorbitol=true;
      }
   }
}
if(fructose) {
      OntClass ft = model.getOntClass( NS + "Fructose_free");
      patient.addProperty(patientHasTreatment, ft);
}
if(gluten) {
      OntClass gt = model.getOntClass( NS + "Gluten_free");
      patient.addProperty(patientHasTreatment, gt);
}
if(lactose) {
      OntClass lt = model.getOntClass( NS + "Lactose_free");
      patient.addProperty(patientHasTreatment, lt);
}
if(sorbitol) {
      OntClass st = model.getOntClass( NS + "Sorbitol_free");
      patient.addProperty(patientHasTreatment, st);
}

if(!fructose && !gluten && !lactose && !sorbitol){
      System.out.print("The patient has not been assigned to any
treatment");
}else {
      StmtIterator i3 = model.getIndividual(NS +
patient.getLocalName()).listProperties();
      while(i3.hasNext()) {
            Statement s = (Statement) i3.next();
            if (!s.getObject().isLiteral()) {
      if(s.getPredicate().getLocalName().equals("patientHasTreatment")
               System.out.println(patient.getLocalName()+"
"+s.getPredicate().getLocalName()+" "+s.getResource().getLocalName());
            }
      }
}
```

Codi 19. Cd'ú *10.ShowPatientTreatment.*

4.4.3.4 *Other functionalities*

Finalment, aquest mòdul permet a l'administrador del sistema incloure noves instàncies de les classes *Food, Nutrient* i *Symptom*, per tal de simplificar el procés com a administrador del sistema.

La implementació d'aquestes funcionalitats no respon a cap dels requisits funcionals específicament però resulta útil i pot facilitar la incorporació de més funcionalitats en un futur.

```
do {
      System.out.println("\n---------FUNCTIONALITIES---------");
      System.out.println("1. Add a new food");
      System.out.println("2. Add a new nutrient");
      System.out.println("3. Add a new symptom");
      System.out.println("4. Close");
      System.out.println("Choose an option: ");

      op = Integer.parseInt(scan.nextLine());

      switch (op) {
      case 1:
            System.out.println("Enter food's name: ");
            String newFood = scan.nextLine();
            System.out.println("Enter food's type: ");
            String type = scan.nextLine();
            OntClass t = model.getOntClass( NS + type);
            Individual i = model.createIndividual( NS + newFood, t);
            break;
      case 2:
            System.out.println("Enter nutrient's name: ");
            String newNutrient = scan.nextLine();
            System.out.println("Enter food's type: ");
            String type2 = scan.nextLine();
            OntClass t2 = model.getOntClass( NS + type2);
            Individual i2 = model.createIndividual( NS + newNutrient,
t2);
            break;
      case 3:
            System.out.println("Enter symptom's name: ");
            String newSymptom = scan.nextLine();
            System.out.println("Enter symptom's type: ");
            String type3 = scan.nextLine();
            OntClass t3 = model.getOntClass( NS + type3);
            Individual i3 = model.createIndividual( NS + newSymptom,
t3);
            break;
      }
} while (op != 4);
```

Codi 20. *Other functionalities.*

5 Avaluació

5.1 Avaluació de l'ontologia

En primer lloc, abans de realitzar una avaluació del sistema basat en el coneixement, cal comprovar la correctesa d'aquesta base que, en aquest cas, és l'ontologia construïda.

5.1.1 *Mètriques*

Per tal de garantir la qualitat d'un projecte s'utilitzen mètriques adients que avaluïn les característiques del sistema o, en aquest cas, la correctesa del model ontològic.

Les mètriques que s'avaluen sobre l'ontologia construïda són les següents:

Coherència	En un sentit lògic, és coherent el que no incorre en contradicció. Per exemple: un argument, una teoria, un conjunt de proposicions, etc., són coherents en la mesura que no comporten contradicció o no es contradiuen entre sí.
Consistència	Quan la coherència de la qual es parla és referent a un sistema formal axiomàtic, s'anomena consistència [28].

Taula 44. Mètriques.

En el cas d'un model ontològic, aquest serà coherent i consistent si l'estructura dissenyada no suposa cap contradicció entre les diferents entitats d'aquest. És a dir, totes les instàncies poden ser definides perfectament, sense produir cap ambigüitat.

5.1.2 *Avaluació*

Per avaluar les mètriques descrites s'utilitzen els raonadors, els quals automatitzen el procés de cerca d'incoherències en els axiomes definits pel sistema i faciliten enormement el procés de depuració d'aquest.

En aquest cas, s'ha utilitzat un Plugin propi de Protégé: OntoDebug [35], i s'han seguit els passos següents per tal de depurar el codi.

1. Iniciar el raonador. En aquest cas s'ha utilitzat Pellet [37], el qual és un raonador OWL 2 de codi obert basat en Java.

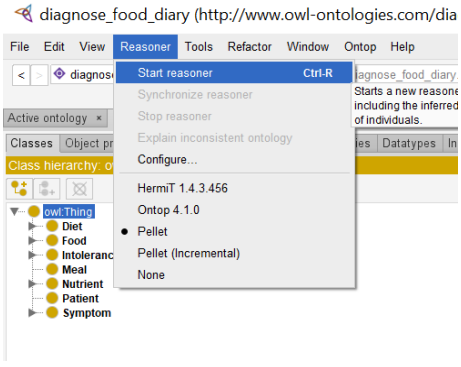


Figura 38. Pellet *Reasoner*.

2. A continuació, s'ha canviat el mode de vista a d'*Asserted* a *Inferred*, el que permet observar amb detall les característiques que el raonador extreu de cada classe, propietat o instància.

Figura 39. Mode *Inferred*.

La vista Debugger ha permès realitzar l'anàlisi de les característiques de l'ontologia que no eren consistents i modificar-les en els casos que era necessari.

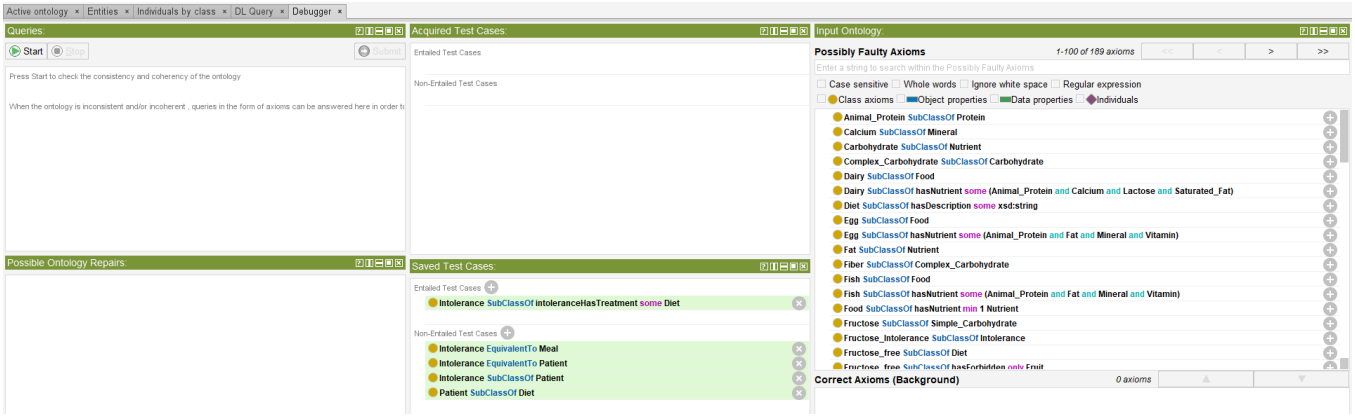


Figura 40. Mode *Debugger*.

Com que el procés de creació d'una ontologia és iteratiu, tal i com s'ha especificat en l'apartat 2.4.1, la realització de certes modificacions no ha suposat un imprevist, ja que la depuració de l'ontologia s'ha realitzat abans d'utilitzar-la en el sistema expert.

3. Finalment, s'ha obtingut la validació de la coherència i la consistència del model ontològic, mètriques comentades al punt 5.1.1.

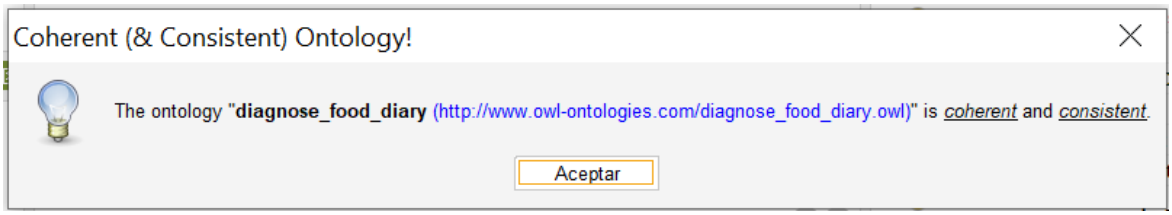


Figura 41. Coherència i consistència de l'ontologia.

5.2 Avaluació del sistema expert

L'avaluació del sistema expert construït es realitza validant el compliment o no dels requisits plantejats en l'apartat 4.1.

A continuació s'enumeren novament.

ID	Requisit
RF-T1	Mostrar els tipus principals d'aliments i la seva classificació.
RF-T2	Mostrar els tipus principals de nutrients i la seva classificació.
RF-T3	Mostrar els tipus principals de simptomatologies i la seva classificació.
RF-T4	Mostrar els tipus principals d'intoleràncies i la seva classificació.
RF-T5	Mostrar els tipus principals de dietes i la seva classificació.
RF-X1	Mostrar els nutrients d'un aliment X.
RF-X2	Mostrar els aliments/nutrients prohibits per una dieta X.
RF-X3	Mostrar el tractament que s'aplica a la reacció adversa (intolerància) a un tipus d'aliment/nutrient X.
RF-P1	Mostrar el nom de la dieta prèvia a l'estudi (en cas que existeixi) i restriccions d'aquesta (en cas que existeixin) realitzada pel pacient.
RF-P2	Mostrar els àpats realitzats pel pacient.
RF-P3	Mostrar el nombre total d'àpats realitzats pel pacient.
RF-P4	Mostrar el nombre total d'aliments ingerits pel pacient.
RF-P5	Mostrar el nombre total de símptomes experimentats pel pacient.
RF-P6	Mostrar els aliment/s i la simptomatologia (en cas que es presenti) de cada àpat.
RF-P7	Mostrar la severitat de la simptomatologia experimentada pel pacient (en cas de presentar-se).
RF-P8	Realitzar un diagnòstic (si es pot concloure) de la simptomatologia del pacient.
RF-P8.1	Calcular el percentatge de probabilitats de patir intolerància a la fructosa.
RF-P8.2	Calcular el percentatge de probabilitats de patir intolerància al gluten.
RF-P8.3	Calcular el percentatge de probabilitats de patir intolerància a la lactosa.
RF-P8.4	Calcular el percentatge de probabilitats de patir intolerància al sorbitol.
RF-P9	Definir quin tractament (dieta) s'aplica al pacient.
RF-U1	El sistema permet a l'usuari identificar-se.

RF-U2	El sistema permet a l'usuari registrar un aliment en un àpat.
RF-U3	El sistema permet a l'usuari registrar un símptoma en un àpat.
RF-U4	El sistema permet a l'usuari afegir un àpat.

Alguns exemples dels outputs obtinguts de cadascuna de les tasques avaluades s'inclouen en l'Annex C.

Requisit	Input	Output esperat	Validació
RF-T1	Ontologia en format RDF/XML	Llistat en forma jeràrquica de les classes que formen la classe *Food*	Correcte
RF-T2	Ontologia en format RDF/XML	Llistat en forma jeràrquica de les classes que formen la classe *Nutrient*	Correcte
RF-T3	Ontologia en format RDF/XML	Llistat en forma jeràrquica de les classes que formen la classe *Symptom*	Correcte
RF-T4	Ontologia en format RDF/XML	Llistat en forma jeràrquica de les classes que formen la classe *Intolerance*	Correcte
RF-T5	Ontologia en format RDF/XML	Llistat en forma jeràrquica de les classes que formen la classe *Diet*	Correcte
RF-X1	Nom d'un aliment[11] en format String	Llistat de nutrients que conté l'aliment	Correcte

[11] L'aliment ha d'estar instanciat al sistema ontològic.

RF-X2	Nom d'una dieta[12] en format String	Llistat d'aliments que prohibeix la dieta	Correcte
RF-X3	Nom d'una intolerància[13] en format String	Dieta que s'usa com a tractament a la intolerància especificada	Correcte
RF-P1	ID del pacient en format String	Dieta que està realitzant, en cas que en realitzi una, i restriccions d'aquesta dieta	Correcte
RF-P2	ID del pacient en format String	Llistat d'àpats realitzats pel pacient	Correcte
RF-P3	ID del pacient en format String	Nombre total d'apats realitzats	Correcte
RF-P4	ID del pacient en format String	Nombre total d'aliments ingerits	Correcte
RF-P5	ID del pacient en format String	Nombre total de símptomes experimentats	Correcte
RF-P6	ID del pacient en format String	Aliment/s i simptomatologia (en cas que es presenti) de cada àpat	Correcte
RF-P7	ID del pacient en format String	Severitat de la simptomatologia, en cas de que es presenti	Correcte
RF-P8	ID del pacient en format String	Diagnòstic (intolerància) que es conclou (si es pot concloure) de la simptomatologia del	Correcte

[12] La dieta ha d'estar instanciada al sistema ontològic.

[13] La intolerancia ha d'estar instanciada al sistema ontològic.

		pacient	
RF-P8.1	ID del pacient en format String	Percentatge de probabilitats de patir intolerància a la fructosa	Correcte
RF-P8.2	ID del pacient en format String	Percentatge de probabilitats de patir intolerància al gluten	Correcte
RF-P8.3	ID del pacient en format String	Percentatge de probabilitats de patir intolerància a la lactosa	Correcte
RF-P8.4	ID del pacient en format String	Percentatge de probabilitats de patir intolerància al sorbitol	Correcte
RF-P9	ID del pacient en format String	Tractament (dieta) que s'aplica al pacient	Correcte
RF-U1	ID del pacient en format String	El pacient s'identifica en el sistema	Correcte
RF-U2	Tipus de l'aliment (seleccionat en un desplegable) i nom de l'aliment en format String	El pacient registra un aliment a un àpat	Correcte
RF-U3	Severitat del símptoma (seleccionat en un desplegable) i nom de del símptoma en format String	El pacient registra un símptoma a un àpat	Correcte
RF-U4	Clic sobre el botó *Add meal*	El pacient afegeix un àpat	Correcte

Taula 45. Avaluació del sistema.

6 Conclusions

6.1 Limitacions

En la realització d'aquest treball he pogut apreciar les limitacions que comporta en un sistema d'aquestes característiques el fet de no poder accedir a una base de dades real i/o de cert volum de dades. Aquest fet impossibilita l'aplicació d'algorismes més potents sobre el sistema i limita l'anàlisi que es podria arribar a realitzar.

Tanmateix, la realització d'aquest treball m'ha permès aprofundir en el coneixement dels sistemes basats en regles i, també, d'adonar-me un cop més de la importància que suposa el *big data* a la nostra societat a dia d'avui; així de com el camp de la intel·ligència artificial pot aprofitar-se'n, aportant grans solucions i millores en molts àmbits de les nostres vides.

6.2 Futur

Per altra banda, desitjo i espero poder continuar aquest projecte, ja que es tracta d'un projecte que pot arribar a ser molt ambiciós i útil per a la societat, però que requereix de més temps i recursos. Així doncs, en cas de disposar de més recursos en un futur, posseint una gran quantitat de dades i pacients, es planteja:

- Donar resposta a preguntes més complexes.
- Representar formalment la informació associada als pacients a partir del sistema construït i extreure'n estadístiques.
- Entrenar algun sistema de *Machine Learning* sobre el conjunt de dades estructurades.

Per altra banda, també es planteja la possibilitat de desenvolupar la interfície d'usuari a nivell de *front-end*, construint una aplicació més còmode d'usar mitjançant tecnologies de disseny web (utilitzant llenguatge HTML5 i CSS3 per al disseny principal, *frameworks* com *Vue.js* o *React* que es basen en *JavaScript* per implementar i/o vincular les diverses funcionalitats i *Bootstrap* per assegurar un disseny responsiu).

6.3 Aprenentatge

Finalment, la realització d'aquest projecte m'ha aportat un gran aprenentatge tant a nivell professional com a nivell personal.

Per una banda, a nivell tècnic he tingut l'ocasió d'aprofundir en el temari del món de la intel·ligència artificial, la qual és una de les branques de la informàtica que més interès i curiositat em desperta. Realitzar aquest treball m'ha permès familiaritzar-me amb instruments com Protégé i metodologies (*Ontology Engineering*) per representar el coneixement; també especialitzar-me en algunes eines que permeten tractar-lo, com Jena o SPARQL.

A més, degut a la naturalesa de l'àmbit d'estudi del projecte, he pogut millorar la meva capacitat de recerca, ja que el projecte requeria d'una investigació prèvia que difereix molt del plantejament dels projectes que havia realitzat fins al moment.

Tanmateix, això no ha impedit que el meu aprenentatge comprengui també el desenvolupament de les tasques principals per a un projecte de software: captura de

requisits, anàlisi, disseny, construcció o avaluació. El treball de fi de grau ha estat una oportunitat per aplicar les tècniques apreses per a la gestió de projectes i la formalització de la informació requerida, mitjançant l'ús de diagrames UML, per exemple.

Per altra banda, és a nivell personal on m'emporto la satisfacció més gran. Tal i com comentava a la introducció d'aquest projecte, la temàtica escollida em resulta de gran interès perquè m'afecta de primera mà. El fet que el projecte hagi nascut d'una inquietud pròpia ha significat que el disseny de tot el projecte es basi en la implementació d'una idea on jo he estat l'encarregada de les decisions essencials en tot moment.

Tot això ha fomentat la meva creativitat, autogestió i presa de decisions. Amb l'ajuda del meu tutor, he definit el projecte des de zero i he anat solucionant totes les dificultats i limitacions que han anat sorgint. Ha estat el primer projecte d'aquesta complexitat que he realitzat de forma individual i m'ha resultat molt útil per millorar la meva organització i autosuficiència a l'hora d'haver de realitzar cada fase, on sovint he hagut de gestionar també amb l'estrès davant de la resolució de problemes.

En conclusió, m'emporto un gran aprenentatge de la realització d'aquest treball que espero aplicar ben aviat en futurs projectes, així com poder aportar tots els coneixements adquirits al món laboral.

7 Referències

7.1 Bibliografia

[1] Thomas R. Gruber. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*.

[2] Obrst, Leo & Ceusters, Werner & Mani, Inderjeet & Ray, Steve & Smith, Barry. (2007). *The Evaluation of Ontologies.*

[3] Guarino, Nicola. (1998). *Formal Ontologies and Information Systems.*

[4] Scientia et Technica Año XVII, No 50, Universidad Tecnológica de Pereira. (2012). *Metologías y métodos para la construcción de ontologías*. ISSN 0122-1701

[5] Alves, Miguel & Damásio, Carlos & Correia, Nuno. (2015). *SPARQL Commands in Jena Rules*. 518. 253-262. 10.1007/978-3-319-24543-0_19.

[6] Gavrilova, Tatiana & Leshcheva, Irina & Strakhovich, Elvira. (2014). *Gestalt principles of creating learning business ontologies for knowledge codification. Knowledge Management Research & Practice.* 13. 10.1057/kmrp.2013.60.

[7] Smith, Barry. (2004). *Beyond Concepts: Ontology as Reality Representation.* In Proceedings of the International Conference on Formal Ontology and Information Systems, Turin, 4-6 November 2004.

[8] Line Pouchard, Nenad Ivezic and Craig Schlenoff. (2000). *Ontology Engineering for Distributed Collaboration in Manufacturing*. In Proceedings of the AIS2000 conference, March 2000.

[9] Angles, Renzo; Gutiérrez, Claudio. (2008). *The Expressive Power of SPARQL.* ISWC 2008, LNCS 5318, Springer , pàg. 114-129.

[10] Cañas, J. J.; Salmerón, L.; Gámez, P. (2001). *El factor humano.* En: J. Lorés (ed.). La interacción persona-ordenador. Lérida: AIPO.

[11] Carroll, J. M. (1997). *Human-computer interaction: psychology as a science of design.* Annual Review of Psychology (n.° 48, págs. 61-83).

[12] Chang, D.; Dooley, L.; Tuovinen, J. E. (2002). *Gestalt theory in visual screen design – a new look at an old subject.* A: 7th World Conference on Computers in Education (del 29 de juliol al 3 d'agost del 2001). Copenhague.

7.2 Recursos web

[13] Zugasti, Ana. *Intolerancia alimentaria.* https://www.elsevier.es/es-revista-endocrinologia-nutricion-12-articulo-intolerancia-alimentaria-S157509220971407X [Consulta: 9/03/2021]

[14] Sánchez, Dennis & de las Heras, Román. *Sistemas expertos basados en reglas.* https://sites.google.com/site/sistemasexpertosunah/home/sistemas-expertos-basados-en-reglas [Consulta: 9/03/2021]

[15] Academia Americana de Médicos de Familia. *Nutrición: cómo llevar un diario de comida.* https://es.familydoctor.org/nutricion-como-llevar-un-diario-de-comida/ [Consulta: 14/03/2021]

[16] TOP DOCTORS. *Intolerancia alimentaria*. https://www.topdoctors.es/diccionario-medico/intolerancia-alimentaria# [Consulta: 14/03/2021]

[17] Franz Inc. *Java Jena API Tutorial for AllegroGraph.* https://franz.com/agraph/support/documentation/current/java-tutorial/jena-tutorial.html [Consulta: 18/03/2021]

[18] Semantic creatures. *Getting Starting with Apache Jena and Eclipse.* https://semanticreatures.com/2015/04/04/getting-started-with-apache-jena-and-eclipse/ [Consulta: 15/03/2021]

[19] Deborah McGuinness and Frank van Harmelen. «OWL Web Ontology Language Overview». *W3C Recommendation for OWL, the Web Ontology Language*. World Wide Web Consortium. https://www.w3.org/TR/2004/REC-owl-features-20040210/ [Consulta: 18/03/2021].

[20] Bañón, José María. Universidad Carlos III de Madrid. (2013). *Estudio del manejo de ontologías para la monitorización de pacientes.* https://core.ac.uk/download/pdf/29405475.pdf [Consulta: 18/03/2021].

[21] W3C. (2012). *OWL 2 Web Ontology Language Quick Reference Guide.* https://www.w3.org/TR/2012/REC-owl2-quick-reference-20121211/#Names.2C_Prefixes.2C_and_Notation [Consulta: 04/04/2021].

[22] W3C (2013). *SPARQL 1.1 Overview.* https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/ [Consulta: 13/04/2021].

[23] McCarthy, Philip. (2015). *Search RDF data with SPARQL.* https://developer.ibm.com/languages/java/articles/j-sparql/ [Consulta: 24/04/2021].

[24] *Uses of Interface com.hp.hpl.jena.query.ResultSet.* thttps://docs.huihoo.com/jena/ARQ/javadoc/com/hp/hpl/jena/query/class-use/ResultSet.html [Consulta: 5/05/2021].

[25] R. Ishida, F. Yergeau, M. J. Düst, M. Wolf, T. Texin, Editors, W3C Recommendation. (2005). *Character Model for the World Wide Web 1.0: Fundamentals.* http://www.w3.org/TR/2005/REC-charmod-20050215/ [Consulta: 5/05/2021].

[26] J. Melton, A. Malhotra, N. Walsh, Editors, W3C Recommendation. (2007). *XQuery 1.0 and XPath 2.0 Functions and Operators*. http://www.w3.org/TR/2007/REC-xpath-functions-20070123/ [Consulta: 5/05/2021].

[27] Wordpress: ianasemanticweb. (2011). *Robotics ontology study*. https://ianasemanticweb.wordpress.com/category/uncategorized/ [Consulta: 10/05/2021].

[28] CENTENO, S. (2020). *"Coherencia"; en: Diccionario filosófico de Centeno.* https://sites.google.com/site/diccionariodecenteno/c/coherencia [Consulta: 12/05/2021].

[29] Visual Paradigm. *Requirement Analysis Techniques.* https://www.visual-paradigm.com/guide/requirements-gathering/requirement-analysis-techniques/#:~:text=Requirement%20Analysis%2C%20also%20known%20as,requirements%20gathering%20or%20requirements%20capturing. [Consulta: 12/05/2021].

[30] Blog: Metodología Gestión de Requerimientos. *Técnicas para Identificar Requisitos Funcionales y No Funcionales*. https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificar-requisitos-funcionales-y-no-funcionales [Consulta: 12/05/2021].

7.3 Ontologia

[31] Ontobee. *Symptom ontology*. Disponible a: http://purl.obolibrary.org/obo/symp.owl [Consulta: 15/03/2021].

7.4 Programari

[32] The Eclipse IDE 2021-03. Disponible a: https://www.eclipse.org/downloads/packages/installer {Consulta: 30/03/2021].

[33] Protégé Desktop v.5.5.0. Disponible a: https://protege.stanford.edu/products.php#desktop-protege/ [Consulta: 23/03/2021].

[34] Apache Jena 3.17. Disponible a: https://jena.apache.org/download/index.cgi [Consulta: 30/03/2021].

[35] OntoDebug 0.2.2 Plugin. Disponible a: https://protegewiki.stanford.edu/wiki/OntoDebug [Consulta: 10/04/2021].

[36] OntoGraf 1.0.1 Plugin. Disponible a: https://protegewiki.stanford.edu/wiki/OntoGraf [Consulta: 10/04/2021].

[37] Pellet Reasoner. Disponible a: https://www.w3.org/2001/sw/wiki/Pellet [Consulta: 14/04/2021].

[38] WindowBuilder. Disponible a: https://www.eclipse.org/windowbuilder/ [Consulta: 20/04/2021].

[39] VisualParadigm 16.2. Disponible a: https://www.visual-paradigm.com/download/ [Consulta: 10/10/2020].

Annexes

Annex A. Manual d'usuari

Aquest manual pretén mostrar les instruccions per realitzar les diverses funcionalitats del sistema en base a un exemple hipotètic. Els passos a realitzar es mostren en els següents subapartats.

Instal·lació de programari

Per poder utilitzar el sistema es necessita haver realitzat prèviament els punts que segueixen:

1. Instal·lació de *Protégé* [33].
2. Instal·lació d'*Eclipse IDE* [32].
3. Descarregar i configurar Apache Jena a Eclipse. L'enllaç per descarregar l'arxiu i els passos per configurar Jena a l'IDE es poden troben a:

   https://semanticreatures.com/2015/04/04/getting-started-with-apache-jena-and-eclipse/ [18].

Registrar pacients al sistema

El primer pas és registrar algun pacient a la base de coneixement. En aquest cas, la base de coneixement correspon a l'ontologia i per afegir un pacient es pot realitzar de manera senzilla i còmoda mitjançant l'ús de *Protégé*. Per fer-ho, cal:

- Obrir l'aplicació d'escriptori de *Protégé* i obrir l'arxiu owl corresponent. Aquesta opció es troba a *File* > *Open*. A continuació, es selecciona l'arxiu *diagnosi_food_diary.owl*.

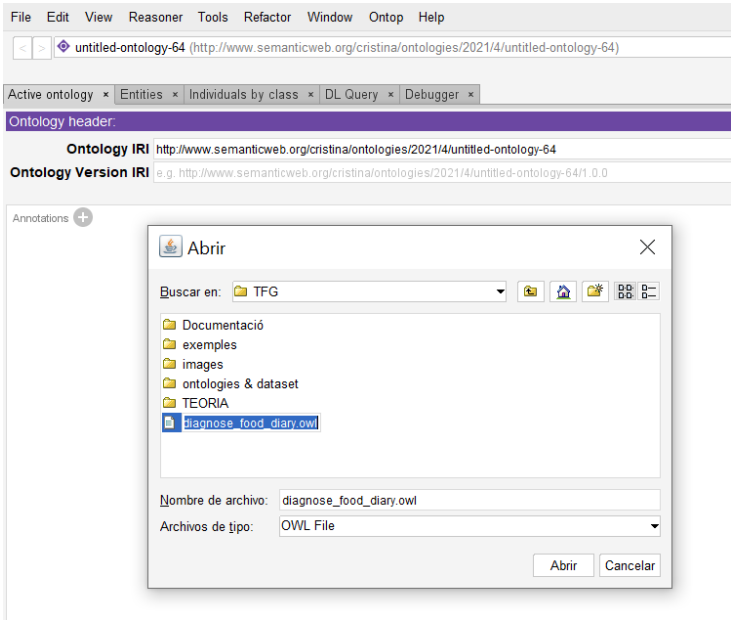


Figura 42. Manual d'ús: Importació de l'axiu *diagnosi_food_diary.owl*.

- Un cop obert l'arxiu de tipus *owl*, es selecciona la pestanya *Entities*. Dins d'aquesta s'observen diverses pestanyes més, on per defecte es mostrarà la de *Classes*. A continuació, es selecciona la classe *Patient* i es selecciona el camp *Instances*, ara buit ja que no hi ha cap pacient registrat.

  A la Figura 43 s'observen les principals classes que formen l'ontologia.

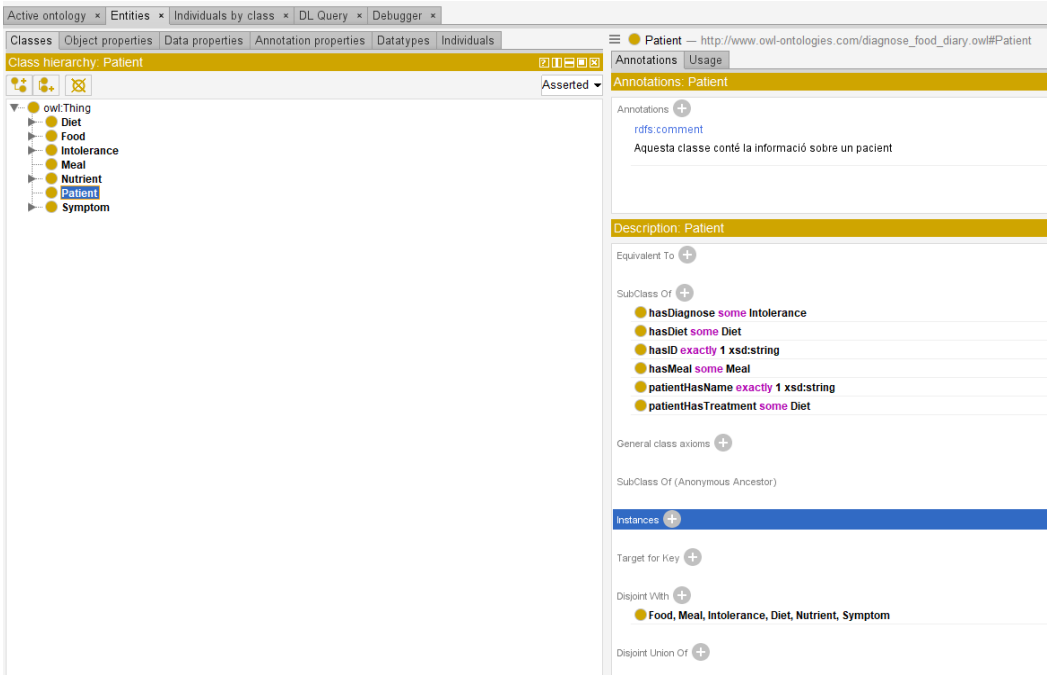


Figura 43. Manual d'ús: Instància d'un pacient.

- Per afegir un nou pacient al sistema es clica sobre la icona amb el símbol + al costat del camp *Instances* i es selecciona la icona per afegir una nova instància a la finestra emergent que ha aparegut.

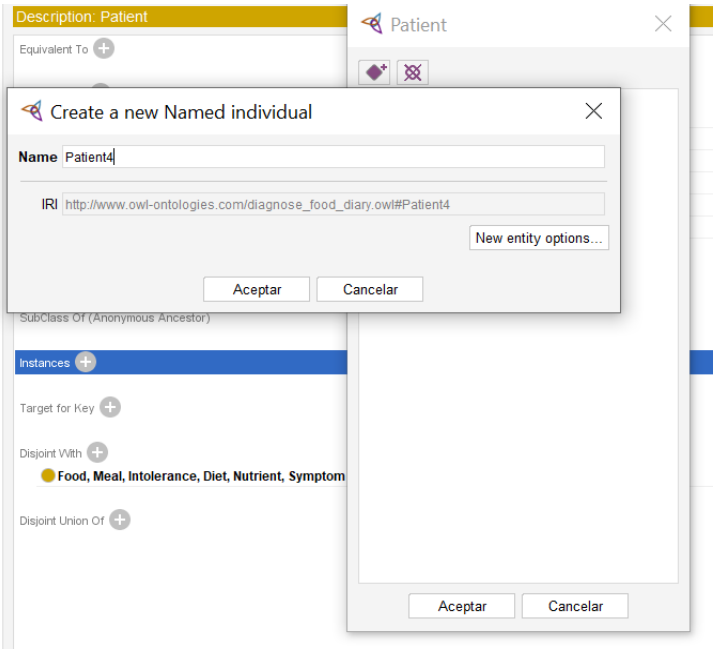


Figura 44. Manual d'ús: Instància d'un pacient II.

Una nova finestra emergent apareixerà i permetrà introduir el nom de la nova instància. En aquest cas, tal i com es mostra a la Figura 44, el nom correspon a *Patient4*.

A continuació es selecciona l'opció *Aceptar* i, un cop més, *Aceptar*. La instància queda afegida al sistema.

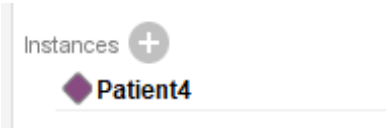


Figura 45. Manual d'ús: Instància d'un pacient III.

De la mateixa manera que s'ha inserta una nova instància d'un pacient al sistema, es pot realitzar per a qualsevol de la resta d'entitats de l'ontologia.

- Si es realitza un clic sobre la instància de pacient creada, l'entorn de treball es dirigeix a la pestanya *Individuals*, la qual presenta diverses opcions d'edició de la instància.

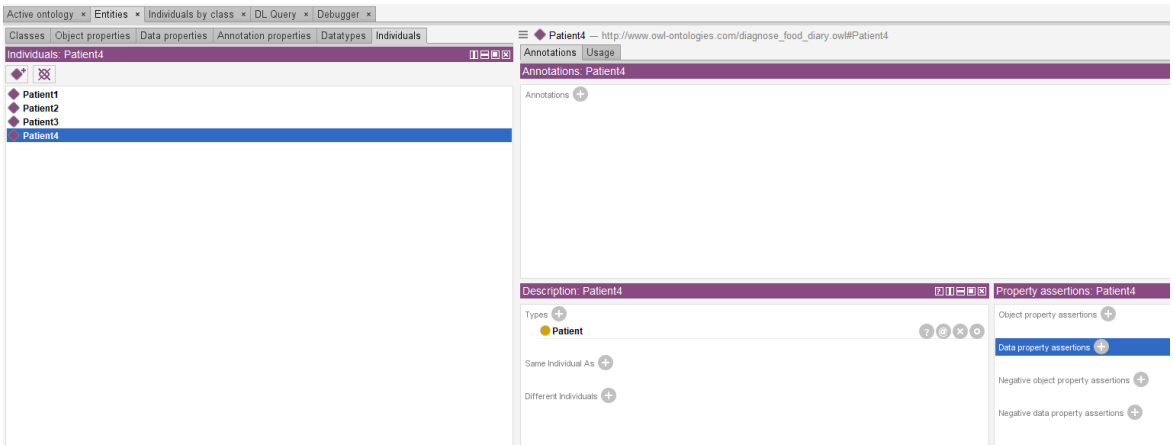


Figura 46. Manual d'ús: Instància d'un pacient IV.

A la dreta de la pantalla s'observa el camp *Property assertions: Patient4*, el qual permet afegir propietats sobre la instància seleccionada.

- En aquest cas es decideix afegir l'identificador del pacient, ja que és essencial per a utilitzar posteriorment el sistema.

  Per fer-ho, es prem el botó amb el símbol + al costat de l'opció *Data property assertions*, ja que en aquest cas es tracta d'una propietat de tipus *data*.

  A continuació apareixerà una finestra emergent que permetrà seleccionar la propietat que es vol afegir d'entre totes les propietats d'aquest tipus definides pel model ontològic. En aquest cas, es selecciona la propietat *hasID*, s'introdueix en el camp de la dreta de la finestra el valor que es vol atribuir a la propietat i es selecciona el tipus (*type*) *string*.

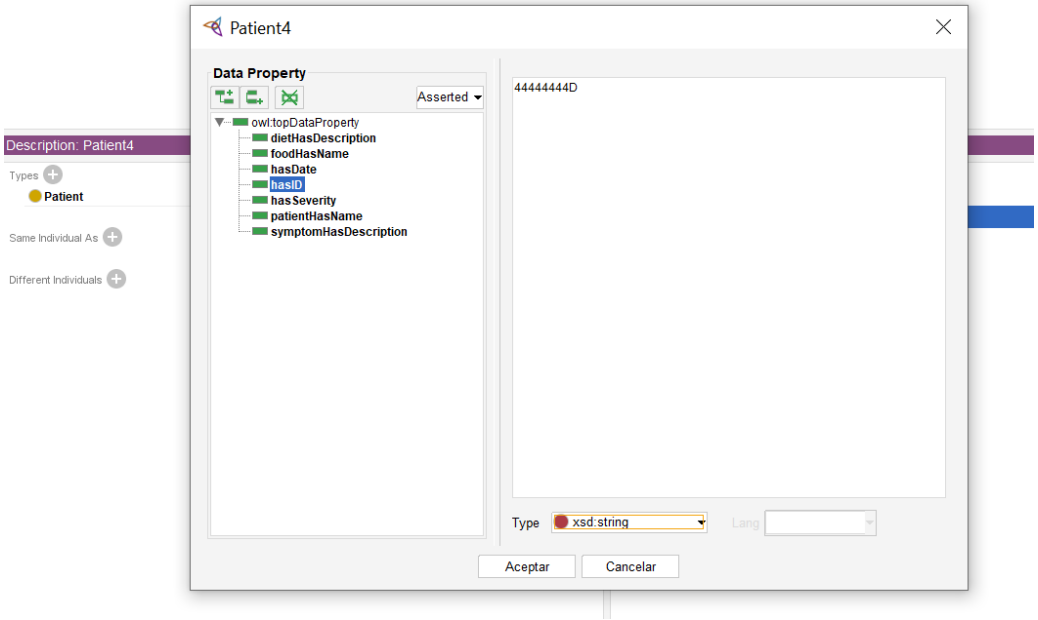


Figura 47. Manual d'ús: Instància d'un pacient V.

Finalment, es selecciona *Aceptar*. La propietat queda definida a la instància.

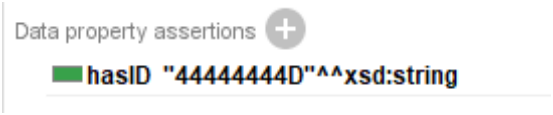


Figura 48. Manual d'ús: Instància d'un pacient VI.

- Per altra banda, es decideix afegir el tipus de dieta que presenta el pacient actualment: una dieta vegetariana. Per fer-ho, es realitza un procés similar a l'anterior.

  Es prem el botó amb el símbol + al costat de l'opció *Object property assertions*, ja que en aquest cas es tracta d'una propietat de tipus *object*.

  A continuació apareixerà una finestra emergent amb dos camps: el primer permet introduir per teclat el nom de la propietat que es vol afegir (aquesta ha d'estar definida pel model ontològic) i el segon introduir el valor (que es tracta d'una instància, ja que es tracta d'una propietat de tipus *object*) atribuïda a la propietat. En aquest cas, s'introdueix la propietat *hasDiet* i el valor *Vegetarian*.

  Finalment es prem *Aceptar* i la propietat queda registrada a la instància del pacient.

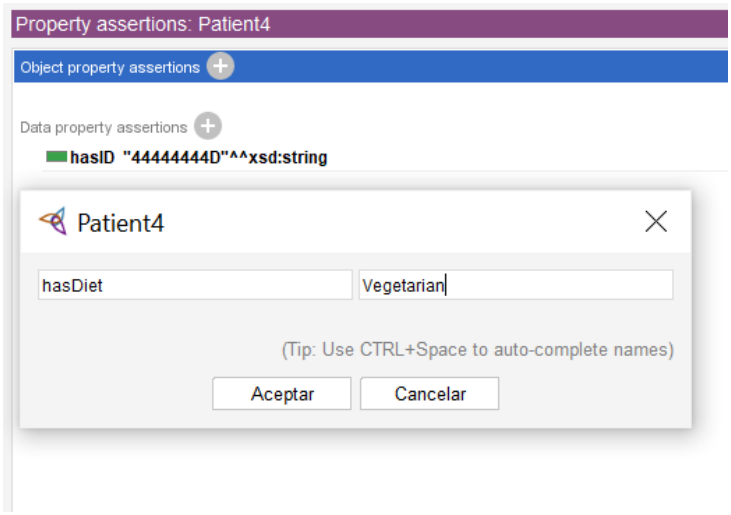


Figura 49. Manual d'ús: Instància d'un pacient VII.

- Per últim, es guarda l'arxiu *owl* en el format RDF/XML. Per fer-ho es selecciona la ruta *File* > *Save as*. Apareix un desplegable on es selecciona el format esmentat i es clica el botó *Aceptar*.

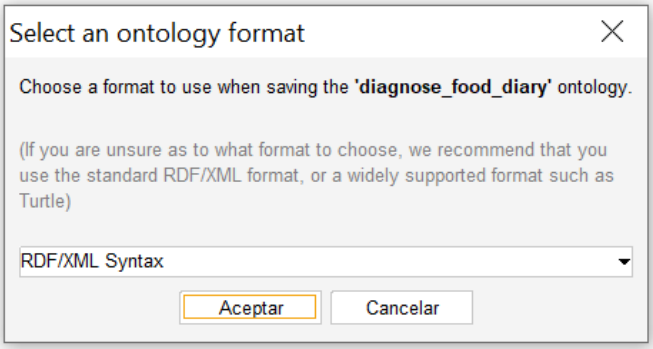


Figura 50. Manual d'ús: Guardar en format RDF/XML.

Execució del sistema

Un cop registrats els pacients a la base de coneixement es procedeix a realitzar els passos generals següents per executar el projecte.

1. Descomprimir el projecte *DiagnoseFoodDiary.zip* i importar-lo a l'entorn d'*Eclipse*.
2. Comprovar que el sistema *JRE System Library* té actualment seleccionat la versió JavaSE-1.8 per assegurar la compatibilitat amb el projecte.

En la següent figura es visualitza de forma general l'estructura del projecte.

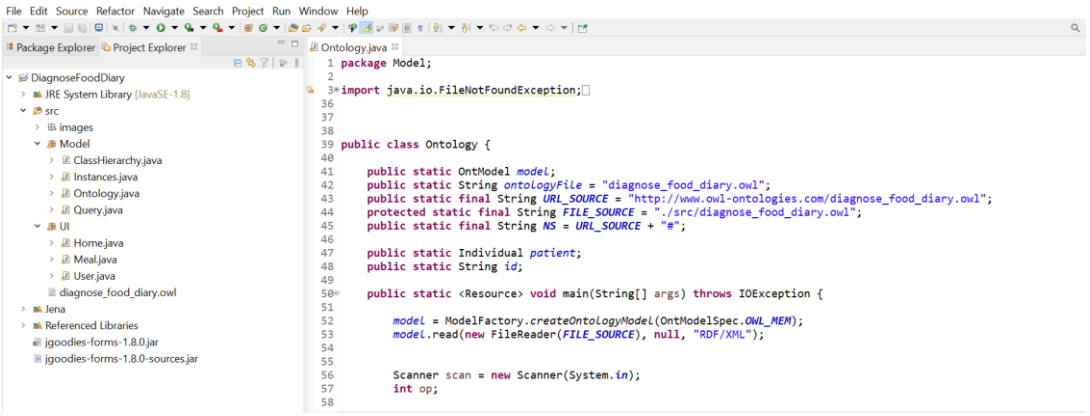


Figura 51. Manual d'ús: Entorn del sistema.

3. Accedir a la carpeta /src.
    3.1 En cas de voler executar la interfície que permet a l'usuari introduir informació a la base de fets (memòria de treball), accedir al package *UI*.
    3.1.1 Compilar i executar l'arxiu *Home.java* utilitzant l'opció *Run as* → *Java Application*.
    3.2 En cas de voler executar el motor d'inferència que permet a l'administrador obtenir la informació general del coneixement proporcionat al sistema i/o tractar amb les dades d'un pacient, accedir al package *Model*.
    3.2.1 Compilar i executar l'arxiu *Ontology.java*.

Afegir instàncies a la base de coneixement

De la mateixa forma que s'han registrat els pacients a la base del sistema des de *Protégé*, per tal d'introduir aliments, nutrients o símptomes específics s'ha implementat l'opció per fer-ho directament des del propi sistema.

Per fer-ho cal seguir els passos següents:

1. Executar l'arxiu *Ontology.java* i seleccionar la quarta opció del menú principal.

2. Seleccionar en el segon menú mostrat sobre quina de les classes principals de l'ontologia es vol introduir una nova instància.

3. El sistema demanarà el nom que es vol assignar a la nova instància i el tipus al que correspon.

```
---------MENU---------
1. General data types
2. Specific data information
3. Patient data
4. Other functionalities
5. Close
Choose an option:
4

---------OTHER FUNCTIONALITIES---------
1. Add a new food
2. Add a new nutrient
3. Add a new symptom
4. Close
Choose an option:
1
Enter food's name:
Tomato
Enter food's type:
Vegetable
```

Figura 52. Manual d'ús: Introduir instàncies.

Informació sobre la base de coneixement

Per tal de conèixer informació bàsica sobre les instàncies afegides al sistema (aquelles que no corresponen a pacients), s'enumeren els passos per executar les accions que s'assignen als següents casos d'ús.

Cd'ú 01. ShowClassInformation

1. Executar l'arxiu *Ontology.java* i seleccionar la primera opció del menú principal.

2. Seleccionar en el segon menú mostrat sobre quina de les classes principals de l'ontologia es vol conèixer la jerarquia de subclasses que la composen.

3. S'obté el resultat per consola. Veure a l'Annex C els *outputs* dels requisits funcionals RF-T1, RF-T2, RF-T3, RF-T4 i RF-T5.

```
---------MENU---------
1. General data types
2. Specific data information
3. Patient data
4. Other functionalities
5. Close
Choose an option:
1

---------GENERAL DATA TYPES---------
1. Food types
2. Nutrient types
3. Symptom types
4. Intolerance types
5. Diet types
6. Close
Choose an option:
```

Figura 53. Manual d'ús del cd'ú 01.

Aquesta funcionalitat permet visualitzar l'estructura actual de l'ontologia importada al projecte, mostrant les classes que la composen de forma jeràrquica.

Els següents tres casos d'ús s'utilitzen per obtenir informació més específica sobre alguns dels aspectes de més interès de l'ontologia, el que permet conèixer les regles sobre les que es basen els diagnòstics i tractaments del pacients.

Cd'ú 02.ShowNutrientsFromFood

1. Executar l'arxiu *Ontology.java* i seleccionar la segona opció del menú principal.

2. Seleccionar la primera opció del segon menú mostrat.

3. Introduir per teclat el nom de l'aliment sobre el qual es volen consultar els nutrients.

4. S'obté el resultat per consola. Veure a l'Annex C l'*output* del requisit funcional RF-X1.

```
---------MENU---------
1. General data types
2. Specific data information
3. Patient data
4. Other functionalities
5. Close
Choose an option:
2

---------SPECIFIC DATA INFORMATION---------
1. Nutrients of a specific food
2. Forbidden in a specific diet
3. Treatment of a specific intolerance
4. Close
Choose an option:
1
Enter food's name:
Apple
```

Figura 54. Manual d'ús del cd'ú 02.

L'aplicació mostrarà els nutrients registrats com a *Object Property* sobre la instància d'un aliment específic.

Cd'ú 03.ShowForbiddenNutrientsFromDiet

1. Executar l'arxiu *Ontology.java* i seleccionar la segona opció del menú principal.

2. Seleccionar la segona opció del segon menú mostrat.

3. Introduir per teclat el nom de la dieta sobre la qual es volen consultar els nutrients prohibits.

4. S'obté el resultat per consola. Veure a l'Annex C l'*output* del requisit funcional RF-X2.

```
---------MENU---------
1. General data types
2. Specific data information
3. Patient data
4. Other functionalities
5. Close
Choose an option:
2

---------SPECIFIC DATA INFORMATION---------
1. Nutrients of a specific food
2. Forbidden in a specific diet
3. Treatment of a specific intolerance
4. Close
Choose an option:
2
Enter diet's name:
Vegan
```

Figura 55. Manual d'ús del cd'ú 03.

L'aplicació mostrarà els nutrients prohibits registrats com a *Object Property* sobre la instància d'una dieta.

Cd'ú 04.ShowTreatmentForIntolerance

1. Executar l'arxiu *Ontology.java* i seleccionar la segona opció del menú principal.

2. Seleccionar la tercera opció del segon menú mostrat.

3. Introduir per teclat el nom de la intolerància sobre la qual es vol consultar el tractament.

4. S'obté el resultat per consola. Veure a l'Annex C l'*output* del requisit funcional RF-X3.

```
---------MENU---------
1. General data types
2. Specific data information
3. Patient data
4. Other functionalities
5. Close
Choose an option:
2

---------SPECIFIC DATA INFORMATION---------
1. Nutrients of a specific food
2. Forbidden in a specific diet
3. Treatment of a specific intolerance
4. Close
Choose an option:
3
Enter intolerance's name:
Lactose_Intolerance
```

Figura 56. Manual d'ús del cd'ú 04.

L'aplicació mostrarà el tractament registrat com a *Object Property* sobre la instància d'una intolerància.

Introducció d'àpats al sistema

Per poder realitzar consultes sobre un pacient cal introduir fets a la memòria de treball. Per fer-ho s'utilitza una interfície simple que, tot i que no es la base del treball, permet proporcionar la informació necessària de forma còmoda.

Així doncs, la implementació d'aquesta interfície ve donada per conveniència per facilitar el procés d'introducció d'informació al sistema. Tanmateix, tal i com es comentava a l'apartat de conclusions, aquest és un dels aspectes del projecte que es planteja desenvolupar a major escala en un futur.

A continuació es detallen els passos a seguir per introduir un àpat al sistema.

En primer lloc, l'usuari s'ha d'identificar al sistema seguint els passos:

1. Executar l'arxiu *Home.java* i introduir l'ID i el *password* en els camps corresponents.

2. Clicar el botó *Sign up*.

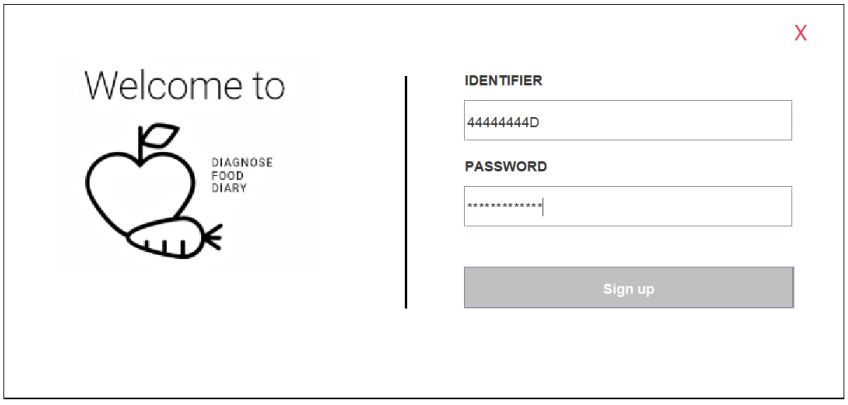


Figura 57. Manual d'ús: cd'ú 11.

Aquesta acció permet identificar que l'usuari està registrat al sistema com a pacient.

El pacient amb identificador 44444444D s'ha registrat al sistema tal i com s'ha mostrat en passos anteriors, per tant, el sistema l'identifica i permet iniciar el procés per registrar àpats.

La Figura 58 mostra la pantalla del perfil de l'usuari, on actualment només existeix l'opció per afegir un àpat.

Figura 58. Manual d'ús: Afegir un àpat.

Un cop seleccionada l'opció *Add meal*, es procedeix a introduir la informació corresponent al nou àpat realitzant els passos següents:

1. Seleccionar el tipus d'aliment utilitzant el desplegable.
2. Introduir el nom de l'aliment.
3. Clicar el botó *Add food.*
4. Repetir el procés tants cops com aliments es desitgi afegir a l'àpat.

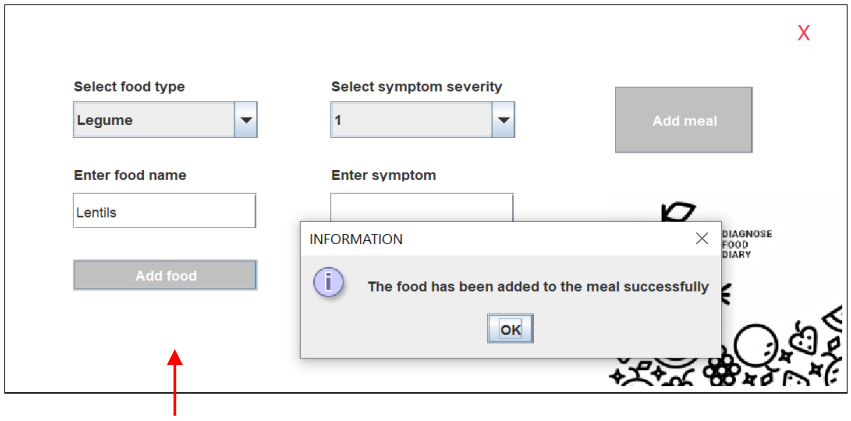


Figura 59. Manual d'ús: cd'ú 12.

Per introduir un símptoma a l'àpat, es realitzen els passos següents:

1. Seleccionar el grau de severitat del símptoma utilitzant el desplegable.
2. Introduir el nom de del símptoma.
3. Clicar el botó *Add symptom.*
4. Repetir el procés tants cops com símptomes es desitgi afegir a l'àpat.

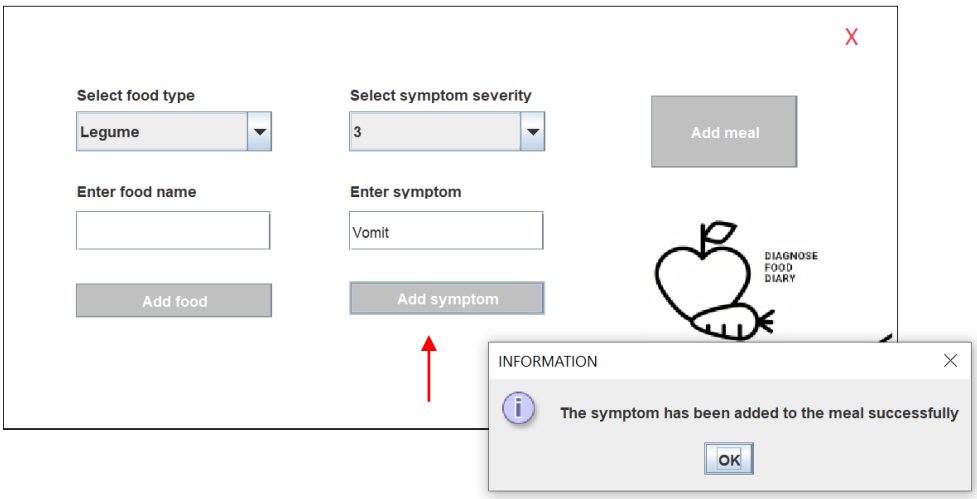


Figura 60. Manual d'ús: cd'ú 13.

Després de realitzar el cd'ú 12 (com a mínim un cop) i el cd'ú 13 tantes vegades com es desitgin, clicar el botó *Add meal* per afegir de forma definitiva l'àpat al sistema.

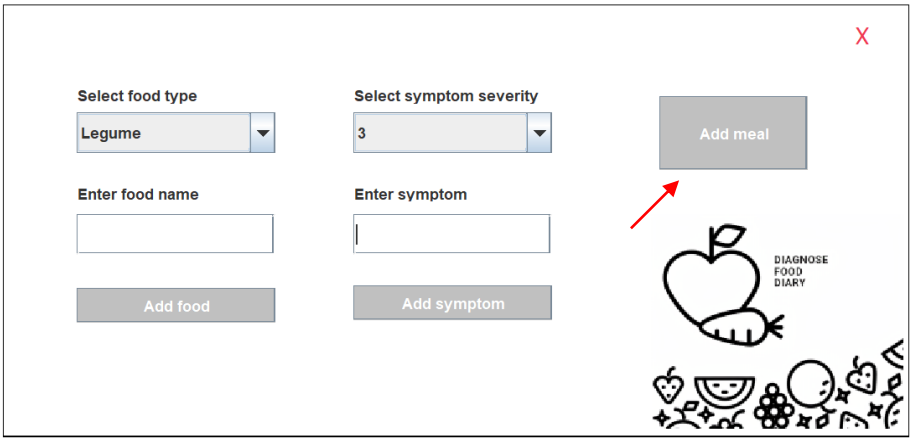


Figura 61. Manual d'ús: cd'ú 14.

Per a aquest exemple, els àpats que s'han afegit al sistema són els que es mostren a la Taula 46. S'han decidit incloure només deu àpats simplificant molt un cas hipotètic degut al volum de mostrar un cas més complex.

Meal	**Food**	**Symptom**
Meal_Patient4_1	Tomato	Vomit
	Apple	
Meal_Patient4_2	Soup	
	Omelette	
	Bread	
	Peanut_Butter	
	Yogurt	
Meal_Patient4_3	Lettuce	
	Carrot	
	Rice	
	Omelette	
	Yogurt	
	Apple	
Meal_Patient4_4	Banana	Gatroenteritis
Meal_Patient4_5	Omelette	
	Bread	
	Yogurt	
Meal_Patient4_6	Syrup	Stomach_Ache

	Pancake Banana	
Meal_Patient4_7	Orange_Juice Peanut_Butter	Nausea
Meal_Patient4_8	Pasta Cheese	
Meal_Patient4_9	Carrot_Cake Cheese	Stomach_Ache
Meal_Patient4_10	Strawberry	Diarrhea

Taula 46. Manual d'ús: Àpats d'un pacient.

D'entre aquests àpats, cal destacar els **nutrients més rellevants** que contenen cada un dels aliments afegits, aquells que poden ser causants de les intoleràncies que es volen detectar.

Food	**Nutrients**
Rice	Gluten
Apple	Fructose Sorbitol
Soup	Gluten
Lettuce	Sorbitol
Banana	Fructose
Yogurt	Lactose
Syrup	Sorbitol Fructose
Pancake	Gluten Sorbitol
Orange_Juice	Fructose
Bread	Gluten
Peanut_Butter	Sorbitol
Pasta	Gluten
Cheese	Lactose
Strawberry	Fructose
Carrot_Cake	Sorbitol

Taula 47. Manual d'ús: Nutrients dels àpats d'un pacient.

Funcionalitats sobre els pacients registrats

Finalment, es detallen les funcions de consulta sobre pacients que es poden realitzar com a administrador del sistema. En aquest cas s'utilitza l'exemple realitzat: pacient amb nom Patient4, identificador "44444444D" i els àpats detallats anteriorment.

En primer lloc, cal identificar el pacient sobre el que es volen realitzar les consultes al sistema. Per fer-ho es realitzen els passos següents:

1. Executar l'arxiu *Ontology.java* i seleccionar la tercera opció del menú principal.
2. Introduir per teclat l'ID del pacient sobre el qual es vol realitzar alguna consulta.

```
---------MENU---------
1. General data types
2. Specific data information
3. Patient data
4. Other functionalities
5. Close
Choose an option:
3
Enter patient's ID:
44444444D
OK: This patient exists
```

Figura 62. Manual d'ús: cd'ú 05.

A continuació, es mostra un menú amb les diverses opcions de consulta disponibles.

```
---------PATIENT OPTIONS---------
1. Patient previous diet
2. Patient meals
3. Patient diagnose
4. Patient treatment
5. Close
Choose an option:
```

Figura 63. Manual d'ús: cd'ú 06.

La primera funcionalitat permet consultar si el pacient està realitzant alguna dieta prèviament al diagnòstic i, en cas afirmatiu, quina i quines restriccions té.

En l'exemple realitzat el resultat obtingut es mostra en la Figura 64.

```
---------PATIENT OPTIONS---------
1. Patient previous diet
2. Patient meals
3. Patient diagnose
4. Patient treatment
5. Close
Choose an option:
1
  1 PREFIX  :     <http://www.owl-ontologies.com/diagnose_food_diary.owl#>
  2
  3 SELECT  ?diet ?restriction
  4 WHERE
  5   { ?x  :hasID    "44444444D" ;
  6         :hasDiet  ?diet
  7     OPTIONAL
  8       { ?diet  :hasForbidden  ?restriction }
  9   }
-----------------------------
| diet        | restriction |
=============================
| :Vegetarian | :Shellfish  |
| :Vegetarian | :Meet       |
| :Vegetarian | :Fish       |
-----------------------------
```

Figura 64. Manual d'ús: cd'ú 07.

Tal i com s'ha afegit en el registre del pacient anteriorment, el resultat mostra que el pacient amb ID "44444444D" està realitzant una dieta de tipus *Vegetarian*. També es mostren les restriccions aplicades a la instància d'aquesta, en aquest cas, al marisc, a la carn i al peix.

A continuació, s'accedeix a la segona opció del menú, on es troba tota la informació respecte als àpats que s'han registrat del pacient.

```
---------PATIENT OPTIONS---------
1. Patient previous diet
2. Patient meals
3. Patient diagnose
4. Patient treatment
5. Close
Choose an option:
2
  1 PREFIX  :     <http://www.owl-ontologies.com/diagnose_food_diary.owl#>
  2 PREFIX  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  3
  4 SELECT  ?meal ?food ?symptom ?severity
  5 WHERE
  6   { ?meal     rdf:type     :Meal .
  7     ?patient  :hasID       "44444444D" .
  8     ?meal     :hasPatient  ?patient ;
  9               :hasFood     ?food
 10     OPTIONAL
 11       { ?meal  :hasSymptom  ?symptom
 12         OPTIONAL
 13           { ?symptom  :hasSeverity  ?severity }
 14       }
 15   }
 16 ORDER BY ?meal
```

meal	food	symptom	severity
:Meal_Patient4_1	:Apple	:Vomit	
:Meal_Patient4_1	:Tomato	:Vomit	
:Meal_Patient4_10	:Strawberry	:Diarrhea	
:Meal_Patient4_2	:Bread		
:Meal_Patient4_2	:Omelette		
:Meal_Patient4_2	:Peanut_Butter		
:Meal_Patient4_2	:Soup		
:Meal_Patient4_2	:Yogurt		
:Meal_Patient4_3	:Apple		
:Meal_Patient4_3	:Bread		
:Meal_Patient4_3	:Carrot		
:Meal_Patient4_3	:Lettuce		
:Meal_Patient4_3	:Omelette		
:Meal_Patient4_3	:Rice		
:Meal_Patient4_3	:Yogurt		
:Meal_Patient4_4	:Banana	:Gastroenteritis	
:Meal_Patient4_5	:Omelette		
:Meal_Patient4_5	:Yogurt		
:Meal_Patient4_6	:Banana	:Stomach_Ache	"2"
:Meal_Patient4_6	:Pancake	:Stomach_Ache	"2"
:Meal_Patient4_6	:Syrup	:Stomach_Ache	"2"
:Meal_Patient4_7	:Orange_Juice	:Nausea	
:Meal_Patient4_7	:Peanut_Butter	:Nausea	
:Meal_Patient4_8	:Cheese		
:Meal_Patient4_8	:Pasta		
:Meal_Patient4_9	:Carrot_Cake	:Stomach_Ache	"2"
:Meal_Patient4_9	:Cheese	:Stomach_Ache	"2"

```
Total of meals: 10
Total of food: 27
Total of symptoms: 6
```

Figura 65. Manual d'ús: cd'ú 08.

En la Figura 65 s'observa tota la informació que s'ha afegit a l'apartat anterior, així com un recompte del total d'àpats, el total d'aliments i el total de símptomes.

Aquesta funcionalitat permet a l'administrador conèixer tota la informació rellevant sobre els àpats, molt útil abans de realitzar un diagnòstic o en cas de voler conèixer algun detall específic de la dieta del pacient.

Finalment, la tercera opció del menú ens permet realitzar un breu diagnòstic sobre els àpats del pacient.

```
---------PATIENT OPTIONS---------
1. Patient previous diet
2. Patient meals
3. Patient diagnose
4. Patient treatment
5. Close
Choose an option:
3
Total meals with fructose: 4
Total meals with symptoms and fructose: 3
Fructose intolerance: 75%


Total meals with gluten: 6
Total meals with symptoms and gluten: 1
Gluten intolerance: 16,67%


Total meals with lactose: 5
Total meals with symptoms and lactose: 1
Lactose intolerance: 20%


Total meals with sorbitol: 8
Total meals with symptoms and sorbitol: 5
Sorbitol intolerance: 62,5%


Patient4 hasDiagnose Sorbitol_Intolerance

Patient4 hasDiagnose Fructose_Intolerance
```

Figura 66. Manual d'ús: cd'ú 09.

En aquest cas, al tractar-se d'un registre d'àpats molt simplificat, el diagnòstic té menys valor de pes, doncs com més gran sigui el nombre de fets aportats a la memòria de treball més precisió i exactitud s'atribuirà al resultat.

Tanmateix, aquest exemple serveix per mostrar de forma senzilla l'anàlisi que es realitza sobre els àpats definits, on actualment (el sistema pretén ampliar el rang d'intoleràncies) estudia les probabilitats de patir quatre de les intoleràncies més comuns.

La Figura 66 mostra els percentatges obtinguts per a cadascuna de les possibles intoleràncies i atribueix un diagnòstic positiu en cas que aquesta superi el 25%, el que s'ha decidit de forma arbitrària i es podria adaptar a les recomanacions de l'expert que usi el software.

Finalment, l'última opció ens permet determinar quin tractament aplicar, el qual ve definit per una dieta específica que prohibeix el nutrient conflictiu segons la intolerància detectada.

```
---------PATIENT OPTIONS---------
1. Patient previous diet
2. Patient meals
3. Patient diagnose
4. Patient treatment
5. Close
Choose an option:
4
Patient4 patientHasTreatment Sorbitol_free
Patient4 patientHasTreatment Fructose_free
```

Figura 67. Manual d'ús: cd'ú 10.

En l'exemple realitzat, el pacient s'ha detectat que pateix intolerància a la fructosa i al sorbitol i, per tant, els tractaments recomanats que es mostren són una dieta lliure de fructosa i lliure de sorbitol.

La informació sobre cada dieta també es pretén seguir desenvolupant ara que l'estructura general de dades està ben definida.

Annex B. Arxius

En aquest apartat s'inclouen com a exemple alguns dels fragments de l'arxiu RDF/XML generat en la creació de l'ontologia mitjançant Protégé.

Prefixos i informació general

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.owl-ontologies.com/diagnose_food_diary.owl#"
     xml:base="http://www.owl-ontologies.com/diagnose_food_diary.owl"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:terms="http://purl.org/dc/terms/"
xmlns:diagnose_food_diary="http://www.owlontologies.com/diagnose_food_diary.owl#">
    <owl:Ontology rdf:about="http://www.owl-ontologies.com/diagnose_food_diary.owl">
        <dc:contributor>Cristina Llort</dc:contributor>
        <terms:description>This ontology defines the basic concepts to diagnose a
patient of a food intolerance. It also provides some treatments as a recommendation of a
specific diet life style.</terms:description>
    </owl:Ontology>
```

Axiomes generals

```
    <!--// General axioms //-->
    <rdf:Description>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#AllDisjointClasses"/>
        <owl:members rdf:parseType="Collection">
            <rdf:Description
rdf:about="http://www.owl-ontologies.com/diagnose_food_diary.owl#Diet"/>
            <rdf:Description
rdf:about="http://www.owl-ontologies.com/diagnose_food_diary.owl#Food"/>
            <rdf:Description
rdf:about="http://www.owl-ontologies.com/diagnose_food_diary.owl#Intolerance"/>
            <rdf:Description
rdf:about="http://www.owl-ontologies.com/diagnose_food_diary.owl#Meal"/>
            <rdf:Description
rdf:about="http://www.owl-ontologies.com/diagnose_food_diary.owl#Nutrient"/>
            <rdf:Description
rdf:about="http://www.owl-ontologies.com/diagnose_food_diary.owl#Patient"/>
            <rdf:Description
rdf:about="http://www.owl-ontologies.com/diagnose_food_diary.owl#Symptom"/>
        </owl:members>
    </rdf:Description>
</rdf:RDF>
```

Annex C. Outputs

En aquest apartat s'aporten els outputs obtinguts durant l'avaluació dels requisits als que respon el sistema expert.

Requisit RF-T1

Tipus principals d'aliments i la seva classificació.

```
Class:Sweet
Class:Meet
Class:Nut
Class:Shellfish
Class:Vegetable
Class:Fish
Class:Legume
Class:Egg
Class:Fruit
Class:Dairy
Class:Seed
```

Taula 48. Requisit RF-T1.

Requisit RF-T2

Tipus principals de nutrients i la seva classificació.

```
Class:Vitamin
Class:Protein
  Class:Plant_Protein
    Class:Gluten
  Class:Animal_Protein
Class:Mineral
  Class:Magnesium
  Class:Iron
  Class:Calcium
  Class:Potassium
Class:Fat
  Class:Unsaturated_Fat
  Class:Saturated_Fat
Class:Carbohydrate
  Class:Simple_Carbohydrate
    Class:Sorbitol
    Class:Lactose
    Class:Glucose
    Class:Fructose
  Class:Complex_Carbohydrate
    Class:Starch
    Class:Glycogen
    Class:Fiber
```

Taula 49. Requisit RF-T2.

Requisit RF-T3

Tipus principals de simptomatologies i la seva classificació.

```
Class:abdominal_symptom
  Class:abdominal_rigidity
  Class:pelvic_symptom
  Class:abdominal_tenderness
  Class:abdominal_lump
  Class:abdominal_discomfort
  Class:abdominal_swelling
  Class:abdominal_cramp
  Class:abdominal_mass
  Class:abdominal_distention
  Class:ascites
Class:digestive_system_symptom
  Class:flatulence
  Class:enteritis
  Class:belching
  Class:salivary_gland_symptom
  Class:distended_loops_of_intestines_on_rectal
  Class:feces_and_droppings_symptom
  Class:dyspepsia
  Class:abnormal_bowel_sound
  Class:bloating
  Class:diminished_gastro-intestinal_motility
  Class:gastroenteritis
  Class:vomiting
  Class:nausea
Class:urinary_system_symptom
  Class:urinary_stream_symptom
  Class:urinary_retention
  Class:urinary_incontinence
  Class:urinary_frequency
  Class:dysuria
Class:skin_and_integumentary_tissue_symptom
  Class:urticaria
  Class:spontaneous_ecchymoses
  Class:rash
    Class:blotchy_red_rash
    Class:ankle_rash
  Class:itching
  Class:edema
Class:respiratory_system_and_chest_symptom
  Class:swelling_in_chest
  Class:sneezing
  Class:respiratory_failure
  Class:painful_respiration
  Class:lump_in_chest
  Class:cough
  Class:congestion
  Class:abnormal_chest_sound
```

Taula 50. Requisit RF-T3.

Requisit RF-T4

Tipus principals d'intoleràncies i la seva classificació.

```
Class:Sorbitol_Intolerance
Class:Lactose_Intolerance
Class:Gluten_Intolerance
Class:Fructose_Intolerance
```

Taula 51. Requisit RF-T4.

Requisit RF-T5

Tipus principals de dietes i la seva classificació.

```
Class:Gluten_free
Class:Plant_based
Class:Vegan
Class:Vegetarian
Class:Sugar_free
Class:Sorbitol_free
Class:Fructose_free
Class:Lactose_free
```

Taula 52. Requisit RF-T5.

Requisit RF-X1

Nutrients d'un aliment X.

```
Enter food's name:
Apple
Apple hasNutrient Fructose

Apple hasNutrient Potassium

Apple hasNutrient VitaminC
```

Taula 53. Requisit RF-X1.

Requisit RF-X2

Aliments/nutrients prohibits per una dieta X.

```
Enter diet's name:
Vegan
Vegan hasForbidden Animal_Protein
```

Taula 54. Requisit RF-X2.

Requisit RF-X3

Tractament que s'aplica a la reacció adversa (intolerància) a u tipus d'aliment/nutrient X.

```
Enter intolerance's name:
Fructose_Intolerance
Fructose_Intolerance intoleranceHasTreatment diet type Fructose_free
```

Taula 55. Requisit RF-X3.

Requisit RF-P1

Nom de la dieta prèvia a l'estudi (en cas que existeixi) i restriccions d'aquesta (en cas que existeixin) realitzada pel pacient.

Si el pacient no està realitzant cap dieta la consulta ens retorna els camps buits.

```
PREFIX  :     <http://www.owl-ontologies.com/diagnose_food_diary.owl#>

SELECT  ?diet ?restriction
WHERE
  { ?x  :hasID    "11111111A" ;
        :hasDiet  ?diet
    OPTIONAL
      { ?diet  :hasForbidden  ?restriction }
  }
----------------------
| diet | restriction |
======================
----------------------
```

Si el pacient està realitzant una dieta però aquesta no té restriccions, la consulta ens retorna el camp de la dieta amb la informació corresponent i el camp de les restriccions buit.

```
PREFIX  :     <http://www.owl-ontologies.com/diagnose_food_diary.owl#>

SELECT  ?diet ?restriction
WHERE
  { ?x  :hasID    "33333333C" ;
        :hasDiet  ?diet
    OPTIONAL
      { ?diet  :hasForbidden  ?restriction }
  }
-----------------------------
| diet        | restriction |
=============================
| :Vegetarian |             |
-----------------------------
```

Si el pacient està realitzant una dieta amb restriccions, la consulta ens retorna els dos camps amb la informació corresponent.

```
PREFIX  :     <http://www.owl-ontologies.com/diagnose_food_diary.owl#>

SELECT  ?diet ?restriction
WHERE
  { ?x  :hasID    "22222222B" ;
        :hasDiet  ?diet
    OPTIONAL
      { ?diet  :hasForbidden  ?restriction }
  }
---------------------------
| diet   | restriction     |
===========================
| :Vegan | :Animal_Protein |
---------------------------
```

Taula 56. Requisit RF-P1.

Requisit RF-P2

Àpats realitzats pel pacient.

```
  4 SELECT  ?meal
  5 WHERE
  6   { ?meal     rdf:type     :Meal .
  7     ?patient  :hasID       "22222222B" .
  8     ?meal     :hasPatient  ?patient
  9   }
---------------------
| meal              |
=====================
| :Meal_Patient2_11 |
| :Meal_Patient2_4  |
| :Meal_Patient2_15 |
| :Meal_Patient2_13 |
| :Meal_Patient2_1  |
| :Meal_Patient2_2  |
| :Meal_Patient2_9  |
| :Meal_Patient2_14 |
| :Meal_Patient2_16 |
| :Meal_Patient2_8  |
| :Meal_Patient2_3  |
| :Meal_Patient2_12 |
| :Meal_Patient2_5  |
| :Meal_Patient2_6  |
| :Meal_Patient2_7  |
| :Meal_Patient2_10 |
---------------------
```

Taula 57. Requisit RF-P2.

Requisit RF-P3

Nombre d'àpats realitzats pel pacient.

Per obtenir el resultat d'aquest exemple s'ha pres la informació del cas anterior mostrat a la Taula 57.

```
  4 SELECT  (COUNT(?meal) AS ?NELEMENTS)
  5 WHERE
  6   { ?meal     rdf:type     :Meal ;
  7               :hasPatient  ?patient .
  8     ?patient  :hasID       "22222222B"
  9   }
Total of meals: 16
```

Taula 58. Requisit RF-P3.

Els següents requisits es realitzen sobre un pacient que ha realitzat els àpats que es mostren a la Figura 68.

meal	food	symptom
:Meal_Patient1_2021-05-12_18:07:14	:Banana	:Vomit
:Meal_Patient1_2021-05-12_18:07:14	:Pasta	:Vomit
:Meal_Patient1_2021-05-12_18:07:47	:Apple	:Vomit
:Meal_Patient1_2021-05-12_18:05:40	:Lentils	:Stomach_Ache
:Meal_Patient1_2021-05-12_18:05:40	:Boiled_Egg	:Stomach_Ache
:Meal_Patient1_2021-05-12_18:05:40	:Orange	:Stomach_Ache

Figura 68. Exemple: Àpats d'un pacient.

Requisit RF-P4

Nombre d'aliments ingerits pel pacient.

```
 4 SELECT  (COUNT(?food) AS ?NELEMENTS)
 5 WHERE
 6   { ?meal     rdf:type     :Meal ;
 7               :hasPatient  ?patient .
 8     ?patient  :hasID       "11111111A" .
 9     ?meal     :hasFood     ?food
10   }

Total of food: 6
```

Taula 59. Requisit RF-P4.

Requisit RF-P5

Nombre de símptomes experimentats pel pacient.

```
 4 SELECT  (COUNT(?symptom) AS ?NELEMENTS)
 5 WHERE
 6   { SELECT DISTINCT  ?meal ?symptom
 7     WHERE
 8       { ?meal     rdf:type     :Meal ;
 9                   :hasPatient  ?patient .
10         ?patient  :hasID       "11111111A" .
11         ?meal     :hasFood     ?food ;
12                   :hasSymptom  ?symptom
13       }
14   }

Total of symptoms: 3
```

Taula 60. Requisit RF-P5.

Requisit RF-P6

Aliment/s i simptomatologia (en cas que es presenti) de cada àpat.

```
 1 PREFIX  :     <http://www.owl-ontologies.com/diagnose_food_diary.owl#>
 2 PREFIX  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
 3
 4 SELECT  ?meal ?food ?symptom
 5 WHERE
 6   { ?meal     rdf:type     :Meal .
 7     ?patient  :hasID       "22222222B" .
 8     ?meal     :hasPatient  ?patient ;
 9               :hasFood     ?food ;
10               :hasSymptom  ?symptom
11   }
-------------------------------------------------------------
| meal                                | food    | symptom   |
=============================================================
| :Meal_Patient2_2021-04-29_15:51:58 | :Milk   | :Vomit    |
| :Meal_Patient2_2021-04-29_15:51:58 | :Almond | :Vomit    |
| :Meal_Patient2_2021-04-29_15:51:38 | :Burger | :Diarrhea |
-------------------------------------------------------------
```

Taula 61. Requisit RF-P6.

Requisit RF-P7

Severitat de la simptomatologia experimentada pel pacient (en cas de presentar-se).

```
 4 SELECT  ?meal ?food ?symptom ?severity
 5 WHERE
 6   { ?meal     rdf:type     :Meal .
 7     ?patient  :hasID       "22222222B" .
 8     ?meal     :hasPatient  ?patient ;
 9               :hasFood     ?food
10     OPTIONAL
11       { ?meal  :hasSymptom  ?symptom
12         OPTIONAL
13           { ?symptom  :hasSeverity  ?severity }
14       }
15   }
-----------------------------------------------------------
| meal              | food    | symptom       | severity |
===========================================================
| :Meal_Patient2_21 | :Apple  | :Stomach_Ache | "4"      |
| :Meal_Patient2_21 | :Banana | :Stomach_Ache | "4"      |
| :Meal_Patient2_22 | :Apple  |               |          |
| :Meal_Patient2_23 | :Banana | :Vomit        | "3"      |
| :Meal_Patient2_23 | :Banana | :Diarrhea     |          |
-----------------------------------------------------------
```

Taula 62. Requisit RF-P7.

Requisit RF-P8

Diagnòstic (intoleràncIa) que es conclou (si es pot concloure) de la simptomatologia del pacient.

```
Total of meals: 395
Total of food: 471
Total of symptoms: 275
---------PATIENT OPTIONS---------
1. Patient previous diet
2. Patient meals
3. Patient diagnose
4. Patient treatment
5. Close
Choose an option:
3
Total meals with fructose: 88
Total meals with symptoms and fructose: 0
The patient has not had any symptom related to fructose.


Total meals with gluten: 335
Total meals with symptoms and gluten: 259
Gluten intolerance: 77,31%


Total meals with lactose: 0
Total meals with symptoms and lactose: 0
The patient has not eaten any food with lactose.


Total meals with sorbitol: 48
Total meals with symptoms and sorbitol: 16
Sorbitol intolerance: 33,33%


Patient2 hasDiagnose Sorbitol_Intolerance
Patient2 hasDiagnose Gluten_Intolerance
```

Taula 63. Requisit RF-P8.

Per a cada un dels requisits que pertanyen al grup de càlcul del percentatge de probabilitats de patir un tipus concret d'intolerància, s'han inclòs casos en que s'exemplifiquen situacions diferents.

Requisit RF-P8.1

Percentatge de probabilitats de patir intolerància a la fructosa.

```
Total meals with fructose: 2
Total meals with symptoms and fructose: 1
Fructose intolerance: 50%
```

Taula 64. Requisit RF-P8.1.

Requisit RF-P8.2

Percentatge de probabilitats de patir intolerància al gluten.

```
Total meals with gluten: 82
Total meals with symptoms and gluten: 0
The patient have not had any symptom related with gluten.
```

Taula 65. Requisit RF-P8.2.

Requisit RF-P8.3

Percentatge de probabilitats de patir intolerància a la lactosa.

```
Total meals with lactose: 0
Total meals with symptoms and lactose: 0
The patient have not eaten any food with lactose.
```

Taula 66. Requisit RF-P8.3.

Requisit RF-P8.4

Percentatge de probabilitats de patir intolerància al sorbitol.

```
Total meals with sorbitol: 57
Total meals with symptoms and sorbitol: 25
Sorbitol intolerance: 43,86%
```

Taula 67. Requisit RF-P8.4.

Requisit RF-P9

Tractament (dieta) que s'aplica al pacient.

L'output següent s'obté també a partir del cas representat en la Figura 68.

```
---------PATIENT OPTIONS---------
1. Patient previous diet
2. Patient meals
3. Patient diagnose
4. Patient treatment
5. Close
Choose an option:
4
Patient2 patientHasTreatment Sorbitol_free
Patient2 patientHasTreatment Gluten_free
```

Taula 68. Requisit RF.P9.