Ontologia del LinkedIn

Primer pas per inferir informació a partir de les ofertes de feina

Marc Escarmís i Arasa
Enginyeria Tècnica en Informàtica de Gestió

Consultor: Joan Anton Perez Braña

Entrega:
9 de gener de 2015

FITXA DEL TREBALL FINAL

Títol del treball:	ONTOLOGIA DE LINKEDIN: EL PRIMER PAS PER INFERIR INFORMACIÓ A PARTIR DE LES OFERTES DE FEINA
Nom de l'autor:	Marc Escarmís Arasa
Nom del consultor:	Joan Anton Perez Braña
Data de lliurament (mm/aaaa):	01/2015
Àrea del Treball Final:	TFC XML-Web semàntica
Titulació:	Enginyeria Tècnica en Informàtica de Gestió

Resum del Treball (màxim 250 paraules):

L'actual creixement de dades en la web fa necessari noves estratègies a l'hora de codificar-ne els continguts. Mitjançant la web semàntica es pretén nodrir d'informació amb significat els seus continguts, d'aquesta manera fer possible que tant el programari com les persones en puguin inferir informació.

Les xarxes socials són un clar exponent d'aquest augment en les dades. Les relacions de dependència de les dades a vegades són molt complicades de discernir. Aquest fet i el seu augment fa indispensable la utilització de software per poder-les processar.

El cas que ens ocupa és la xarxa social LinkedIn. La quantitat d'ofertes de feina i demandes de feina fan a vegades difícil extreure'n la informació necessària. Es vol crear una ontologia per tal de nodrir les dades presents en la web LinkedIn amb informació semàntica. Mitjançant el programari Protégé es pretén crear aquesta ontologia, i mitjançant les API que ofereix LinkedIn un mitjà per inferir informació respecte a les ofertes i demandes de feina.

Aquest treball d'investigació pretén fer una primera aproximació a la web semàntica sobre la xarxa social LinkedIn.

Abstract (in English, 250 words or less):

The current growth of the web data makes necessary new strategies for encoding their content. Using semantic web aims to feed the meaning all these content, in order to make it accessible for both software and people.

Social networks are a clear example of this increase in the data. The data dependency relationships are sometimes very difficult to discern. This fact and its increase makes essential the use of software to process them.

The present case is LinkedIn social network. The amount of job demands and job vacancies makes sometimes difficult to extract the necessary information. We want to create an ontology to feed the data present in the LinkedIn web with semantic information. Protégé software is used to create the ontology and LinkedIn APIs provides meaning (sense) to infer information about the supply and demand of work.

This research aims to make a first approach to semantic web over social network LinkedIn.

Paraules clau (entre 4 i 8):

web, semàntica, XML, RDF, OWL, Ontologia, linkedIn

Index

Index **3**

Index de figures **5**

1 Introducció **6**

1.1 Context i justificació del Treball 6

1.1.1 Context 6

1.1.2 Pila tecnològica de la web semàntica 8

XML (eXtended Markup Language) 8

RDF (Resource Description Framework) 8

RDF Triples 9

OWL (Web Ontology Language) 9

Ontologies 10

SPARQL 10

SWRL 10

1.1.3 Justificació del treball 10

1.2 Antecedents i objectius del Treball 10

1.2.1 Antecedents 10

1.2.2 Objectius del treball 11

1.3 Enfocament i mètode seguit 11

1.4 Planificació del projecte 12

1.4.1 Introducció 12

1.4.2 Tasques i temporització 12

1.4.3 Diagrama de Gantt 14

1.4.4 Possibles incidències 15

1.4.5 Consideracions respecte les versions del software Protégé 15

1.5 Breu sumari de productes obtinguts 15

1.5.1 Productes obtinguts 15

1.5.2 Fases de producció pels productes obtinguts 16

2 Disseny de l'ontologia del LinkedIn **17**

2.1 Determinar el domini i abast de l'ontologia 17

2.2 Enumeració de les entitats més rellevants del LinkedIn 18

2.3 Definir classes i jerarquies 19

2.4 Definir propietats de les classes 20

2.4.1 Principals classes i les seves propietats 20

2.4.2 Classes i propietats (Class & Slots) 20

Class Jobs....20
Class JobPosting....21
Class Companies....21
Class People....23
Class CompanyWorker....24
Class Independent....24
Class Groups....24
Class GroupMemberShip....25
2.5 Representació de les classes amb el Protégé....25
2.5.1 Esquemes Ontograph....26
2.6 Simplificació....26
2.7 Nomenclatura....27
2.8 Creació d'instàncies....27

3 Desenvolupament de l'aplicació....29
3.1 Accés a les API del LinkedIn....29
3.2 Proves de les API....29
3.3 Propietats de l'ontologia i de les API....31
3.4 Programari necessari pel desenvolupament....31
3.5 Lògica de l'aplicació....33
3.6 Disseny de l'aplicació (UML)....34
3.7 Fitxer principal serializeLinkedin.py....36
3.8 Fitxer RDF creat....42
3.9 Resultat de l'execució....44

4 Tractament de la informació, consultes SPARQL....46
4.1 Visió general, creació d'instàncies....46
4.2 Consultes SPARQL....48

5 Conclusions....53

6 Glossari....55

7 Bibliografia recomanada....56
7.1 Documents....56
7.2 Llibres de text....56
7.3 Alguns Enllaços d'interès a internet....56

8 Annexos....59
8.1 Edició i execució de l'script alternatives, editor PyCharm....59

Index de figures

Il·lustració 1: Comparativa web actual i web semàntica: [2].....7
Il·lustració 2: Evolució de la web.....7
Il·lustració 3: Capes tecnològiques involucrades en la web semàntica.....8
Il·lustració 4: Diferents tipus de triples en RDF.....9
Il·lustració 5: Diagrama de Gantt.....14
Il·lustració 6: Diagrama de classes.....19
Il·lustració 7: Protégé. Classe Jobs.....21
Il·lustració 8: Protégé. Classe Companies.....22
Il·lustració 9: Protégé. Classe People.....23
Il·lustració 10: Protégé. Classe Groups.....25
Il·lustració 11: Ontograph expandit.....26
Il·lustració 12: Creació de individuals/instàncies. Instància de la classe Jobs.....27
Il·lustració 13: Creació de individuals/instàncies. Instància de la classe Companies...28
Il·lustració 14: Consola OAuth del LinkedIn per probar consultes a les API.....31
Il·lustració 15: Entorn gràfic d'execució de Python. GUI.....32
Il·lustració 16: Diagrama de flux.....33
Il·lustració 17: Diagrama de classes. UML.....34
Il·lustració 18: Diagrama d'estats. UML.....35
Il·lustració 19: RDF. Classe jobs.....42
Il·lustració 20: RDF. Classe companies.....43
Il·lustració 21: RDF. Classe CompanyWorker.....43
Il·lustració 22: Resultat de l'execució de l'script 1.....44
Il·lustració 23: Resultat de l'execució de l'script 2.....45
Il·lustració 24: Instàncies d'objectes Jobs.....46
Il·lustració 25: Instàncies d'objectes Companies.....47
Il·lustració 26: Instàncies d'objectes People->CompanyWorker.....47
Il·lustració 27: Protégé. Consulta SPARQL 1.....49
Il·lustració 28: Protégé. Consulta SPARQL 2.....50
Il·lustració 29: Protégé. Consulta SPARQL 3.....51
Il·lustració 30: Protégé. Consulta SPARQL 4.....52
Il·lustració 31: Pantalla principal de l'editor PyCharm.....59
Il·lustració 32: Selecció de la versió Python en l'editor PyCharm.....60
Il·lustració 33: Sortida de l'execució en l'editor PyCharm.....60

1 Introducció

1.1 Context i justificació del Treball

1.1.1 Context

La ingent quantitat d'informació que hi ha actualment en la Web[1], fa cada cop és més costós poder inferir informació de qualitat sense que el fet d'inferir-la ens suposi un relatiu esforç humà per trobar el que hi busquem. El creixement exponencial dels continguts disponibles fa que cada cop aquestes tasques siguin més costoses. Aquestes quantitats són calculades sobre la *web superficial*[2], formada per documents estàtics. S'ha calculat que la *web profunda*, constituïda per base de dades no directament accessibles, pot ser centenars de cops major. [2].

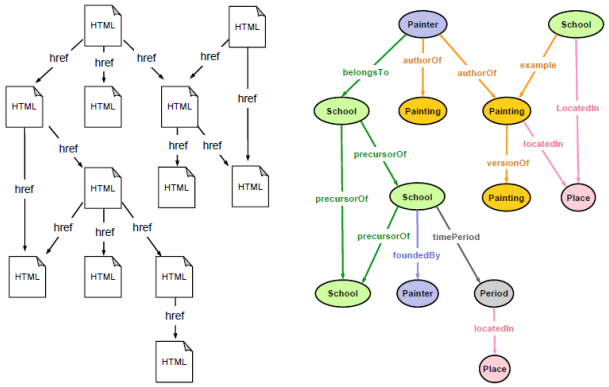
També, les xarxes socials augmenten la quantitat d'informació disponible, així com un extensa varietat conceptual i de temàtiques. Aquestes dades sovint estan relacionades entre sí, però la mancança d'un significat semàntic de les mateixes molts cops fa humanament i automàticament impossible relacionar-les i extreure'n informació. [4].

Es en aquest context i moment que entra en joc la web semàntica. Dotant les dades de contingut semàntic podem fer-les accessibles i intel·ligibles als sistemes informàtics i al mateix temps ser interpretades per éssers humans. Podríem dir que és una extensió de la web actual.

En la figura següent, la primera imatge correspon a la web actual i la segona a la web semàntica. Com veiem en la primera no es distingeixen entitats diferents, tot son pàgines web. També les relacions entre diferents documents son indistingibles, son un simple enllaç. En la web semàntica els documents son identificats com diferents entitats i les seves relacions son particulars i distingibles entre elles, ens dóna molta més informació dels continguts.

De la web semàntica en podríem dir la web 3.0. És un nou paradigma web.

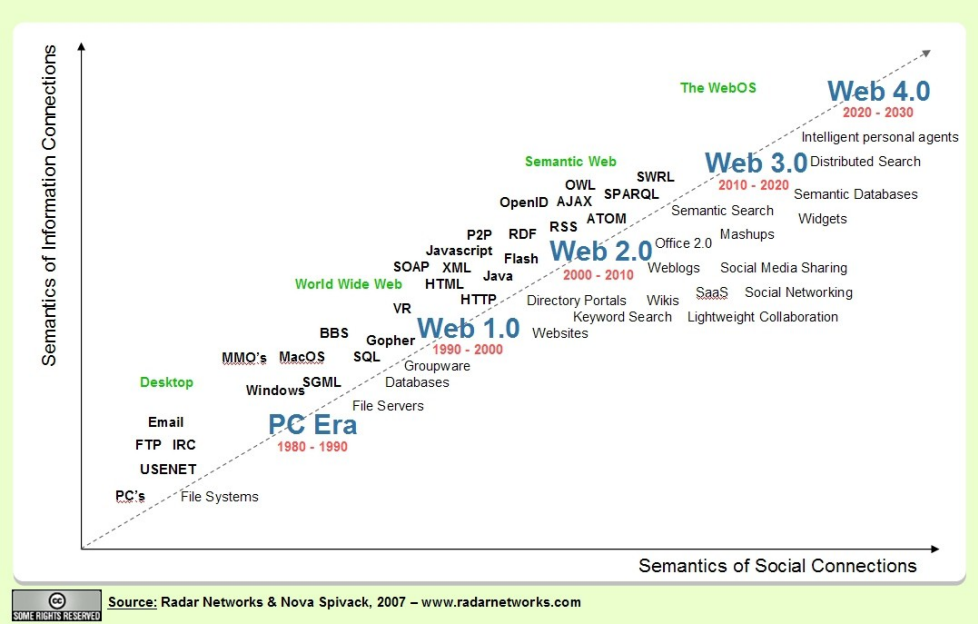
1 entenem per Web el conjunt de pàgines/llocs webs accessibles des d'Internet, es calcula que existeixen $7{\cdot}10^9$ documents web. [1].
2 Es coneix com a web superficial la web o la porció de pàgines d'internet que pot ser indexada per robots. La part que no pot ser indexada es coneix com a web profunda.



Il·lustració 1: Comparativa web actual i web semàntica: [2]

Actualment ja s'han desenvolupat especificacions que garanteixen la interoperabilitat entre aplicacions, s'han desenvolupat eines, magatzems semàntics, interfícies de programació (API) i entorns de treball que permeten a les persones participar de la web semàntica i inferir-ne informació rellevant per els seus interessos.

En el següent esquema es mostra una evolució de la web durant el temps, on s'hi veuen les eines disponibles, tipus de dades utilitzades i la seva previsible evolució. [1]

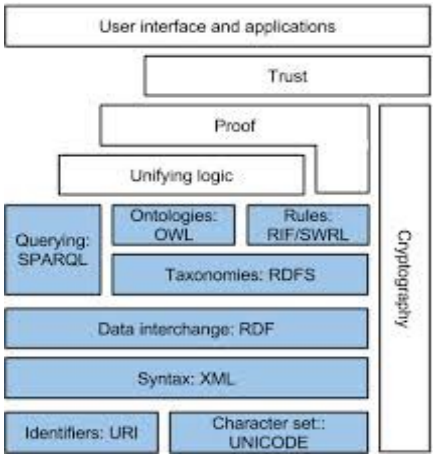


Il·lustració 2: Evolució de la web

1.1.2 Pila tecnològica de la web semàntica

Les tecnologies involucrades en el desenvolupament de la web semàntica les podem veure com una combinació de capes, on cada capa té la seva funcionalitat indispensable per la correcta aplicació de la capa següent.

En aquest treball ens ocuparem essencialment de les capes de color blau en l'il·lustració següent.



Il·lustració 3: Capes tecnològiques involucrades en la web semàntica

Podem presentar les tecnologies implicades en la web semàntica com una estructura de capes. A continuació s'expliquen algunes de les tecnologies principals implicades. [4].

***XML** (eXtended Markup Language)*

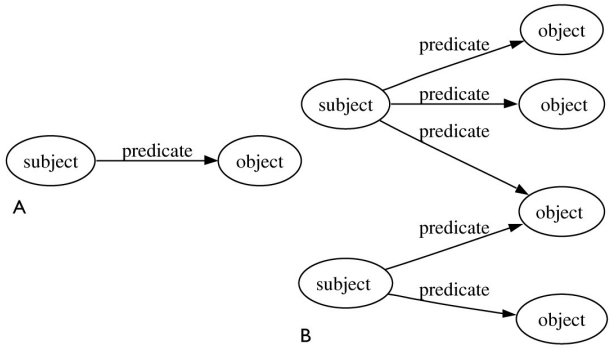
El podríem situar en la capa més baixa del esquema de la web semàntica. Aporta estructuració de les dades però no ens aporta massa significat semàntic de les dades. Mitjançant els esquemes (XML-Scheme) o les DTD (DataType Definition) podem dotar d'estructura aquest XML, aquestes dades. [5].

RDF (**Resource Description Framework**)

Aporta un model de dades per objectes i les relacions entre ells, nodrint-les d' una semàntica que es pot representar amb sintaxi XML. El podem situar en un segon nivell en aquest esquema. [5][6].

RDF Triples

Esta format per els conceptes subjecte-predicat-objecte i marca la relació (*predicat*) que té un *subjecte* amb un *objecte*.



Illustració 4: Diferents tipus de triples en RDF

OWL (*Web Ontology Language****)***

És un llenguatge dissenyat per representar coneixement de les entitats, grups d'entitats i relacions entre elles. OWL aporta més vocabulari per descriure propietats i classes. [5]. Hi ha tres nivells de formats OWL:

- **OWL Lite**, que és la versió més senzilla d'ontologia i, per tant, la més restringida en expressivitat; ofereix només la definició d'una jerarquia i restriccions simples de cardinalitat (0 o 1).
- **OWL DL**, aporta ampliacions respecte d'OWL Lite i està dissenyada per a tenir la màxima expressivitat possible però garantint la completesa computacional (totes les seves conclusions són computables) i decidibilitat (el temps de raonament sempre serà finit). Com a contrapartida a poder expressar propietats més complexes, el temps de raonament és superior que en el cas d'OWL Lite.
- **OWL Full,** aquest nivell presenta un canvi conceptual considerable respecte els nivells Lite i DL. Tot i que pel que fa a la sintaxi no augmenta respecte a OWL DL, sí que permet mantenir la mateixa relaxació de l'RDF Schema. [3].

Ontologies

El terme ontologia en el marc de la informàtica fa referència a un esquema conceptual molt detallat i amb rigor dintre un d'un o diversos dominis, amb la finalitat de facilitar la cerca o intercanvi d'informació entre diversos sistemes o entitats.

Els programes informàtics poden usar aquestes ontologies per una varietat de propòsits com el raonament inductiu, classificació, presa de decisions, etc..

SPARQL

És un llenguatge utilitzat per consultar diversos tipus de fonts de dades expressades nativament en format RDF. Ve a ser el SQL de la web semàntica.

SWRL

És una extensió del conjunt d'axiomes del OWL per tal d'incloure-hi regles de tipus condicional, són clàusules del tipus *...si....llavors....* L'exemplificació de SWRL queda fora de l'abast d'aquest treball. [20]

1.1.3 Justificació del treball

És en el context de inferir informació i del nou paradigma que es conceptualitza de la web semàntica que té justificació el present treball.

Es volen utilitzar les eines que ofereix la xarxa LinkedIn per extrapolar-ne informació sobre les ofertes i demandes de feines. Tota aquesta informació extreta mitjançant les API que ens ofereix LinkedIn ha de servir per el desenvolupament d'un programari que pugui utilitzar les dades. El programari obté les dades formatades en XML i posteriorment fa ús d'aquesta informació mitjançant SPARQL.

1.2 Antecedents i objectius del Treball

1.2.1 Antecedents

Els objectius perseguits per la present memòria els podríem dividir en dos: Per un costat te l'objectiu d'assolir els coneixements propis de l'assignatura, l'elaboració d'una memòria acurada sobre l'estat de l'art. I per l'altre la de desenvolupar una ontologia per la consulta d'ofertes i demandes de treball en la xarxa social LinkedIn, més concretament, fer servir ontologies per

emmagatzemar la informació rellevant de les ofertes de feina difoses a LinkedIn i de les companyies que les ofereixen.

1.2.2 Objectius del treball

En caràcter general el principal objectiu és la realització d'un treball de síntesi dels coneixements adquirits en les diferents assignatures dels estudis de Enginyeria tècnica en informàtica de gestió.

En detall, aquest treball pretén crear un sistema que permeti emmagatzemar informació d'ofertes de feina i les empreses que les ofereixen en ontologies i explotar aquesta informació. Per tant, el treball tindrà els següents objectius:

1. Crear una ontologia per emmagatzemar ofertes de feina (també es podrà reutilitzar ontologies existents si es troba adient).

2. Crear un programa que permeti poblar l'ontologia a partir de les ofertes de feina (Jobs) i les empreses (Companies) de LinkedIn.

3. Crear un programa o un conjunt de regles sobre l'ontologia que permeti extreure informació de les ofertes de feina.

Per assolir aquests objectius es contemplen les següents etapes:

- Analitzar un problema complex de tipus pràctic transformant-lo en un projecte informàtic.
- Planificar i estructurar el desenvolupament del projecte mitjançant l'elaboració d'un pla de treball aplicant una metodologia adient.
- Treballar a fons els aspectes formals del desenvolupament de projectes.
- Sintetitzar una solució viable i realista al problema proposat.
- Elaborar una memòria del projecte segons una estructura prefixada.
- Elaborar una presentació del desenvolupament i resultats finals del projecte.

1.3 Enfocament i mètode seguit

Per tal d'assolir els objectius plantejats es pot encarar el pla de treball en dos vessants. ***a)*** adaptació d'un producte ja existent en el mercat; ***b)*** desenvolupament d'un producte nou.

La primera opció passa per una tasca de recerca web en els diferent cercadors d'ontologies com poden ser: DAML, Ontolingua, Swoogle o Hakia. No s'ha trobat cap recurs reutilitzable per el present projecte. La segona opció es basa en la creació des de zero d'una ontologia basada en les dades que ens ofereixen les API de LinkedIn.

L'estratègia a seguir escollida ha estat crear des de zero una ontologia per poder-hi desar informació de la xarxa LinkedIn.

1.4 Planificació del projecte

1.4.1 Introducció

Les tasques a realitzar es detallen en el quadre següent. Tanmateix també es detalla una primera temporització del desenvolupament dels continguts de la memòria, així com una programació mitjançant els diagrama de Gantt. Per finalitzar considerem algunes causes de possibles incidències temporals.

1.4.2 Tasques i temporització

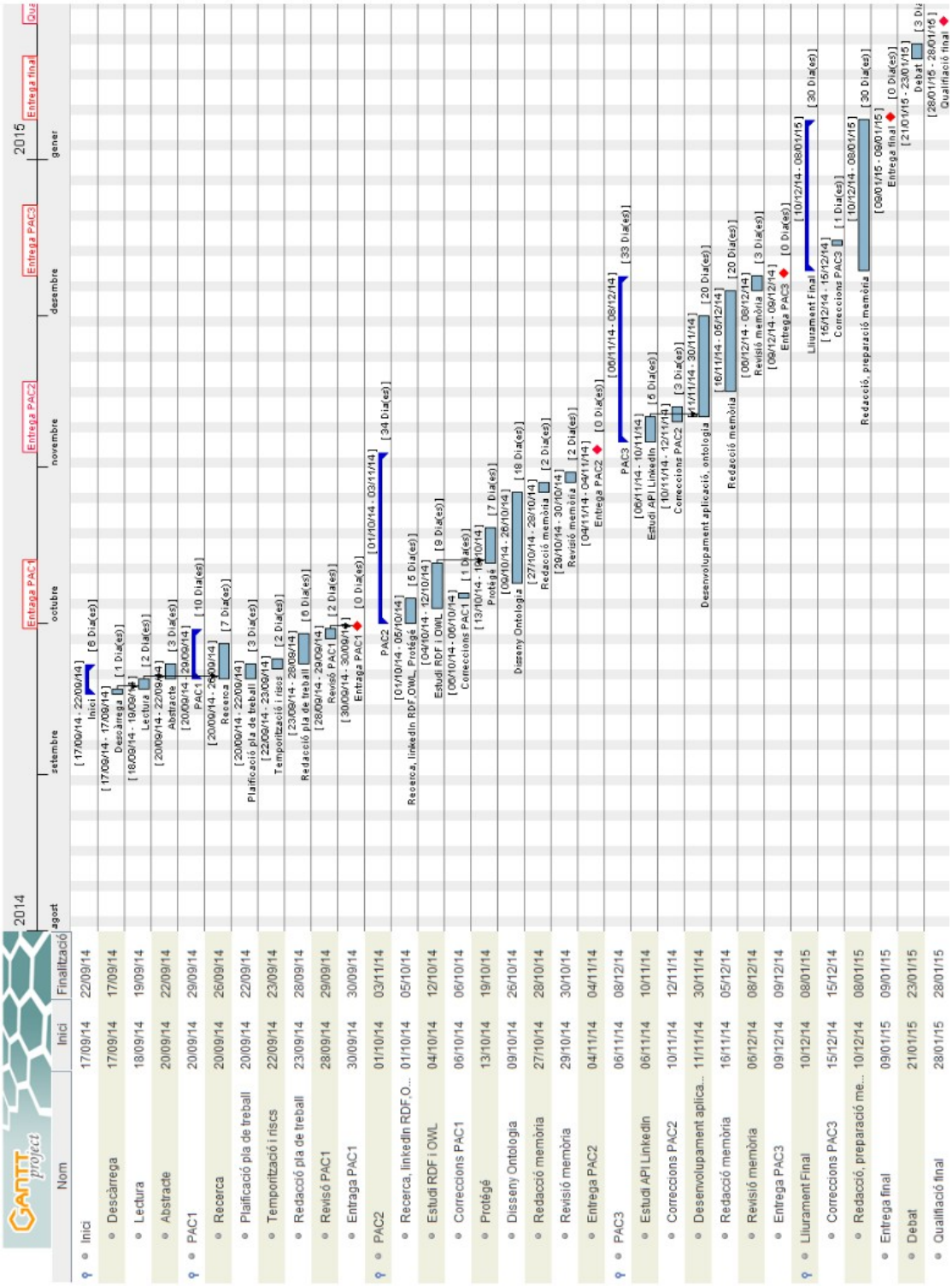
Les principals tasques detectades es veuen resumides en el següent quadre. Aquestes tasques poden ser ampliades o anul·lades a mesura que avança el projecte.

La dedicació prevista la desenvolupament de l'activitat és de 10 hores setmanals, repartides bàsicament els caps de setmana i els dimecres tarda. No obstant, depenent de la càrrega setmanal de la tasca es preveu una dedicació extraordinària de entre una i dues hores la resta de dies de la setmana, pujant la dedicació a 15 hores setmanals.

Nom	Hores	Inici	Fi	Notes
Inici	4	17/09/14	22/09/14	
Descàrrega		17/09/14	17/09/14	
Lectura		18/09/14	19/09/14	Lectura dels mòduls 1 i 2 teòrics
Abstracte		20/09/14	22/09/14	Primeres notes del Pla de treball
PAC1	15	20/09/14	29/09/14	
Recerca		20/09/14	26/09/14	

Plaificació pla de treball		20/09/14	22/09/14	
Temporització i riscs		22/09/14	23/09/14	
Redacció pla de treball		23/09/14	28/09/14	
Revisó PAC1		28/09/14	29/09/14	
Entraga PAC1		30/09/14	30/09/14	
PAC2	70	01/10/14	03/11/14	
Recerca, LinkedIn RDF,OWL, Protégé		01/10/14	05/10/14	Recerca de possibles ontologies ja creades sobre LinkedIn. Recerca general de OWL i Protégé. Recerca general RDF.
Estudi RDF i OWL		04/10/14	12/10/14	
Correccions PAC1		06/10/14	06/10/14	
Protégé		13/10/14	19/10/14	
Disseny Ontologia		09/10/14	26/10/14	
Redacció memòria		27/10/14	28/10/14	
Revisió memòria		29/10/14	30/10/14	
Entrega PAC2		04/11/14	04/11/14	
PAC3	80	06/11/14	08/12/14	
Estudi API LinkedIn		06/11/14	10/11/14	
Correccions PAC2		10/11/14	12/11/14	
Desenvolupament aplicació, ontologia		11/11/14	30/11/14	Desenvolupament de l'aplicació i refinament de l'ontologia. Utilització de les API de LinkedIn.
Redacció memòria		16/11/14	05/12/14	
Revisió memòria		06/12/14	08/12/14	
Entrega PAC3		09/12/14	09/12/14	
Lliurament Final	40	10/12/14	08/01/15	
Correccions PAC3		15/12/14	15/12/14	
Redacció, preparació memòria		10/12/14	08/01/15	
Entrega final	25	09/01/15	09/01/15	
Debat		21/01/15	23/01/15	

1.4.3 Diagrama de Gantt



Il·lustració 5: Diagrama de Gantt

1.4.4 Possibles incidències

Les circumstàncies familiars porten a tenir previst algunes contingències de responsabilitat. En aquests casos es preveu una dedicació intersetmanal extraordinària.

Altres contingències que hagin pogut sorgir durant el desenvolupament del project i han afectat el correcte procés, han estat solucionades amb una dedicació intersetmanal extraordinària.

1.4.5 Consideracions respecte les versions del software Protégé

Per el disseny de l'ontologia s'ha utilitzat la versió 3.5 (Build 663) per a Windows. Els gràfics s'han generat amb la versió 5.0 (Beta). La decisió ha estat presa degut a que el format OWL en que desa les ontologies les versions 4.x i 5.x és diferent que les versions anteriors. En canvi la versió 3.5 no té la funcionalitat de presentació gràfica d'ontologies. Per tant, s'ha generat un fitxer *.owl* amb la versió 3.5 que es pot obrir perfectament amb la versió 5.o per tal de veure gràficament l'ontologia.

1.5 Breu sumari de productes obtinguts

1.5.1 Productes obtinguts

S'han obtingut uns productes que representen la informació del domini estudiat de la xarxa social LinkedIn. El primer producte és una ontologia mitjançant la qual es representa certa informació del LinkedIn, en concret, permet definir un conjunt d'ofertes de feina, detalls sobre les ofertes i les companyies que les ofereixen.

Un segon producte és una aplicació capaç demanar informació al LinkedIn mitjançant les seves pròpies API i que també és capaç de poblar l'anterior ontologia. El llenguatge de programació escollit per elaborar aquesta aplicació es el **Python**, el qual disposa de llibreries tant per autenticar-se en la xarxa social com per a la utilització de les seves API. També s'han elaborat un sèrie de consultes SPARQL que infereixen informació en la ontologia poblada.

1.5.2 Fases de producció pels productes obtinguts

Tasca	Comentari
Inici	Fase inicial del projecte. Plantejament del problema per part del consultor.
Descàrrega	Descàrrega de material a disposició.
Lectura	Primera lectura dels mòduls 1 i 2, "Representació del coneixement" i "Web semàntica, ontologies".
Abstracte	Planificació resum del pla de treball.
PAC1	**Producte obtingut:** Memòria inicial i pla de treball detallat.
PAC2	**Producte obtingut:** Ontologia del LinkedIn.
PAC3	**Producte obtingut:** Aplicació per la consulta d'informació en la xarxa social LinkedIn i per la població de l'ontologia creada en l'anterior fase. Conjunt de sentències en SPARQL per inferir informació a l'ontologia, Memòria final gairebé enllestida.
Entrega final	**Producte obtingut:** Memòria del treball i un vídeo presentació basat en diapositives.

2 Disseny de l'ontologia del LinkedIn

El procés de implementació d'ontologies no ve marcat per un sol patró. La creació d'ontologies es pot encarar de diferents maneres, segons l'enfocament que li vulgui donar el creador o el mateix abast que hagi de cobrir l'ontologia.

El fonamental a l'hora d'encarar-ne la creació és tenir clar les entitats que es representaran en la ontologia. Aquestes entitats ens vindran acotades per l'enfocament que vulguem donar al domini estudiat, pot ser d'espectre molt ampli o d'espectre acotat.

Una altra consideració a tenir en compte és la metodologia a l'hora del disseny: *top-down, bottom-up* o una *combinació* d'ambdues.

Segons la guia editada per **Stanford University** [7] podríem acotar el procediment de l'ontologia en els següents passos:

- Determinar l'abast de l'ontologia dintre el domini estudiat
- Considerar la reutilització d'alguna ontologia existent
- Delimitar les entitats importants del domini
- Definir les classes extretes del pas anterior
- Definir les propietats de les classes
- Crear instancies o "*individuals*" i poblar-les

S'ha seguit com a guia el document [8] creat per la **University of Manchester** de com utilitzar el Protégé.

2.1 Determinar el domini i abast de l'ontologia

El primer pas segons aquesta guia és determinar-ne el domini i ens indica quins aspectes hem de tenir en compte:

- Quin és el domini que ha de cobrir l'ontologia
- Quin és l'objectiu de l'ontologia
- Quines preguntes hauria de poder contestar la ontologia

- Qui usarà i mantindrà la ontologia

El domini ens el marcaran els elements que hem d'estudiar del LinkedIn, els objectes/entitats **Jobs** i **Companies,** però concretament en l'entorn de l'entitat **Jobs**. També és cert que hi han altres objectes implicats en el domini, però que no tindrem en compte en aquest estudi.

L'objectiu de l'ontologia és poder inferir certa informació de les feines (**Jobs**) que s'anuncien en el LinkedIn.

Les preguntes que se li podrien fer a l'ontologia podrien ser per exemple:

- Quantes ofertes de feina s'ofereixen?
- A quina empresa pertany la oferta "*tal"*?
- Quina empresa oferix més feines?
- ...

Aquestes preguntes, també anomenades *preguntes de competència*, ens serviran com a control a l'hora de verificar el correcte comportament de l'ontologia.

2.2 Enumeració de les entitats més rellevants del LinkedIn

En aquest pas el primer que caldria fer és enumerar els elements del domini que ens poden ser rellevants per contestar les preguntes de competència. Dintre d'aquest termes a anomenar també s'hi poden incloure les propietats dels termes o característiques que ens agradarien saber del elements.

Possibles termes: *job, company, location (country), duration, salary, poster, description, skills, benefits, industry.*

En aquesta primera selecció hi trobem tant entitats (*classes*), com propietats dels mateixos. A més hi ha propietats com *location* que també poden esdevenir classe degut a que contenen altres propietats.

Entitats més rellevants:

Jobs

Companies

Groups

People *(Amb les subclasses CompanyWorker, IndependentWorker i GropuMemberShip)*

Altres entitats:

JobPosting

CompanyWorker

IndependentWorker

2.3 Definir classes i jerarquies

Les classes que usarem en aquesta otologia es poden dels principals objectes que ens ofereix les API's del LinkedIn. També hi afegiré dues classes per parametreitzar millor els objectes que ens ofereix el domini. Aquestes classes afegides també les podríem usar com a *Datatype properties* del OWL. En un primer estat de l'ontologia farem servir les següents classes.

Principals classes:

Jobs

JobPosting *(Subclasse)*

Companies

Groups

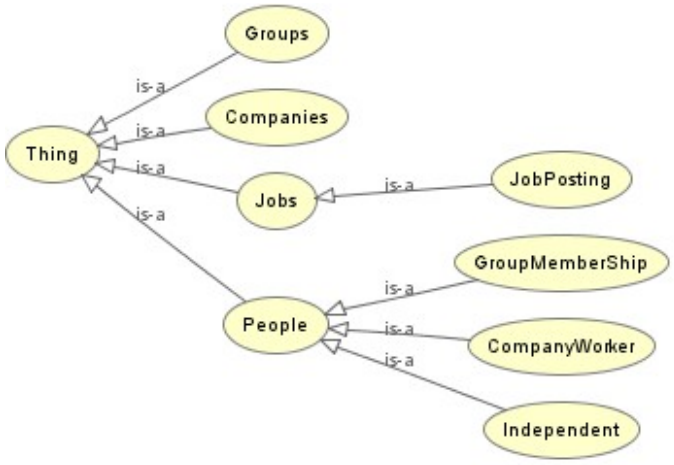
People

CompanyWorker *(Subclasse)*

Independent *(Subclasse)*

GroupMemberShip *(Subclasse)*

En el següent diagrama es pot veure d'una forma gràfica aquesta classificació:



Illustració 6: Diagrama de classes

2.4 Definir propietats de les classes

Cal tenir en compte la classe de propietat que és. OWL contempla diverses variants de propietats: *fuctional, Inverse functional, transitive, Symetric, asymetric,reflexive i Irreflexive*.

També distingirem entre les *Object properties* i les *Datataype properties*. Les primeres fan referència a les relacions entre classes o "*individuals*", i les segones entre una instància/*individual* i els valors que pot prendre.

2.4.1 Principals classes i les seves propietats

El LinkedIn ens ofereix moltes propietats que poden tenir els seus objectes, en aquest document en presentarem alguns però sols en contemplarem un subconjunt. El subconjunt ens vindrà marcat per el domini de l'ontologia i que facin possible respondre les preguntes de competència.

2.4.2 Classes i propietats *(Class & Slots)*

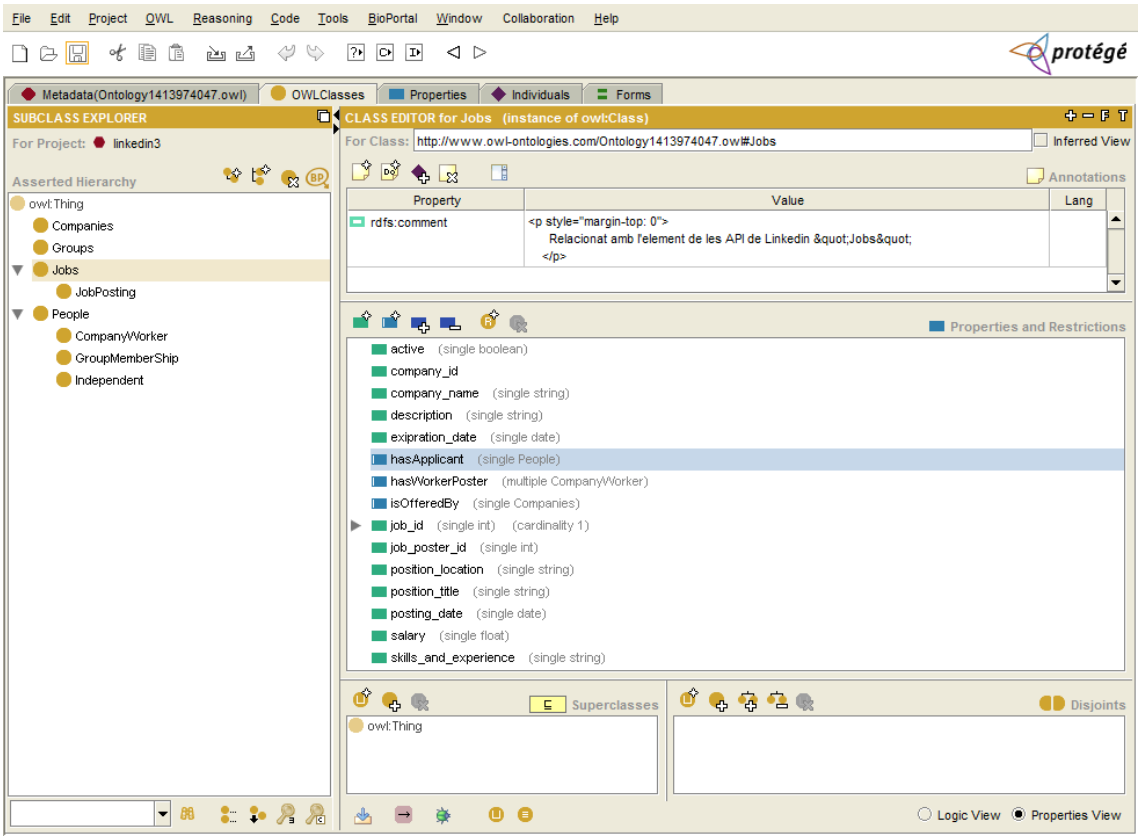
Class Jobs

Aquesta classe s'anomena igual que l'objecte del LinkedIn i és la classe principal al voltant de la qual volem crear l'ontologia. Les propietats de la classe que ens ofereix el LinkedIn és molt més extensa que les que utilitzarem aquí.

Més informació dels camps contemplats per LinkedIn [11].

propietat	**tipus**	**tipus dada**	**descripció**
job-id	Datatype	int	Identificador del *Job*. Dada oferta per el LinkedIn
active	Datatype	boolean	Ens indica si el Job està actiu o ha caducat
posting-date	Datatype	date	Data de creació de l'oferta de feina
expiration-date	Datatype	date	Data de caducitat de l'oferta de feina
company-id	Datatype	int	Identificador únic de la companyia que ofereix la feina
comapy-name	Datatype	string	Nom de l'empresa ofertant
position-title	Datatype	string	Lloc ocupat per la l'oferta de feina
position-location	Datatype	string	Ubicació geogràfica de l'oferta de feina
skills-and-experience	Datatype	string	Coneixements i experiència que ha de tenir l'aspirant
salary	Datatype	float	Remuneració econòmica de l'oferta de feina
job-poster-id	Datatype	int	Identificador únic de la persona que ha creat l'oferta de feina. Aquesta persona pot o no pertànyer a la

			companyia ofertant
description	Datatype	string	Descripció de la feina
isOfferedBy	Object		Relació entre les classes ***Companies*** i ***Jobs***,
hasOffer	Object		Funcionalitat inversa a isOfferedBy
hasApplicant	Object		Relaciona aspirants amb la feina



Il·lustració 7: Protégé. Classe Jobs

Class JobPosting

Classe usada per el LinkedIn a l'hora de fer un post (donar d'alta) una feina.

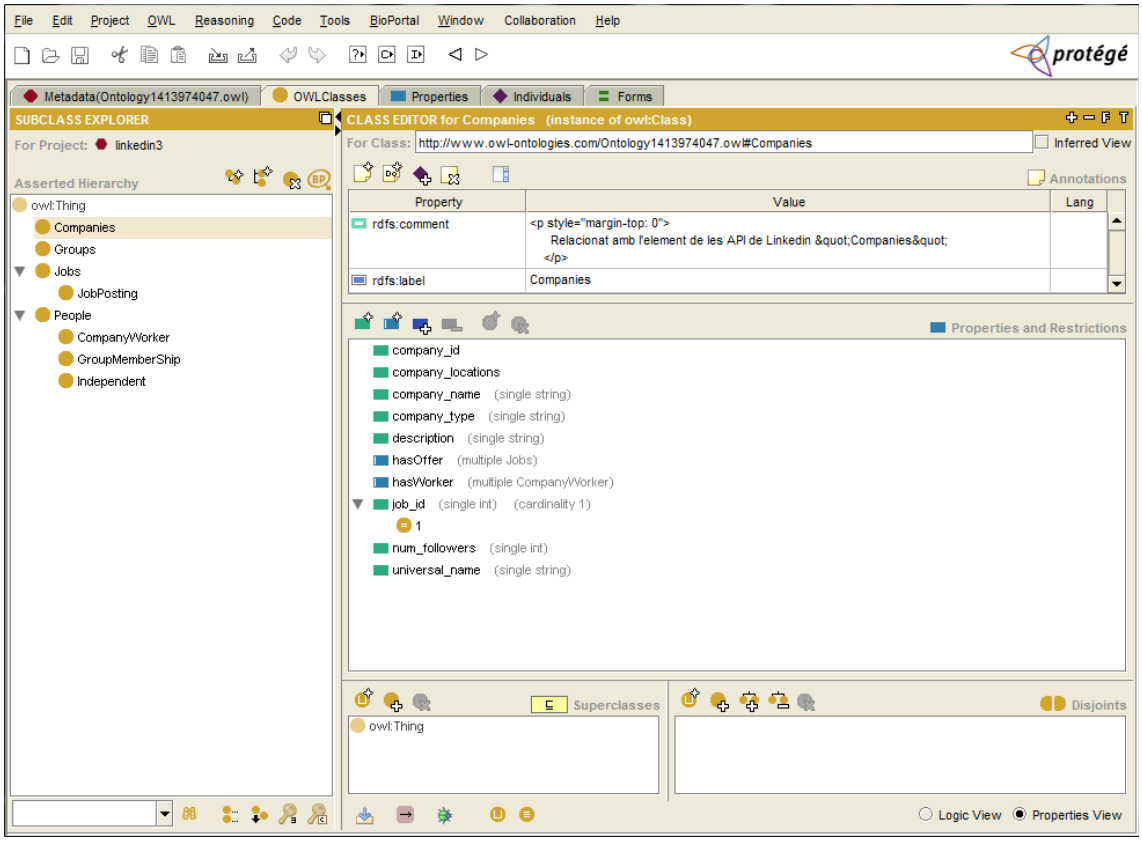
Té certs camps compartits amb la classe mare **Jobs**. La contemplem de moment, a l'hora de refinar l'ontologia potser caurà degut a que la volem orientar a inferir dades d'ofertes de feina ja creades.

Class Companies

En aquesta classe només hem contemplat alguns dels camps que ens ofereix el LinkedIn. Hem mantingut tant sols els que creiem que poden ser de més utilitat a l'ontologia.

Més informació dels camps contemplats per LinkedIn [12].

proopietat	tipus	tipus dada	descripció
id	Datatype	int	identificador de la *Company*. Dada oferta per el LinkedIn
name	Datatype	string	Nom de la companyia
universal-name	Datatype	string	nom universal únic per identificar la companyia
company-type	Datatype	string	identificador del tipus de servei que ofereix la companyia.
locations	Datatype	string	Emplaçament de les diferents delegacions de l'empresa
description	Datatype	string	Descripció de l'empresa
num-followers	Datatype	int	Nombre de seguidors que te l'empresa. Objectes *People*
isOfferedBy	Object		Relació entre les classes ***Companies*** i ***Jobs***, ens indica quina companyia (*individual*) ha creat el lloc de treball
hasOffer	Object		Funcionalitat inversa a isOfferedBy



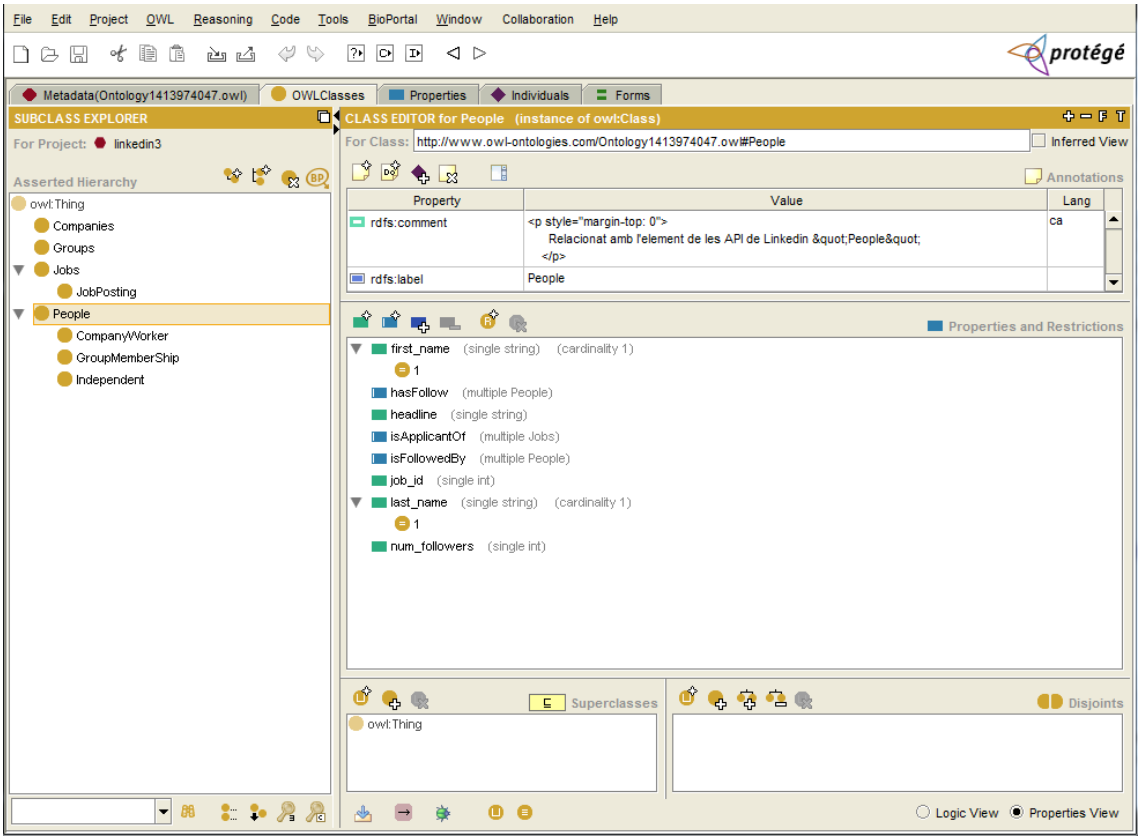
Il·lustració 8: Protégé. Classe Companies

Class People

També és una de les principals classes que ens ofereix el LinkedIn.

Més informació dels camps contemplats per LinkedIn [13].

propietat	tipus	tipus dada	descripció
id	Datatype	int	Identificador de la *People*. Dada oferta per el LinkedIn
first-name	Datatype	string	Nom de la persona
last-name	Datatype	string	Nom de la persona
headline	Datatype	string	Nom de la descripció del seu perfil professional
location-name	Datatype	string	Ciutat de residència
location-country	Datatype	string	País de residència
num-followers	Datatype	int	Nombre de seguidors que te l'empresa. Objectes *People*
isLocatedOn	Object		Realció entre les classes ***Companies*** i ***Jobs***, ens indica quein companyia (*individual*) ha creat el lloc de treball
hasLocation	Object		Funcionalitat inversa a isLocatedOn



Il·lustració 9: Protégé. Classe People

Class CompanyWorker

Subclasse derivada de **People** que engloba els usuaris pertanyents a una empresa.

Class Independent

Subclasse derivada de People que engloba la resta d'usuaris del LinkedIn, pertanyin o no a una empresa.

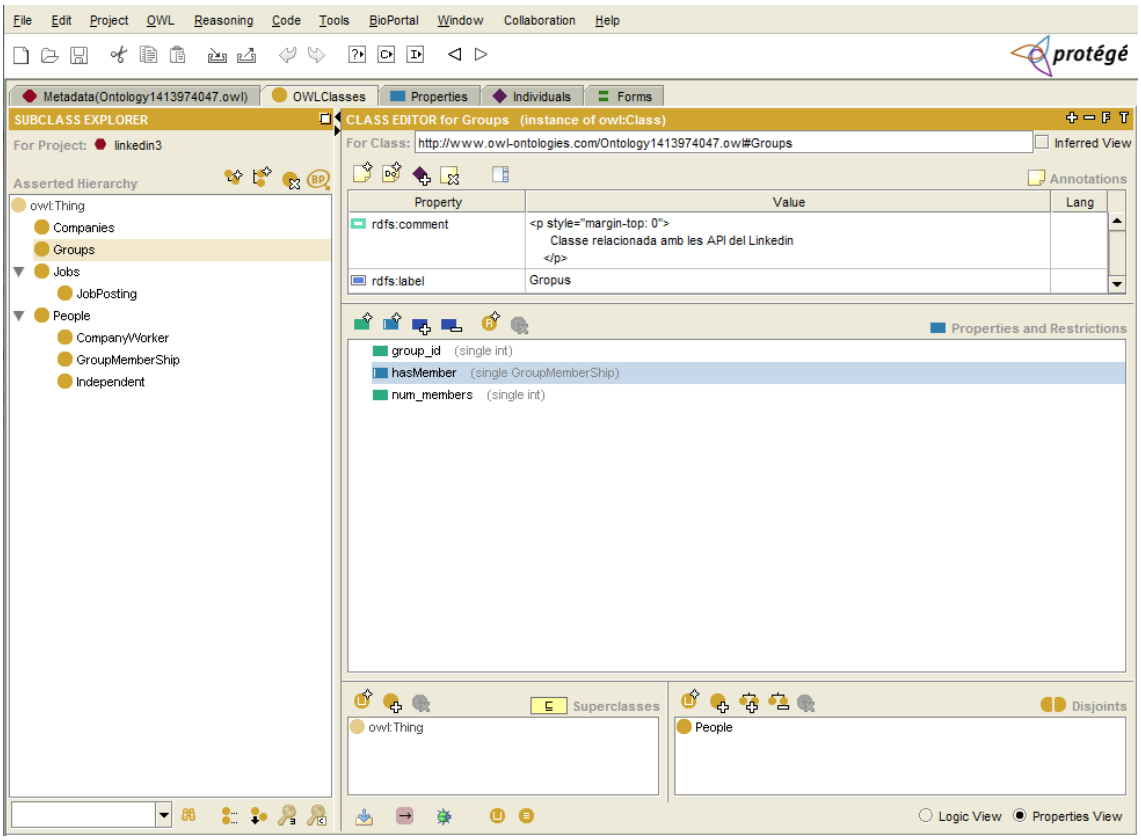
Class Groups

La classe *Groups* segons les especificacions de l'API conté diversos camps de dades. He decidit crear altres classes a partir d'aquestes dades i només deixaré les dades més rellevants per definir el *Groups*.

Les classes que he derivat, que no les he fet subclasse, son *GroupMemberShip*, que seria una subclasse de *People.*

Més informació dels camps contemplats per LinkedIn [14].

propietat	**tipus**	**tipus dada**	**descripció**
id	Datatype	int	Identificador del *Groups*. Dada oferta per el LinkedIn
name	Datatype	string	Nom del *Grup*
description	Datatype	string	Descripció del *Group*
posts	Datatype	int	Nom de la descripció del seu perfil professional
num-members	Datatype	int	Quantitat de membres que te el *Group*



Il·lustració 10: Protégé. Classe Groups

Class GroupMemberShip

Classe derivada de **People**. Porta cera informació relacionada amb al grup a que pertany.

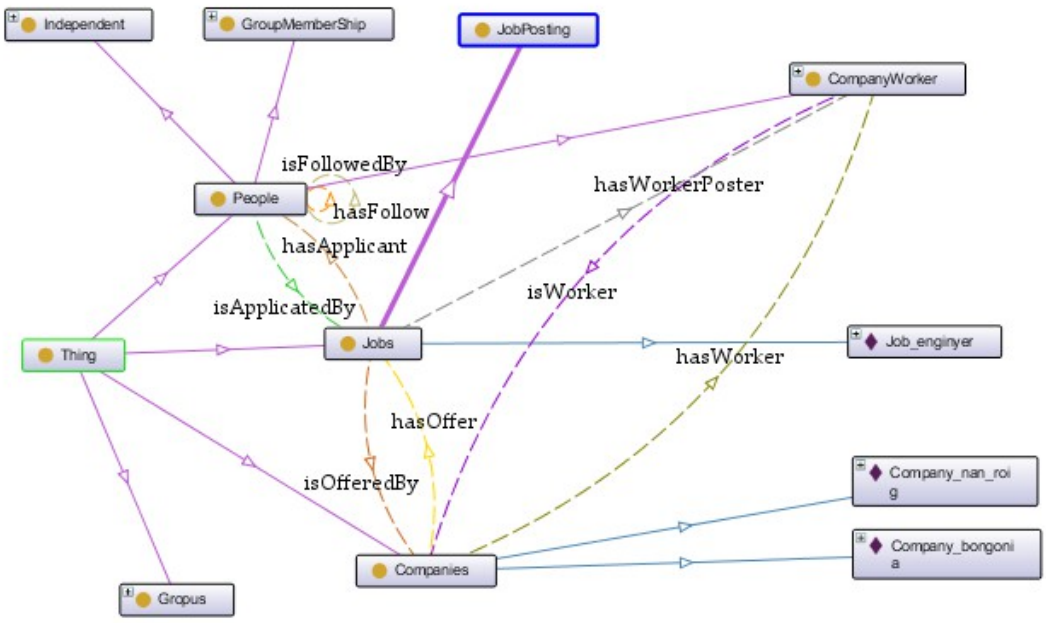
propietat	**tipus**	**tipus dada**	**descripció**
person	Datatype	string	Nom de la persona pertanyent al grup
group-id	Datatype	int	Identificador del grup al que pertany

2.5 Representació de les classes amb el Protégé

El programari Protégé disposa d'una eina per visualitzar gràficament el graf creat per l'ontologia, les seves relacions i les propietats d'objecte.

2.5.1 Esquemes Ontograph

Esquema desenvolupat de les interrelacions entre les classes **Companies**, **Jobs**, **People** *(en representació de les seves instàncies/indviduals):*



Il·lustració 11: Ontograph expandit.

En el ontograf de la figura anterior es despleguen les entitats principals **Jobs**, **Companies** i **People**. També hi observem les instàncies creades de **Companies**, *Company_nan_roig* i *Company_bongonia* i les instàncies de **Jobs**, Job_enginyer.

Podem observar-hi també les relacions d'objecte que es generen[3].

2.6 Simplificació

Per simplificar i acotar l'ontologia a les necessitats del domini estudiat, les classes *Posts* i *Comments*, relatives a informació d'un Group en un principi les deixarem fora del domini, per tant no les desenvoluparem.

També no hem contemplat les entitats del LinkedIn *Person* i *Position*, pel mateix motiu.

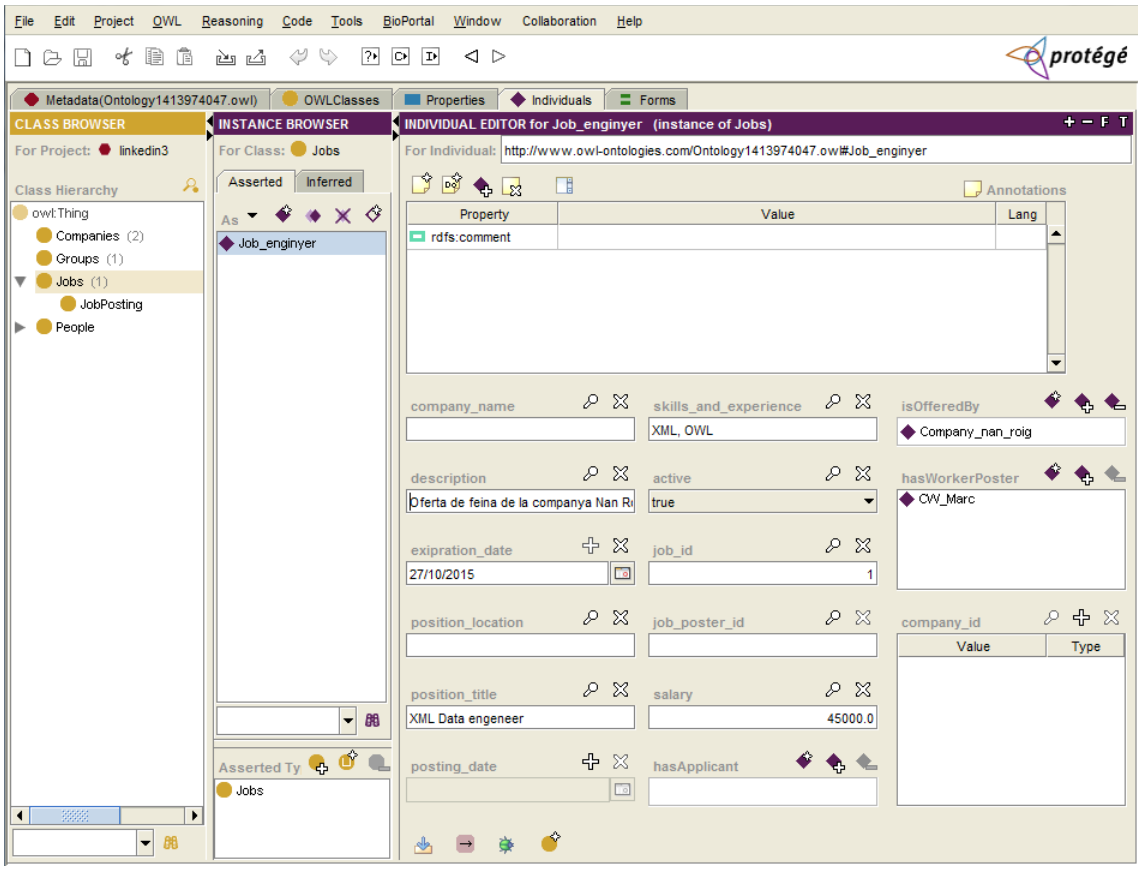
3 Fletxes amb línies discontínues.

2.7 Nomenclatura

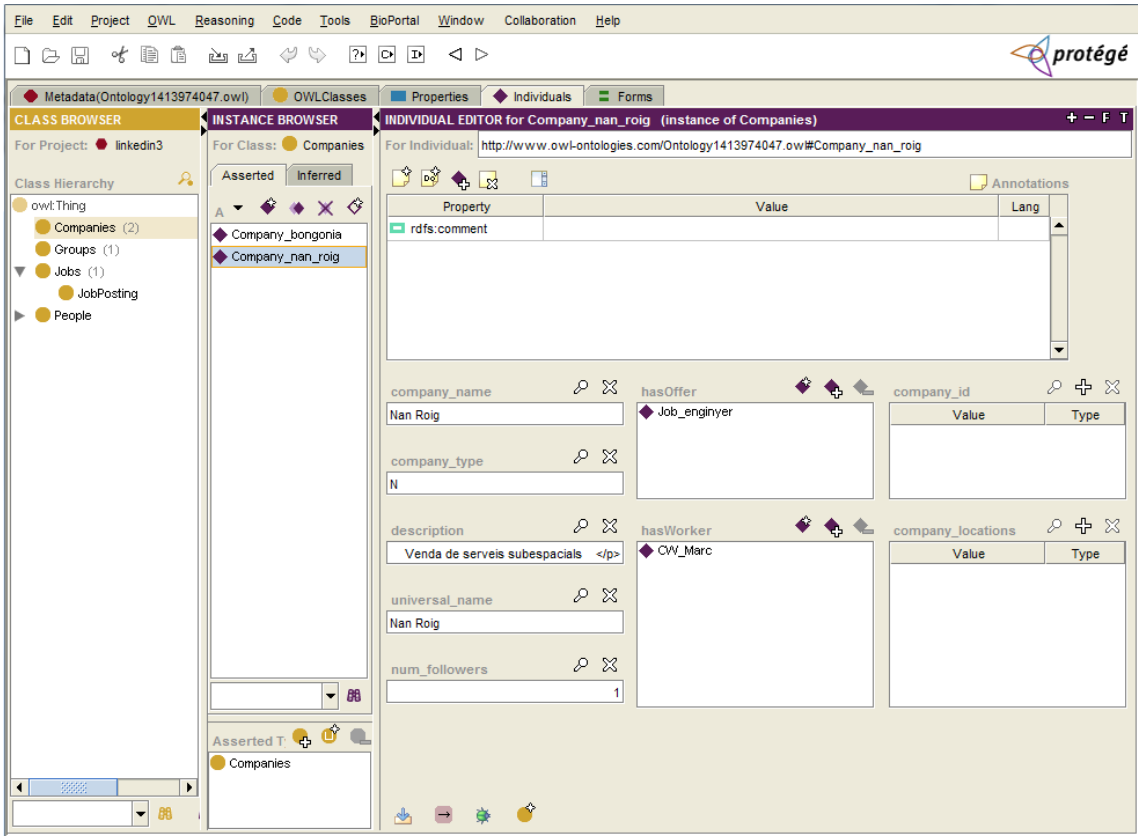
Es convenient anomenar les propietats amb el mateix nom de propietat que ens ofereix les API. D'aquesta manera a l'hora de poblar l'ontologia el mapatge de camps API - Ontologia serà més descriptiu.

2.8 Creació d'instàncies

S'han creat instàncies per mirar de provar el bon funcionament de l'ontologia. La instància principal es fa sobre la classe **Jobs**, i una altre de molt important també seria la de la classe **Companies**.



Il·lustració 12: Creació de individuals/instàncies. Instància de la classe Jobs.



Il·lustració 13: Creació de individuals/instàncies. Instància de la classe Companies.

3 Desenvolupament de l'aplicació

3.1 Accés a les API del LinkedIn

Per tal de poder accedir a les API del LinkedIn s'ha de demanar accés mitjançant el sistema d'autenticació OAUTH 2.0. És un procediment que sembla senzill però port portar a confusions en alguns procediments. Un cop LinkedIn et dóna accés se t'assignen unes claus, però posteriorment has de demanar un "token" per tal de poder accedir a les dades. Aquest token ja serà demanat mitjançant el programari desenvolupat.

Per una altra banda certes API del LinkedIn, en concret l'API de **Jobs** i **JobPosting** s'ha de demanar accés particular amb les teves credencials.

3.2 Proves de les API

Cal tenir un compte LinkedIn per tal de poder utilitzar les API's que posen a disposició. Un cop creat l'usuari és convenient demanar accés a les API de la classe **Jobs** i **JobsPosting**, ja que LinkedIn les ofereix prèvia demanda.

Les consultes que es faran des de l'aplicatiu són:

Per consultar els **Jobs** que existeixen, en consultarem 30:

```
https://api.linkedin.com/v1/job-search:(jobs:
(id,active,posting-date,expiration-date,skills-and-
experience,salary,company:(id),position:
(title,location),job-poster:(id)))?count=30
```

Mitjançant aquesta **url** demanem a les API de **Jobs** del LinkedIn que ens remeti les dades següents: el `id` de la feina, si la feina esta o no activa (`active`), la data que en que es va donar d'alta la oferta de feina i la data que expira la oferta (`posting-date,expiration-date`), els coneixements que ha de tenir el candidat (`skills-and-experience`), el salari ofert (`salary`) i diversa informació de la companyia ofertant. També informem a LinkedIn que ens envii 30 ofertes de feina (`count=30`).

Per consultar les **Companies** implicades:

```
http://api.linkedin.com/v1/companies/'+node_company.text  +':
(id,name,description,universal-name,company-type,
locations,num-followers)
```

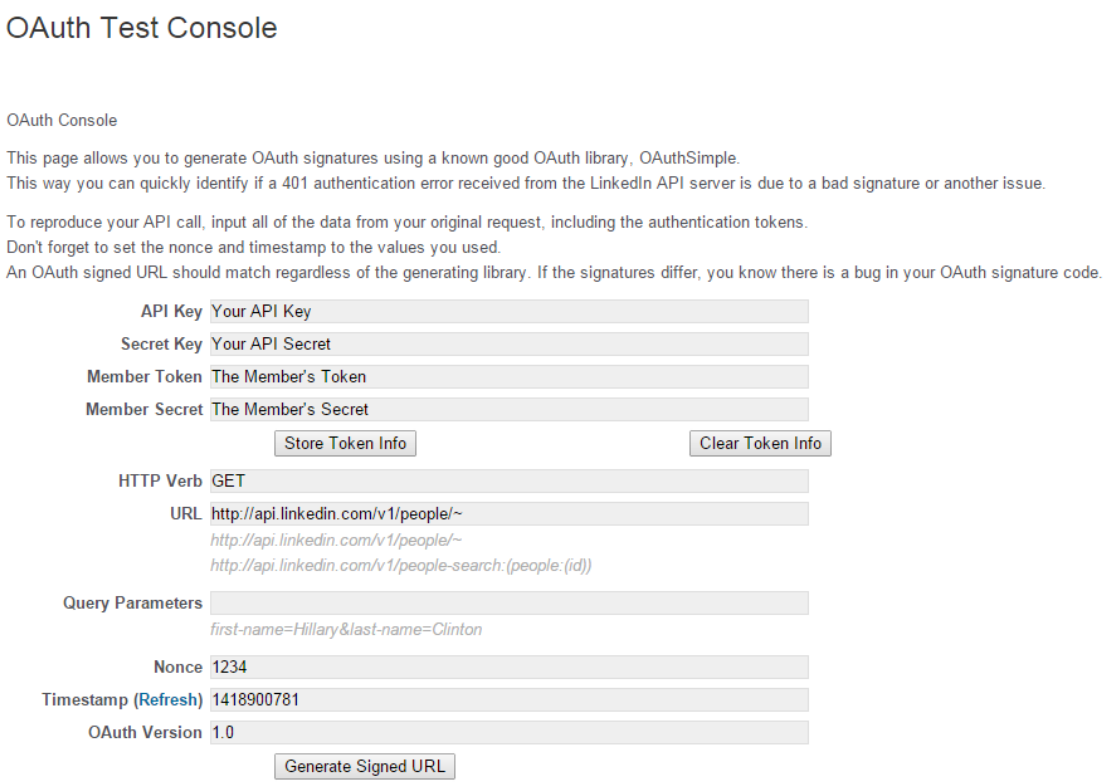
La cadena **url** anterior demana a les API de **Companies** del LinkedIn informació de l'empresa amb el `id` de la companyia extret de la consulta anterior i que li passem com a paràmetre (`node_company.text`), en concret demanem el nom de la companyia (`name`), una descripció (`description`), el nom genèric dintre del linkedIn (`universal-name`), el tipus que companyia que és (`company-type`), les ubicacions que pot tenir aquesta companyia (`locations`) i el nombre de seguidors dintre del LinkedIn que té (`num-followers`).

Per les **People** implicades:

```
http://api.linkedin.com/v1/people/'+node_job_poster.text+':
(id,first-name,last-name,headline)
```

Es demanen dades de la persona que ha fet el post de la feina com son el nom el cognom (`first-name,last.name`), i el títol de la feina que exerceix (`headline`).

LinkedIn ofereix una consola HTTP on et permet provar les funcionalitats i permisos que el teu usuari té. [10]. En aquesta consola que mostrem en la figura següent, hi informem en el camp **url** les consultes http o url's que hem descrit anteriorment. Hem de tindre en compte de seleccionar oauth 2.0, que és el mètode que hem seguit per autenticar-nos a LinkedIn.



Il·lustració 14: Consola OAuth del LinkedIn per probar consultes a les API.

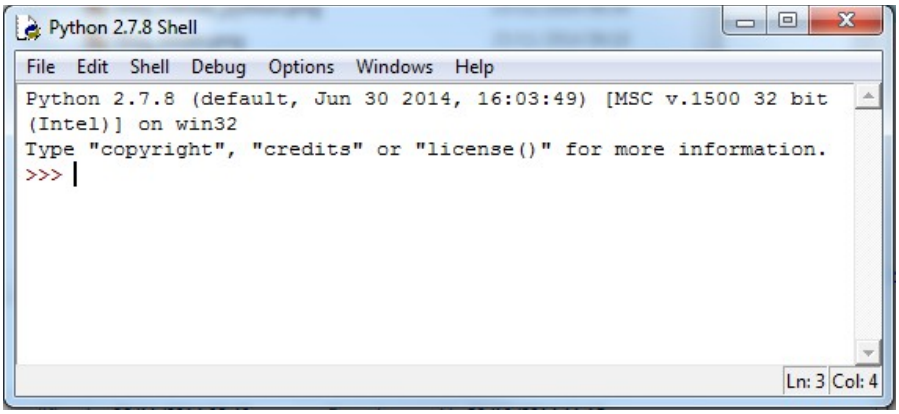
3.3 Propietats de l'ontologia i de les API

És molt important haver decidit quins dels camps que ens ofereixen les diferents API's s'utilitzaran per la nostra ontologia. Seria convenient limitar el nombre d'slots (propietats) que utilitzarem. Hem de descartar aquelles propietats que no ens donin resposta al problema acotat al nostre domini.

Es convenient anomenar les propietats amb el mateix nom de propietat que ens ofereix les API. D'aquesta manera a l'hora de poblar l'ontologia el mapatge camps API - Ontologia serà més descriptiu.

3.4 Programari necessari pel desenvolupament

El desenvolupament es farà amb el llenguatge Python, versió 2.7.8 [15]. Per l'execució usarem la consola gràfica IDLE (Python GUI), en la seva versió per windows.



Il·lustració 15: Entorn gràfic d'execució de Python. GUI.

Python a més de ser un llenguatge d'alt nivell i encara que interpretat és molt àgil. Disposa de moltes llibreries que facilitaran la feina en la fase de desenvolupament.

Llibreries:

oauth2: Facilitarà la feina d'aconseguir les credencials del LinkedIn. [16]

python-linkedin: Facilitarà les consultes a les API. [17]

rdflib: S'usarà a l'hora de construir el graf de l'ontologia. [18]

xml.etree.ElementTree: Servirà per tractar l'arbre XML de les API. [19]

Totes les llibreries s'han d'instal·lar mitjançant l'aplicació ***pip***

```
c:\pip install oauth

c:\pip install python-linkedin

c:\pip install rdflib
```

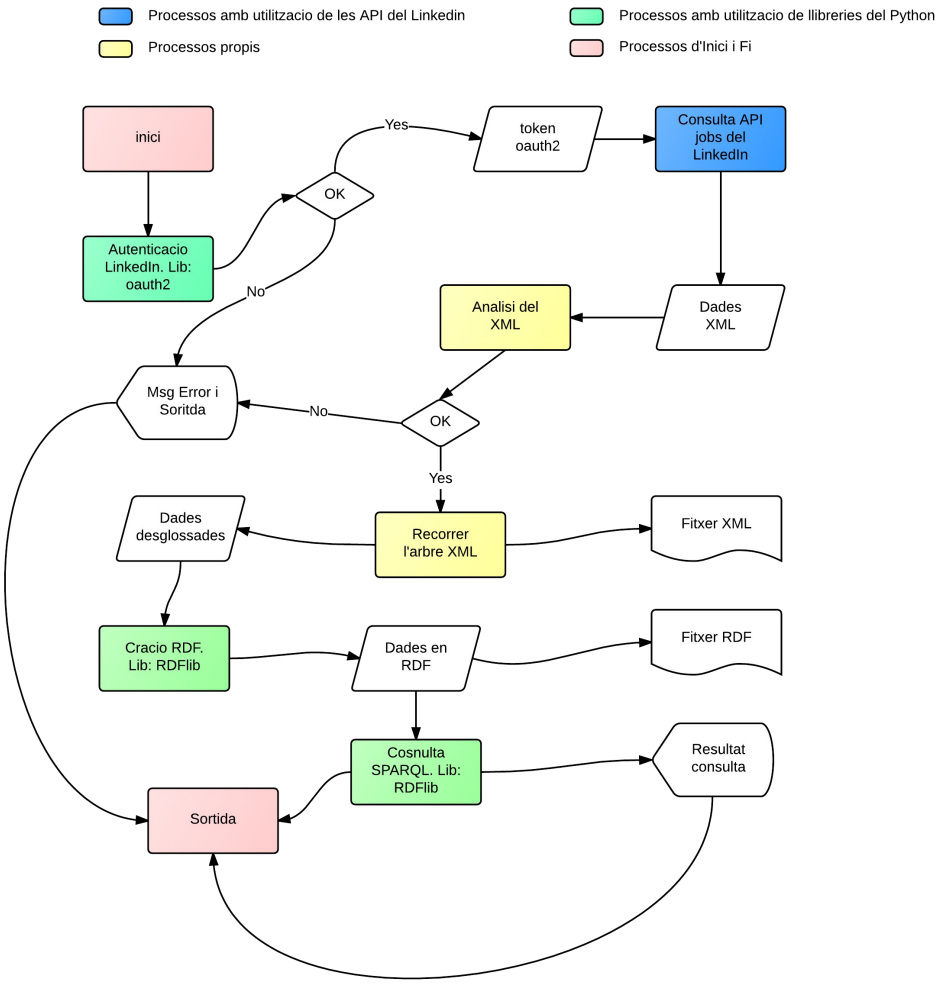
Si tot ha anat bé hauríem de veure una sortida similar a aquesta:

```
Downloading/unpacking python-linkedin

Downloading python-linkedin-4.1.tar.gz

Running setup.py
(path:c:\users\mescar~1.ana\appdata\local\temp\pip_build_mescarmis\python-
linkedin\setup.py) egg_info for package python-linkedin

Requirement already satisfied (use --upgrade to upgrade): requests>=1.1.0
in c:\python27\lib\site-packages (from python-linkedin)

Requirement already satisfied (use --upgrade to upgrade): requests-
oauthlib>=0.3 in c:\python27\lib\site-packages (from python-linkedin)

Installing collected packages: python-linkedin

  Running setup.py install for python-linkedin

Successfully installed python-linkedin

Cleaning up...
```

3.5 Lògica de l'aplicació

Per una òptima execució de l'script, es llança l'aplicació mitjançant la consola GUI del Python, prement F5. També és possible llençar l'script des de la consola del sistema.

En els següent diagrama de flux es pot veure la lògica de l'script:

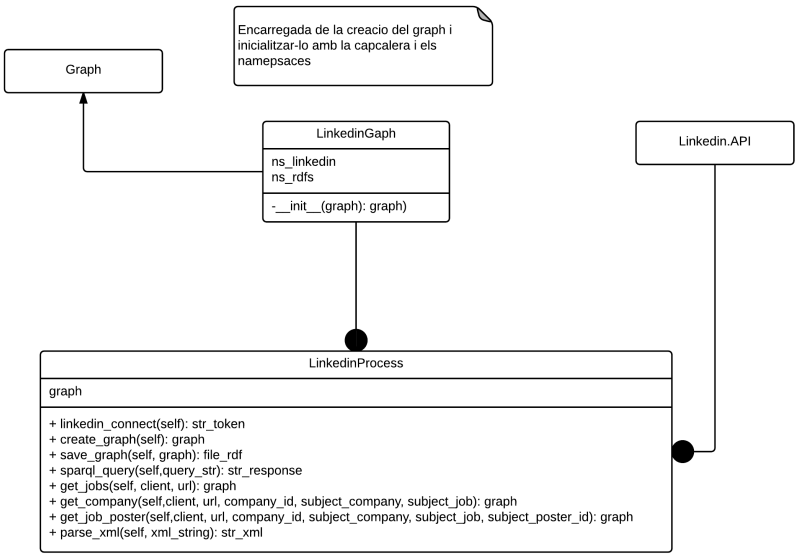


Il·lustració 16: Diagrama de flux

3.6 Disseny de l'aplicació (UML)

Amb els següents diagrames UML es presenta el disseny de l'aplicació des de la prespectiva de tècniques en desenvolupament de programari.

Amb el diagrama de classes mostrem les dues classes implicades en l'script. Val a dir que al ser una aplicació simple s'hagués pogut dissenyar sota el concepte de programació estructurada.

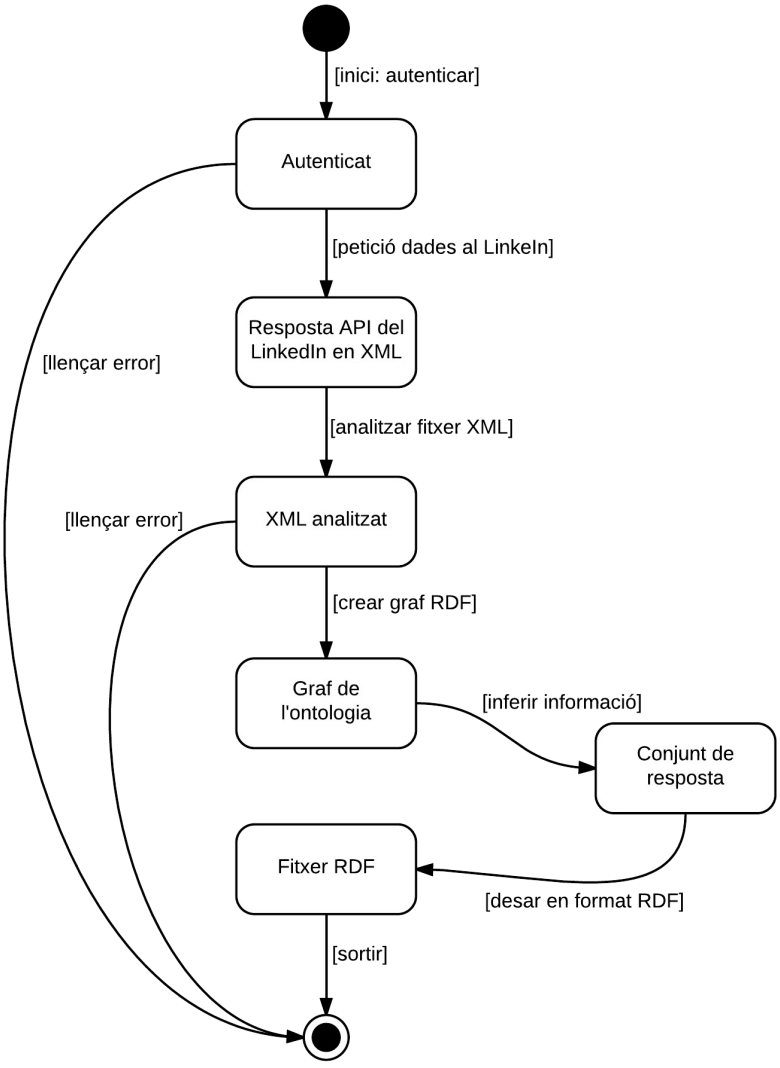


Il·lustració 17: Diagrama de classes. UML.

La classe que porta a terme tots els procediments per la població de l'ontologia és la classe **LinkedinProcess**, la qual disposa dels mètodes necessaris per l'autenticació mitjançant oauth2.0, consulta a les API del LinkedIn, recorregut de l'arbre XML i la creació i desat de l'ontologia poblada en format RDF/XML. També fa ús de les classes de les API del LinkedIn. També crea un objecte de la classe Graph del tipus LinkedinGraph on s'hi aniran desant totes dades del graph.

La classe **LinkedinGraph**, simplement correspon al graph RDF i hereta la classe Graph de la llibreria rdflib, és l'encarregada d'inicialitzar el graf.

Per últim i malgrat haver mostrat ja el diagrama de flux es presenta el següent diagrama d'estats on es mostren els estats que va assolint l'aplicació a mesura que va executant les diferents consultes a les API i va conformant el fitxer *.rdf* de sortida:



Il·lustració 18: Diagrama d'estats. UML.

3.7 Fitxer principal *serializeLinkedin.py*

L'únic fitxer de l'aplicació és fet en Python i es diu **serializeLinkedin.py**.

L'apartat o cos principal és la funció `__main__`que s'encarrega d'instanciar la classe `LinkedinProcess` que fa tota l'activitat del programa. I en la part final també fa crides a mètodes de la mateixa classe per executat les consultes SPARQL:

```
if __name__ == '__main__':
    print u'>>> Iniciant procés'
    try:
        # Instancia la classe que farà tot el procés
        linkedin_process = LinkedinProcess()
        # Obtenim les credencials per poder connectar-nos i fer consultes al LinkedIn
        client = linkedin_process.linkedin_connect()
        # Obtenim el primer arbre XML amb la informació de JOBS
           url    =    "https://api.linkedin.com/v1/job-search:(jobs:(id,active,posting-
date,expiration-date," \
               "skills-and-experience,salary,company:(id),position:(title,location),job-
poster:(id)))?count=30"
        # Processa el XML base aportat per la consulta feta a l'api de LinkedIn
        jobs = linkedin_process.get_jobs(client,url)
        linkedin_process.create_graph()
        tree = ET.fromstring(jobs)
        linkedin_process.parse_xml(tree)
        # Desa el graph
        linkedin_process.save_graph()
          # Fem una sèrie de consultes SPARQL per comprobar que s'han desat dades en
l'ontologia
        print u'\n>>> Consultes SPARQL:'

        query_str = 'select ?h where {?h rdf:type linkedin:Jobs}'
        response = linkedin_process.sparql_query(query_str)
        print '\tNombre de Jobs desats: ' + str(len(response))
        for row in response:
            print '\t\t',
            print str(row)

.....
```

Un cop instanciada la classe LinkedinProcess, comença el procés per poblar l'ontologia amb la informació provinent de les API de LinkedIn.

El primer pas és autenticar-nos mitjançant la crida `client = linkedin_process.linkedin_connect()`. En cas que hàgim pogut autenticar-nos sense problemes ens retornarà l'objecte `client` amb la informació pertinent de la connexió autoritzada:

```
def linkedin_connect(self):
    #*****************************************************************
    # Aconsegueix les credencials necessaries per poder fer consultes a les
    # API del LinkedIn. Ho fa mitjançant oauth2 i ussant la llibrerria de python
    # oauth2
    # Sortida:
    # Objecte oauth amb el  token necessari per fer consultes
    #*****************************************************************
    print u'>>> Connectant amb LinkedIn '
    try:
        CONSUMER_KEY            = "???????" #claus del linkedin
        CONSUMER_SECRET         = "???????" #claus del linkedin
        OAUTH_TOKEN             = "???????" #token del oauth
        OAUTH_TOKEN_SECRET      = "???????" #token del oauth
        consumer = oauth.Consumer(CONSUMER_KEY, CONSUMER_SECRET)
        client = oauth.Client(consumer)

        consumer = oauth.Consumer(
             key=CONSUMER_KEY,
             secret=CONSUMER_SECRET)

        token = oauth.Token(
             key=OAUTH_TOKEN,
             secret=OAUTH_TOKEN_SECRET)
        client = oauth.Client(consumer, token)
        return(client)
    except:
        print u'>>> Possible error de connexió', sys.exc_info()[0]
```

Un cop autenticats es demana a l'API **Jobs** de LinkedIn que ens retorni informació de 30 feines i es començarà el poblament de l'ontologia:

```
 url = "https://api.linkedin.com/v1/job-search:(jobs:(id,active,posting-date,expiration-
date,"skills-and-experience,salary,company:(id),position:(title,location),job-poster:
(id)))?count=30"
# Processa el XML base aportat per la consulta feta a l'api de LinkedIn
jobs = linkedin_process.get_jobs(client,url)
linkedin_process.create_graph()
tree = ET.fromstring(jobs)
linkedin_process.parse_xml(tree)
```

El mètode `linkedin_process.create_graph()` crea el objecte Graph de la classe importada **rdflib**.

La cadena `url` conté la consulta a les API de JOBS. Posteriorment es crida al mètode `get_jobs()` que obté l'arbre XML amb el resultat de la consulta. I aquest resultat es passa en forma de cadena a l'objecte de la llibreria **xml.etree** per parsejar-ne el contingut.

```
def get_jobs(self,client,url):
    resp, content = client.request(url)
    return(content)
```

Amb el mètode `parse_xml()` es recorre l'arbre i es comença a omplir l'ontologia. L'encapçalament del mètode és el següent:

```
def parse_xml(self,content):
    #*****************************************************************
    #    Processa el XML base aportat per les API de LinkedIn.
    #
    #    Entrda:
    #       content: XML retornat per les API del LinkedIn
    #
    #    Sortida:
    #       Graph RDF actualitzat amb la info de Jobs
    #*****************************************************************

    print u'>>> Processant arbre XML obtingut de LinkedIn'
    subject_company = ''
    subject_job = ''
```

```
    for jobs in content.iterfind('./jobs'):
        for job in jobs.getchildren():
            if job.tag == 'job':
                subjectBN = BNode()
                print u'>>> Afegint informació de \'Jobs\' al Graph ',
                for node in job.getchildren():
                    if node.tag == 'id':
                        subject_job = URIRef(self.graph.ns_linkedin['job_' + node.text])
                        print 'job_' + node.text
                        self.graph.add((subject_job, RDF.type, self.graph.ns_linkedin.
Jobs))
                        self.graph.add((subject_job, self.graph.ns_linkedin.job_id,
Literal(node.text)))


......
```

El següent codi recorre l'element xml **jobs** i tots els seus fills:

```
for jobs in content.iterfind('./jobs'):
    for job in jobs.getchildren():
        if job.tag == 'job':
.......
```

El codi:

```
self.graph.add((subject_job, RDF.type, self.graph.ns_linkedin.Jobs))
```

crea una entrada en el Graph afegint-hi un objecte JOB. Aquest codi l'executa la llibreria **rdflib**.

En aquest mateix mètode es fan altres crides a altres mètodes per anar poblant l'ontologia amb la informació de les COMPANIES i PEOPLE extretes de noves consultes a les API.

El següent codi recorre l'element **company** i n'extreu l'identificador per tal de poder fer la consulta a les API de COMPANIES per obtenir informació de la companyia que ha publicat la feina:

```

if node.tag == 'company':
    for node_company in node.getchildren():
        if node_company.tag == 'id':
            company_id = node_company.text
```

```
            url = 'http://api.linkedin.com/v1/companies/' + node_company.text + ':' \
                  '(id,name,description,universal-name,company-type,locations,num-
followers)'
            subject_company = URIRef(self.graph.ns_linkedin['company_' + company_id])
            print u'>>> Afegint informació de \'Companies\' al Graph ',
            print 'company_' + company_id
            #fa la crida al procés per obtenir la informació de l'empresa

LinkedinProcess.get_company(self,client,url,company_id,subject_company,subject_job)
            self.graph.add((subject_job, self.graph.ns_linkedin.isOfferedBy,
subject_company))
```

La línia `LinkedinProcess.get_company(self,client,url,company_id, subject_company, subject_job)` crida al mètode get company amb vàris paràmetres el qual s'encarregarà de poblar la ontologia amb les instàncies d'objectes **companies**.

```
def get_company(self,client,url,company_id,subject_company,subject_job):
    resp, content = client.request(url)
    subjectBN = BNode()
    self.graph.add((subject_company, RDF.type, self.graph.ns_linkedin.Companies))
    self.graph.add((subject_company, self.graph.ns_linkedin.HasOffer, subject_job))
    self.graph.add((subject_company, self.graph.ns_linkedin.company_id,
Literal(company_id)))
    self.graph.add((subject_job, self.graph.ns_linkedin.company_id,
Literal(company_id)))
    sub_xml = ET.fromstring(content)
    for company in sub_xml.getchildren():
        if company.tag == 'name':
            self.graph.add((subject_company, self.graph.ns_linkedin.company_name,
Literal(company.text)))
            self.graph.add((subject_job, self.graph.ns_linkedin.company_id,
Literal(company_id)))
        if company.tag == 'description':
            self.graph.add((subject_company, self.graph.ns_linkedin.description,
Literal(company.text)))
        if company.tag == 'universal-name':
            self.graph.add((subject_company, self.graph.ns_linkedin.universal_name,
Literal(company.text)))
        if company.tag == 'num-followers':
            self.graph.add((subject_company, self.graph.ns_linkedin.num_followers,
Literal(company.text)))
        if company.tag == 'company-type':
```

```
            for code in company.iter():
                if code.tag == 'name':
                    self.graph.add((subject_company,
self.graph.ns_linkedin.company_type, Literal(code.text)))
        if company.tag == 'locations':
            for location in company.find('location'):
                if location.tag == 'address':
                    str_adr = ''
                    for address in location.getchildren():
                        if address.tag == 'street1':
                            if address.text:
                                str_adr = str_adr + address.text
                        if address.tag == 'city':
                            if address.text:
                                str_adr = str_adr + ', ' + address.text
                        if address.tag == 'postal-code':
                            if address.text:
                                str_adr = str_adr + ', ' + address.text
            self.graph.add((subject_company, self.graph.ns_linkedin.company_locations,
Literal(str_adr)))
......
```

Un cop tenim el graf complert amb tota la informació que es desitja es procedeix a serialitzar-lo en un fitxer RDF/XML. La crida es fa en el mètode main de l'sctript amb el codi `linkedin_process.save_graph()`.que crida al mètode `save_graph()` de la instància `linkedin_process`.

Un cop tenim el graf desat en format *rdf* ja podem fer unes primeres consultes SPARQL, per exemple aquesta que ens dóna el nombre de instàncies de l'objecte **people** que té el graph

```
query_str = 'select ?h where {?h rdf:type linkedin:People}'
response = linkedin_process.sparql_query(query_str)
print '\tNombre de People desats: ' + str(len(response))
for row in response:
    print "\t\t",
    print str(row)
.......
```

3.8 Fitxer RDF creat

El fitxer RDF creat conté diverses instancies dels objectes de l'ontologia.

Classe J**obs**:

```
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413974047.owl#job_12440824">
  <linkedin:posting_date>2014/11/26</linkedin:posting_date>
  <linkedin:skills_and_experience>&lt;ul&gt;&lt;li&gt;Good knowledge of display advertising,
  Affiliation, Ad networks and retargeting &lt;/li&gt;&lt;li&gt;Good knowledge in MS Excel
  &lt;/li&gt;&lt;li&gt;Strong analytics capabilities &lt;/li&gt;&lt;li&gt;Basic knowledge on Web
  analytics tools (i.e. Google analytics, Omniture …) &lt;/li&gt;&lt;li&gt;Good understanding on online
  marketing &amp;amp; KPI's &lt;/li&gt;&lt;li&gt;English
  &lt;p&gt;&lt;/p&gt;&lt;/li&gt;&lt;/ul&gt;&lt;p&gt;&lt;strong&gt;Person specification:&lt;/strong&gt;
  &lt;/p&gt;&lt;ul&gt;&lt;li&gt;Detail-oriented and resolutive person &lt;/li&gt;&lt;li&gt;Autonomous
  &amp;amp; ressourceful &lt;/li&gt;&lt;li&gt;Passionned for data analysis, online marketing and new
  technologies&lt;/li&gt;&lt;/ul&gt;</linkedin:skills_and_experience>
  <rdf:type rdf:resource="http://www.owl-ontologies.com/Ontology1413974047.owl#Jobs"/>
  <linkedin:expiration_date>2014/12/26</linkedin:expiration_date>
  <linkedin:position_title>Performance marketing Executive (display&amp;affiliates)
  </linkedin:position_title>
  <linkedin:company_id>20741</linkedin:company_id>
  <linkedin:hasWorkerPoster
  rdf:resource="http://www.owl-ontologies.com/Ontology1413974047.owl#companyworker_FYePsS517n"/>
  <linkedin:active>true</linkedin:active>
  <linkedin:job_id>12440824</linkedin:job_id>
  <linkedin:isOfferedBy
  rdf:resource="http://www.owl-ontologies.com/Ontology1413974047.owl#company_20741"/>
</rdf:Description>
```

Il·lustració 19: RDF. Classe jobs.

Classe **Companies**:

```
<rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1413974047.owl#company_20741">
  <linkedin:company_type>Privately Held</linkedin:company_type>
  <linkedin:HasOffer rdf:resource="http://www.owl-ontologies.com/Ontology1413974047.owl#job_12441143"/>
  <linkedin:HasOffer rdf:resource="http://www.owl-ontologies.com/Ontology1413974047.owl#job_12440700"/>
  <linkedin:hasWorker
  rdf:resource="http://www.owl-ontologies.com/Ontology1413974047.owl#companyworker_FYePsS517n"/>
  <linkedin:num_followers>20790</linkedin:num_followers>
  <rdf:type rdf:resource="http://www.owl-ontologies.com/Ontology1413974047.owl#Companies"/>
  <linkedin:description>eDreams is the largest independent european online travel agency. Its
  head-office is situated in Barcelona, Spain and it has also office in Italy. It focuses on offering
  the best selection of flights, hotels and vacation packages through the use of innovative search,
  packaging and booking engines via its websites (in English, French, Spanish, Italian, Portuguese,
  German and Turkish). Clients are offered all these services in a comfortable, simple manner through
  state-of-the-art booking and search engine technology. They include traditional airline and low cost
  flights, offering the best prices in the market to its clients.

More than six million clients have reserved their hotels, flights and holiday package deals at eDreams
since the company first started its activities.

eDreams is the largest online travel agency in terms of sales in Southern Europe.</linkedin:description>
  <linkedin:company_id>20741</linkedin:company_id>
  <linkedin:company_locations>World Trade Center, Barcelona, 08039</linkedin:company_locations>
  <linkedin:HasOffer rdf:resource="http://www.owl-ontologies.com/Ontology1413974047.owl#job_12440824"/>
  <linkedin:universal_name>edreams</linkedin:universal_name>
  <linkedin:company_name>eDreams</linkedin:company_name>
</rdf:Description>
```

Il·lustració 20: RDF. Classe companies.

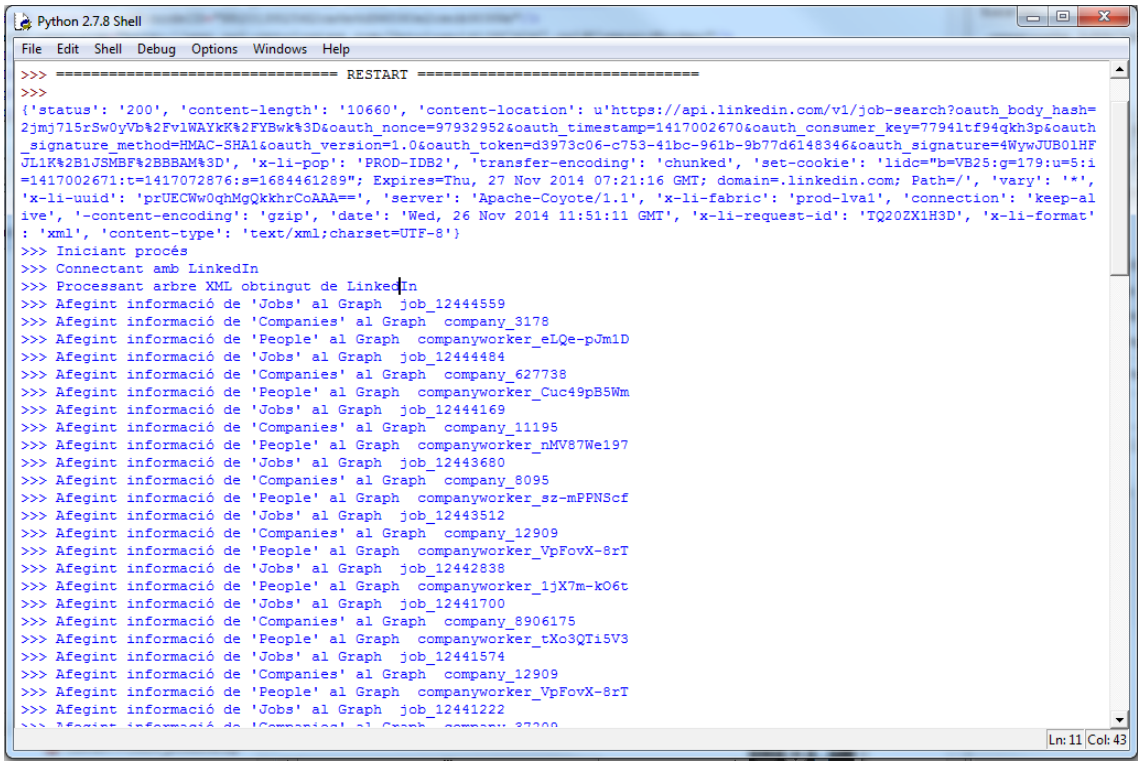
Classe **CompanyWorker**:

```
<rdf:Description
rdf:about="http://www.owl-ontologies.com/Ontology1413974047.owl#companyworker_FYePsS517n">
  <linkedin:IsWorkerPoster
  rdf:resource="http://www.owl-ontologies.com/Ontology1413974047.owl#job_12440700"/>
  <linkedin:isFollowedBy rdf:nodeID="N080e01f41dd7402492214f82a41dbe8f"/>
  <linkedin:IsWorkerPoster
  rdf:resource="http://www.owl-ontologies.com/Ontology1413974047.owl#job_12440824"/>
  <linkedin:headline>HR Support  en eDreams ODIGEO</linkedin:headline>
  <linkedin:isFollowedBy rdf:nodeID="N5b42479790964172aedaf54f82cc949b"/>
  <linkedin:IsWorkerPoster
  rdf:resource="http://www.owl-ontologies.com/Ontology1413974047.owl#job_12441143"/>
  <linkedin:first_name>Marc</linkedin:first_name>
  <linkedin:hasFollow rdf:nodeID="N080e01f41dd7402492214f82a41dbe8f"/>
  <rdf:type rdf:resource="http://www.owl-ontologies.com/Ontology1413974047.owl#CompanyWorker"/>
  <linkedin:isFollowedBy rdf:nodeID="N5cf362b32faa4f489b30bf0946d3e2f8"/>
  <linkedin:hasFollow rdf:nodeID="N5cf362b32faa4f489b30bf0946d3e2f8"/>
  <linkedin:isApplicantOf rdf:nodeID="N5cf362b32faa4f489b30bf0946d3e2f8"/>
  <linkedin:hasFollow rdf:nodeID="N5b42479790964172aedaf54f82cc949b"/>
  <linkedin:isApplicantOf rdf:nodeID="N5b42479790964172aedaf54f82cc949b"/>
  <linkedin:IsWorkerPoster
  rdf:resource="http://www.owl-ontologies.com/Ontology1413974047.owl#company_20741"/>
  <linkedin:last_name>G.</linkedin:last_name>
  <linkedin:isApplicantOf rdf:nodeID="N080e01f41dd7402492214f82a41dbe8f"/>
</rdf:Description>
```

Il·lustració 21: RDF. Classe CompanyWorker.

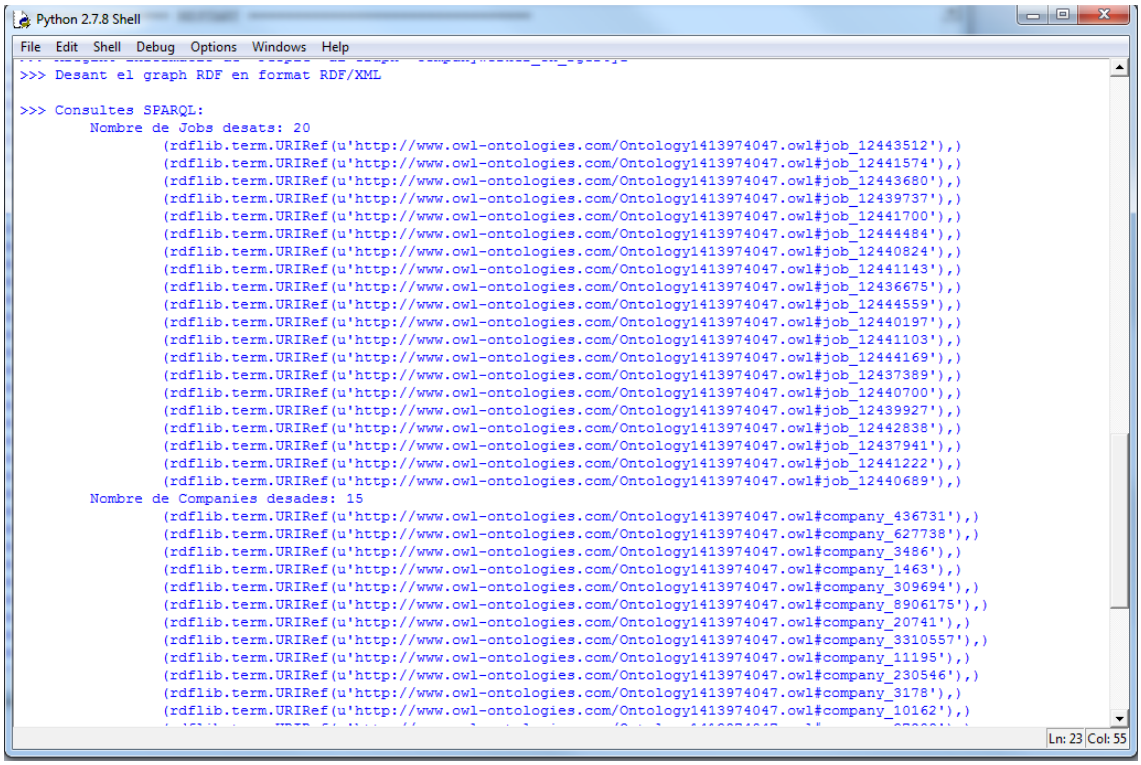
3.9 Resultat de l'execució

En la primera captura podem observar els missatges que l'aplicació ens va llençant com a resultat de les tasques que va executant.



Il·lustració 22: Resultat de l'execució de l'script 1.

I en la següent hi veiem el resultat de les consultes SPARQL llençades des del mateix script.



Il·lustració 23: Resultat de l'execució de l'script 2.

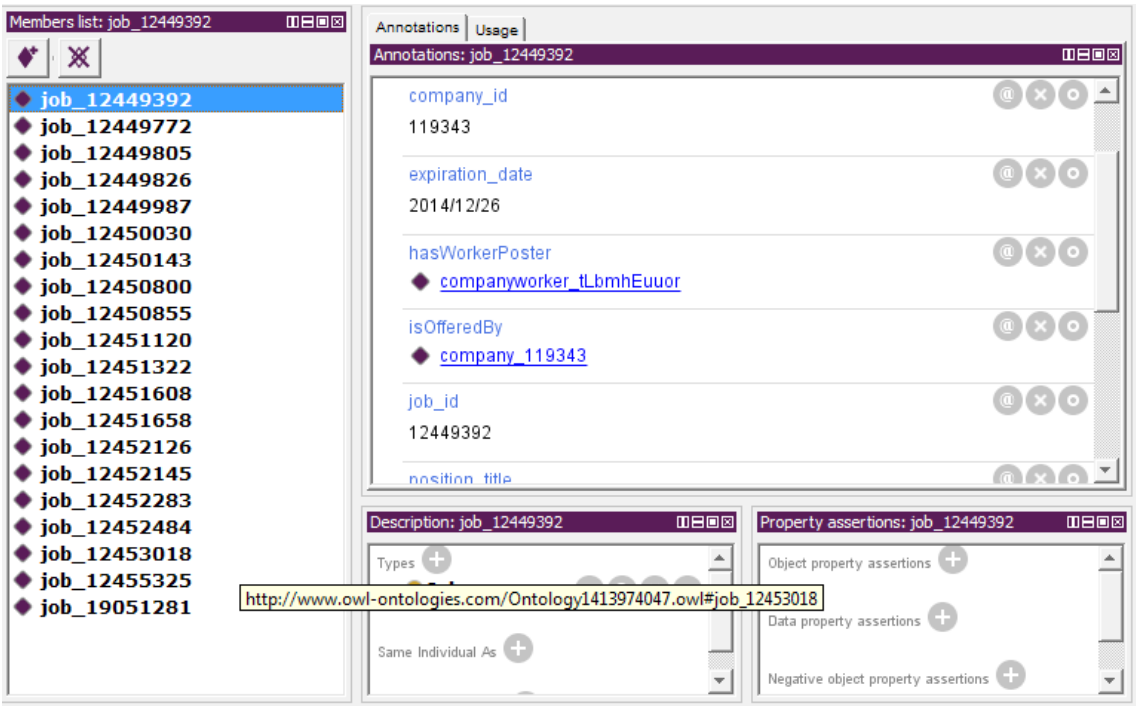
4 Tractament de la informació, consultes SPARQL

4.1 Visió general, creació d'instàncies

Com a objectiu principal s'ha assolit poder navegar mitjançant les instancies creades i inferir-hi informació a través de consultes SPARQL.

Amb el Protégé 5.0 obrim el fitxer RDF creat per veure'n les instàncies creades. En la pestanya Individuals del Protégé se'ns mostra les classes que s'han instanciat en el fitxer RDF podent-hi navegar per veure'n les propietats que tenen. A continuació presentem un resum del que podem veure amb el Protégé.

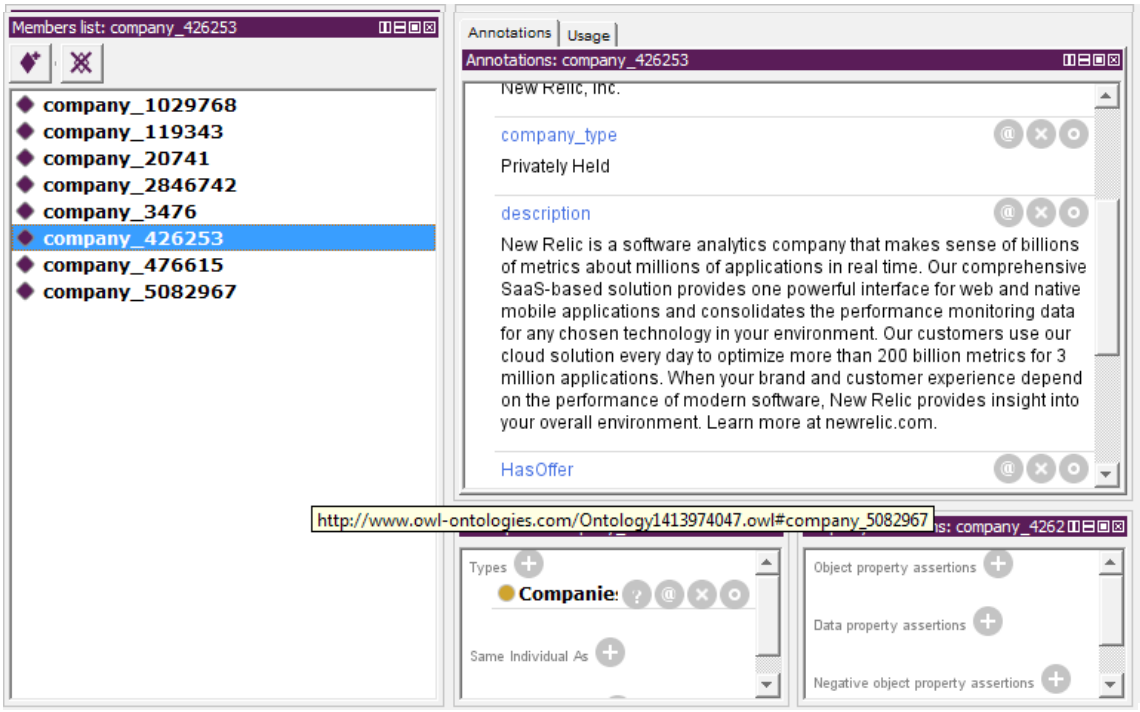
Instàncies creades de l'objecte **Jobs**:



Il·lustració 24: Instàncies d'objectes Jobs.

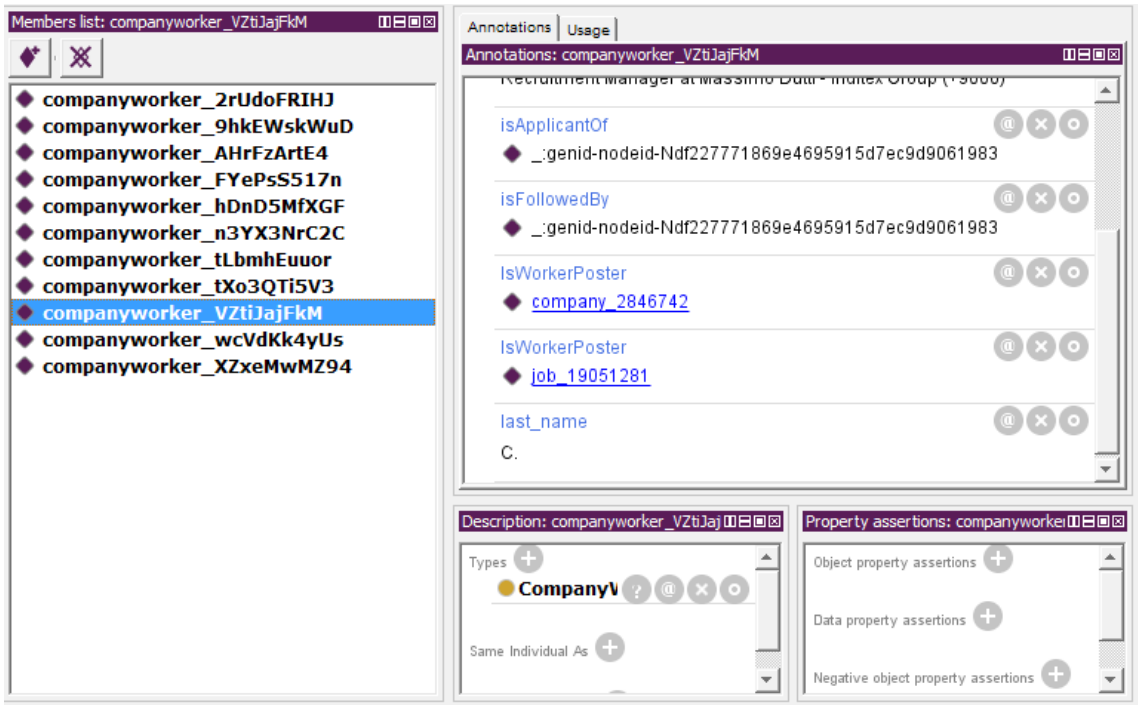
A l'esquerra de la figura hi veiem les instàncies creades. A la banda dreta podem navegar entre les seves propietats, tant d'objecte com de dades.

Instàncies creades de objectes **Companies**:



Il·lustració 25: Instàncies d'objectes Companies.

Instàncies creades de objectes **CompanyWorker**:



Il·lustració 26: Instàncies d'objectes People->CompanyWorker

4.2 Consultes SPARQL

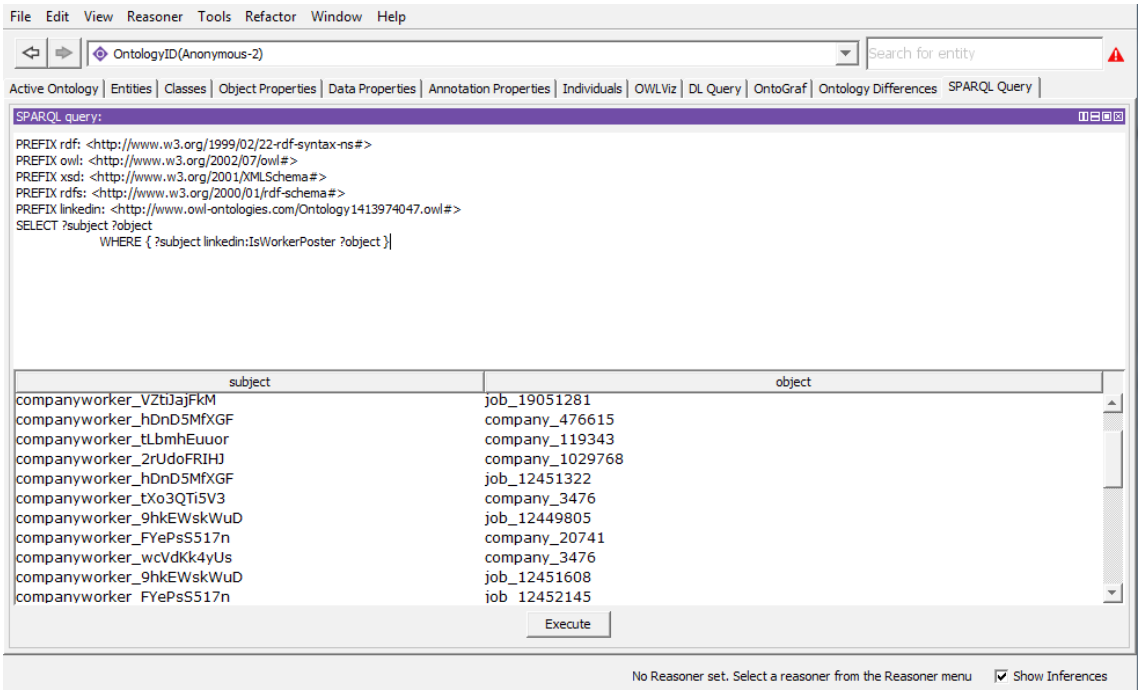
El llenguatge de consultes SPARQL és una adaptació del llenguatge SQL clàssic de consultes a base de dades però amb adaptacions per poder inferir informació de fitxers RDF, a ontologies.

Mitjançant la ontologia poblada amb dades del LinkedIn podem iniciar-nos en els conceptes bàsics de la creació de consultes SPARQL. El present treball no pretén fer un aprofundiment del llenguatge de consultes SPARQL, tan sols vol marcar-ne uns conceptes bàsics per provar el correcte funcionament de l'ontologia.

En la següent consulta SPARQL podem inferir quins treballadors pertanyen a quines companyies i quins treballadors han publicat la feina, mitjançant les clàusules PREFIX acotem els espais de noms on actuarà l'ontologia. En aquest exemple es veu molt clar el funcionament de la triple subjecte-predicat-objecte. `?subject` fa referència a instàncies de la classe `CompanyWorker` de **People** amb el predicat IsWorkerPoster i com a `?object` la companyia a la qual pertany.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX linkedin: <http://www.owl-ontologies.com/Ontology1413974047.owl#>

SELECT ?subject ?object
       WHERE { ?subject linkedin:IsWorkerPoster ?object }
```

La consulta s'ha aplicat mitjançant el Protégé 5.0, i és el següent:

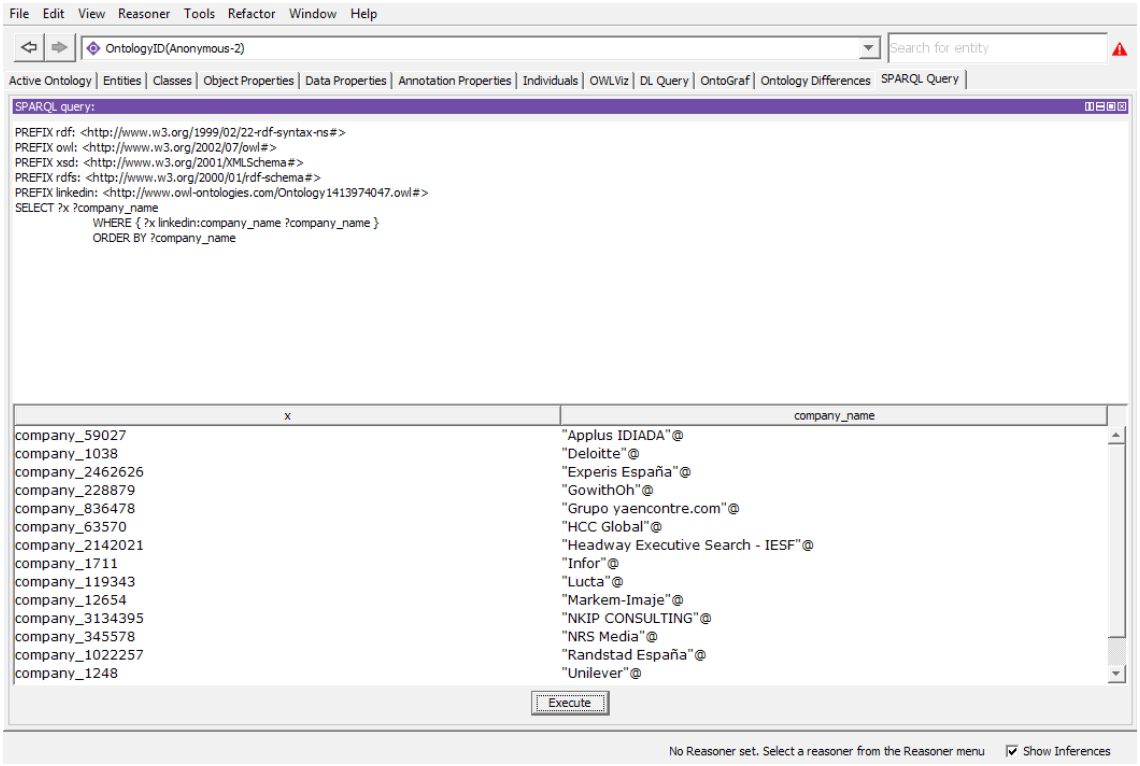


Il·lustració 27: Protégé. Consulta SPARQL 1.

Per il·lustrar una mica més en profunditat el funcionament de SPARQL us proposem un seguit de consultes més avançades.

En la següent consulta extraiem els noms de les companyies que ofereixen feina i les ordenem alfabèticament. En aquest exemple el subjecte `?x` ens donarà l'identificador de la companyia i com a objecte el nom `?company_name` de la companyia.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX linkedin: <http://www.owl-ontologies.com/Ontology1413974047.owl#>

SELECT ?x ?company_name
       WHERE {?x linkedin:company_name ?company_name}
       ORDER BY ?company_name
```



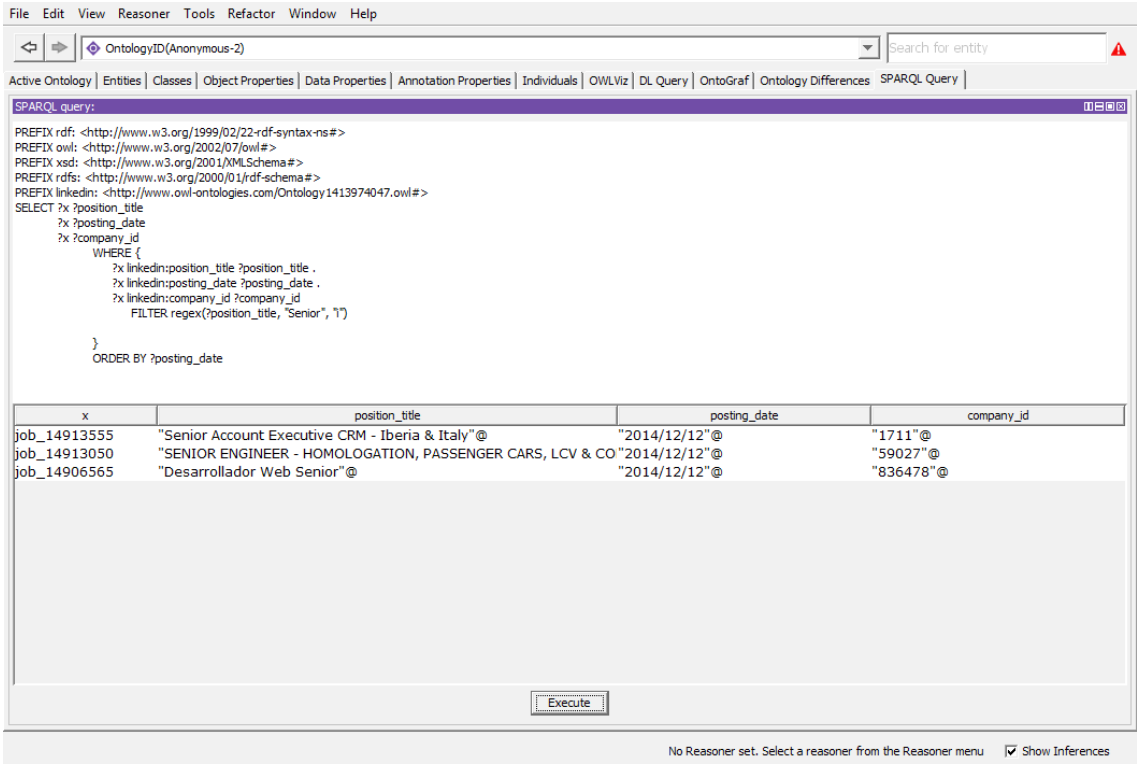
Il·lustració 28: Protégé. Consulta SPARQL 2.

En la següent consulta es pretén cercar totes les feines que en el seu epígraf del nom de la feina hi surti la paraula *senior.* Per aconseguir-ho introduïm la clàusula `FILTER` amb la funció ***regex*** la qual avalua una expressió regular. El modificador `i` de la instrucció la fa *key insensitive*.

El resultat és un seguit de registres amb l'identificador de la feina, el títol que se li ha donat a la feina i l'identificador de la companyia que la ofereix.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX linkedin: <http://www.owl-ontologies.com/Ontology1413974047.owl#>

SELECT ?x ?position_title
       ?x ?posting_date
       ?x ?company_id
              WHERE {?x linkedin:position_title ?position_title .
                     ?x linkedin:posting_date ?posting_date .
                     ?x linkedin:company_id ?company_id
                            FILTER regex(?position_title, "Senior", "i")
```

```
}
                ORDER BY ?posting_date
```



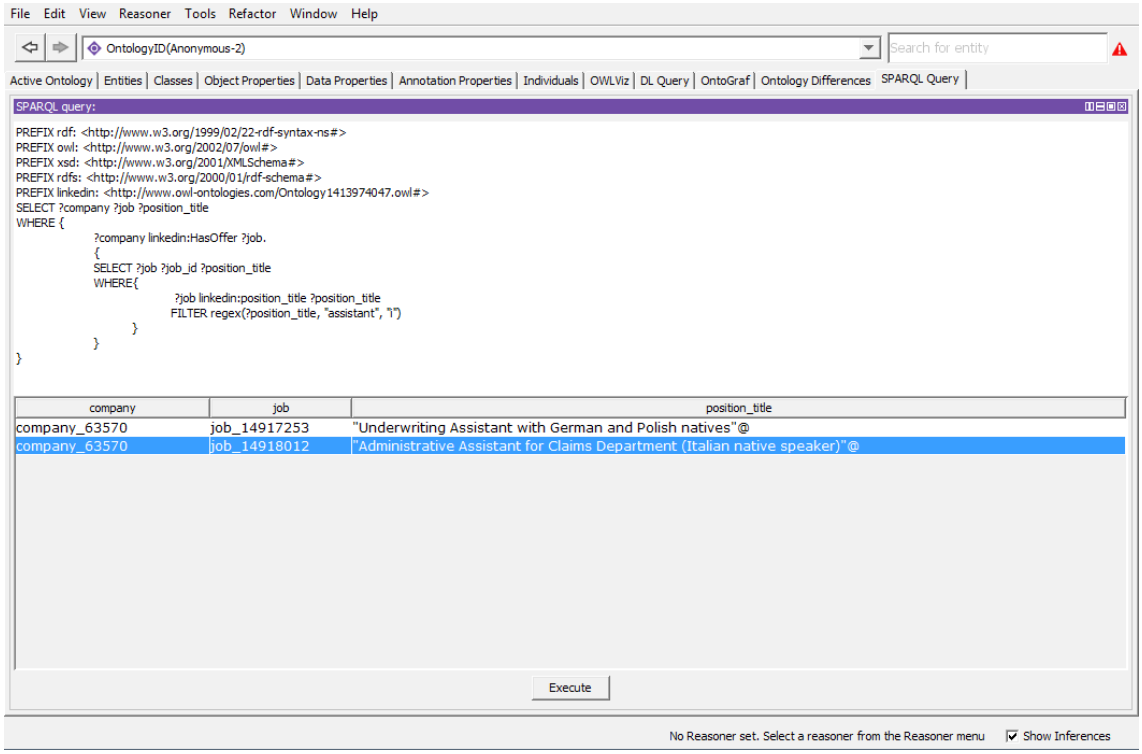
Il·lustració 29: Protégé. Consulta SPARQL 3.

Com a últim exemple introduïm el concepte de consulta nidificada. Farem servir les clàusules `SELECT` nidificats.

En aquest cas busquem feines relacionades amb el concepte *assistant* mitjançant la clàusula ja coneguda `FILTER` i en volem saber les companyies que les ofereixen i en quines ofertes de feina apareixen.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX linkedin: <http://www.owl-ontologies.com/Ontology1413974047.owl#>

SELECT ?company ?job ?position_title
WHERE {?company linkedin:hasOffer ?job.
       {
```

```
        SELECT ?job ?job_id ?position_title
        WHERE{
                ?job linkedin:posting_title ?posting_title
                FILTER regex(?position_title, "Senior", "i")
                }
        }
}
```



Il·lustració 30: Protégé. Consulta SPARQL 4.

5 Conclusions

Encara que el concepte de la web semàntica fa més de 10 anys que s'està estudiant, amb aquest treball s'ha pogut constatar que encara està en una etapa molt prematura pel que fa a la implementació i utilització. Això s'ha pogut detectar per exemple en les versions del programari Protégé. Per crear l' ontologia hem usat la seva versió 3.x, ja que les posteriors desen les ontologies en formats diferents que les fan incompatibles en versions prèvies.

Amb el treball s'ha aprofundit en el coneixement dels conceptes relacionats amb la web semàntica. Considero que s'han assolit uns coneixements bàsics del domini estudiat. S'ha pogut aprofundir en el coneixement de vàries tecnologies, llenguatges i estructures relacionades amb la web semàntica com RDF, XML, OWL, SPARQL.

El treball de recerca que s'hi ha fet ha estat exhaustiu i profund, degut al desconeixement inicial que es tenia de la temàtica. Al mateix temps s'han explorat els diversos llenguatges de programació que millor s'adaptaven o que disposaven de prou llibreries per poder dur a terme l'objectiu del treball. S'han avaluat els llenguatges PHP, Javascript, Java i Python. Escollint finalment Python per la seva simplicitat i claredat.

Un altre dels punts que s'han hagut d'estudiar a fons han estat les API de la xarxa social LinkedIn, assumint-ne les funcionalitats i la lògica implícita. Durant el desenvolupament del treball la mateixa xarxa LinkedIn ha canviat els seus recursos, la xarxa LinkedIn Developers ha canviat desplaçant informació a la xarxa StackOverflow, especialitzada en recursos de programació.

La metodologia seguida durant tot el treball s'ha assolit, fins hi tot escurçant-ne els terminis del seu desenvolupament.

La realització del treball ha permès fixar la metodologia en el desenvolupament del software, sobretot a l'hora a de plantejar i donar solucions a un problema en un àmbit no del tot familiar dintre dels propis coneixements.

El projecte pot marcar unes línies de futur orientades en una possible ampliació de les capacitats del script en el poblament de l'ontologia, en el sentit de incloure-hi noves relacions entre objectes del domini estudiat o fins i tot

ampliar l'ontologia amb conceptes d'altres àmbits. En aquest sentit es podria ampliar les relacions de les persones (classe **People**) amb altres xarxes socials com poden ser el Twitter o Facebook, usant les respectives API. Això ens permetria crear una ontologia que abastés diversos dominis dintre les xarxes socials.

També es podria ampliar la informació que s'infereix a l'ontologia poblada fent ús de la tecnologia SWRL (http://www.w3.org/Submission/SWRL/).

Una altre millora es podria encaminar en incloure una interfície gràfica per facilitar-ne l'ús o simplement que es permetés iniciar l'script amb paràmetres d'entrada com el nombre de **Jobs** a consultar o fins i tot cercar certs tipus de feines indicant-li el sector industrial en que s'ofereixen.

Amb tot això el treball té moltes possibilitats d'expansió i creixement dintre del domini estudiat com en d'altres.

6 Glossari

Bottom-up: Metodologia del disseny de classes que va de les classes derivades a les superclasses. Generalització.

Interfície: Contracte que pren una classe java, que defineix les accions que ha de implementar sense tenir en compte com s'ha fet aquesta implementació.

Ontologia: Rigorós esquema conceptual explícit que afecta a un determinat domini del coneixement.

OWL: És un llenguatge dissenyat per representar ontologies.

Protégé: Software utilitzat per disseny visual d'ontologies.

Python: Llenguatge de programació d'alt nivell interpretat.

RDF: Llenguatge que aporta un model de dades per objectes i les relacions entre ells.

RDFS: Representació esquemàtica per la representació explícita de fitxers RFD.

SPARQL: És un llenguatge utilitzat per consultar diversos tipus de fonts de dades expressades nativament en format RDF.

SQL: Structured Query Language. Llenguatge per consultar a bases de dades relacionals.

TFC: Sigles de Treball final de carrera.

Top-down: Metodologia de disseny de classes que va des de les superclasses a les classes derivades. Especialització.

XML: Llenguatge d'etiquetes per emmagatzemar dades en format humanament llegible.

7 Bibliografia recomanada

7.1 Documents

"La web semántica" [2]
Pablo Castells
Universidad Autónoma de Madrid
pablo.castells@uam.es

"Ontologies i web semàntica" [3]
Jordi Duran Cals, Jordi Conesa i Caralt, Robert Clarisó Viladrosa.
UOC

7.2 Llibres de text

"Nosotros, los constructores de la web semàntica"[4]
Epub
Luís Criado-Ferandez
http://www.amazon.es/Nosotros-los-constructores-Web-Semántica-ebook/dp/B00DC8IAJA/ref=sr_1_1?ie=UTF8&qid=1411480988&sr=8-1&keywords=web+semantica

7.3 Alguns Enllaços d'interès a internet

"Webadicto.net" [1]
http://webadicto.net/post/Cuantas-Paginas-Web-Existen-en-Internet-Reporte-de-Agosto-2013

"Semantic Web"[2]
http://semanticweb.org/

"Nova Spivak - Minding the planet"[3]
http://www.novaspivack.com/
http://www.novaspivack.com/technology/how-the-webos-evolves

"Blog de web semántica y ontología" [4]
http://sones.blogs.uv.es/

"W3C - Semantic Web" [5]
http://www.w3.org/standards/semanticweb/

http://www.w3.org/2001/sw/wiki/OWL
http://www.w3.org/2009/sparql/wiki/Main_Page

"SAPRQL, Lenguaje de consulta RDF" [6]
http://skos.um.es/TR/rdf-sparql-query/

"Ontology Development 101: A Guide to Creating Your First Ontology" [7]

http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html

"A Practical Guide To Building OWL Ontologies Using Prot´eg´e 4 and CO-ODE Tools" [8]
http://130.88.198.11/tutorials/protegeowltutorial/

"The DBpedia Ontology (2014)" [9]

http://wiki.dbpedia.org/Ontology

"OAuth Test Console" [10]
https://developer.LinkedIn.com/oauth-test-console

"LinkedIn Jobs"[11]
https://developer.linkedin.com/documents/jobs

"LinkedIn Companies"[12]
https://developer.linkedin.com/documents/companies

"LinkedIn People"[13]
https://developer.linkedin.com/documents/people

"LinkedIn Groups"[14]
https://developer.linkedin.com/documents/groups

"Download Python"[15]
https://www.python.org/downloads/release/python-278/

llibreria OAUTH de Python [16]
https://pypi.python.org/pypi/oauth/1.0.1
https://developer.linkedin.com/documents/getting-oauth-token-python

llibreria python-linkedin4.1 de Python [17]
https://pypi.python.org/pypi/python-linkedin/4.1

llibreria rdflib de Python [18]
https://pypi.python.org/pypi/rdflib/4.1.2
http://rdflib.readthedocs.org/en/latest/intro_to_creating_rdf.html#

llibreria xml.etree de Python [19]
https://pypi.python.org/pypi/ElementTreeFactory/1.0
https://docs.python.org/2/library/xml.etree.elementtree.html#supported-xpath-syntax
http://luisartola.com/software/2010/easy-xml-in-python/

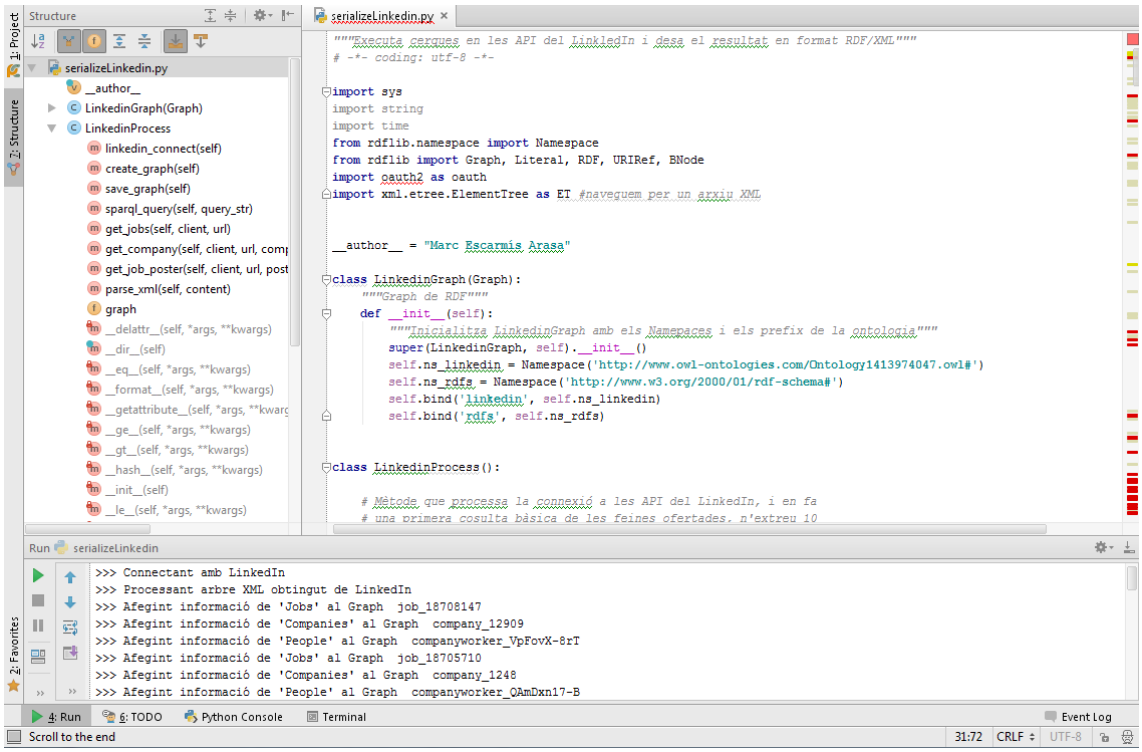
SWRL: A Semantic Web Rule Language Combining OWL and RuleML
http://www.w3.org/Submission/SWRL/ [20]

8 Annexos

8.1 Edició i execució de l'script alternatives, editor PyCharm

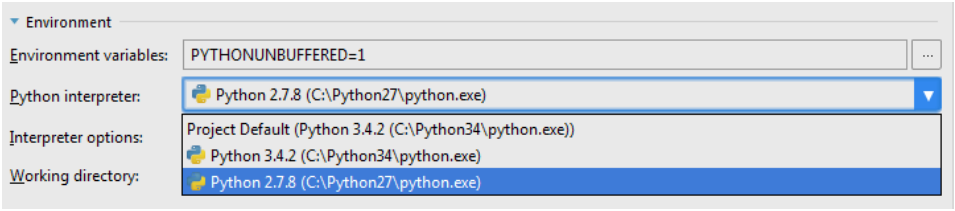
En aquest estudi s'ha fet servir la consola IDE que ens ofereix el programari Python per la plataforma Windows, però hi ha altres editors i mètodes d'execució dels scripts de Python.

Un editor de codi Python, que te una versió *Community*, és el PyCharm, del la companyia JetBrains (https://www.jetbrains.com/pycharm/), disponible per MacOS, Windows i Linux. Aquest editor té ajuda popup a l'hora d'escriure codi, també ens permet utilitzar diferents versions del Python, podent-la escollir a l'hora de l'execució.



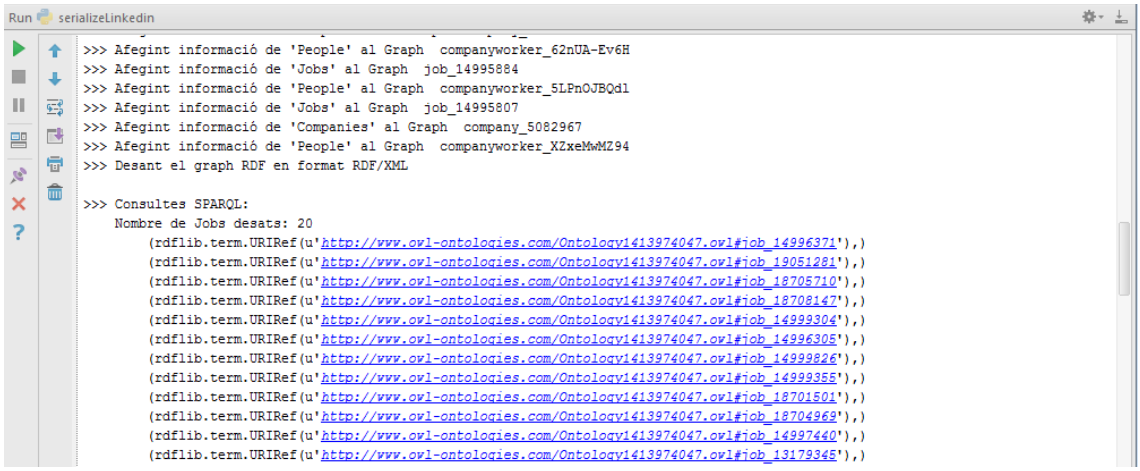
Il·lustració 31: Pantalla principal de l'editor PyCharm

Elecció de la versió Python instal·lades en el sistema:



Il·lustració 32: Selecció de la versió Python en l'editor PyCharm.

Sortida de l'execució de l'script:



Il·lustració 33: Sortida de l'execució en l'editor PyCharm.