eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Ingeniaritzako Gradua

Konputazioa

Gradu Amaierako Proiektua

Grafoen bistaratzea eta edizioa

Egilea

Unai Arrieta

informatika fakultatea
facultad de informática

2017 iraila

Laburpena

Proiektu honen helburu nagusia LiDom Builder sistemak sortzen duen domeinu-moduluaren ontologia grafo-egitura bat erabiliz bistaratzea eta editatzea ahalbidetzen duen web-aplikazio bat garatzea da. Ontologia hori automatikoki erauzten du LiDom Builder-ek testu liburuetatik abiatuta. Grafoari esker, domeinuari lotutako ontologia, hau da, edukiak eta edukien arteko erlazioak, bistaratzea eta editatzea posible izango da. Ontologia editatzeaz gain, eduki edo edukien arteko erlazio berriak gehitzeko eta ezabatzeko aukera egongo da. Baita edukiei lotutako hainbat datu eguneratzeko ere.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

Kodearen aurkibidea

1. KAPITULUA

Analisia eta planifikazioa

1.1 Testuingurua eta problematika

Informazio eta Komunikazio Teknologiek izandako iraultzak hezkuntzan ere eragin du, ikasketa eta ikaskuntza hobetzeko bideak eta tresnak eskainiz. Tresna hauek ezinbesteko bihurtu dira, ez soilik hezkuntza-erakundeetan, adibidez *Moodle*, baita ikasleak bere aldetik ikasi nahi duenean ere. Teknologian Oinarritutako Hezkuntzarako Tresnen erabilerak ikasketaren emaitzetan ez ezik, ikasleen motibazioan ere eragin dezake; ikasteko era berriak sortu dira, adibidez, elkarlanean oinarritutako ikaste-prozesuetan oinarritutakoak.

Teknologian Oinarritutako Hezkuntzarako Tresnek ikasi edo irakatsi beharreko domeinuaren adierazpena behar dute nahi eta nahi ez. Ikasleak landu behar dituen gaiak eta baliabideak antolatuta egotea komeni da, gainera. Tresna horietan, guztietan, domeinu-moduluak jasoko du, landu beharreko topikoaz gain, horien arteko erlazio pedagogikoak eta ikasteko erabiliko diren baliabideak. Erlazio pedagogikoek adieraziko dute, besteak beste, ikasketa-saioak nola planifikatu. Adibidez, zer eduki landu edo menperatu behar den, nahi eta nahi ez, beste eduki batera pasa aurretik, edo zer eduki dagoen beste batetik oso gertu. Baliabideek, aldiz, ikasteko erabiliko diren edukiak jasotzen dituzte, besteak beste, definizioak, ariketak, adibideak... Domeinu-modulua egokia ez bada, ikasketa-prozesu eraginkorra burutzea ezinezkoa izango da.

Domeinu-modulua sortzea ez da lan arina, ordea. Sortze-prozesua zaila izanik, are eta zailagoa da domeinu-modulua era automatikoan edo erdiautomatikoan erauztea, dokumentu elektronikoetatik abiatuta. Teknologiaren bilakaerari esker, egiteko horren inguruan

egin dira, dagoeneko, saiakera batzuk (ikus 1.2 atala). Baina, automatikoki erauzitako domeinu-modulu horiek bistaratu eta gainbegiratu egin behar dira. Sortze-prozesua automatikoa denez, erabiltzaileen gainbegiratzea ezinbestekoa da domeinu-moduluak izan dituen akatsak ikusi eta zuzendu ahal izateko, edo besterik gabe bada ere, domeinu-modulua erabiltzaileen helburuetara egokitzeko.

Lan honetan aurkezten den proiektuaren helburu nagusia, automatikoki sortu den domeinu-moduluaren egitura bistaratzea eta editatzea ahalbidetzen duen web-aplikazio bat garatzea izango da.

1.2 Aurrekariak

Atal honetan aurkeztuko dira proiekturako aurrekari gisa erabiliko diren tresnak: DOM-Sortze [Larrañaga et al., 2013] eta LiDom Builder [Conde et al., 2017], biak EHUko GaLan[1] ikerkuntza taldean garatuak.

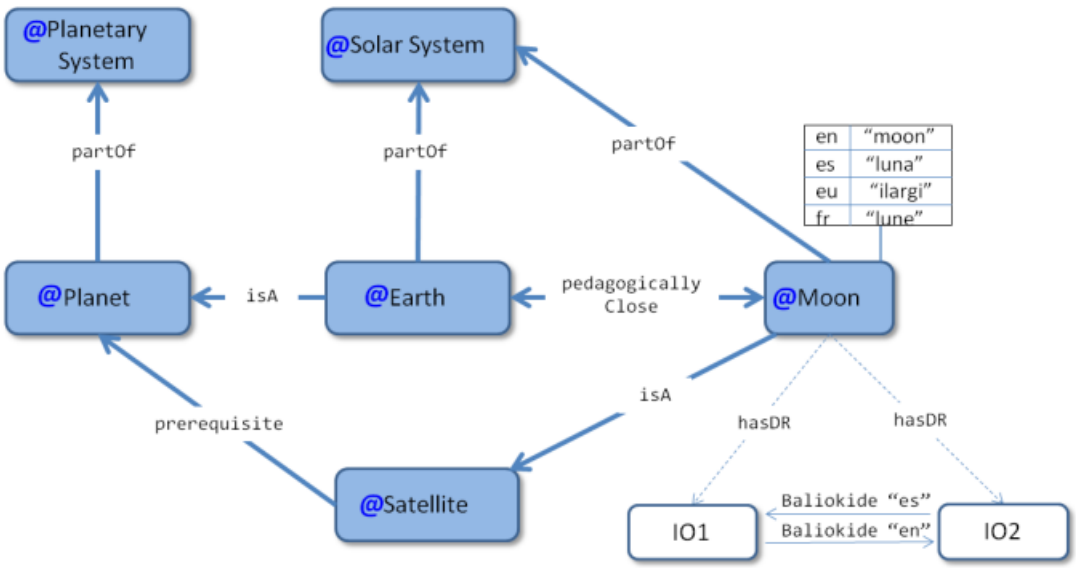
Euskaraz idatzitako testuliburu elektronikoetatik domeinu-modulua era erdiautomatikoan erauzteko tresna da DOM Sortze eta domeinuarekiko independentea da. Domeinu-modulu elebakarretik domeinu-modulu eleaniztunerako bidean, LiDom Builder tresna DOM Sortze ingurunearen bilakaera dela esan genezake. Domeinu-modulua adierazteko orduan, bi sistemek jasotzen dute, batetik, Ikaste Domeinuaren Ontologia (IDO), non topiko edo gaiak eta horien arteko erlazioa pedagogikoak jasotzen baitira eta, bestetik, Ikaste Objektuak, hau da, hezkuntzarako erabiliko diren edukiak, metadatuekin etiketatuak (definizioak, adibideak, ariketak...).

LiDom Builderek aukera ematen du onartutako hizkuntza guztietan domeinuaren gaiak adierazteko. Gai bakoitza lotuta dago hizkuntza bakoitzean dagokion etiketa baliokidearekin.

Domeinu-modulu eleaniztun zati baten adibidea erakusten da 1.1 irudian. Bertan adierazten dira domeinuko gai nagusiak —*Planetary System*, *Solar System*, *Planet*, *Earth*, *Moon*, *Satellite*— eta horien itzulpenak —adibidean, Moonen itzulpenak: euskaraz, *Ilargi*; gaztelaniaz, *Luna*, eta frantsesez, *Lune*). Horrez gain, gai nagusien arteko erlazio pedagogikoak ere zehazten dira. Lau erlazio pedagogikorekin lan egiten du: *isA*rekin eta *partOf*ekin, egitura adierazteko; *prerequisite*ekin, ordena adierazteko, eta *pedagogicallyClose*rekin, gertutasun pedagogikoa adierazteko. *Earth isA Planet* erlazioak adierazten du *Earth* gaia

[1] `http://galan.ehu.es/Galan`

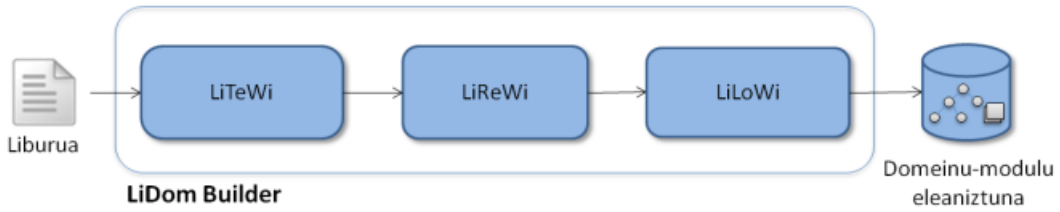
*Planet*en mota zehatz bat dela. *Planet partOf Planetary System* erlazioak, ordea, *Planet* gaia *Planetary System*aren zati bat dela erakusten du, hots, *Planetary System* landutzat emateko, *Planet* ikasi behar diren gaietako bat dela. Halaber, *Satellite prerequisite Planet* erlazioak adierazten du, *Satellite* ikasten hasi aurretik, ikasleak *Planet* dagoeneko landuta izan behar duela, eta *Earth pedagogicallyClose Moon* erlazioak, berriz, bi gaiak oso gertu daudela adierazten du eta litekeena dela, adibidez, biak batera lantzea. Gai eta erlazio pedagogikoez gain, IO eleaniztunak ere jaso behar dira domeinu-modulu eleaniztunean. IO bakoitzak lotuta egon behar du beste hizkuntzetan baliokideak diren IOei. Helburu hori betetzeko, IO bakoitzaren metadatuak aberastu dira, baliokideen arteko loturak deskribatzeko.



1.1 Irudia: Domeinu-modulu eleaniztun zati baten adibidea

Testuliburuetatik domeinu-modulu eleaniztunak eraikitzeko 3 modulu nagusitan oinarritzen da LiDom Builder, ikusi 1.2 irudia.

- LITEWI: Ikaste-domeinutako testuliburu batetik abiatuta, identifikatzen ditu Ikaste Domeinuaren Ontologia (IDO) bati dagozkion hainbat gai edo topiko.
- LIREWI: Erlazio pedagogikoez aberastuko du IDOa, beti ere, testuliburua abiapuntu gisa erabilita.
- LILOWI: IOak lau urratzetan erauziko ditu, abiapuntuko testuliburutik ez ezik Wikipedia edo WordNet moduko ezagutza-baseetatik ere.



1.2 Irudia: Domeinu-modulu eleaniztunaren sorkuntza-prozesua

1.3 Helburuak

Proiektu honen helburu nagusia LiDom Builderrek sortzen duen domeinu-moduluaren ontologia (LDO) grafo-egitura bat erabiliz bistaratzea eta editatzea ahalbidetzen duen web-aplikazio bat garatzea da. Grafoak domeinuari lotutako edukiak eta edukien arteko erlazioak adieraziko ditu. Bai grafoko adabegiek bai erlazioek informazio gehigarria daukate erantsita, besteak beste, ziurtasun maila. Aplikazioak grafoko elementuak gehitzea, ezabatzea eta eguneratzea ahalbidetuko du. Gainera, bistaratze-mekanismo desberdinak erabiltzea ahalbidetuko du. Grafoak jasotzen dituen adabegi eta erlazio kopuru handia dela eta, eta bistaratzea ahalik eta ulergarriena izateko helburua lor dadin, beharrezkoa izango da grafoen bistaratze-mekanismo egokienen aztertzea. Adibidez, ertzak ahalik eta gutxien gurutzatzeko, edo eta 3D bistaratzea txertatzea merezi ote duen aztertzeko. Bestalde, grafoan identifikatutako erlazio eta erlazio horien semantika kontuan hartuta, grafoa eraldatzeko aukera aztertzea ere interesgarritzat jotzen da. Grafoko elementuak gehitzea, ezabatzea eta eguneratzea ahalbidetuko du. Baita ere, grafoko elementuen hainbat ezaugarri eguneratzea ere.

1.4 Irismena

1.4.1 Lanaren deskonposaketa-egitura (LDE)

Jarraian proiektuaren lanaren deskonposaketa egitura (LDE) aurkezten da, ikus 1.3 irudia.

1.4.2 Atazen definizioa

Atal honetan proiektuaren ataza edo lan-pakete ezberdinak deskribatuko dira:

- Plangintza
  Ataza honetan proiektua garatzeko erabili den planifikazioa ageri da. Bertan proiektuaren helburuak eta egingo diren atazak finkatzen dira. Proiektuaren mugarriak, arrisku-plana eta jarraipena ere bertan garatuko dira.

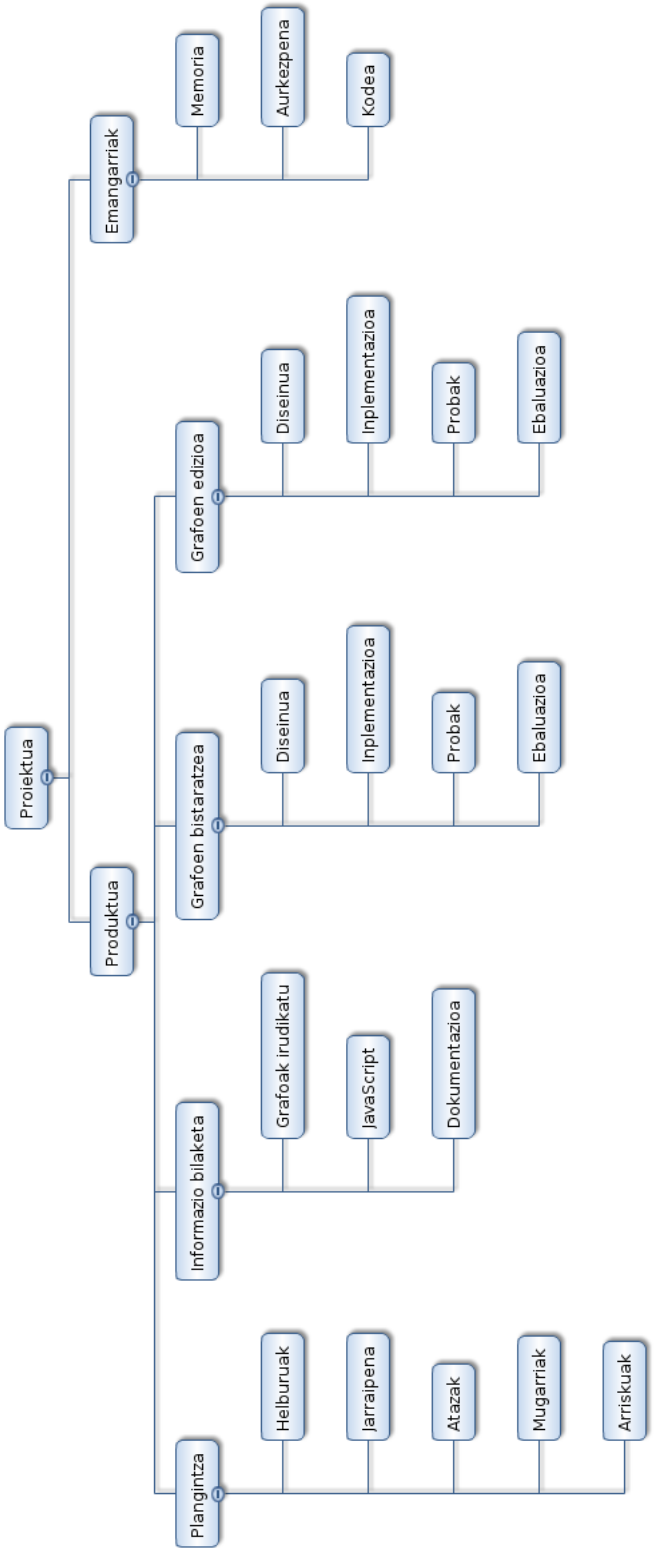
- Informazio bilaketa
  Ataza honetan proiektua garatzeko beharrezko informazioa bilatuko da. Hiru gai nagusiri buruzko informazioa bilatu beharko da:

  – Grafoak irudikatu.
    Grafoak irudikatzeko inplementaturik dauden liburutegiak aurkitu eta aztertu beharko ditugu. Behin bat aukeratuta, hura erabiltzen ikasi beharko da.

  – JavaScript.
    Proiektu honetan JavaScript liburutegiak erabiliko direnez, eta konputazio espezialitatean JavaScript ez denez erabiltzen, lengoaia menperatu beharko da.

  – Dokumentazioa.
    Proiektu handia denez, dokumentazioa garatzeko eta dokumentazio txukun bat egiteko informazioa bilatu behar da. Baita ere hura Latex-en garatzeko teknikak ikasi.

- Grafoen bistaratzea
  Ataza honetan grafoak bistaratzeko liburutegi egokiak aurkitu beharko dira, hauen artean konparazioak egin eta gure beharretarako hobeto datorkiguna aukeratu. Ondoren honen diseinua eta inplementazioa egin beharko dira.



1.3 Irudia: LDE diagrama

- Grafoen edizioa
  Ataza honek bistaraturiko grafoaren edizioa ahalbidetzea du helburu. Bezeroari interesatzen zaizkion grafo-eraldaketak diseinatu eta inplementatuko dira.

1.4.3 Kronograma

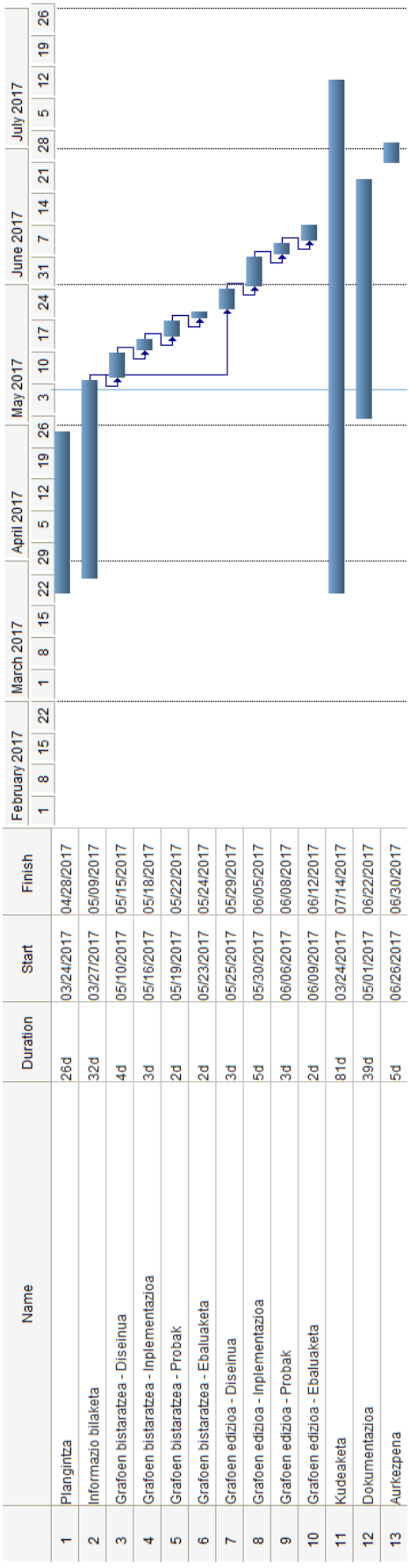
Atal honetan, egingo diren atazen adierazpen grafikoa *Gantt* diagrama baten bitartez aurkezten da, ikus 1.4 irudia. Bertan ataza bakoitzaren hasiera- eta bukaera-datak finkatzen dira.

1.4.4 Mugarriak

Proiektua aurrera ongi eramateko baldintzatzen duten zenbait mugarri ezarri dira, bai kanpo-mugarriak zein barne-mugarriak, ikus 1.5 irudia. Horretaz gain, proiektuaren garapenak bi fase edo etapa izango ditu.

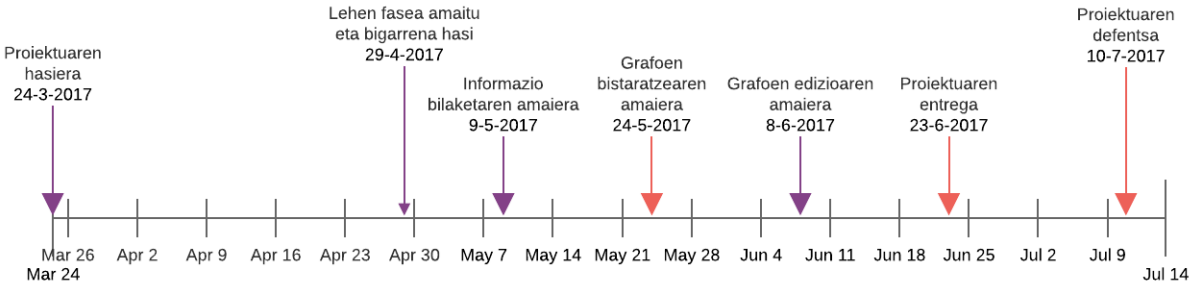
- Lehenengo fasea: Informazio bilaketa (2017/03/24 - 2017/04/30). Fase honetan, ikasleak, plangintza- eta informazio-bilaketan lan egingo du.
- Bigarren fasea: Inplementazioa (2017/05/01 - 2017/07/14). Fase honi informazio bilaketa amaitzean emango zaio hasiera. Fase honetan aukeratu den liburutegiaren bistaratzea eta edizioa tratatuko dira.
- Barne-mugarriak
  Mugarri hauek erreferentzia gisa erabiliko dira jakiteko ea bide onetik goazen ala ez. Irudian more kolorearekin adierazita daude.

  1. Informazio bilaketaren amaiera (2017/05/09).
     Egun honetarako proiektua osatu ahal izateko beharrezkoa izango den informazioa eskura egon behar da.
  2. Grafoen bistaratzearen amaiera (2017/05/24).
     Egun honetarako eskatu zaigun grafoa bistaratzeko inplementazioa eginda egon behar du, baita ere zuzen dabilela bermatu.
  3. Grafoen edizioaren amaiera (2017/06/08).
     Egun honetarako bistaraturiko grafoa editatzeko aukera inplementaturik eta funtzionamenduan egon behar du.



	Name	Duration	Start	Finish
1	Plangintza	26d	03/24/2017	04/28/2017
2	Informazio bilaketa	32d	03/27/2017	05/09/2017
3	Grafoen bistaratzea - Diseinua	4d	05/10/2017	05/15/2017
4	Grafoen bistaratzea - Inplementazioa	3d	05/16/2017	05/18/2017
5	Grafoen bistaratzea - Probak	2d	05/19/2017	05/22/2017
6	Grafoen bistaratzea - Ebaluaketa	2d	05/23/2017	05/24/2017
7	Grafoen edizioa - Diseinua	3d	05/25/2017	05/29/2017
8	Grafoen edizioa - Inplementazioa	5d	05/30/2017	06/05/2017
9	Grafoen edizioa - Probak	3d	06/06/2017	06/08/2017
10	Grafoen edizioa - Ebaluaketa	2d	06/09/2017	06/12/2017
11	Kudeaketa	81d	03/24/2017	07/14/2017
12	Dokumentazioa	39d	05/01/2017	06/22/2017
13	Aurkezpena	5d	06/26/2017	06/30/2017

1.4 Irudia: *Gantt* diagrama



1.5 Irudia: Mugarriak

- Kanpo-mugarriak
  Mugarri hauek Gradu Amaierako Proiektuen egutegi ofizialak zehaztuak dira eta derrigorrez bete behar dira zehaztutako eguna baino lehen. Irudian kolore laranja kolorearekin adierazita daude.

  1. Defentsa-eskaera (2017/06/16).
     Egun honetarako, ikasleak, proiektua GAURen matrikulatu behar du eta defentsa eskaera egin zuzendariak defentsaren baimen txostena idatzi dezan.
  2. Proiektuaren entrega (2017/06/23).
     Egun honetarako proiektua ADDI plataforman eskuragarri utzi behar da.
  3. Proiektuaren defentsa (2017/07/10-13).
     Egun hauetan proiektua epai-mahi baten aurrean defendatu beharko da, beraz dagokion aurkezpena eginda egon behar da.

1.4.5 Emangarriak

Proiektuan zehar honako emangarriak sortuko dira:

1. Kodea
   Emangarri honetan proiektuan zehar inplementatu den JavaScript kode guztia dokumentaturik egongo da.

2. Memoria
   Emangarri hau proiektuan zehar egin den lan guztiaren deskribapena duen dokumentu bat izango da. Bertan, egindako plangintza, aurkituriko informazioa, arazoen analisia eta soluzioa, lorturiko emaitzak eta ateratako ondorioak bilduko dira.

3. Aurkezpena
   Emangarri hau proiektuaren defentsan erabiliko den dokumentuari dagokio. Bertan, proiektuan egindako lanaren laburpen bat bilduko da.

1.5 Arriskuak

1.5.1 Arriskuen analisia

Proiektu batean zehar hainbat arrisku sor daitezke. Arrisku hauek proiektuaren arrakasta baldintzatu dezakete. Horregatik, garrantzitsua da arrisku hauek garaiz identifikatzea eta horiek kudeatzeko kontingentzia planak eratzea. Jarraian proiektuan zehar sortu daitezkeen arriskuak identifikatu eta bakoitzarentzat erabiliko den kontingentzia-plana zehazten da.

Identifikaturiko arriskuak

1. Aukeraturiko liburutegia gure proiektura moldatu ezin izatea.
2. Proiektuari buruzko informazioaren galera.
3. Proiektuaren planifikazioa betetzeko gai ez izatea proiektuaren tamaina, irakasgaien lan-karga edo sortu diren arazoak direla medio.

Arriskuen kontingentzia planak

1. Liburutegiak aztertzerakoan hainbat liburutegi aztertuko dira, beraz aukeratutakoak huts eginez gero beste bat egongo da eskura. Honekin, liburutegi bilaketan denbora gehiago ez galtzea lortuko da.
2. Proiektua ikaslearen konputagailu pertsonalean garatuko da. Konputagailua matxuratuz gero informazio guztia ez galtzeko periodikoki segurtasun-kopiak egingo dira eta *google drive* plataformara igo.

3. Proiektua garatzen hastean irakasgai gehienak amaitzea lortu da, beraz, irakasgaiek ez lukete arazo gehiegi eman beharko. Hala ere, lehen aipatutako arazoek asko atzeratu dezakete proiektuaren garapena eta proiektua ekainean aurkeztea galarazi. Hori gertatuz gero, proiektuaren defentsa irailera mugitu beharko litzake.

2. KAPITULUA

Grafoak bistaratzeko liburutegiak

Lehen azaldu den bezala, proiektu honen helburua betetzeko, JavaScript lengoaiako liburutegiez baliatuko gara. 11 liburutegi desberdin aztertu dira eta bakoitzeko, deskribapen labur bat eratu da eta 10x10-ko sare baten bistaratzearen adibide batekin. Liburutegi guztiek kode askea dute eta Github-en eskuragarri daude.

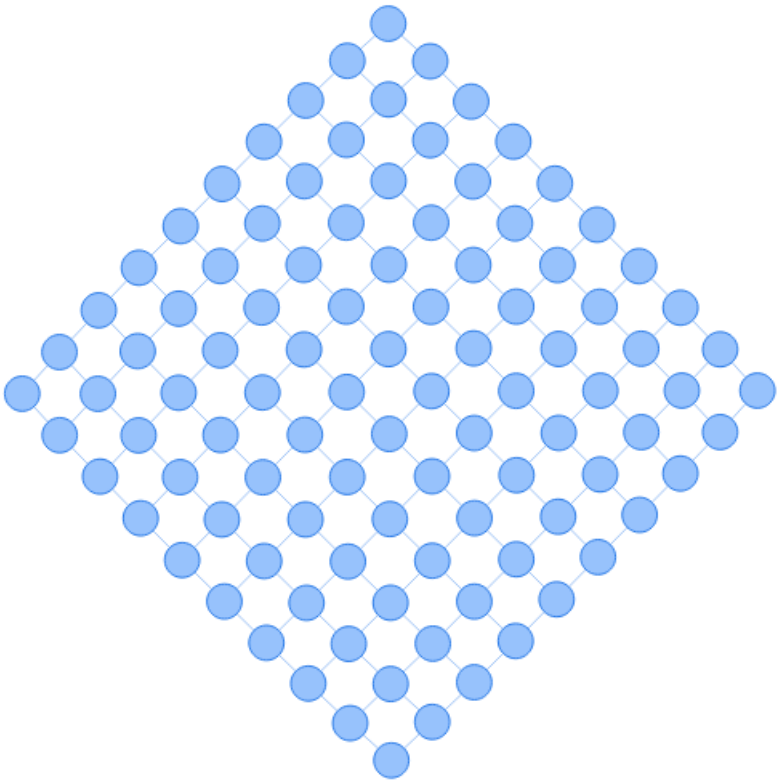
2.1 Viz

Viz[1] liburutegia, nabigatzailean oinarrituriko liburutegi dinamiko bat da eta npm plataforman eskuragarri dago (npm plataformari buruzko informazioa ikusteko jo 3.2 atalera). Datu kopuru handiekin erabiltzeko erraza eta datuen maneiua eta elkarrekintza ahalbidetzeko diseinatuta dago. Sareak adabegi eta ertzekin bistaratzen ditu. 2D grafoak sortzeko aukerak ditu eta baita ere datuak 3D grafo baten adierazteko baliabideak.
2.1 irudian liburutegi honek sortu dezakeen grafoaren adibide bat ageri da.

2.2 Ngraph

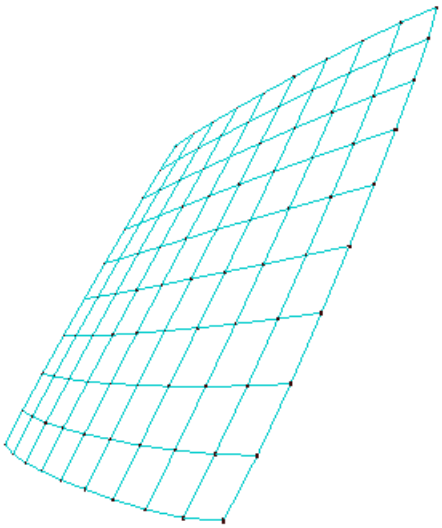
Ngraph[2] liburutegia, grafoei bakarrik zuzendurik dagoen liburutegi bat da eta osagai guztiak npm plataforman eskuragarri ditu. Nabigatzailean edo zerbitzari aldean erabil daiteke.

[1] `http://visjs.org`
[2] `https://github.com/anvaka/ngraph`



2.1 Irudia: Viz liburutegiaren adibidea

Grafoak 2D edo 3D egituran bistaratzeko aukera du. Grafoen errenderizazioa liburutegi desberdinekin egitea dago.
2.2 irudian liburutegi honek sortu dezakeen grafoaren adibide bat ageri da.



2.2 Irudia: Ngraph liburutegiaren adibidea

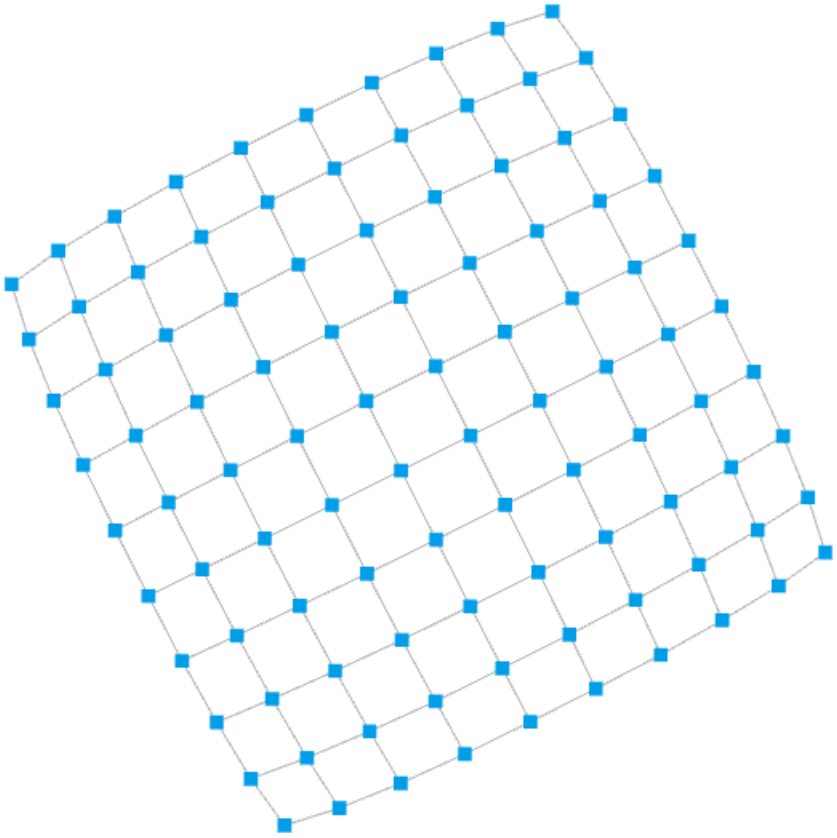
2.3 VivaGraphJS

VivaGraphJS[3] liburutegia, sareak eratzeko eta adabegietan oinarritutako grafoak sortzen dituen liburutegia da. Grafo txikienetatik, ehunaka edo milaka adabegi dituzten grafoak eratzeko gai da. Hainbat errenderizazio motorrekin lan egin dezake eta diseinu algoritmo desberdinak erabil ditzake.
2.3 irudian liburutegi honek sortu dezakeen grafoaren adibide bat ageri da.

2.4 Dagre

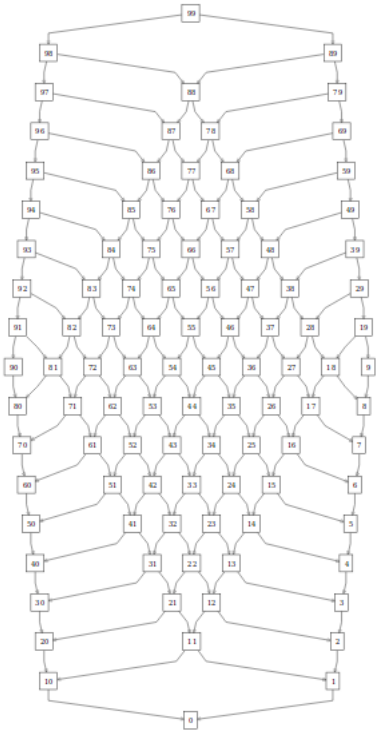
Dagre[4] liburutegia, bezeroaren aldean grafo zuzenduak sortzeko aukera eskaintzen duen liburutegi bat da, liburutegia npm plataforman eskuragarri egonda. Grafoa errenderizatze-

[3] (https://github.com/anvaka/VivaGraphJS
[4] https://github.com/cpettitt/dagre



2.3 Irudia: VivaGraph liburutegiaren adibidea

ko aukera ugari ditu, horietako bat D3 liburutegia izanik. Liburutegi hau ez dago modu aktiboan garatzen edota mantentzen gaur egun. 2.4 irudian liburutegi honek sortu dezakeen grafoaren adibide bat ageri da.

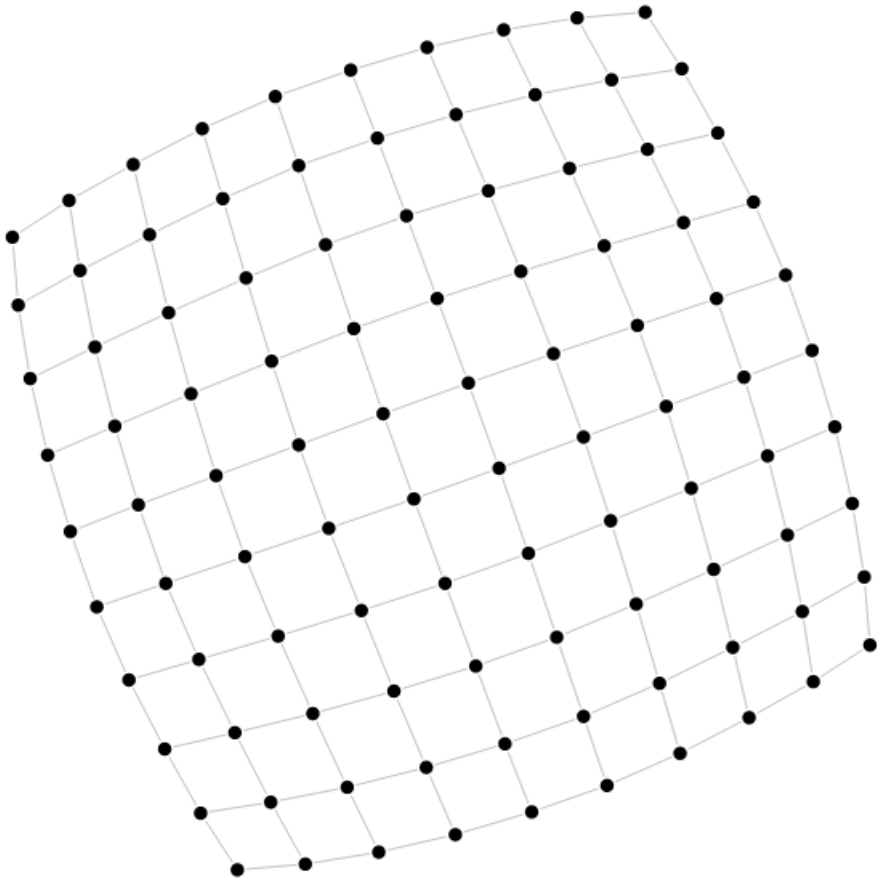


2.4 Irudia: Dagre liburutegiaren adibidea

2.5 D3

D3[5] liburutegiak, datuetan oinarrituriko fitxategiak maneiatzeko erabiltzen da eta npm plataforman eskuragarri dago. Oso arina, datu multzo handiak erabili ditzake eta portaera dinamikoa eskaintzen du animazio eta elkarrekintzetarako.
2.5 irudian liburutegi honek sortu dezakeen grafoaren adibide bat ageri da.

[5] `https://d3js.org/`



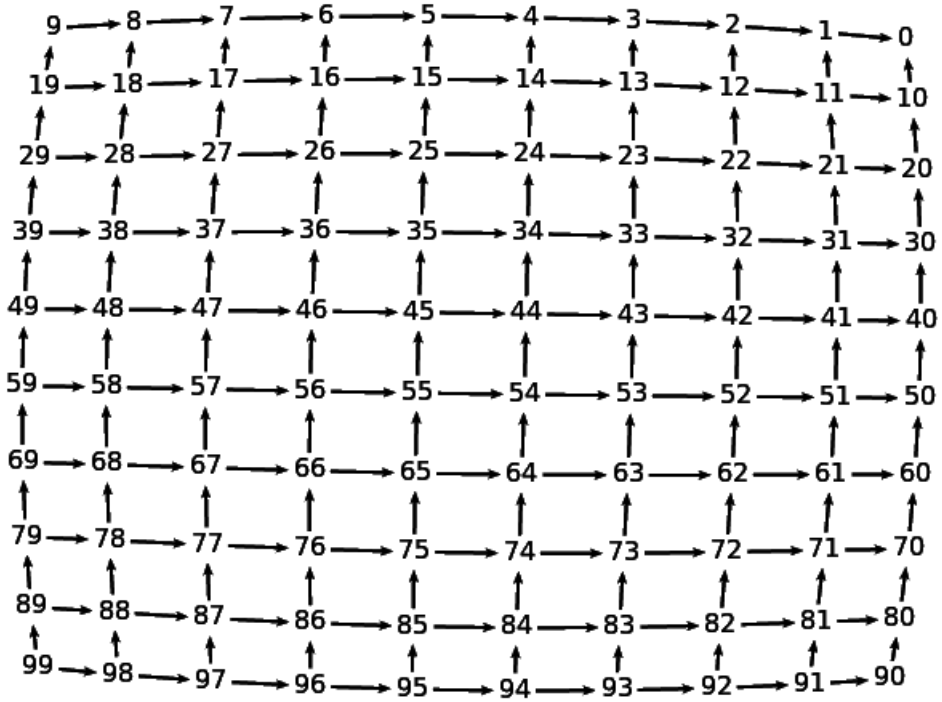
2.5 Irudia: *D3* liburutegiaren adibidea

2.6 Cytoscape

Cytoscape[6] liburutegia, molekula sareen elkarrekintza eta baliabide biologikoak bistaratzeko eta grafo hauek informazioz betetzeko erabiltzen den liburutegi bat da. Hasiera batean ikerkuntza biologikoari zuzenduta zegoen arren, gaur egun grafoen analisi eta bistaratzerako erabiltzen da. npm plataforman eskuragarri.
Ezin izan da errenderizazioaren adibiderik exekutatu.

2.7 Springy

Springy[7] liburutegia, grafo zuzenduak diseinatzeko algoritmoak eskaintzen ditu. Grafoen diseinua tratatzen du baina grafoekin elkarrekintzarik ez dauka inplementaturik.
2.6 irudian liburutegi honek sortu dezakeen grafoaren adibide bat ageri da.

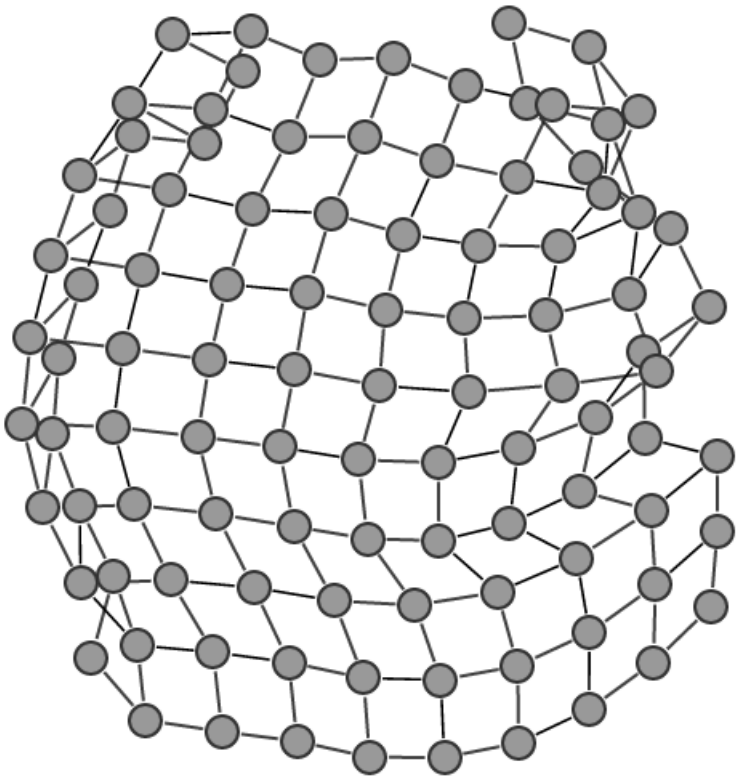


2.6 Irudia: Springy liburutegiaren adibidea

[6] `http://js.cytoscape.org/`
[7] `http://getspringy.com/`

2.8 JSNetworkX

JSNetworkX[8] liburutegiak, grafoak eraiki, prozesatu eta analizatzeko aukera ematen du eta npm plataforman eskuragarri dago. Python lengoaian idatzita dagoen *NetworkX* liburutegiaren bertsio bat da baina JavaScript lengoaiara moldatua. D3 liburutegiarekin batera lan egin dezake nabigatzailean grafoak bistaratzeko.
2.7 irudian liburutegi honek sortu dezakeen grafoaren adibide bat ageri da.



2.7 Irudia: JSNetwork liburutegiaren adibidea

2.9 Alchemy

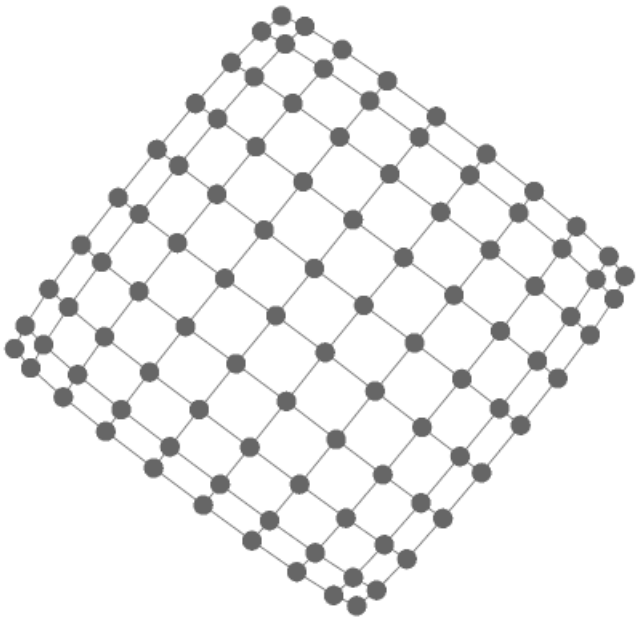
Alchemy[9] liburutegiak, grafoak bistaratzen ditu, D3 liburutegiaz baliatzen da eta npm plataforman eskuragarri dago. Liburutegiaren konfigurazioa aldatzeko, balio lehenetsiak

[8] `http://jsnetworkx.org/`
[9] `http://graphalchemist.github.io/Alchemy/#/`

aldatzearekin nahikoa da, horrela *JavaScript*ekin egin beharreko inplementazioa minimizatuz. Liburutegi hau D3 liburutegiarekin eratua dagoenez honen osagaiak gehitzea ahalbidetzen du.
Ezin izan da errenderizazioaren adibiderik exekutatu.

2.10 Sigma

Sigma[10] liburutegiak, grafoak bistaratzeko erabiltzen da eta npm plataforman eskuragarri dago. Web aplikazioak sortzeko eta egokitzeko diseinatua. Liburutegi honek errenderizazio motorra eskaintzen du bakarrik, horrela grafoen algoritmoak nahi dituzunak sar ditzakezu. 2.8 irudian liburutegi honek sortu dezakeen grafoaren adibide bat ageri da.

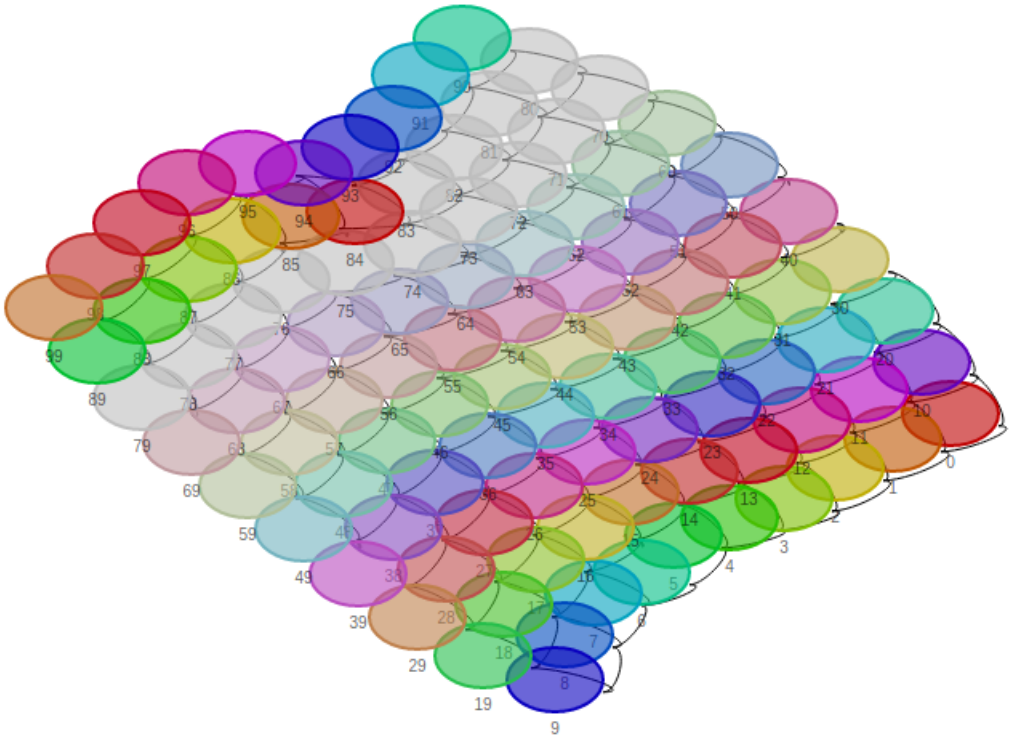


2.8 Irudia: Sigma liburutegiaren adibidea

[10] `http://sigmajs.org/`

2.11 Dracula

Dracula[11] liburutegia, sareak bistaratzeko eta diseinu interaktiboa ahalbidetzen duen liburutegia da. Grafoen teoriako hainbat algoritmo eskuragarri ditu.
2.9 irudian liburutegi honek sortu dezakeen grafoaren adibide bat ageri da.



2.9 Irudia: *Dracula* liburutegiaren adibidea

2.12 Konparazioa

Honako ezaugarriak aztertu dira liburutegietatik:

- npm.
  Liburutegia npm plataforman eskuragarri dagoen edo ez.

[11] `https://www.graphdracula.net/`

- Dokumentazio garatua.
  Liburutegiak dokumentazio nahikoa duen grafoekin lan egin ahal izateko, hala nola, adabegiak eta ertzak gehitzeko informazioa, instalatzeko pausoak eta errenderizazio nola martxan jarri.

- Adibideak eskuragarri.
  Grafoei egin ahal zaizkien aldaketak, egitura mota desberdinak eta errenderizazio motor desberdinak erabiltzeko adibideak dauden.

- Errenderizazio adibidea.
  Liburutegiaren errenderizazio emaitza sinple bat exekutatu ahal izatea honen itxura ikusteko. Lehen adierazi den moduan 10x10-ko sare bat errenderizatu da adibide gisan.

- 3D diseinua.
  Grafoak 3D egituran bistaratzeko gaitasuna.

- 3D adibidea.
  Liburutegiaren 3D bistaratzearen adibideak eskura egotea.

Liburutegia	npm	Dokumentazio garatua	Adibideak eskuragarri	Errenderizazio adibidea	3D diseinua	3D adibidea
Viz	X	X	X	X	X	
Ngraph	X	X	X	X	X	X
VivaGraphJS	X	X	X	X		
Dagre	X	X	X	X	X	
Cytoscape	X	X	X			
Springy	X	X	X	X		
JSNetworkX	X	X	X	X		
Alchemy	X	X	X		X	
Sigma	X	X	X	X		
D3	X	X	X	X	X	
Dracula		X	X	X		

2.1 Taula: Liburutegien konparazioa

Lehenik eta behin, liburutegiek npm plataforman duten eskuragarritasuna aztertu da. Baldintza hau ez da derrigorrezko proiektuan aurrera egiteko baina liburutegien instalazioan asko laguntzen du. *Dracula* liburutegiak ez du baldintza hori betetzen.
Dokumentazioaren aldetik, liburutegi guztiek daukate dokumentazio landua atal guztiak

betetzeko, baita ere atal guztiek daukate adibideren bat hobeto azaldu ahal izateko.
Liburutegiek daukaten grafoen bistaratzea aztertzeko, liburutegi bakoitzarekin 10x10 adabegiko sare bat eratu da. Sare honen bistaratzea liburutegi bakoitzaren deskribapenean dago, *Alchemy* eta *Cytoscape* liburutegiekin ez ezik beste guztiekin sarea errenderizatzea lortu da. Honek ez du esan nahi liburutegiek akatsen bat dutenik, baizik eta ez dutela adibideak modu arin batean probatzeko aukerarik.
Viz, *Ngraph* eta *D3* liburutegiek daukate grafoak 3D egitura batean bistaratzeko aukera. *Alchemy* eta *Dagre* liburutegiek aukera hau daukate baina *D3* liburutegiaz baliatzen direlako, beraz, grafoak 3D egituran bistaratzeko *D3* liburutegia azken bi hauek baino hobea izango da. Hala ere, 3D bistaratzerako *Ngraph* liburutegian bakarrik aurkitu da honen adibidea.

Aztertu diren liburutegietatik, bi izan dira ikaslearentzat interes gehien izan dutenak.

- D3 liburutegia.
  Liburutegi oso handia da, ez dago bakarrik grafoekin lan egiteko egina. Kronogramak, grafikoak, grafoak, kartogramak, zuhaitzak eta mugikorreko aplikazioekin lan egiteko aukera eskaintzen du. Orain dela 7 urte sortua izan arren gaur egun garatzen jarraitzen du. Github-en oso ezaguna, aztertu direnen artean diferentziaz izar gehien dituena da 65.000 izarrekin.

- Ngraph liburutegia.
  Ez da D3 liburutegia bezain erraldoia baina daukan guztia grafoei zuzendua dago, hau da, liburutegi hau grafoentzat esklusiboki eratua izan da. Errenderizatzeko hainbat aukera ditu, beste liburutegi batzuk erabiliz, adibidez PIXI.js[12], edota 3D errenderizazioa. Grafoen egitura sortzeko, 3.7 atalean azalduko den eredu fisiko batean oinarritzen da, horrela adabegi eta ertzak argi bistaratzeko.

Bi liburutegiak ondo dokumentatuta daude, ikaslearentzat intereserako diren grafoen errenderizazioa eskaintzen dute eta adibideak dituzte liburutegia erabiltzen ikasteko. Hala ere, D3 liburutegia ez dagoenez grafoentzako esklusiboki egina, grafoen ediziorako ez dago gauza askorik inplementatuta. Ngraph liburutegiak grafoentzat jadanik hainbat aukera inplementaturik dituenez liburutegia hau aukeratu da.

[12] http://www.pixijs.com/

3. KAPITULUA

Ngraph liburutegia eta Ngraph erabiltzeko tresnak

Behin ngraph liburutegia aukeratu dugula, zein erreminta desberdinez baliatzen den aztertu behar da, hauek erabiltzen ikasi behar baita. Lehenik eta behin, grafoak bistaratu eta editatzeko zein erreminta erabili beharko diren aztertu da. Honako tresnak ikertu dira:

- Node.js
- npm
- Browserify
- UglifyJs
- Three.js
- dat.GUI

3.1 Node.js

Node.js[1] liburutegi bat izateaz aparte, sarrera/irteera gertaerek zuzenduriko exekuzio-ingurune asinkrono bat da. Kode librea du eta JavaScript lengoaian oinarrituta dago. JavaScript-en ohiko lana nabigatzailean ataza txikiak burutzea izan den arren, gaur egun Java edo C bezalako lengoaiak egin ditzaketen atazak burutzeko gai da. Zerbitzariaren

[1] `https://nodejs.org`

aldean lan egiteko diseinatu zen, hain zuzen sareko programa handiak egiteko zuzendua, baina ez dago bakarrik horretara mugatua.
Gaur egun, zerbitzarien arazo nagusia konexio kopuruan dago, adibidez, Java bezalako lengoaia batean sortzen den konexio bakoitzeko, prozesu berri bat sortzen da normalean, 2MB-eko pisua duena, beraz, 16GB-eko memoria RAM-a duen sistema batek 8.000 bezeroren konexioa jasateko gai da. Konexio kopurua handiagotu nahi bada zerbitzari gehiago gehitu beharko dira. Node-k arazo hau beste modu batean tratatzen du, konexio bakoitzak gertaera-prozesu bat sortzen du Node-ren prozesuen motorrean. Baina zerbitzari-aplikazio bat izan arren ez du esan nahi Apache zerbitzari baten lana egiten duenik, konexio seguruak lortzeko SSL modulu bat instalatzea besterik ez da egin behar.
Node.js-ek helburu berdina du konparatzen baldin bada Python-en Twisted edo Tornado, Perl-en Perl Object Environment, C-ren libevent, Ruby-ren EventMachine eta Java-ren JEE.

3.1.1 Erabilera

Atal honetan Node-n inportatzeko eta esportatzeko erabiltzen diren funtzioak azalduko dira

- *require*.
  *Node*n require() funtzioa erabiltzen da beste fitxategi bateko kodea inportatzeko. 3.1 kodean adierazten den bezala erabiltzen da.

```
var uniq = require('uniq');
var nums = [ 5, 2, 1, 3, 2, 5, 4, 2, 0, 1 ];
console.log(uniq(nums));
```

3.1 Kodea: *require* funtzioaren erabilera

  Baita ere direktorio erlatiboan dauden fitxategiak erabiltzea dago. 3.2 kodean adierazten den bezala erabiltzen da.

```
var foo = require('./foo.js');
console.log(foo(4));
```

3.2 Kodea: *require* funtzioaren erabilera fitxategi erlatiboak lortzeko

Inportatu dugun fitxategiak funtzio bat baino gehiago inportatzen baldin badu, zein funtzio erabili nahi dugun adierazi behar da, ikus 3.3 kodea.

```
var foo = require('./foo.js');
console.log(foo.beep(4));
console.log(foo.boop(4));
```

3.3 Kodea: Inportatu nahi den funtzioa adierazi

- *export*
  Beste fitxategi batetik kode zati bat atzigarri egoteko *module.export* aldagaian gorde beharra dago, ikus 3.4 kodea.

```
module.exports = function (n) {
    return n * 111
};
```

3.4 Kodea: Funtzioak esportatu

Edozein balio esportatu daiteke, ez bakarrik funtzioak. Normalean aldagai edo funtzio bakar bat esportatzearekin ez da nahikoa izaten. Hainbat objektu esportatzeko beraien izena ezarri beharra dago 3.5 kodean adierazten den bezala.

```
module.exports.beep = function (n) { return n * 1000 }
module.exports.boop = 555
```

3.5 Kodea: Hainbat funtzio edo balio esportatu

3.2 npm

Node.js-ren pakete edo moduluen kudeatzailea da[2]. JavaScript lengoaian garaturiko aplikazioen menpekotasunak kudeatzen laguntzeko sortu zen. Pakete bakoitzak *package.json* izeneko fitxategi bat dauka non moduluaren metadatuak dituen, hala nola, honen dependentziak eta lizentzia. 3.6 kodean *package.json* fitxategi baten adibidea ageri da, aplikazioak dituen dependentziak *dependencies* parametroan ageri dira. *Script* parametroaren edukia hurrengo atalean azalduko da.

```
"scripts": { "start": "node_modules/.bin/browserify -s ngraph index.js
    | uglifyjs > bundle.js" },
"dependencies": {
    "networkjs": "^0.1.1",
    "ngraph.generators": "~0.0",
    "ngraph.three": "0.0.11",
    "query-string": "~0.1.1",
    "browserify": "~12.0.0",
    "uglify-js": "^2.4.13"
}
```

3.6 Kodea: *package.json* fitxategi baten adibidea

npm-ek modulu asko batzen ditu, gaur egun 350.000 modulu eskuragarri daude. Gero eta gehiago hedatzen ari da, astero 11.000 modulu berri igotzen dira eta hainbeste modulu izanik, astero 1,3 mila milioi deskarga izatera heltzen da. Plataforma hau oso handia da baina horrek ez du esan nahi dauden pakete guztiak kalitate handikoak direnik. Paketeen kalitatea aztertzeko, pakete jaitsienei begirada bat botatzearekin nahikoa da.

1. Bakarrik % 88,7-ek dute *readme* fitxategia. Pakete guztien oinarrizko dokumentazioa readme izeneko fitxategian biltzen da eta liburutegia ondo ulertu ahal izateko beharrezkoa.

2. Bakarrik % 57,4-ek dute ezarria *license* eremua beraien package.json fitxategian. Lizentziarik gabeko paketeek ez dute egilearen eskubiderik, beraz erabil ezinak dira ingurune komertzialetan.

[2] `https://www.npmjs.com/`

3. Bakarrik % 52,9-a dago loturik bere git biltegira. Hori gabe zaila da kodeari iritzia helaraztea eta hobekuntzak egitea.

4. Bakarrik % 41,7-ak dauka test *script*a. Hau ez da guztiz beharrezkoa baina ondo dator beste erabiltzaileak ezarritako aldaketak probatzeko.

Irizpide hau erabiliz, 32.768 pakete jaitsienetatik, bakarrik 13.675 kontsideratu daitezke kalitate onekoak direla.
Modulu bat igo nahi bada, npm-ko zerbitzaritako erregistrora igo beharko da. Behin erregistroan dagoelarik, beste erabiltzaileentzat atzigarri egongo da. Baina erregistro honek ez du balioztatze-prozesurik pasa behar, beraz baliteke moduluren bat ez-segurua edo maltzurra izatea. Balioztatze-prozesu bat ezarri beharrean, npm bere erabiltzaileen iritzietan oinarritzen da arauak betetzen ez dituzten moduluak plataformatik kentzeko.

3.2.1 Erabilera

Node.js plataforma instalatzearekin batera instalatzen da. npm erabil erraza da, komandolerrotik erabiltzea dago. Hauek dira komando erabilienak:

- *npm search [paketearenIzena]*

  Komando honekin ezarritako 'paketearenIzena' modulua bilatuko du bere erregistroan. 3.1 taulan bilaketaren emaitzaren adibide bat ageri da.

- npm install [*paketearenIzena*]
  npm install -g [*paketearenIzena*]
  npm install [*paketearenIzena*]@[*Bertsioa*]

  Komando honekin *paketearenIzena* modulua instalatuko da uneko direktorioan, hain zuzen, *node_modules* direktorioan gordeko dira. Komandoan -g aukera gehitzen badiogu, paketea sistema osorako instalatuko du, ez bakarrik uneko direktoriorako. Baita ere 'paketearenIzena'-ren ondoren @ ezarriz nahi dugun bertsioa jaitsi dezakegu. Ez bada ezer ezartzen bertsio berriena jaitsiko du.

NAME	DESCRIPTION	AUTHOR	DATE	VERSION
ngraph.agmgen	Graph generator based on affiliation graph model (AGM)	=anvaka	2015-10-01	0.1.5
ngraph.ascii	A small to render graph with text	=anvaka	2015-10-03	0.0.5
ngraph.coarsen	Given a community strcture creates a coarse graph	=anvaka	2016-07-17	1.2.0
ngraph.cw	Chinese Whispers Graph Clustering Algotihm	=anvaka	2015-11-03	1.0.0
ngraph.centrality	Module to calculate graph centrality metrics	=anvaka	2017-07-01	0.1.6

3.1 Taula: *search* komandoaren emaitzaren adibidea

- npm install

  Komando honekin, ez baldin bada paketerik espezifikatzen, package.json fitxategian ageri diren dependentzia guztiak instalatzen ahaleginduko da. GitHub-eko proiektuekin horrela egiten da zeren honen dependentziak ez dira proiektuarekin batera etortzen.

- npm start

  Komando honekin *package.json* fitxategian dagoen *script* objektuaren *start* parametroan dagoen edukia exekutatuko da. Kasu honetan, honako deia exekutatuko da:
  node_modules/.bin/browserify -s ngraph index.js | uglifyjs > bundle.js

- npm uninstall [*paketearenIzena*]
  npm uninstall -g [*paketearenIzena*]

  Komando honekin 'paketearenIzena' modulua kenduko du eta komandoan -g aukera ezarriz globalki ezabatuko du.

3.3 Browserify

Browserify[3] JavaScript-en idatzitako erreminta bat da, *Node* motako moduluak konpilatzeko gai da gero nabigatzailean erabiliak izateko. Horrela, npm-en dauden paketeak eta Node-n erabiltzeko eginak izan direnak nabigatzailean ere funtzionatu ahalko dute. Normalean zerbitzariaren aldean egiten du lan.
Browserify-ri, ezartzen zaion sarrerako fitxategian hasten da exekutatzen eta hortik aurrera, *require()* motako deiak bilatzen ditu errekurtsiboki menpekotasun-zuhaitza eratu arte. Menpekotasun-zuhaitzean aurkitzen den fitxategi bakoitza JavaScript fitxategi bakarrean kateatzen da. Sortzen den fitxategiak lan egiteko behar duen guztia dauka autonomo bihurtuz eta lan-kargarik gabe.

3.3.1 Erabilera

Browserify erabiltzerako orduan, sarrerako fitxategi bat ezarri behar zaio, sortuko den menpekotasun zuhaitza fitxategi horretatik hasiko da. Sortuko den fitxategi trinkoa normalean 'bundle.js' bezala erazagutzen da, ikus 3.7 kodea.

```
browserify main.js > bundle.js
```

3.7 Kodea: Browserify liburutegiaren erabilera

3.4 UglifyJS

UglifyJS[4] liburutegia JavaScript fitxategiak txikiagotzeko erabiltzen da, hau da, kodeak izan ditzakeen lerro baliogabeak kentzeko edo adierazpen batzuk laburtzeko. Hona hemen UglifyJS-k egiten dituen laburtze batzuk:

- foo["bar"] ⇒ foo.bar
- Jarraian dauden adierazpenak batu.
  var a = 10; var b = 20; ⇒ var a=10,b=20

[3] http://browserify.org/
[4] https://github.com/mishoo/UglifyJS

- Adierazpen konstante sinpleak ebatzi.
  1+2*3 ⇒ 7
  Ordezkapenak bakarrik egiten dira emaitzak byte gutxiago erabiltzen baditu. Adibidez, 1/3 -ren emaitza 0.333333 bezala adieraziko litzatekeenez ez litzake ordezkatuko.

- If baldintzaren sintaxiaren optimizazioa.
  - if (foo) bar();else baz(); ⇒ foo?bar():baz();
  - if(!foo) bar(); else baz(); ⇒ foo?baz():bar();
  - if (foo) bar(); ⇒ foo&&bar();
  - if (!foo) bar(); ⇒ foo||bar();
  - if (foo) return bar(); else return baz(); ⇒ return foo?bar():baz();
  - if (foo) return bar(); else something(); ⇒ {if(foo)return bar();something()}
- Exekutatzera helduko ez den kodea kendu, *return*, *throw*, *break* edo *continue* adierazpenen ondoren dagoen kodea alegia.

3.5 Three.js

Three.js[5] JavaScript-en idatzita dagoen eta 3D grafikoak nabigatzailean sortu eta bistaratzeko erabiltzen den liburutegi bat da. WebGL, SVG edota canvas-ekin erabili daiteke. Gaur egun, WebGL-ren bitartez animazioak egiteko erabiliena da.

3.6 dat.GUI

dat.GUI[6] liburutegia grafoaren edizioaren atalean erabiliko da eta erabiltzailearentzako interfaze grafiko bat eratzen du JavaScript-eko parametroak aldatzeko. Interfaze honekin erabiltzaileak grafoa eraldatzeko aukera izango du.

[5] `https://threejs.org/`
[6] `https://github.com/dataarts/dat.gui`

3.6.1 Erabilera

Interfaze berri bat eratzeko *exdat* modulua kargatu behar dugu eta *gui* berri bat, interfaze bat, eratu (ikus 3.8 kodea).

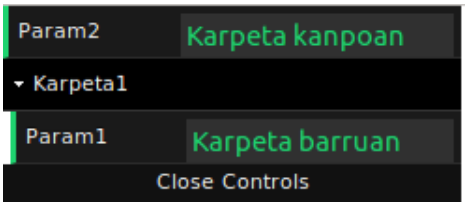
```
var dat = require('exdat');
var gui = new dat.GUI();
```

3.8 Kodea: Interfazearen hasieraketa

Multzoak eratu daitezke ezaugarriak taldekatzeko eta hauek ezkutatzeko, ikus 3.1 irudia.

3.1 Irudia: dat.GUI multzoak

Elementuak *gui* elementuan gehitu daitezke edo karpeta baten barruan, ikus 3.2 irudia.



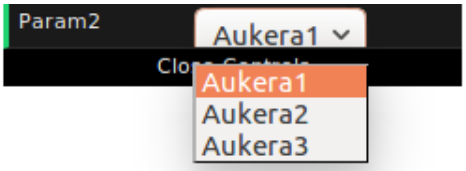
3.2 Irudia: dat.GUI parametroak karpeta barruan eta kanpoan

Liburutegiak hurrengo erremintak eratu ditzake:

- Idazteko eremuak.
  Idazteko eremuetan nahi den balioa idazteko askatasuna ematen du. 3.2 irudian agertzen diren eremuek idazteko aukera ematen dute.
- Zerrendak.
  Zerrenda hedagarriak sortu daitezke hainbat aukeraketa eskaintzeko. Nahi den aukerekin zerrenda bat sortzearekin nahikoa dam ikus 3.3 irudia eta 3.9 kodea.

```
gui.add(obj, 'Param2', ['Aukera1', 'Aukera2', 'Aukera3']);
```

3.9 Kodea: Zerrenda motako eremua sortu

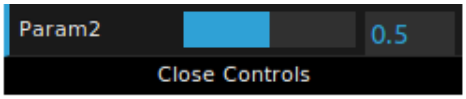


3.3 Irudia: dat.GUI zerrenda-elementua

- Korritze-barra.
  Korritze-barrekin, parametro batek hartu ditzakeen balioak mugatu daitezke, ikus 3.4 irudia eta 3.10 kodea.

```
gui.add(obj, 'Param2', 0, 1);
```

3.10 Kodea: Korritze-barra eremua sortu

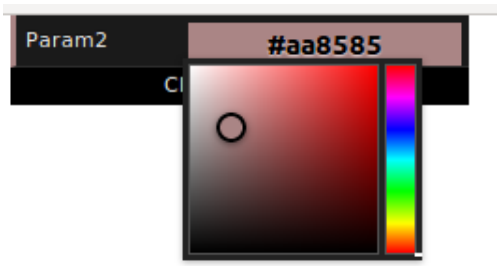


3.4 Irudia: dat.GUI korritze-barra elementua

- Koloreak.
  Koloreak editatzeko aukera ematen duen erreminta dago, ikus 3.5 irudia. Erreminta honetarako *add()*funtzioa erabili ordez, *addColor()* erabili behar da, ikus 3.11 kodea.

```
gui.addColor(obj, 'Param2');
```

3.11 Kodea: Kolorearen eremua sortu



3.5 Irudia: dat.GUI kolorea elementua

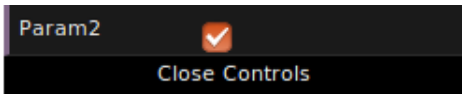
Kontuan izan behar da erreminta honen koloreak '#000000' formatua duela eta Three.js-k koloreak '0x000000' formatuan tratatzen dituela. Beraz, ingurune batetik bestera pasatzean kolorearen formatua aldatu beharko da.

- Egiaztatze-kutxa.
  Egiaztatze-kutxa bat ezartzeko aukera dauka *true* eta *false* balioak tratatzeko, ikus 3.6 irudia.

```
var obj = {
    Param2: true
};
gui.add(obj, 'Param2');
```

3.12 Kodea: Egiaztatze-kutxa eremua sortu

Ezarriko den parametroa definitzerakoan *true* edo *false* balio ezartzearekin nahikoa da hura sortzeko, ikus 3.12 kodea.



3.6 Irudia: dat.GUI egiaztatze-kutxa elementua

- Botoia.
  Botoi bat sakatzean funtzio bati deitzeko aukera dago. Horrela, hura sakatzean gertaera bat sortzen da nahi dugun funtzioa exekutatuz, ikus 3.7 irudia.

  Deitu nahi dugun funtzioa *this.Botoia* aldagaian inplementatu beharko da, ikus 3.13 kodea.

3.7 Irudia: dat.GUI botoi-elementua

```
var obj = new gertaera();
gui.add(obj, 'Botoia');
function gertaera(){
    this.Botoia = function(){}
}
```

3.13 Kodea: Botoiaren eremua sortu

2.2 atalean azaldu den bezala, ngraph grafoak bistaratzeko JavaScript-en eratutako liburutegi bat da eta npm plataforman atzigarri dago. Liburutegi horren egileak beste liburutegi bat sortu du, VivaGraphJS izenekoa, grafoak bistaratzeko. Liburutegi hori modu trinkoan eginda dagoenez honek arazoak ekar ditzake beste aplikazio bat sartu nahi badiogu, ez baitauka malgutasun handirik. ngraph liburutegiarekin, berriz, npm-n honen moduluak atzigarri egonda beste liburutegi bateko moduluak erabiltzeko aukera ematen du aplikazioa guztiz aldatu gabe.
Bestalde, hainbat errenderizazio mota eskaintzen ditu grafoak modu desberdinetan errenderizatzeko.

3.7 Eredu fisikoa

Ngraph liburutegiak modulu bat dauka grafoaren eredu fisikoa simulatzeko. Bere eginkizun nagusia indarretan oinarritutako diseinu grafikoa eratzea da adabegiak eta ertzak kontrolatuz. Simulatzaileak gorputz bakoitzean eragiten duten indarrak kalkulatzen ditu eta ondoren beraien posizioa deduzitzen du Newton-en legea aplikatuz. Newton-en legearen ekuazio diferentzial arruntak ebazteko Euler-en metodoa erabiliko da. Hiru indar nagusik eragiten dute sisteman.

- Hooke-ren legea.
  Malgukien indarraren bitartez adabegiak lotu egiten dira.

- Coulomb-ren legea.
  Gorputz bakoitzak besteak aldaratzen ditu.

- Diseinu sistema egonkor bat lortuko dugula bermatzeko, arraste-indar bat gehitzen zaio simulazioa geldotzeko.

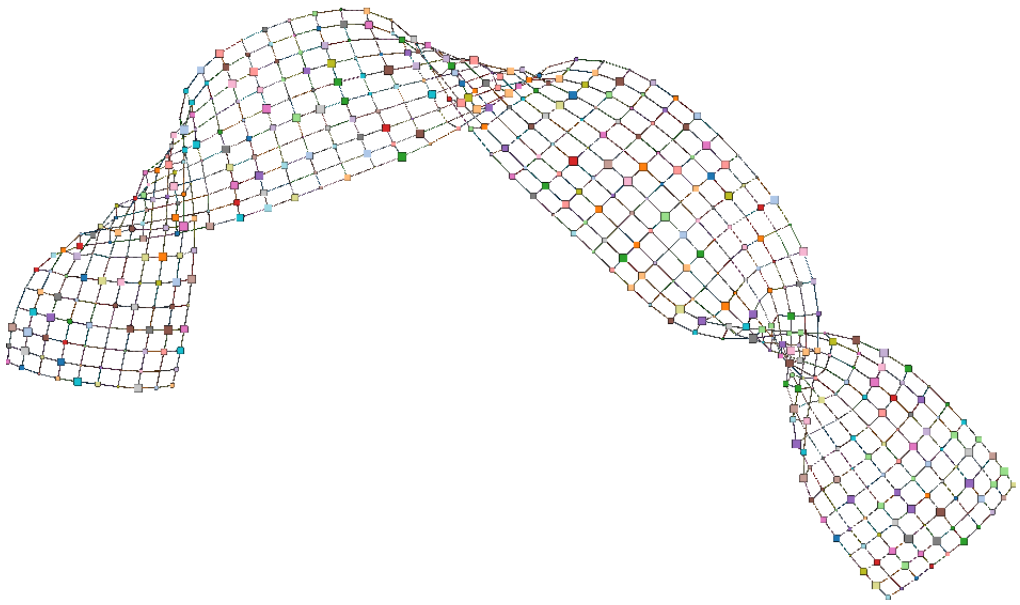
Gorputzen indarren kalkuluak Barnes-Hut algoritmoaren bitartez kalkulatzen da eta horren konplexutasuna O(n log n) da. Oso erabilia da astrofisikaren eremuan. Normalean, % 1-eko doitasuna bilatzen denean erabiltzen da eta doitasun hau nahikoa izaten da astrofisikako kalkuluetarako.

3.8 Errenderizazioa

Jarraian, ngraph liburutegiak eskaintzen dituen errenderizazio desberdinak azalduko dira, errenderizazioaren irudi bat atxikirik.

3.8.1 Pixi.js

Pixi.js[7] grafiko elkarreragileak, multi-plataformako aplikazioak eta jokoak sortzea ahalbidetzen duen 2D errenderizazio liburutegia da. Erabiltzeko erraza da WebGL API-a eta nabigatzailearekin egin beharreko lana minimoa delako, horrela erabiltzailearentzat erosoagoa bihurtuz. npm-n atzigarri dago eta elkarte handia du liburutegiaren zalantzak argitzeko eta eztabaidatzeko. Baita ere hainbat adibide ditu aplikazioa nola funtzionatzen duen ikusteko eta nola erabiltzen den ikasteko.

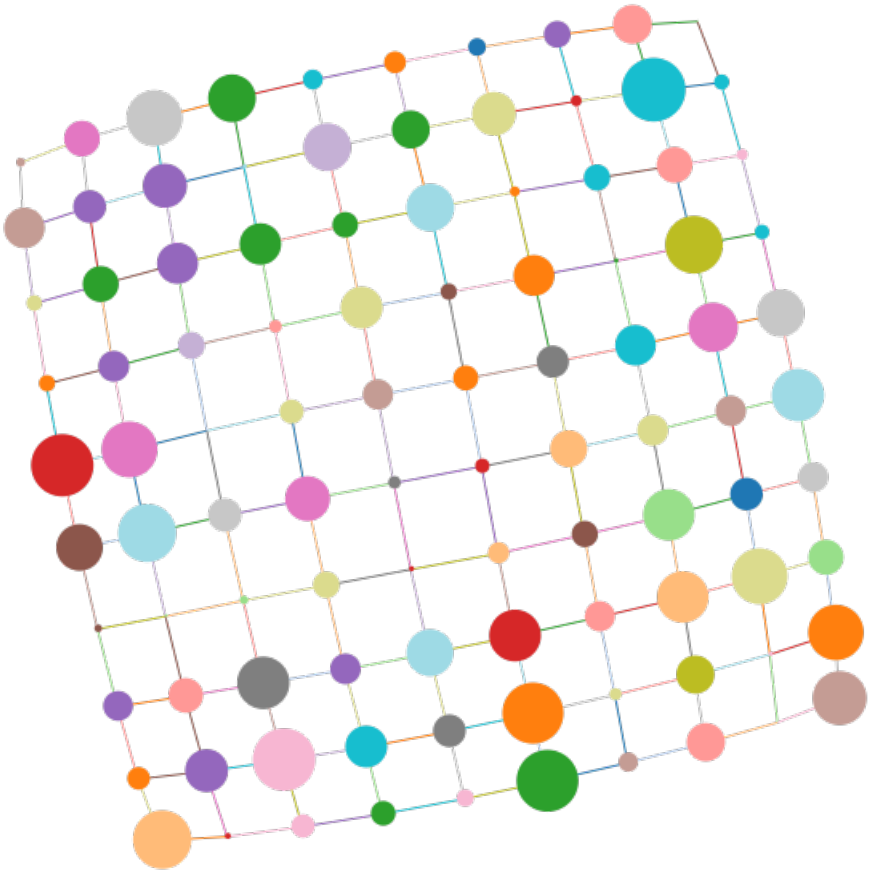


3.8 Irudia: Pixi.js-ren errenderizazioa

[7] `http://www.pixijs.com/`

3.8.2 Fabric.js

Fabric.js[8] canvas-ekin lan egiten duen JavaScript liburutegi indartsu bat da. Canvas-ek webgunean grafikoak egiteko aukera ematen du. Baina grafiko hauekin elkarrekintzaren bat gehitu nahi badugu, hala nola, irudia aldatu edota forma konplexuagoak irudikatu arazoak izaten ditugu. Fabric-en helburua arazo hau konpontzea da. Hau egiteko, Fabric-ek objektuekin lan egiten du.

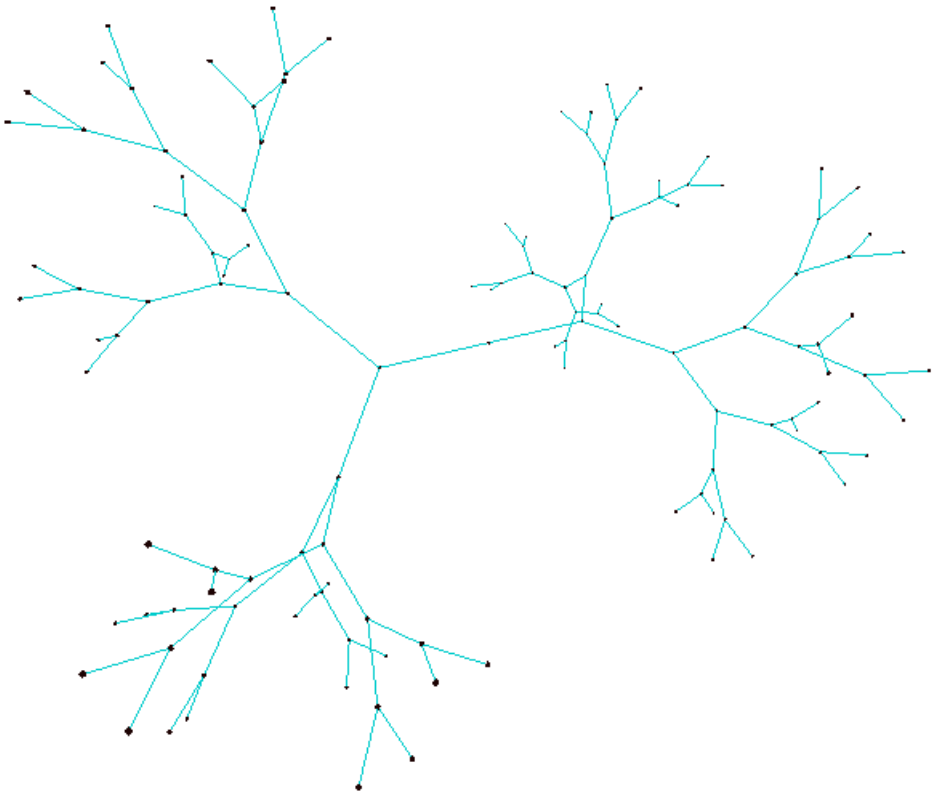


3.9 Irudia: Fabric.js-ren errenderizazioa

[8] `http://fabricjs.com/`

3.8.3 Three.js

3.5 atalean azaldu den Three.js liburutegia erabiliko da errenderizazio hau egiteko. Liburutegi honek 3D-n errenderizatzeko ahalmena du. Bi modulu eginak daude errenderizazio motor honekin lan egiteko ngraph liburutegian, *ngraph.pixel* eta *ngraph.three*, 3.9 atalean aztertuko dira.



3.10 Irudia: *Three.js* errenderizazioa

3.8.4 Konparazioa

Hona hemen errenderizazio desberdinen arteko konparazioa egiteko kontuan hartu diren ezaugarriak:

- Bistaratze ona.
  Errenderizazioaren emaitza bisualki ona den ala ez.

- Errenderizazio arina.
  Adabegi kantitate handiko grafoak errenderizatzeko arintasuna kontuan izan da. Proba hauek egiteko, 100x20 adabegiko sareak errenderizatu dira (2.000 adabegi).

- Dokumentazio zehatza.
  Dokumentazio zabala eta ondo eratua duten ala ez.

- Ereduak.
  Beraiekin lan egiteko adibideak dituzten ala ez.

- 3D.
  Errenderizazioa eskaintzen duten ala ez.

3.2 taulan, azaldu diren ezaugarrien konparaketa ageri da.

Pixi, Fabric eta Three liburutegiek bistaratze argia, dokumentazio ona, errenderizazio azkarra eta ereduak dituzte beraiekin lan egiteko. Hiru liburutegiak liburutegi onak eta oso erabiliak dira, ordea, Three.js da hauetatik 3D errenderizazioa eskaintzen duen bakarra. Ezaugarri honengatik izan da aukeratua errenderizazio-motortzat erabiltzeko.

Liburutegia	Bistaratze ona	Errenderizazio arina	Dokumentazio zehatza	Ereduak	3D
Pixi	X	X	X	X	
Fabric	X	X	X	X	
Three	X	X	X	X	X

3.2 Taula: Errenderizazioaren konparazioa.

3.9 Three.js errenderizazioa

Three.js liburutegia erabiliz errenderizatu ahal izateko bi modulu daude inplementatuta ngraph proiektuan, *ngraph.three* eta *ngraph.pixel*. Biek errenderizazio antzerakoa eskaintzen dute baina diferentzia txiki batzuk dituzte. Jarraian, Three.js errenderizazioaren ezaugarriak azalduko dira eta ezaugarri bakoitzean, bi modulu hauen arteko desberdintasunak adieraziko dira.

3.9.1 Ingurunea

3D ingurune batean gaudenez, honetan mugitzeko gai izan behar gara, hau da, grafoa mugitzeko edo bere inguruan mugitzeko gai izan behar gara.

ngraph.three

Modulu honetan, ingurunean zehar ibiltzeko, kamera mugitzeko aukera dago. Kamera hau beti puntu finko bati begira egongo da eta egiten duen errotazio-mugimendua puntu finko honen inguruan izango da. Puntu finkoa sortu den grafoaren erdian egongo da. Hona hemen ingurunean mugitzeko eskaintzen dituen kontrolak:

- Saguko ezker botoia. Botoi honekin kamera puntu finkoaren inguruan mugituko da, beti distantzia berdina mantenduz. Kontrolak alderantzikatuta daude, hau da, sagua beherantz mugitzean kamera gorantz mugituko da.
- Saguko eskuin botoia. Botoi honekin kamera alboetara edo gora eta behera mugituko da.
- Saguko erdiko botoia. Botoi honekin kamera aurrera eta atzera mugituko da, grafoarekiko distantzia txikituz.

ngraph.pixel

Modulu honetan, ingurunean zehar ibiltzeko, kamera mugitzeko aukera dago. Kasu honetan, kamera ez du grafoaren inguruan biratuko baizik eta biraketa kameraren inguruan egingo da. Hona hemen ingurunean zehar mugitzeko eskaintzen dituen kontrolak:

- Kamera mugitu. Kamera mugitzeko teklatua erabiltzen da. W eta S-rekin kameraren Z ardatzean mugitzen da, W aurrera mugitzeko eta S atzerantz. A eta D-rekin kameraren X ardatzean mugitzen da, A ezkerrera mugitzeko eta Z eskumarantz. R eta F-rekin kameraren Y ardatzean mugitzen da, R gora mugitzeko eta F beherantz. Desplazamendu guztiak kameraren erreferentzia-sisteman egiten dira.
- Kamera biratu. Kamera biratzeko teklatua zein sagua erabili daiteke. Ezker eta eskuin geziekin kamera Y ardatzaren inguruan biratzen du. Gora eta behera geziekin kamera X ardatzaren inguruan biratzen du. Q eta E teklekin kamera Z ardatzaren

inguruan biratzen du, Q erlojuaren kontrako norabidean mugitzeko eta E alderantziz biratzeko.
Saguarekin ezker botoia sakatuz biratu daiteke, hala ere sagua 2D-ko plano batean mugitzen denez ez dauka aukerarik hiru planotan mugimenduak egiteko, beraz bakarrik X eta Y ardatzen inguruan biratu dezake.

Teklatua eta sagua batera erabili daitezke.

3.9.2 ShaderMaterial

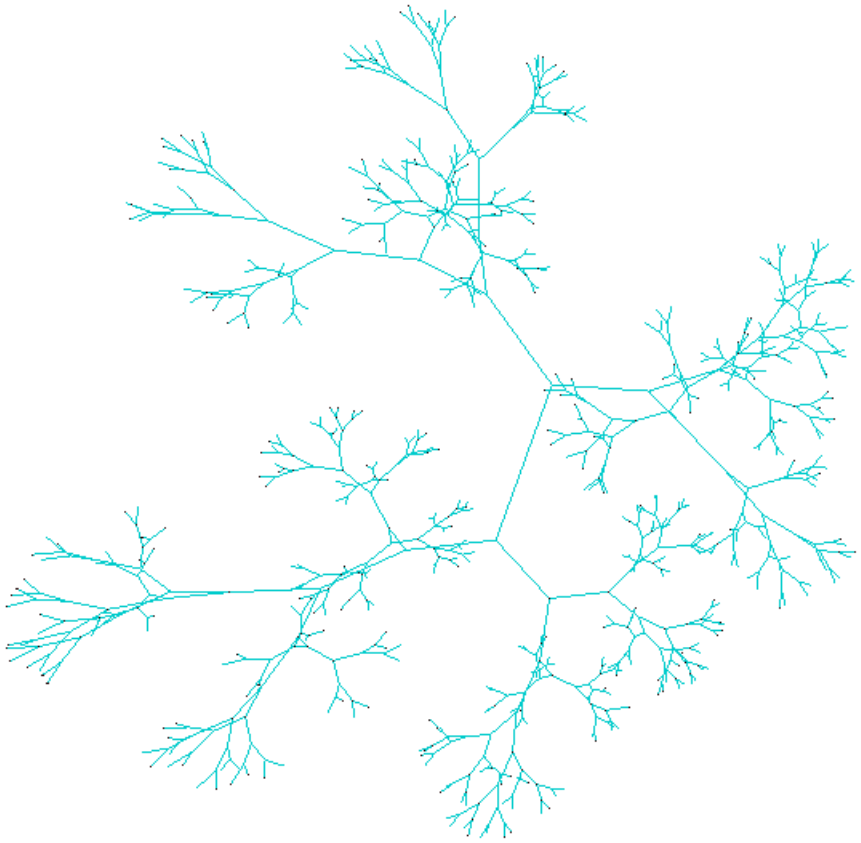
Shader-ak konputagailuen CPU-ean exekutatzen diren GLSL lengoaiako programak dira. Hauek Three.js-ko materialak egiteko erabiltzen dira, zenbat eta shader sinpleagoa orduan eta errenderizazio arinagoa. *ngraph.pixel* moduluak erabiltzen duen shader-a sinpleagoa denez, errenderizazioa arinago gauzatu dezake, hala ere, honek materialaren kalitatean galera bat ekar dezake.

3.9.3 Ereduak eta inplementazioa

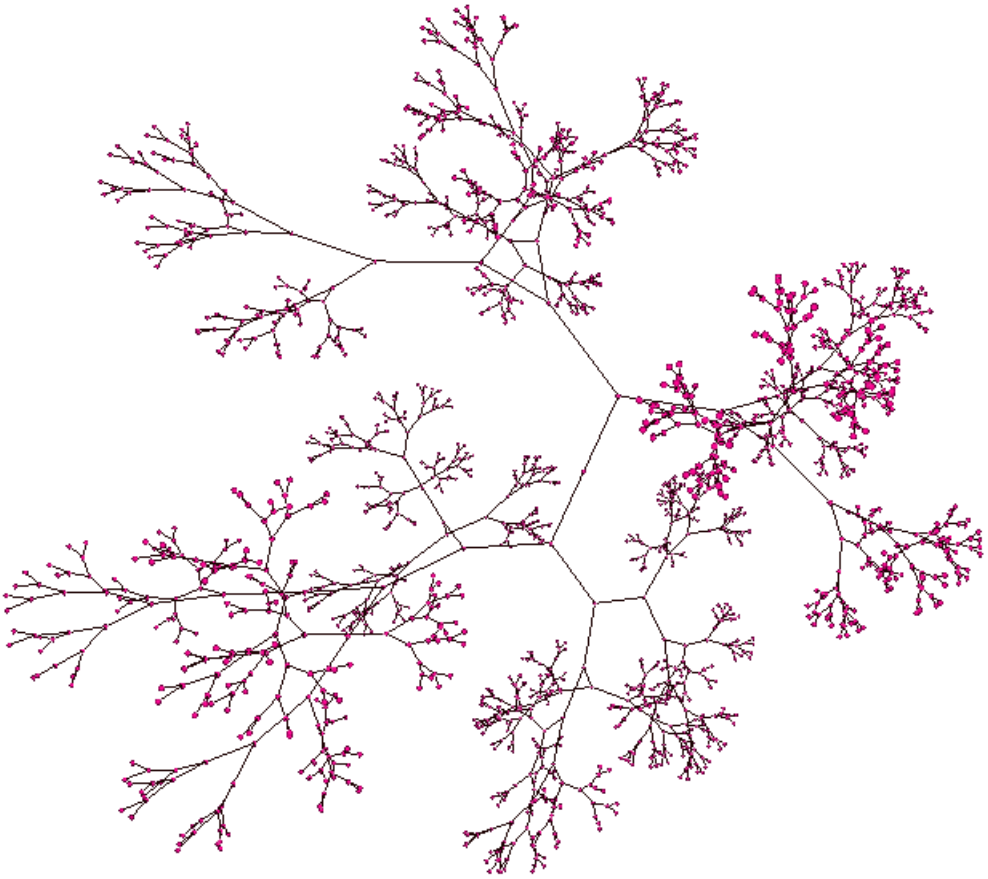
Bi moduluak ondo dokumentatuta daude eta erabiltzen ikasteko eta ulertzeko eredu nahikoa dago grafoen bistaratze aldetik. Grafoen edizioari erreparatzen badiogu, ordea, *ngraph.pixel* moduluak baditu grafoak editatzeko aukera batzuk inplementatuta.

3.9.4 Konparazioa

Bi moduluen arteko bistaratzea oso antzekoa dela ikusi daiteke 3.11 eta 3.12 irudietan. Bistaratze aldetik daukaten diferentzia nabarmenena kamera mugitzeko era da eta, hala ere, ez da garrantzia handiegirik duen ezaugarri bat. Dena dela *ngraph.pixel* modulua erabiltzea aukeratu da, beste konparaketetan ez bezala grafoen bistaratzea ez da izan ezaugarri erabakigarri bat. Kasu honetan, grafoen edizioa hartu da kontuan aukeraketa egiteko, zeren *ngraph.pixel* moduluak editatzeko aukerak inplementatuta ditu eta *ngraph.three* moduluak, ostera ez. Aukeraketa honekin lortu da denbora aurreztea grafoen edizioa inplementatzerakoan.



3.11 Irudia: Zuhaitz bitar baten errenderizazio *ngraph.three* erabiliz



3.12 Irudia: Zuhaitz bitar baten errenderizazio *ngraph.pixel* erabiliz

4. KAPITULUA

Bistaratzea

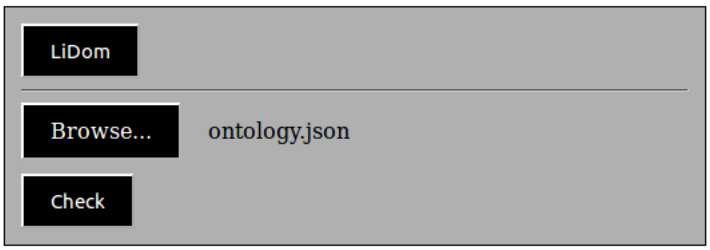
Lehen azaldu den bezala *ngraph.pixel* modulua erabiliko da grafoa eratzeko eta bistaratzeko. Hala ere, ezin da grafoa bistaratu honen egitura ezagutu gabe, hau da, grafoaren adabegi eta ertzen informaziorik gabe.

4.1 Grafoa eratu

Eraikiko dugun grafoaren informazioa JSON fitxategi batean entregatuko zaigu. Fitxategi hau 1.2 atalean azaldu den LiDom Builder aplikazioak eratuko du ontologiaren informazioarekin eta informazio honekin, grafoaren adabegiak eta ertzak eratuko dira. Erabiliko den ontologia Wikipediako Solar System ingelesezko artikulutik eratorrita dago eta aurrerago agertuko diren eredu guztiak ontologia horretan oinarrituta egongo dira.
JSON fitxategia kargatzeko bi aukera izango ditugu, lehenengoa gure ordenagailu pertsonalean daukagun ontologia bat kargatzeko aukera edota LiDom Builder aplikazioak sorturiko datuekin eratu daitekeen fitxategia. 4.1 irudian fitxategi aukeraketa ageri da.

4.1.1 Datuen irakurketa

Datuak irakurtzeko aurreko atalean azaldu den *require()* funtzioa erabiliko da 4.1 kodean adierazten den bezala.



4.1 Irudia: Fitxategi-aukeraketa

```
var json = require('./data.json');
```

4.1 Kodea: JSON fitxategia kargatu

Fitxategi horretan hiru objektu nagusi daude: adabegiak, loturak eta hizkuntza.

- Adabegiak.
  Objektu honetan adabegien informazioa daukagu. Bi adabegi mota bereizten dira: kontzeptuak eta erlazio motakoak. 4.2 kode zatian kontzeptu motako adabegien egitura azaltzen da.

```
{
    "name":"Science",
    "type":1,
    "url":"Science",
    "wikiId":26700,
    "full_name":"Science",
    "entity":"concept",
    "slug":0.39999998,
    "kind":""
}
```

4.2 Kodea: Kontzeptu motako adabegia

  4.3 kode zatian, berriz, erlazio motako adabegien egitura agertzen da.

```
{
"name":"isA",
"type":4,
"url":"Planet###Science",
"wikiId":-1,
"full_name":"WikiTax",
"entity":"relation",
"slug":0.39999998,
"kind":"isA"
}
```

4.3 Kodea: Erlazio motako adabegia

Jarraian, parametro hauek ematen diguten informazioa adierazten da:

- Name.
  Kontzeptu adabegietan parametro honek adabegiaren izena adierazten du. Erlazio adabegietan, berriz, erlazio mota adierazten du (ikusi 1.2 atala erlazio mota desberdinak ikusteko).
- url.
  Kontzeptu adabegietan 'name' parametroak ematen digun informazio bera ematen du, baina kasu honetan, izen konposatuak '_' batekin lotuak daude url bezala erabili ahal izateko. Erlazio adabegietan erlazioak zer bi kontzeptu lotzen dituen adierazten du. Bi kontzeptu horiek '###' karaktereen bitartez banandurik daude.
- Entity.
  Parametro honek zehaztuko digu zer adabegi mota den. Balio posible bakarrak *concept* eta *relation* dira.
- Slug.
  Parametro honetan adabegiaren konfiantza maila agertzen da, hau da, LiDomek ontologia eratzean, zer konfiantza mailarekin eratu duen adabegi hori. 0 eta

1 arteko balioak hartzen ditu, gero eta hurbilago egon 1 baliotik orduan eta seguruago dago aplikazioa adabegi horren zuzentasunaz.

– Kind.
Kontzeptu motako adabegietan eremu hau beti utzik egongo da.
Erlazio adabegietan erlazio mota adierazten du. Parametro honek 'name' parametroak ematen duen informazio berdina ematen du.

Type, *wikiId* eta *full_name* parametroek, bistaratzeari erreparatuz gero, garrantzirik gabeko informazio ematen dute.

Erlazio motako adabegien izena 4.4 kodean adierazten den bezala eratzen da.

```
name = json.nodes[n].url.split('###');
nodes[n].name = name[0] + ' ' + nodes[n].kind + ' ' + name[1];
```

4.4 Kodea: *Relation* adabegien izena eratu

url parametroan dagoen balioa *split()* funtzioarekin banantzen da, horrela erlazionatuta dauden adabegien izenak lortuz. 4.3 adibidean agertzen den egiturarekin honako izena lortuko genuke: *Planet isA Science*. Baita ere bi parametro berri ezarriko ditugu, garrantzia eta zailtasun maila. LiDom-ek ez dauka bi balio horiek eratzeko gaitasuna, baina bezeroari interesatzen zaio edizio atalean bi balio horiek aldatzea. Balio lehenetsia bezala, *slug* parametroaren balio berdina ezarriko zaie.

- Loturak.
Adabegiak lotzen dituzten informazioa dute.

```
{
    "source":115,
    "target":1,
    "type":1,
    "distance":3.0
}
```

4.5 Kodea: *links* parametroaren egitura

  - Source. Ertza zein adabegian sortzen den adierazten du.
  - Target. Ertza zein adabegira doan adierazten du.
  - Type. Ertz mota adierazten du. Bi balio izan ditzake, 1 edo 10.
  - Distance. Ertzaren luzera adierazten du.

  Loturak sortzeko ez da informazioa hori erabiliko, horren ordez, honako metodoa jarraituko da: Kontzeptu motako bi adabegi konektaturik egoteko, beti erlazio bat egon behar da medio, horrek esan nahiko du erlazio motako adabegietan kontzeptuak elkartzeko behar den informazio guztia dagoela. Erlazio adabegi bat eratzean bi lotura eratu behar dira, lehenengoa *url* parametroaren lehen adabegitik erlazio adabegira eta bigarrena, erlazio adabegitik *url* parametroko bigarren adabegira.

- Hizkuntza.
  Parametro honetan sorturiko ontologiaren hizkuntza adierazten da.

```
"language":"en"
```

4.6 Kodea: *language* parametroaren egitura

4.1.2 Grafoaren egitura

Grafoa bistaratzeko 3 objektu eratuko dira JSON fitxategiko datuak erabiliz: adabegien lista, erlazioen datuak eta hizkuntza.

- Adabegien lista.
  Lista honetan grafoa osatuko duten adabegi guztien datuak daude. Adabegien artean, kontzeptu eta erlazio motek egitura desberdina izango dute. 4.7 kodean kontzeptu motako egitura baten adibidea adierazten da eta 4.8 kodean, berriz erlazio motako batena. Jarraian, parametro hauek adierazten duten informazioa azaltzen da:

  - *Confidence*.
    JSON fitxategian dagoen Slug parametroa. Bertan adabegiaren konfiantza maila adierazten da.

– *Importance* eta *Difficulty*.
Aplikazioak sortzen dituen bi parametro berri: adabegiaren garrantzia eta zailtasun maila. Slug parametroaren balioa bera izango dute. 5.1.4 atalean aplikazioak adabegien garrantzia kalkulatuko du balio hau ordeztuz.

```
    "confidence": 0.39999998,
    "importance": 0.39999998,
    "difficulty": 0.39999998,
    "type": "concept",
    "kind": "concept",
    "name": "Science"
```

4.7 Kodea: *concept* motako adabegien datu-egitura

```
{
    "confidence": 0.39999998,
    "importance": 0.39999998,
    "difficulty": 0.39999998,
    "type": "relation",
    "kind": "isA",
    "source": "Planet",
    "target": "Science",
    "name": "Planet isA Science"
}
```

4.8 Kodea: *relation* motako adabegien datu-egitura

– *Type*.
Entity parametroaren balioa izango du, beraz *concept* edo *relation* balioa izango du.

– *Kind*.
Erlazio motako adabegi bat baldin bada *kind* parametroaren balio berdina izango du, hau da, erlazio mota adieraziko du. Kontzeptu motako adabegiek balio hutsa izan beharrean, oraingoan *concept* balioa esleituko zaio.

  - Name. Adabegiaren izena izango da. Kontzeptu motako adabegi bat bada *name* parametrotik hartuko da balioa, erlazio motakoek, aldiz, *url* parametrotik eratorriko da izena.

  - *Source*.
    Erlazio motako adabegiek bakarrik izango dute parametro hau eta erlazioa zein adabegitik hasten den adierazten du.

  - *Target*.
    Erlazio motako adabegiek bakarrik izango dute parametro hau eta erlazioa zein adabegian amaitzen den adierazten du.

- Erlazioen datuak.
  Objektu honetan erlazioak adierazteko erabiliko diren adabegien tamaina eta kolorea zehazten da. JSON fitxategiak datu hauek ez baldin baditu, 4.9 kodean agertzen diren balio lehenetsiak erabiliko dira. Honen helburua kolore eta tamainen bitartez adabegiak desberdintzea da beraien identifikazioa errazteko.

```
var relationType = ['concept', 'isA', 'partOf', 'prerequisite',
 'pedagogicallyClose'];
var relationColor = ['0x2d74b2', '0xf92222', '0x3ae178',
'0xf0ff00', '0xff00f4'];
var relationSize = [40, 20, 20, 20, 20];
```

4.9 Kodea: Erlazioak adierazteko datuak

- Hizkuntza.
  Ontologiaren hizkuntza adierazten du, 5.1.1 atalean interfazearen hizkuntza zehazteko erabiliko da.

4.2 Grafoa sortu

Behin datuen irakurketa eginik dagoenean, grafoa sortu daiteke. *ngraph.graph* modulua erabiliko dugu grafoa eratzeko, 4.10 kodean azaltzen den bezala, *require* funtzioaren bitartez egiten da.

```
var createGraph = require('ngraph.graph');
var graph = createGraph();
```

4.10 Kodea: Grafoa sortu

Agindu hauekin grafoa hasieraturik dago eta adabegiak eta ertzak sartzea bakarrik falta zaio. Adabegiak gehitzeko *addNode()* funtzioaz baliatuko gara eta ertzentzat, berriz, *addLink()*funtzioaz.

- *addNode(id, data)*
  Id parametroan adabegiaren izena doa eta *data* parametroan adabegi horren datuak. Adabegi bakoitzari lotuta 4 datu gordeko dira. Lehenengoa, adabegiaren *kind* parametroa, hau *concept* motakoa baldin bada *concept* dela adieraztea, hau da, adabegien *kind* parametroaren balio pasako zaio. Beste 3 parametroak konfiantza, garrantzia eta zailtasun maila izango dira, hurrenez hurren.

- *addLink(fromId, toId, data)*
  FromId parametroan ertza zein adabegian hasten den adierazten da eta *toId* parametroan, berriz, zein adabegian amaitzen den. Sarturiko adabegia ez bada existitzen, adabegi berri bat sortuko du datu horiekin, beraz garrantzitsua da lehenengo adabegiak sortzea, era honetan sortzen diren adabegiek ez baitute daturik beraien baitan eta jadanik existitzen den adabegi bat sortzean datuak ez direnez eguneratzen ez dira berridatziko. Ertzek datuak gordetzeko aukera ere badute, baina ez zaigunez interesatzen, ez da erabiliko.

4.3 Grafoa errenderizatu

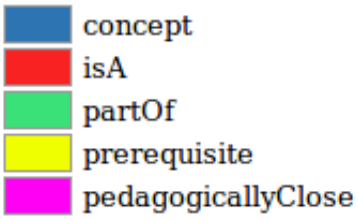
Behin grafoa sortuta eta bistaratu ahal izateko, errenderizatu egin beharko da. Errenderizazioa egiteaz, 3.9 atalean azaldu den bezala, *ngraph.pixel* modulua arduratuko da. Errenderizazioa egitean baita ere, adabegien eta ertzen tamaina eta atzealdearen kolorea zehaztu ahalko dira. 4.11 kodean ageri da zelan gaitzen den errenderizazioa.

```
var renderGraph = require('ngraph.pixel');
var renderer = renderGraph(graph, {
    node: createNodeUI,
    link: createLinkUI,
    container: document.querySelector('.container')
});
```

4.11 Kodea: Grafoa errenderizatu

Adabegiak eta ertzak eratzeko *node* eta *link* parametroetan hurrenez hurren hauek eratuko dituzten funtzioak ezarri behar dira. Grafoaren atzealdea ezartzeko, html fitxategian dagoen *container* parametroko eremuaren izena beharko dugu.

- *createNodeUI(node)*
  Hainbat erlazio mota ditugunez adabegiak ondo desberdintzea ezinbestekoa da, beraz kolore eta tamaina desberdinak esleituko zaizkie. 4.9 kodean adierazten diren datuak erabiliko dira horretarako. Erlazio bakoitzak bere kolore propioa izango du eta tamaina aldetik, guztiek tamaina bera izango dute kontzeptu motako adabegiak izan ezik, hauek bi aldiz handiagoak izango dira beraien garrantzia adierazteko. 4.2 irudian ikus daiteke zein kolore erabili diren erlazio bakoitza adierazteko.



4.2 Irudia: Adabegi eta erlazio desberdinak bistaratzeko erabilitako koloreak

Kolorea eta tamainak definiturik daudela, adabegi bakoitzeko datuetan honen mota bilatu behar da eta dagokion balioak itzuli, ikus 4.12 kodea.

```
if(node.data[0] == relationOptions[0][n]){
    return {
        color: struct[1][1][n],
        size: struct[1][2][n],
        data: node.data };
}
```

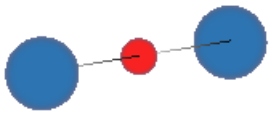
4.12 Kodea: Adabegiei dagokien kolorea, tamaina eta datuak ezarri

- *createLinkUI(link)*
  Ertzetan, loturaren hasiera eta amaiera desberdintzeko koloreen bitartez adierazi-ko ditugu. Loturaren hasiera zuria izango da eta amaiera, berriz, beltza, ikus 4.13 kodea.

```
{
    fromColor: 0xffffff,
    toColor: 0x000000
}
```

4.13 Kodea: Ertzen kolorea

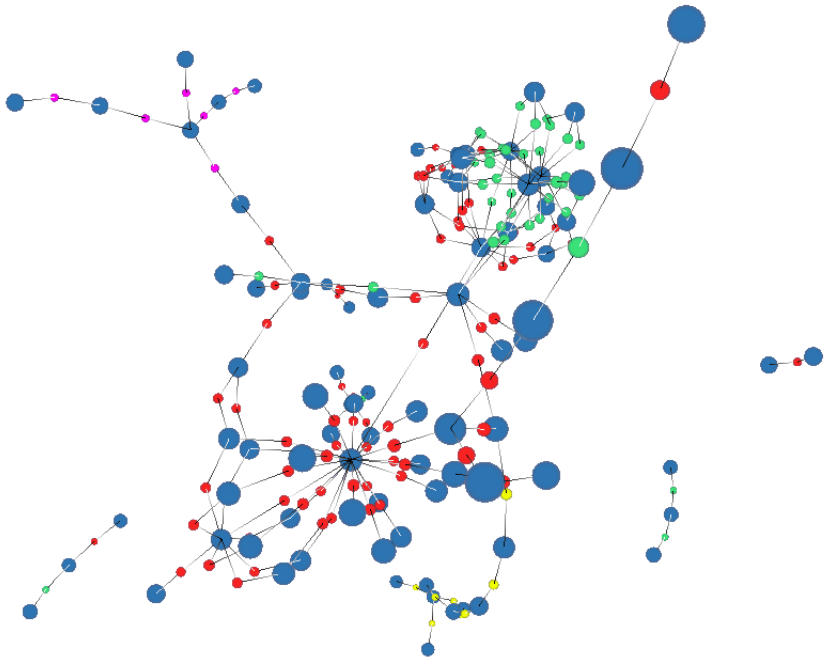
4.3 irudian ertza errenderizatzean izango duen itxura ikus daiteke.



4.3 Irudia: Bi kontzeptuz osaturiko lotura sinple bat

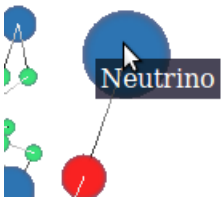
4.4 Errenderizazioaren emaitza

Jarraian, 4.4 irudian eman zaigun Wikipediako ’Solar System’ orrialdeko datuekin eratu den grafoa ageri da.



4.4 Irudia: Wikipediako ’Solas System’ orrialdeko emaitzaren errenderizazioa

Grafoko adabegien izena ikusteko kurtsorea gainetik pasatzearekin nahikoa da, 4.5 irudian ageri den bezala.

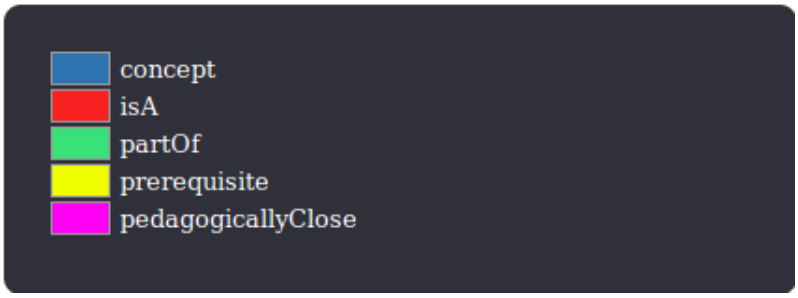


4.5 Irudia: Adabegiaren izena bistaratzea

Hori lortzeko, erabiltzen ari garen moduluak gertaerak kontrolatzeko aukera dauka. Kasu horretan, kurtsorea adabegi baten gainetik pasatzean *’nodeover’* gertaera aktibatzen da eta

adabegiaren *id* parametroaren edukia bistaratzen du. Gertaera honetaz at, *'nodeclick'* eta *'nodedbclick'* gertaerak ere badaude.

Erabiltzailearentzat grafoaren ulertzea erosoagoa izateko testu bat gehitu da, non erlazio bakoitzaren koloreak eta izenak ageri diren. Horrela erabiltzailea ez da adabegiz adabegi joan behar ikusteko zer erlazio mota ageri den, legenda begiratzearekin nahikoa dauka.

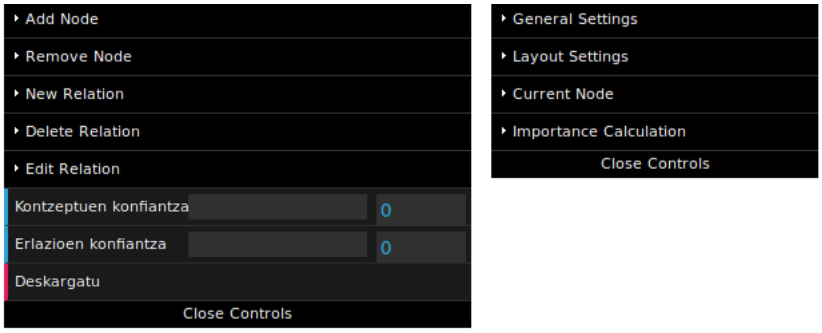


4.6 Irudia: Grafoaren legenda

5. KAPITULUA

Edizioa

Behin grafoa bistaratzea lortu eta gero, edizioaren atalera pasako gara. Grafoa editatzeko datuak sartu eta aldatzeko gai izan behar gara, horretarako menu edo interfaze bat eratu beharko dugu erabiltzaileak grafoaren egitura aldatzeko. Interfaze berri bat zerotik sortu beharrean npm-ek eskura jartzen dituen liburutegiez baliatuko gara. Interfazea eraikitzeko 3.6 atalean azaldu den dat.GUI liburutegia erabiliko da. 5.1 irudian, ediziorako eratu den interfazea ageri da.



5.1 Irudia: dat.GUI liburutegiarekin eratutako interfazea

Bi panel sortuko dira. Lehenengo panela 4 multzoz osatuta egongo da.

- General Settings.
  Multzo honetan, aplikazioaren ezaugarri orokorrak aldatuko dira. Hala nola, hizkuntza, diseinua gelditzeko aukera, 3D edo 2D errenderizazioa gaitu eta atzealdearen kolorea.

- Layout Settings.
  Multzo honetan, diseinuaren itxura kontrolatu ahalko da eredu fisikoa editatuz.

- Current Node.
  Multzo honetan, 4.4 atalean azaldu den *nodeclick* gertaeren bitartez adabegi baten informazio ikusteko eta editatzeko aukera izango dugu. Adabegiaren izena, kolorea, tamaina, konfiantza, garrantzia eta zailtasun maila ikusteko eta editatzeko aukera egongo da.

- Importance Calculation.
  Multzo honetan, adabegien garrantzia kalkulatzeko 3 aukera desberdin proposatuko dira. Hauek dira proposaturiko aukerak: *degree centrality*, *betweenness centrality* eta *eigenvector centrality*.

Bigarren panelean beste 4 Multzo izateaz gain, iragazketa bat egiteko eta ontologiaren datuak gordetzeko aukera egongo da.

- Add Node.
  Multzo honetan, grafoari adabegi berriak eta beraien arteko erlazioak ezartzeko aukera egongo da.

- Remove Node.
  Multzo honetan, grafoak dituen adabegiak ezabatzeko aukera egongo da.

- New Relation.
  Multzo honetan, erlazio berriak eratzeko aukera egongo da.

- Delete Relation.
  Multzo honetan, sortu diren erlazio berriak ezabatzeko aukera egongo da.

- Edit Relation.
  Multzo honetan, jadanik sortuta dauden erlazioen ezaugarriak aldatzeko aukera egongo da.

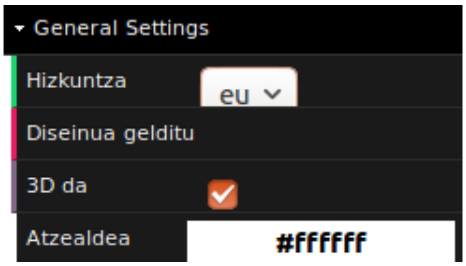
5.1 Lehenengo panela

Lehen esan bezala panel honetan 4 multzo egongo dira: *general settings*, *layout settings*, *current node* eta *importance calculation*.

5.1.1 General Settings

Atal honetan aplikazioaren ezaugarri orokorrak aldatuko dira. 5.2 irudian karpeta honen itxura ageri da. Bertan aukera hauek eskaintzen dira:

- Hizkuntza
- Diseinua gelditu
- 2D/3D
- Atzealdea



5.2 Irudia: *General Settings* multzoa

Hizkuntza

Hiru hizkuntza egongo dira erabilgarri interfazean: euskara, ingelesa eta gaztelania. 4.1.1 atalean lortu dugun hizkuntza hiru hauetako bat baldin bada, interfazean hizkuntza hori ezarriko da. Detektatutako hizkuntza hiru hauetako bat ez bada, hizkuntza lehenetsitzat ingelesa ezarriko da.
Atal hau egiterakoan bi bertsio inplementatu dira.

- Lehen bertsioa.
  Hizkuntza berria eratzeko interfazeak ezabatzea erabaki da, zeren dat.GUI moduluko interfazeko parametroen izenak ezin dira balio bat esleituz aldatu, parametroaren izena aldagaia bera da. Parametro baten izena aldatu ahal izateko parametro berri bat sortu behar da eta nahi zaion izena ezarri, aurrekoa ezabatuz. Parametro guztiak banan-banan sortzen eta ezabatzen joan beharrean, arinagoa da guztiak batera

ezabatzea eta interesatzen zaizkigunak sortzea. Beraz, hizkuntzaren arabera fitxategi desberdinak erabiliko dira, fitxategi bakoitzean dauden parametroak dagokien hizkuntzaren araberakoak dira.

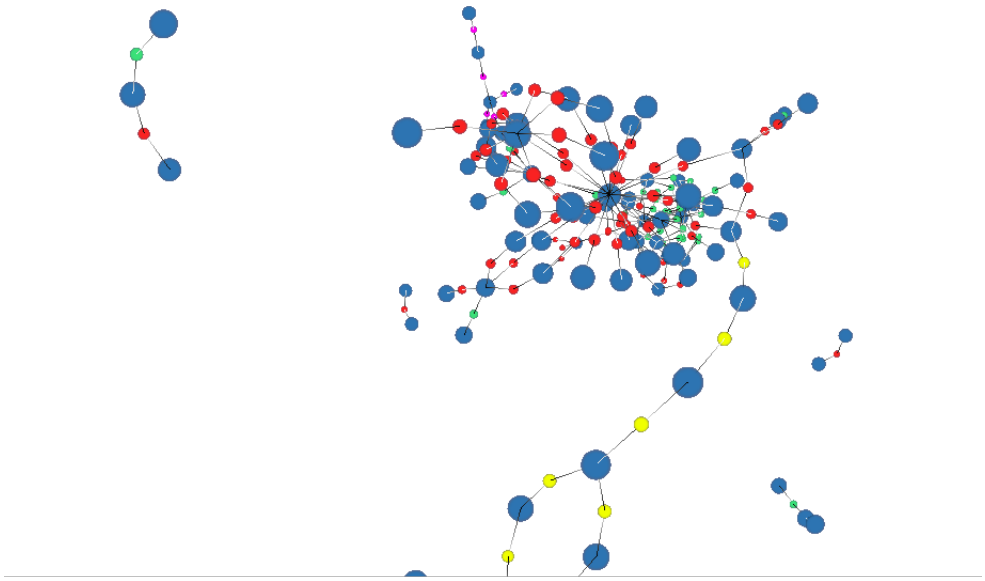
- Bigarren bertsioa.
  Bigarren bertsioan, liburutegia gehiago aztertuz, karpeta baten barruan dauden eremuek *name(newName)* funtzio bat dutela aurkitu da. Funtzio honetan eremuak izango duen izena zehaztu daiteke, beraz, eremuak gordez gero beraien izenak aldatu daitezke nahi beste aldiz. Aurreko bertsioa baino sinpleagoa da ez delako fitxategi bat behar hizkuntza bakoitzerako eta ez da interfazea ezabatu eta berriz eratu behar hizkuntzaz aldatzen den bakoitzean. Hala ere, multzoen izena ezin izan da aldatu ez dutelako honelako parametrorik. Beraz, multzoen izenak ingelesez idatzita egongo dira.

Diseinua gelditu

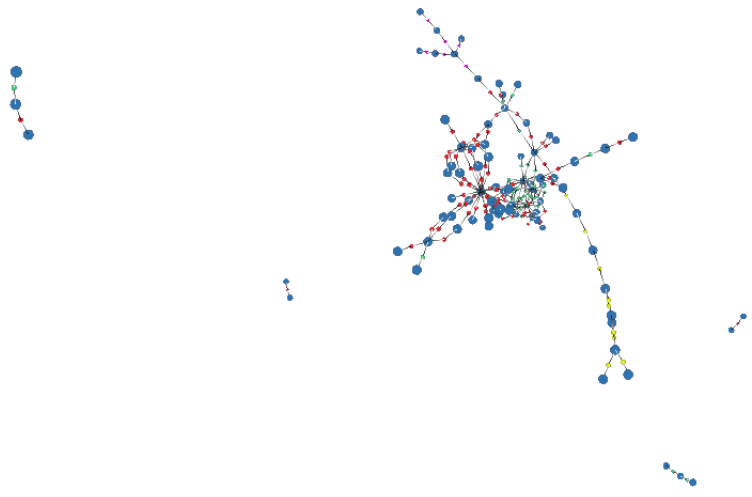
3.7 atalean azaldu den bezala, grafoaren diseinua denbora guztian hedatzen doan modelo bat da. 5.3 irudian grafoak hasieran duen itxura agertzen da eta 5.4 irudian, berriz, 10 segundo ondoren grafoak duen itxura. Nahi izanez gero, modelo hau gelditzea badago. Diseinua geldirik dagoen edo ez jakiteko, *ngraph.pixel* liburutegiak *isStable* izeneko aldagaia du eta honen balioa *true* bada, diseinua estatiko geratuko da, bestela, dinamikoa izango da. Aldagai honen balioa aldatzeko *stable(stableValue)* funtzioa erabiltzea dago. Hala ere, diseinua geldirik egoteak ez du baldintzatzen ingurunean zehar mugitzea.

3D/2D

Grafoaren bistaratzea 3D izan arren, horrek ez du esan nahi beti erosoa izango denik horrela lan egitea, grafoaren arabera 2D bistaratzera pasatzea komenigarriagoa izan daiteke. 5.5 eta 5.6 irudietan grafoaren 3D eta 2D bistaratzearen adibideak daude, bi irudietan kameraren posizioa berdina da. Grafoa 2D-n errenderizatzeak ez du ingurunea aldatuko, kamera askatasun berdinaz mugitu ahalko dugu. 2D itxurara pasatzean grafoko elementu guztiak Z ardatzaren balio berdinean kokatzen dira, hau da, beraien posizioa adierazteko X eta Y koordenatuak erabiltzen dituzte bakarrik, Z-ren koordenatu-balioa konstante bihurtuz.
Hau guztia egiteaz *renderer* parametroaren *layout* objektua arduratzen da. *Layout* objektua *pixel.layout* modulutik dator eta bertan *is3D* izeneko aldagai bat du kontrolatzeko ea



5.3 Irudia: Grafoaren diseinua errenderizazioaren hasieran



5.4 Irudia: Grafoaren diseinua errenderizatu eta 10 segundotara

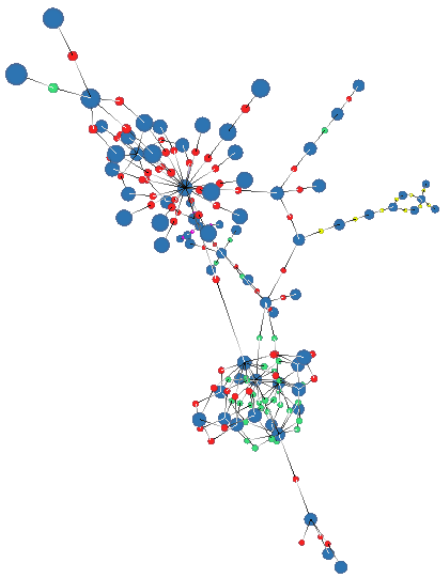
3D diseinua erabili behar den edo ez, 5.1 kode zatian ageri da nola aldatzen den diseinua batetik bestera. Aldagai honen arabera jakingo du diseinua noiz aldatu. Aldagai hau aldatzeko, modulu honetan inplementatuta dagoen *mode3d(newMode)* funtzioa erabiltzen da.

Atzealdea

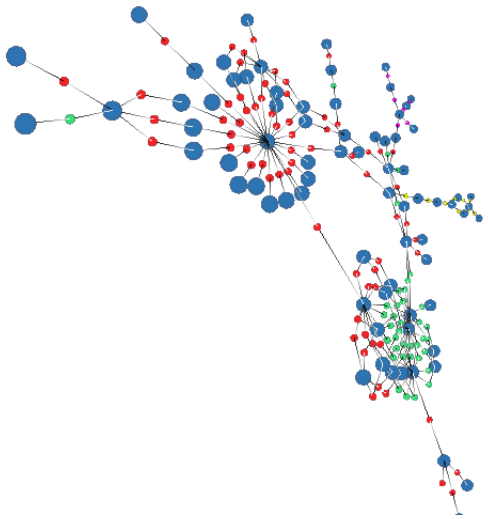
Atzealdea aldatzeko kolore panel bat egongo da. Kolorea zehazteko liburutegiak duen *clearColor* funtzioa erabiltzen da. Kolore lehenetsi bezala, #b4acdb kolorea ezarrita dago.

```
if (is3d) {
    layout = layout3d(graph, physics);
} else {
    layout = layout2d(graph, physics);
}
```

5.1 Kodea: 3D/2D errenderizazioa gaitu



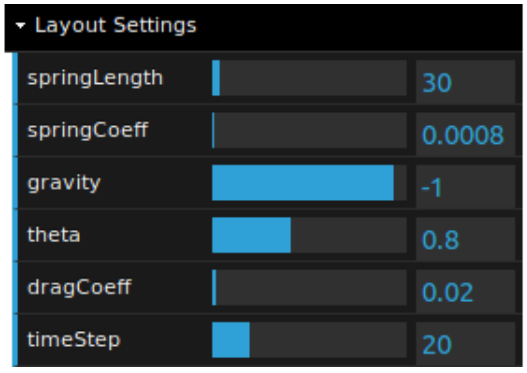
5.5 Irudia: Grafoa 3D-n bistaratua



5.6 Irudia: Grafoa 2D-n bistaratua

5.1.2 Layout Settings

3.7 atalean azaldu den bezala, grafoaren diseinua denborarekin hedatuz doa. Hori simulatzeko errenderizazioak modelo fisiko bat behar du. Atal honetan, simulazio fisikoaren parametroak aldatzeko aukera izango dugu. 5.7 irudian multzoaren egitura azaltzen da. Azaltzen diren balioak aplikazioak dituen balio lehenetsiak dira.



5.7 Irudia: *Layout Settings* multzoa

- *springLength*.
  Parametro honek ertzen luzera zehazten du. Balio lehenetsia, 30.

- *springCoeff*
  Hook-en legearen koefizientea. 0 eta 1 arteko balioak hartzen ditu. Gero eta hurbilago 1-etik orduan eta solidoagoak ertzak, gero eta urrunago orduan eta elastikoagoak izango dira. Balio lehenetsia, 0,0008.

- *gravity*
  Coulomb-en legearen koefizientea. Adabegiak beraien artean aldaratzeko erabiltzen da, beraz, bere balioa negatiboa izan beharko litzake. Positiboa baldin bada, adabegiak beraien artean erakarriko dira. Balio lehenetsia, -1,2.

- *theta*
  Barnes Hut-en simulazioaren theta koefizientea. 0 eta 1 arteko balioak hartzen ditu. Gero eta hurbilago egon 1 baliotik orduan eta adabegi gehiago izango ditu kontuan algoritmoak. Honek Barnes Hut simulazioa indarren 'brute-force' kalkulua bihurtzen du. Balio lehenetsia, 0,8.

- *dragCoeff*
  Herrestatze indarraren koefizientea. Diseinua geldotzeko erabiltzen da. 0 baliotik hurbilago egoteak diseinua estua ez izatea ekartzen du. Balio lehenetsia, 0,02.

- *timeStep*
  Indarrak integratzeko behar den denbora. Zenbat eta hurbilago 0-tik orduan eta geldoago hedatuko da grafoa. Balio lehenetsia, 20.

5.1.3 Current Node

Multzo honetan, klikatu dugun adabegiaren informazio ikusi ahalko dugu, ikus 5.8 irudia. Hori egiteko, 4.4 atalean azaldu diren gertaerak erabiliko dira, kasu honetan *nodeclick* gertaera. Aplikazioak *nodeclick* gertaera aktibatzen duenean *showNodeDetails(node)* funtzioari klikaturiko adabegia pasatzen dio. Adabegiaren *id* parametroaren bitartez rendererak adabegia bere osotasunean lortu dezake, 5.2 kodean ageri da zelan egiten den. Hurrengo ezaugarriak izango ditu interfazeak:

Izena

Bertan adabegiaren izena agertuko da. Eremu honen helburua izena agertzea besterik ez da, ez dauka izena aldatzeko aukerarik.

Kolorea

Aukeraturiko adabegiaren kolorea aldatu.

Tamaina

Aukeraturiko adabegiaren tamaina aldatu.

Geldirik

Aukeraturiko adabegia ingurunean finkatzeko aukera ematen du, hau da, geldirik mantentzeko. Defektuz ez da geldirik egongo.

Konfiantza

Adabegiaren konfiantza maila adierazten du.

Garrantzia

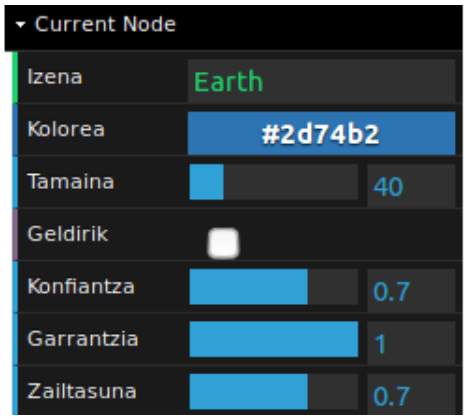
Adabegiaren garrantzia adierazten du.

Zailtasun maila

Adabegiaren zailtasun maila adierazten du.

```
renderer.on('nodeclick', showNodeDetails);
function showNodeDetails(node) {
    var nodeUI = renderer.getNode(node.id);
    nodeSettings.setUI(nodeUI);
}
```

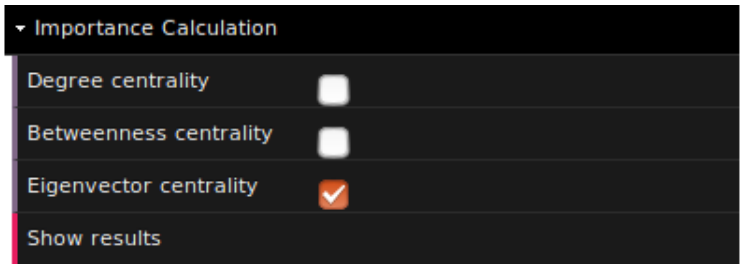
5.2 Kodea: *nodeclick* gertaeraren aktibazioaren tratamendua



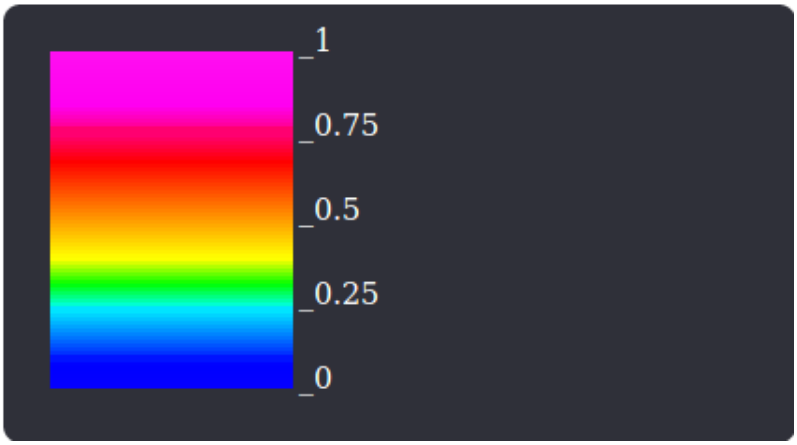
5.8 Irudia: *Current Node* multzoa

5.1.4 Importance Calculation

Ontologian dauden edukiak garrantzi desberdinak izan ditzakete. Hipotesi bezala, lotura asko dauzkatenak izan daitezke garrantzitsuenak eta horregatik zentralitatea erabiliko da hori neurtzeko ([Borgatti, 2005]), ikus 5.9 irudian multzoaren itxura. Grafoen analisian erabiltzen diren hiru neurri hauek aztertu ditugu: *degree centrality*, *betweenness centrality* eta *eigenvector centrality*. Kontuan izan behar da kalkuluak egiteko orduan, erlazioak ez direla kontuan izan, hau da, grafoa kontzeptuek bakarrik osatuko balute bezala erabili da kalkuluak errazteko asmoz. Analisiaren emaitza bistaratu ahal izateko emaitzak normalizatu egingo dira eta balio horren arabera kolore berri bat ezarriko zaio. Balio altuenek kolore beroak izango dituzte eta balio baxuek, berriz, kolore hotzak. 5.10 irudian koloreen eskala adierazten da. Erlazio motako adabegiak ez direnez kontuan hartzen #454545 kolorea, gris kolorea, ezarriko zaie ez dezaten koloreen bistaratzea nahasi.



5.9 Irudia: *Importance Calculation* multzoa



5.10 Irudia: Garrantzia adierazteko koloreen eskala

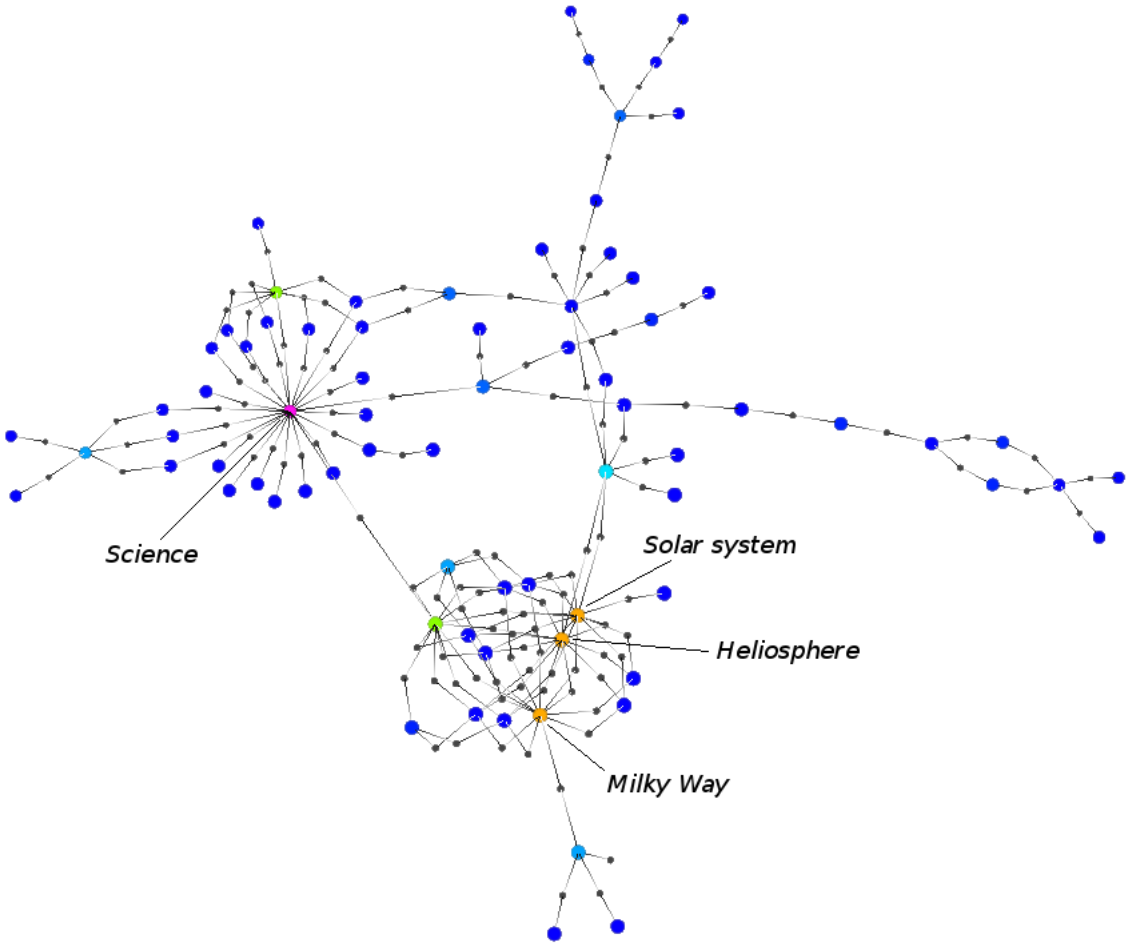
Degree centrality

Zentralitatea kalkulatzeko modurik sinpleena da, adabegi bakoitzak dituen ertzak kontuan hartzen dira bakarrik. Grafo zuzenduetan beste bi modutan kalkulatu daiteke, alde batetik sartzen diren ertzak eta beste aldetik, irteten diren ertzak. Gure grafoaren kasurako bakarrik sarrerako ertzak hartuko ditugu kontuan, honek adabegi hedatuena zein den esango baitigu.
Kalkulua egiteko eratu den algoritmoa O(V) ordenakoa da, zeren *relation* motako adabegietan badago adabegiak lotzen dituen informazioa eta beraz, ez dago zertan auzokidetza-matrizerik erabili beharrik.
5.11 irudiko grafoan *degree centrality*-ren emaitza ageri da. Emaitzei erreparatuz, ikus daiteke adabegirik hedatuena *Science* dela 1-eko garrantziarekin. Hurrengo adabegirik edatuenak *Solar System*, *Heliosphere* eta *Milky Way* dira, 0,47-ko garrantziarekin hirurak.

Betweenness centrality

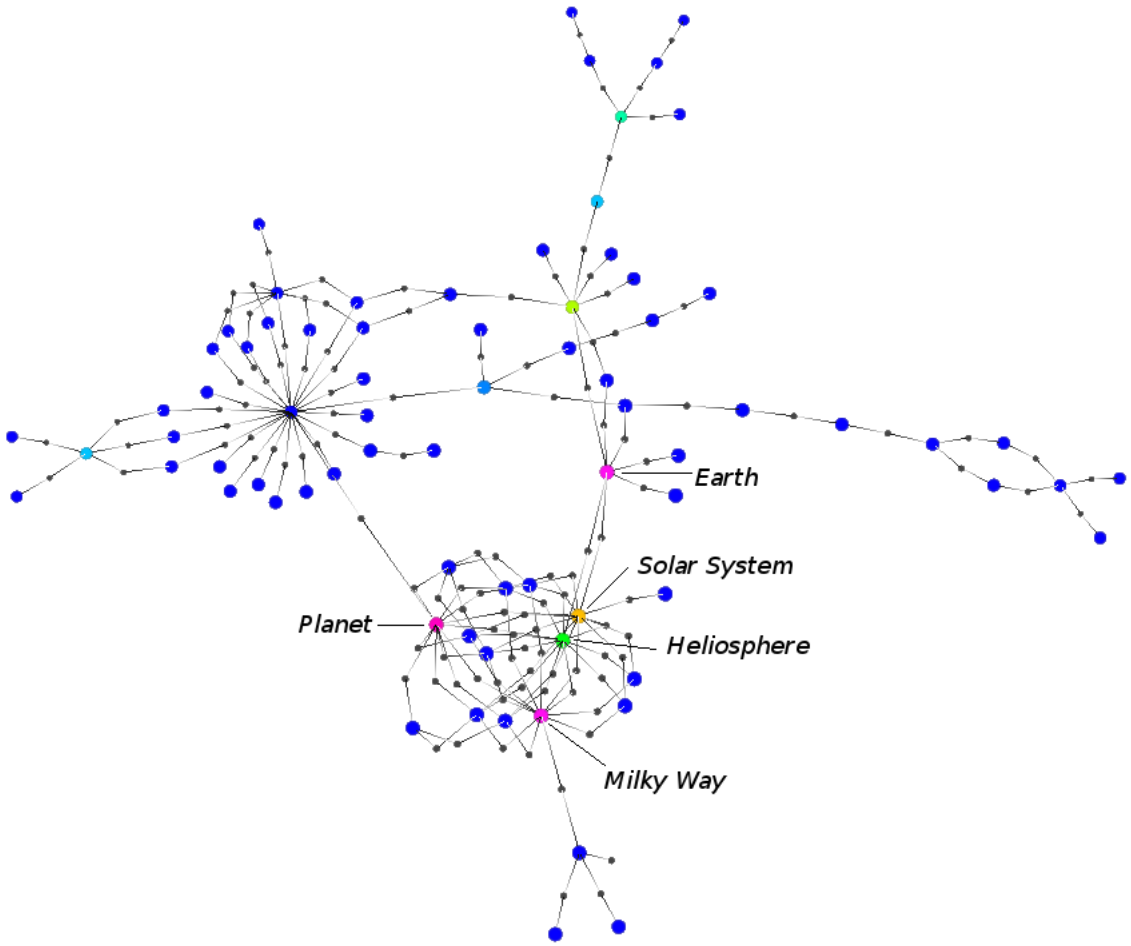
Betweenness ([Brandes, 2001]) neurriak kontuan hartzen du adabegi batek zenbat aldiz jarduten duen zubi bezala bi adabegiren arteko biderik laburrenean . Kalkulu hau egiteko Brandes-en algoritmoa erabili da, O(VE) ordenakoa eta (V + E) espazio behar du pisurik gabeko grafoetan non V grafoaren adabegiak diren eta E ertzak. Algoritmo oso arina da bere aurrekariekin konparatuz gero, O(V^3) ordenakoa eta (V^2) espazio behar dute. Kalkulua egiteko npm-n eskuragarri dagoen Betweenness[1] liburutegia erabili da.

[1] `https://www.npmjs.com/package/betweenness`



5.11 Irudia: *Degree centrality* analisiaren emaitza

5.12 irudiko grafoan *betweenness centrality*-ren emaitza ageri da. Emaitzei erreparatuz, ikus daiteke adabegirik zentralenak *Milky Way* eta *Earth* direla. Hurrengo adabegirik zentralenak *Planet*, *Solar System* eta *Heliosphere* dira 0,8, 0,45 eta 0,3-ko garrantziarekin, hurrenez hurren. Grafoaren egiturari erreparatuz, ikusi dezakegu lortu ditugun emaitzak egiturarekin bat datozela.

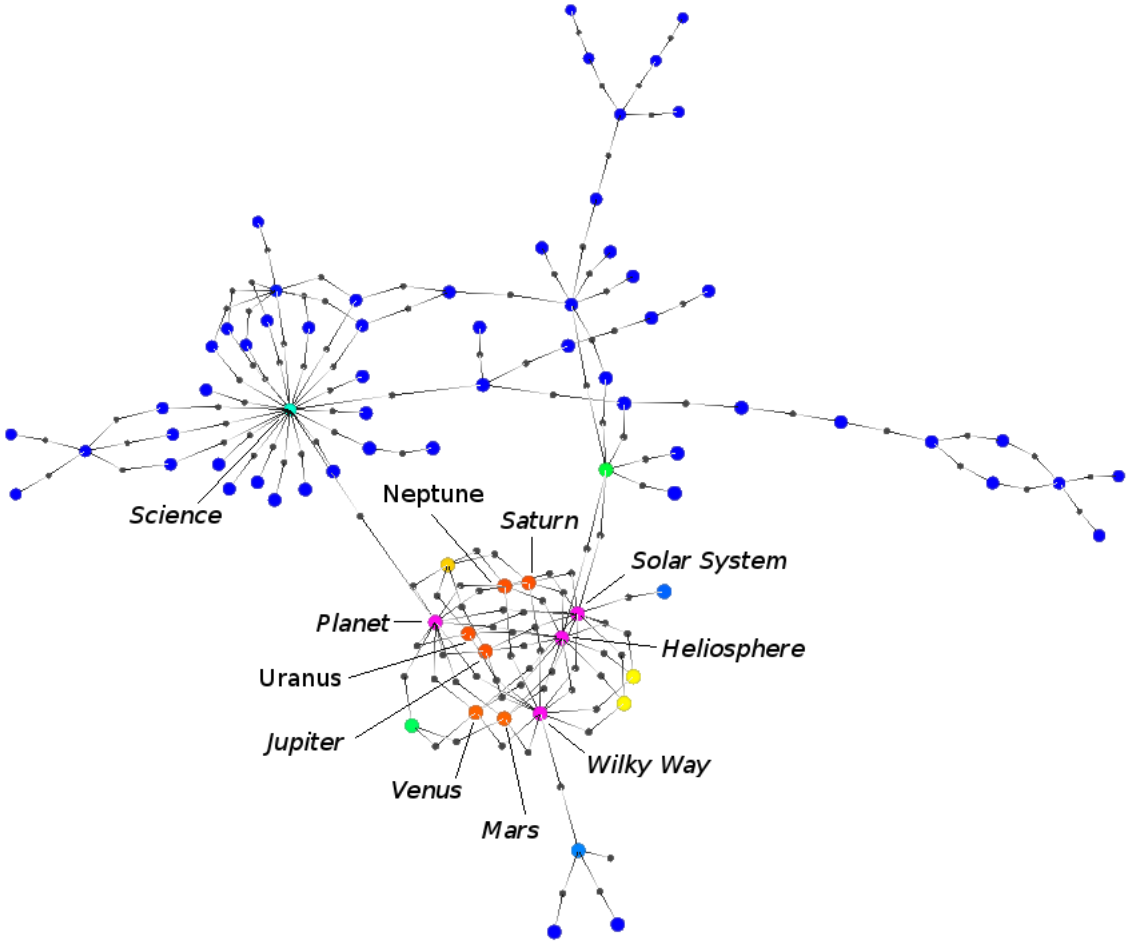


5.12 Irudia: *Betweenness centrality* analisiaren emaitza

Eigenvector centrality

Eigenvector neurriak, grafoan adabegiek duten eragina kalkulatzen du. Analisi honetan, balio altua dituzten adabegiek lotura asko dituzte eta aldi berean adabegi hauek ere lotura ugari dituzte ([Newman, 2006]). Analisi honetarako bektore eta balio propioak kalkulatu

behar dira, hau egiteko npm-n eskuragarri dagoen Networkjs[2] liburutegia erabili da eta algoritmo honen konplexutasuna O(V^3) ordenakoa da non V erpin kopurua den.
5.13 irudiko grafoan *eigenvector centrality*-ren emaitza ageri da. Emaitzei erreparatuz, ikus daiteke grafoan eragin gehien daukaten adabegiak *Solar System*, *Heliosphere*, *Milky Way* eta *Planet* direla. Hurrengo adabegi garrantzitsuenak *Venus*, *Mars*, *Jupiter*, *Saturn*, *Uranus* eta *Neptune* dira 0,55-eko garrantziarekin. *Degree centrality* atalarekin konparatuta, *Science* adabegiak 0,23-ko garrantzia dauka zeren erlazionatuta dagoen adabegiek ez dituzte erlazio erlazio asko.



5.13 Irudia: *Eigenvector centrality* analisiaren emaitza

[2] `https://www.npmjs.com/package/networkjs`

5.2 Bigarren panela

Bigarren panelean 4 multzo izango ditugu: *Add Node*, *Remove Node*, *New Relation* eta *Edit Relation*. Honetaz aparte, adabegien konfiantza mailaren balioarekin iragazketa bat egiteko aukera eta ontologiaren datuak deskargatzeko aukera egongo da.

5.2.1 Add Node

Multzo honetan bi adabegiren arteko erlazioak eratzeko aukera egongo da, multzoaren itxura 5.14 irudian ikus daiteke. Bi adabegiren artean erlazio berri bat sortzeko, bi adabegi berri edo lehendik sorturik zeudenak behar dira, ondoren, bi adabegi horiek lotuko dituen erlazio bat behar da. Eremu hauek izango ditu:

Nondik

Adabegiaren izena jarri beharko da eta erlazioa adabegi honetan hasiko da. Adabegiaren izena idatzi daiteke edota existitzen den adabegi bat klikatzearekin honen izena idatziko da eremuan.

Nora

Erlazioaren helburua den adabegiaren izena ezarri beharko da. Adabegiaren izena idatzi daiteke edota existitzen den adabegi bati klik bikoitzarekin honen izena idatziko da eremuan.

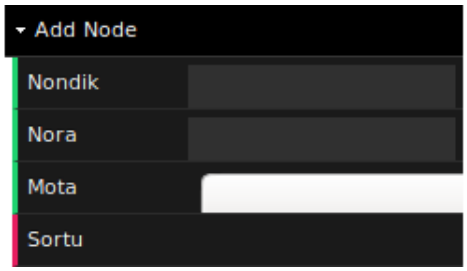
Mota

Erlazio mota adierazi behar da. Lista baten agertuko dira existitzen diren erlazioak.

Sortu

Botoi hau sakatzean beste eremuetan dagoen informazioaz baliatuko da adabegi berria eratzeko.

Sarturiko izena jadanik existitzen den adabegi bati badagokio, ez du adabegi berririk sortuko ezta honen datuak berridatziko ere. Adabegia ez baldin bada existitzen, hau eratzeaz aparte, honen konfiantza, garrantzia eta zailtasun mailaren informazioa 0.5 balio lehenetsiarekin hasieratuko dira. Kontuan izan behar da grafoak hizki larriak eta xeheak desberdintzen dituela, hau da, *Science* izeneko adabegi bat baldin badago, *science* izena sartzean beste adabegi berri bat bezala identifikatuko du eta berri bat sortu.



5.14 Irudia: *Add Node* multzoa

5.2.2 Remove Node

Multzo honetan bi adabegiren arteko erlazioak ezabatzeko aukera egongo da, interfazearen itxura 5.15 irudian ikus daiteke. Multzo honetan bai kontzeptu zein erlazio motako adabegiak ezabatzeko aukera dago. Erlazio motako bat ezabatzean, lotzen dituen bi kontzeptuak askatzen dira, hau da, ez daudela bata bestearekin erlazionatuta. Kontzeptu motako adabegi bat ezabatzean, ordea, adabegi hori ezabatzeaz gain, honi erlazionaturik dauden erlazio guztiak ezabatuko dira, adabegi honekin jada ez baitaude erlazionatuta. Eremu hauek izango ditu:

Adabegiaren izena

Ezabatu behar den adabegiaren izena idatzi behar da. Idatzi den adabegia ez baldin bada existitzen, mezu bat pantailaratuko da hau esanez. Kontuan izan behar da, aurreko atalean azaldu den bezala, grafoak letra larriak eta xeheak desberdintzen dituela. Adabegiaren izena idatzi daiteke edota existitzen den adabegi bat klikatzearekin honen izena idatziko da eremuan.

Ezabatu

Botoia sakatuz zehaztutako adabegia ezabatuko da.

5.15 Irudia: *Remove Node* multzo

5.2.3 New Relation

Multzo honetan erlazioak gehitzeko aukera egongo da, 5.16 irudian ageri den interfazea erabiliz. Erlazio berri bat sortuz adabegi berriei erlazio hau zehaztu ahalko zaie eta erlazioen legendan gehitua izango da. Eremu hauek izango ditu:

Izena

Erlazio berria adieraziko duen izena.

Kolorea

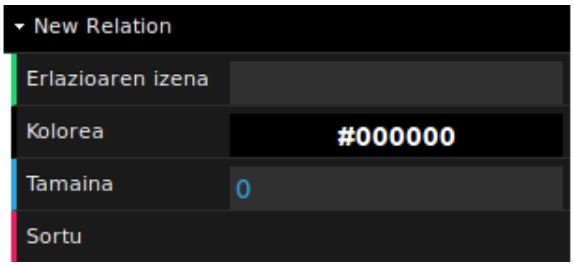
Erlazio honen adabegiek izango duten kolorea.

Tamaina

Erlazio honen adabegiek izango duten tamaina.

Sortu

Zehaztu den erlazioa gehitu egingo da aurreko parametroen balioak erabiliz.



5.16 Irudia: *New Relation* multzoa

5.2.4 Delete Relation

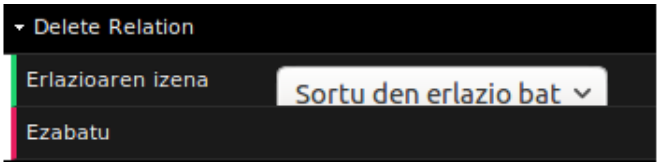
Multzo honetan sortu diren erlazioak ezabatzeko aukera egongo da, 5.17 irudian ageri den interfazea erabiliz. Eremu hauek izango ditu:

Erlazioaren izena

Ezabatu daitezkeen erlazioen lista agertuko da. Ezabatu daitezkeen erlazioen artean, aplikazioaren bitartez sortu diren erlazioak bakarrik ezabatu ahalko dira, hau da, *isA*, *partOf*, *prerequisite* eta *pedagogicallyClose* erlazioak ezin izango dira ezabatu zeren ontologia bat ondo osatzeko beharrezkoak dira.

Ezabatu

Aukeratu den erlazio ezabatuko da. Ez baldin badago erlaziorik aukeratuta mezu bat pantailaratuko da hau esanez. Aukeraturiko erlazioak ontologiako terminoren bat lotzen baldin badu, mezu bat pantailaratuko da esanez erlazio hau ezin dela ezabatu eta lehendabizi termino hauek ezabatu beharko direla erlazioa ezabatzeko.



5.17 Irudia: *Delete Relation* multzoa

5.2.5 Edit Relation

Multzo honetan erlazioak editatzeko aukera egongo da 5.18 irudian ageri den interfazea erabiliz. Sortu berri diren edota jadanik sortuta zeuden erlazioak editatzeaz gain, *concept* motako adabegiak aldatzea badago. Eremu hauek izango ditu.

Tamaina

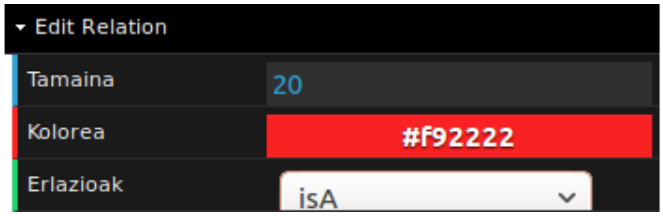
Aukeratu den erlazioaren adabegien tamaina aldatu.

Kolorea

Aukeratu den erlazioaren adabegien kolorea aldatu.

Erlazioak

Editatzeko dauden erlazioen lista.



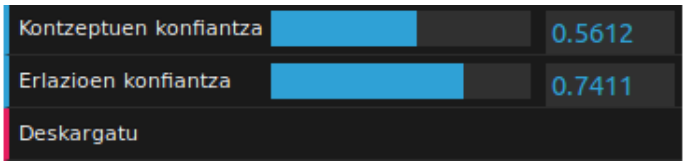
5.18 Irudia: *Edit Relation* multzoa

5.2.6 Kontzeptuen eta erlazioen konfiantza

Kontzeptuen eta erlazioen konfiantzarekin iragazketa bat egingo da. Iragazketa hori egiteko, adabegien konfiantza parametroa erabiliko da. Iragazketaren atalasea ezartzeko, 5.19 irudian agertzen den korritze-barren bidez egingo da.

Kontzeptuen konfiantza

Iragazketa hau egiteko, 'Kontzeptuen konfiantza' izeneko korritze-barra erabiliko da eta kontzeptu motako adabegiak bakarrik izango dira kontuan. Balio lehenetsia 0 izango da.



5.19 Irudia: Iragazketarako korritze-barrak eta deskarga botoia

Balioa aldatzean, atalase honen azpian dauden kontzeptuek aldaketa bat izan behar dute, adieraziz, adabegi hori atalasearen azpitik dagoela.

Erlazioen konfiantza

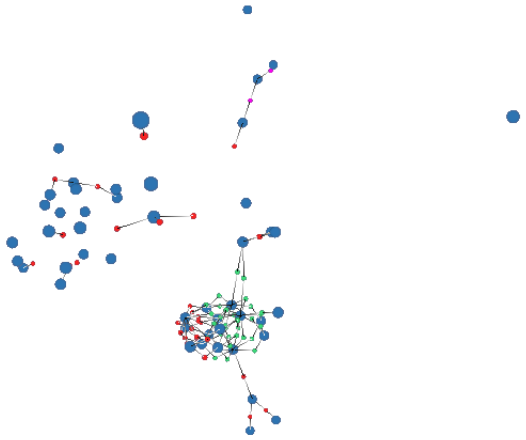
Iragazketa hau egiteko, 'Erlazioen konfiantza' izeneko korritze-barra erabiliko da eta erlazio motako adabegiak bakarrik izango dira kontuan. Balio lehenetsia 0 izango da. Balioa aldatzean atalase honen azpian dauden erlazioek aldaketa bat izan behar dute, adieraziz adabegi hori atalasearen azpitik dagoela.

Bertsio desberdinak

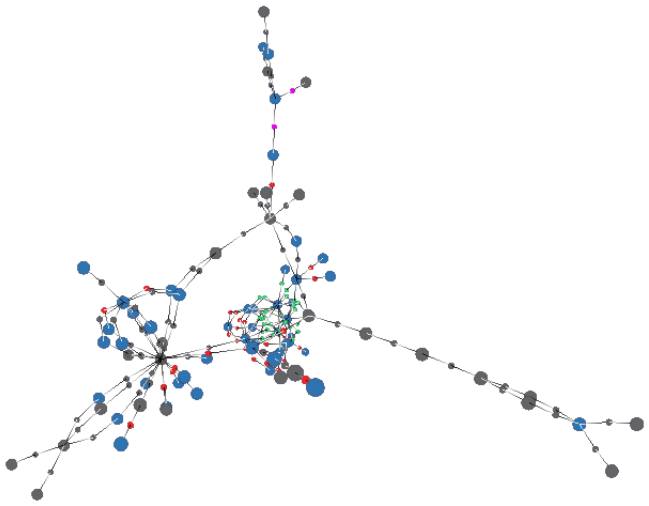
Adabegien aldaketak adierazteko bi bertsio inplementatu dira.

- Lehen bertsioa. Bertsio honetan, atalasea gainditzen ez zuten adabegiak ez errenderizatzea erabaki zen. Alde batetik, hori egin ahal izateko, grafoaren zati bat berriro errenderizatu behar da, ez da operazio oso pisutsua baina aldaketa asko jarraian eginez gero, 3D bistaratzean, antzeman da aplikazioa geldotzen dela. Beste aldetik, 5.20 irudian adabegiak eta ertzak bistaratzeko modua ez da bezeroak bilatzen zuena, zeren atalasea gainditzen ez duten adabegiak desagertu egiten dira. Beraz, arrazoi hauek direla medio, beste modu batean egitea erabaki da.

- Bigarren bertsioa. Bigarren bertsioan adabegien koloreak aldatzea bilatu zen. Hasiera batean, adabegien kolorea gardentzea erabaki zen, baina errenderizatzeko erabiltzen ari garen moduluak ez du gardentasunaren *alpha* balioa onartzen. Beraz, adabegien kolorea iluntzea aukeratu zen, kasu honetan '#646464' kolorea hain zuzen. 5.21 irudian azken bertsioaren eredu bat ageri da.



5.20 Irudia: Lehen bertsioa 0,6-ko atalasea ezarriz



5.21 Irudia: Bigarren bertsioa 0,6-ko atalasea ezarriz

5.2.7 Ontologia JSON fitxategian gorde

Aukera hau erabiliz, ontologian egin diren aldaketak ordenagailu pertsonalean gorde ahalko ditugu, JSON fitxategi batean hain zuzen. 5.19 irudian agertzen den deskarga botoia sakatzean, deskargatuko den JSON fitxategiaren egitura sortuko da. Egitura horrek 3 parametro nagusi izango ditu: adabegien lista, erlazioen datuak eta hizkuntza.

Adabegien lista

Adabegien lista eratzean, kontuan izan behar da aurreko ataleko korritze-barrek duten balioa. Ezarritako atalasea gainditzen duten adabegiak bakarrik sartuko ditugu listan, adibidez, 0,7-ko atalasea ezartzen baldin badugu, balio hori baino gutxiago duten adabegiak ez dira listan egongo. Kontuan izan behar da, baita ere, erlazio bat listan sartzeko, atalasea gainditzeaz gain, erlazioa osatzen duten bi kontzeptuek ere atalasea gainditu behar dutela. 5.19 irudia adibide bezala erabiliz, kontzeptuen atalasea 0,5612 balioan ezarrita dago eta erlazioena, berriz, 0,7411 balioan. *Milky Way* eta *Galaxy* kontzeptuen konfiantza maila 0,55 eta 0,7 dira, hurrenez hurren, eta *Milky Way partOf Galaxy* erlazioaren konfiantza maila 0,8. Balio hauek izango bagenitu, erlazioa ezingo genuke adabegien listan sartu, nahiz eta berari ezarritako atalasea gainditu (0,8 > 0,7411), *Milky Way* kontzeptuak ez baitu zegokion atalasea gainditu ($0.55 \ngtr 0.5612$). Kasu honetan, listan, bakarrik *Galaxy* kontzeptua sartu ahalko genuke.
Arazo hau konpontzeko lehendabizi kontzeptu motako adabegiak sartuko ditugu listan eta ondoren erlazioak. Erlazioek dituzten kontzeptuak listan sarturik dauden edo ez jakiteko, *Map* motako datu-egituraz baliatuko gara. Kontzeptuak *Map* egituran baldin badaude listan sartuak izan dira. *Map* egitura erabiltzearen arrazoia lista bat denbora guztian korritu behar ez izatea izan da.
5.3 kodean kontzeptu motako egitura ageri da eta 5.4 kodean, berriz, erlazio motako adabegi batena.

```
        "confidence": 0.7262953,
        "importance": "0.556",
        "difficulty": 0.7262953,
        "type": "concept",
        "kind": "concept",
        "name": "Venus"
```

5.3 Kodea: Kontzeptu motako adabegi baten egitura

```
        "confidence": 0.7,
        "importance": 0.7,
        "difficulty": 0.7,
        "type": "relation",
        "kind": "partOf",
        "source": "Venus",
        "target": "Milky Way",
        "name": "Venus partOf Milky Way"
```

5.4 Kodea: Erlazio motako adabegi baten egitura

Erlazioen datuak

Parametro honetan erlazio mota eta hau adierazteko kolorea eta tamainaren datuak gordeko dira, horrela, 5.2.3 atalean gehitu diren erlazio berriak gordetzeko aukera ematen du. 5.5 kodean erlazioen datu lehenetsiak zelan gordetzen diren adierazten da.

Hizkuntza

Ontologia zein hizkuntzatan dagoen adierazten du. Fitxategia hurrengoan zabaltzean hizkuntza honekin zabalduko da.

```
"relations": {
"relationType": ["concept", "isA", "partOf", "prerequisite", "
pedagogicallyClose"],
"relationColor": ["0x2d74b2", "0xf92222", "0x3ae178", "0xf0ff00", "0
xff00f4"],
"relationSize": [40, 20, 20, 20, 20 ]
}
```

5.5 Kodea: Erlazioen datuak

6. KAPITULUA

Ondorioak eta etorkizuneko lana

6.1 Ondorioak

Behin proiektua bukatuta, proiektuaren laburpen bat eta proiektuan zehar izan diren gorabehera guztiak komentatuko dira. Lehenengo proiektu mailakoak eta ondoren, maila pertsonalekoak.

6.1.1 Laburpena

Proiektu honetan egin den lana laburbilduz, honako ataza hauek azpimarratuko genituzke:

- Grafoak bistaratzeko hainbat liburutegien analisia.
- Aukeratu den Ngraph liburutegiaren analisia eta hau erabiltzeko tresnak.
- Ngraph liburutegiarekin grafoak errenderizatzeko jarraitu beharreko pausuak.
- Errenderizatu den grafoaren edizio aukerak.

6.1.2 Proiektuko ondorioak

Proiektu honetan ontologia bat erabiliz, honen grafoa bistaratzeko eta editatzeko gai den web-aplikazio bat eratu da. Aplikazio eratu baino lehen, JavaScript lengoaian idatzitako

hainbat liburutegi aztertu dira. Nire sorpresarako, aztertu diren liburutegi guztiak kode librea zuten, espero nuen doakoa ez zen liburutegiren bat topatzea baina ez da horrela izan. Guztiak GitHub-en daude eta gehienek erabiltzaileen iradokizunak kontuan hartu eta aurkitzen dituzten erroreak konpontzen dituzte. Baita ere, erabiltzaileak animatzen dituzte gauza berriak inplementatzera eta egindako aldaketak mundu guztiarekin konpartitzera. Liburutegi batzuek komunitate handiak dituztenez, foroak sortzen dituzte beraien artean laguntzeko eta garatzaile batzuek beraien korreoa ematen dute beraiekin harremanetan jartzeko dudak kontsultatu ahal izateko.

Aztertu diren liburutegietatik, Ngraph liburutegia erabiltzea aukeratu da. Ngraph liburutegia erabiltzean hainbat erreminta erabiltzen ikasi dut. Ez nekien Node.js bezalako aplikaziorik zegoenik JavaScript zerbitzari aldean exekutatzeko, orain arte bezeroaren aldean bakarrik erabili baitugu. Npm plataformak JavaScript liburutegien kudeaketa asko erraztu dit, liburutegien eta beraien dependentzien instalazioa komando bakar batekin lortu daiteke eta gaur egun, JavaScript liburutegiak kudeatzeko gero eta gehiago erabiltzen da, hala ere mundu honetatik at, ez da oso ezaguna, nik behintzat orain arte ez nekien existitzen zenik. Beti pentsatu izan dut JavaScript web-aplikazio handiak egiteko ez zuela etorkizun handirik, bezeroari fitxategi kantitate handia pasatu beharko zitzaiolako hau exekutatu ahal izateko. Arazo hau ordea, Browserify liburutegia erabiliz saihestu daiteke, liburutegi honek fitxategi nagusiak dituen menpekotasun guztiak fitxategi bakarrean batzen baititu. Hala ere, metodo honek arazoak ekar ditzake liburutegi batzuekin, adibidez fitxategiak idazteko eta irakurtzeko erabiltzen diren liburutegi arruntak ez dira bateragarriak eta, bereziki, Browserify-rekin lan egiteko eginda dauden liburutegiak erabili behar dira hauen ordez. Orain arte JavaScript-ekin egin ditudan lanetan ataza txikiak eta sinpleak egin ditudanez, ez nuen uste aplikazio handietan erabilgarria izango zenik. Three.js liburutegia aztertzean lengoaia hau C++ edo Java lengoaiak bezain lengoai indartsua dela ikusi nuen. Liburutegi honetan dauden adibideen artean, Particle Love[1] aplikazio aurkitu dezakegu lengoaia honen indarraren adibide gisan.

Bestalde, adabegien garrantzia kalkulatzerakoan, grafoaren teorien arloan gauza asko ikasi dut. Uste nuen grafoei loturiko algoritmoei buruz jakin behar zen guztia karreran zehar jadanik ikusi genuela, baina ez da horrela izan. Gaur egun, grafoak arlo askotan erabili daitezke eta, ez bakarrik, informatikarekin loturiko arloetan. Adibidez, biologian grafo batek pertsonen populazio bat adierazi dezake eta adabegiek, berriz, indibiduoak. Ada-

[1] `http://particle-love.com/`

begien garrantziaren kalkuluak adierazi diezaguke, adibidez, zenbateko arriskua duen indibiduo horrek infektatua izateko, beraz grafoek aplikazio oso praktikoak izan ditzakete eguneroko bizitzan.

Azkenean, ez da proiektua ekainean entregatzea lortu espero zen bezala. Hala ere, aukera hau aprobetxatu da proiektua fintzeko eta proiektuari pare bat gauza gehitzeko. Proiektuaren amaiera ekainaren 23an izatea espero zen arren, honen amaiera 2 aste atzeratu da eta, beraz, irailerako entregatzeak ez du arazorik suposatuko.

6.1.3 Ondorio pertsonalak

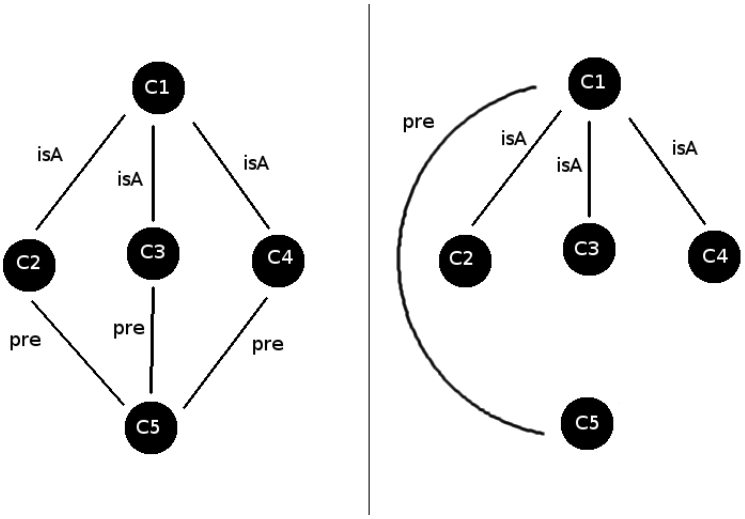
Ondorio pertsonalei dagokionez, proiektu hau garatu izana oso aberasgarria izan dela uste dut. Azken finean, ingurune birtual batean lan egiten aritu naiz, hori da niri gehien gustatzen zaidan eremua eta proiektu honekin interes hori handiagotzea eragin du. Proiektu honekin, konputazio arloko ikasgaiak erabiltzeaz aparte, software-ingeniaritzatik hurbil dauden irakasgaiak landu ditut.
Nire formakuntzari dagokionez, proiektu hau garatzen egon naizen bitartean aurkitu ditudan arazo eta oztopoak, lan-munduan aurki ditzakedanen antzerakoak izan daitezke. Beraz, etorkizunean enpresa edo beste nonbaiten lan egiteko prestakuntza bezala erabili dut. Baita ere, oso lagungarria iruditu zait proiektuan parte hartu duten irakasleek izan duten zuzendari (Ana Arruarte) eta bezero (Mikel Larrañaga) rola, eguneroko bizitzan sarri aurkitu daitekeen egoera bat da eta ahalik eta probetxu handiena ateratzen saiatu naiz. Proiektua erosoagoa egiteaz aparte, irakasleek proiektuaren hasieratik amaiera arte izan ditudan zalantzak argitu dizkidate, behar nuenean laguntza eskaini didate eta bilerak egiteko prest egon dira beti.
Laburbilduz, batzuetan aurrera egitea kostatu den arren, disfrutatu eta asko ikasi dut proiektua garatzen. Espero dut egindako esfortzua eta lana norbaitentzat erabilgarria izatea.

6.2 Etorkizunerako lanak

Proiektuaren hasierako helburuak beti diren arren, inplementatu daitezkeen hainbat puntu identifikatu dira, proiektuak amaiera bat izan behar duenez alde batera utzi behar izan direnak. Jarraian, aurrerago inplementatu daitezkeen atazak zerrendatzen dira:

- Aldagai batzuen balioak kalkulatzeko modu berriak.
  Kontzeptuak sailkatzeko beste metodoren bat inplementatu daiteke grafoen teoria erabiliz, *closenness centrality*, *Harmonic centrality* eta *Katz centrality* besteak beste.

- Erlazioen arteko patroiak identifikatu ontologia sinplifikatzeko helburuarekin.
  Erlazioen artean patroiak aurkitzen baldin baditugu, ontologiak dituen erlazio kopurua murriztea lortuko genuke. Adibide bezala 6.1 irudia daukagu, bertan, C1 kontzeptua, C2, C3 eta C4 kontzeptuez osatuta dago eta C5 hiru hauen aurre-baldintza da. Beraz, C1 kontzeptua, C2, C3 eta C4 kontzeptuez osatuta dagoenez, C5 C1-en aurre-baldintza da.



6.1 Irudia: Ontologia sinplifikatzeko modu bat

- Liburutegi berriak gehitu eta liburutegien artean aukeratzeko aukera eman.
  Liburutegi berriak gehitzeak grafoak bistaratzeko aukera gehiago eskaintzen ditu errenderizatzeko modu desberdinak emanez, horrela bistaratzea aberastuz.

Bibliografia

[Borgatti, 2005] Borgatti, S. P. Centrality and network flow. *Social Network*, Vol. 27, pp. 51-71, 2005.

[Brandes, 2001] Brandes, U. *A Faster Algorithm for Betweenness Centrality*, 2001.

[Conde et al., 2017] Angel Conde, Ana Arruarte, Mikel Larrañaga, Jon A. Elorriaga, Ruben Urizar. Testuliburuetatik domeinu-modulu eleaniztunak eraikitzen. *EKAIA*, ISSN 0214-9001, Vol. 31, pp. 117-132, 2017.

[Larrañaga et al., 2013] M. Larrañaga, A. Conde, I. Calvo, A. Arruarte, and J. A. Elorriaga. Ikaste-domeinuaren sorkuntza erdiautomatikoa. *EKAIA*, ISSN 0214-9001, Vol. 26, pp. 419-437, 2013.

[Newman, 2006] Newman, M. E. J. *The Mathematics of Networks*, 2006.