eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Informatika Ingeniaritzako Gradua

Software Ingeniaritza

Gradu Amaierako Lana

Software Produktu-Lerroen (SPL) kontzeptu-mapak sortzeko tresna laguntzailea

Egilea

Mikel Galarza Garcia

2020

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Informatika Ingeniaritzako Gradua

Software Ingeniaritza

Gradu Amaierako Lana

Software Produktu-Lerroen (SPL) kontzeptu-mapak sortzeko tresna laguntzailea

Egilea

Mikel Galarza Garcia

Zuzendaria

Arantza Irastorza

Laburpena

Gradu Amaierako proiektu honen helburu nagusia Software Produktu-Lerroen (SPL) inguruan lan egiten duten garatzaile berriei eskainiko zaien kontzeptu mapa egiten lagunduko duen aplikazioa sortzea da. Garatzaile berri hauek SPL-aren ezaugarri (arkitektura, domeinua, funtzionamendua, etab.) eta betekizunak aztertu behar dituzte, eta prozesu hau errazteko, kontzeptu mapa bat eskainiko zaie, eta aplikazio honek, kontzeptu mapa hau sortzeko eta diseinatzeko laguntza eskainiko du.

Horretarako anotazioen bidezko SPL-ak erabili dira oinarri gisa, Onekin ikerkuntza taldeak garatutako *WacLine* SPL-a esaterako. Garatu beharreko aplikazioak, *WacLine* proiektua arakatuko du, eta arakatze prozesu horretatik kontzeptu mapa eraikitzen lagunduko duen kontzeptuen bilduma emango dio, kontzeptu mapa hau diseinatuko duen pertsonari.

Aplikazioa bi zatitan banatzen dela esan genezake, lehena, arakatze eta aztertze prozesu osoa, non proiektuaren kode eta fitxategi guztien meatzaritza-lana egingo den eta bigarrena, behin arakatze edo meatze prozesu horren ostean lortutako kontzeptu bilduma antolatu eta bisualizatzen duena.

Memoria honetan aplikazioa garatzeko prozesu oso azaltzen da, hasierako proposamen eta plangintzatik amaierarako kalitate probatara.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

Esker onak

Gradu amaierako proiektu honen xehetasunak azaltzen hasi baino lehen, proiektu hau aurrera eramaten lagundu duten pertsonei eskerrak ematea gustatuko litzaidake:

- Arantza Irastorzari proiektu honetan lan egiteko aukera emateagatik. Gainera, proiektuaren garapenean zehar Gradu Amaierako Lana arrakastaz bukatzeko emandako aholku eta laguntzagatik.
- Iosu Salaberri ikaskideari, berak garatutako proiektuan oinarritu dudalako proiektu hau, eta bere proiekturik gabe ez litzatekeelako hau existituko.
- Familiako kide guztiei, hauek emandako laguntza eta euskarriagatik.
- Lagunei eta graduan zehar ezagututako jendeari, bereziki, pisukideei.

1. KAPITULUA

Sarrera eta Helburuak

Memoriaren lehen atal honetan, proiektuaren sarrera aurkeztuko da. Bertan, proiektu hau aurrera eramateko motibazioa azaldu, eta gainera, proiektu honek dituen helburuei buruz hitz egingo da.

1.1 Motibazioa

Gaur egun, softwarea tresna oso garrantzitsua da gure gizartearentzako. Hain da garrantzitsua, softwarerik gabe, gure egunerokoan egiten ditugun gauza asko ezingo genituzkeela egin. Gizartearen ikuspegitik, softwareak malgutasuna, adimena eta segurtasuna eskaintzen dizkie gure gizarteko funtsezko azpiegitura desberdinak onartzen eta kontrolatzen dituzten sistema konplexu guztiei: garraioa, komunikazioak, energia, industria, negozioak, gobernua, osasuna, entretenimendua, etab.

Softwarea oso zabalduta dago, baina honek arazoak ekar ditzake softwarearen garapenaren arloan. Izan ere, softwarea garatzen den mundua gero eta azkarrago doa, bai hutsetik hasitako software berria sortuz eta bai, jadanik garatutako software batean oinarrituz, hobekuntza edota aurrerapen ezberdinak garatuz.

Beste alde batetik, industria heldu batean, kostuen presioak handitzen jarraitzen du, eta epeak zaildu egiten dira. Eraginkortasuna hobetzea eta merkatura iristeko denbora murriztea biziraupen-kontua da industria horretako edozein erakunderentzat. Softwarearen garapenerako metodologia ezberdinak existitzen dira, eta orokorrena, garapenaren hasieratik amaierara aplikazio bakar bat emaitzatzat lortzen duena da. Ordea, existitzen da beste software garapen metodologia bat, ezaugarri komun batzuk konpartitzen dituzten hainbat aplikazio ezberdin emaitza moduan lortzen dituena, eta hau Software Produktu-Lerroak dira.

Software Produktu-Lerroen printzipioak hurrengoko ideian oinarritzen dira, merkatu edo eginkizun eremu jakin bat betetzen duten ezaugarri komunen multzoa partekatzen duten hainbat software sistema eraikitzen dira. Eta software desberdin horien antzekotasunak eta berezitasunak egitura bakar batean jasotzen dira.

Hala ere, ezaugarri asko eta asko konpartitzen dituzten aplikazioak garatzerakoan, proiektuaren eta SPL-aren egitura asko konplika daiteke, eta lehenago aipatutako kostu presioen inguruan, proiektuak garatzerako orduan, denbora kostua diru kostuan itzultzen da. Beraz garapen denbora hori murriztuen duen edozein tresna interesgarria izango da, edonongo software garapen

talderentzako. Eta hau da gure kasuan proiektua aurrera eramateko izan dugun motibazio nagusia. SPL-ak garatzea lan luzea izan ohi da, askotan urteak irauten duen lana, izan ere, ezaugarriak gehitu edota moldatu egiten dira. Horren ondorioz, lan-talde handiak eta aldakorrak izaten dira eta horregatik lan-talde hauetan sartu berriak diren garatzaileei laguntza eman behar zaie, orain arte eginda dagoena eta lan egiteko modua ulertzeko. Honen ondorioz, UPV/EHU-ko Onekin lan taldean kontzeptu mapak erabiltzearen proposamena lantzen ari dira.

Eta testuinguru horretan, Iosu Salaberriren “Software Produktu-Lerroen (SPL) arkitektura bistaratzeko aplikazioaren garapena” GrAL-a aurkeztu zen aurreko ikasturtean, SPL-aren domeinua eta funtzionamendua modu errazagoan azaltzen laguntzen zuen softwarea garatu zelarik.

Beraz, garatzaile berri horrentzako berria izango den garapen taldearen lan egiteko modura eta talde horretan garatzen ari diren aplikazioaren egitura, domeinua eta funtzionamendura ulertzeko prozesuan lan esfortzua era nabarmen batean murriztuko duelakoan garatuko da. Beraz, garapen talde honen Software Produktu-Lerro konplexua ulertzen lagunduko dion kontzeptu mapa batekin egingo du topo, eta ez zuzenean SPL-aren ezaugarri diagramarekin. Horrela, informazioaren zati bat kontzeptu maparekin erakutsi zen, eta hortik, kontzeptu mapa horiek sortzeko laguntzaren beharra ere agertu da, eta hemen kokatzen da lan honen oinarria.

Eta gradu amaierako lan honetan garatuko den aplikazioak, kontzeptu mapa hori diseinatzeko laguntza emango du. Hau da, kontzeptu mapa hau diseinatzen duen garatzaileari, kontzeptuak, garapen taldearen proiektuko kodetik eta dokumentaziotik abiatuz, adierazi eta aurkeztuko dizkio. Aplikazio honek prozesu hau nola egingo duen aurrerago azalduko da.

1.2 Helburuak

Proiektu honen helburu nagusia, garatzaile berriei eskainiko zaien kontzeptu mapa sortzen lagunduko duen aplikazio bat garatzea da. Zehazkiago esanda, SPL-aren kodea eta proiektuaren dokumentazioa arakatu egingo da, paketeen izenak, klaseen izenak, funtzioak, metodoak, aldagaiak… eta hauen testuingurua gordeko da, hots, proiektuaren zein tokitan aurkitzen diren. Eta hortik abiatuz, kontzeptuen eta hauen arteko erlazioen arabera lehen proposamena eskaintzen duen modulua garatuko da. Eta ondoren, modulu honen bitartez lortutako kontzeptu bilduma, SPL-aren garatzaile berriari modu egokian aurkeztuko dion bistaratze modulua garatuko da.

Bigarren mailako helburu bezala, software garatzaile moduan teknologia berrien inguruan ezagutza berriak jasotzea egongo litzateke. Gradu amaierako proiektu honetan garatuko den aplikazioa gauzatzeko, teknologia ezberdinak erabili behar izango dira eta teknologia berri hauen erabileraz, ezagutza zehatz batzuk bereganatuko dira.

Azkenik, bigarren mailako beste helburu bat da, hasieratik bukaerara maila honetako proiektu bat egiteko eta kudeatzeko gaitasuna erakustea da, batez ere bakarka. Ezaugarri zehatz batzuk izango dituen proiektu bat planifikatu, antolatu eta garatzeak agerian uzten du autosufizientzia eta lau urteko ikasketa-gaitasuna, lan-mundura igarotzeko ezaugarri behar beharrezkoa izango dena.

2. KAPITULUA

Proiektuaren kudeaketa-plana

Atal honetan, 1.2 atalean aurkeztutako helburuak arrakastaz lortzeko, osatu den kudeaketa-plana azalduko da. Hasieran, proiektu honen irismena jorratuko da: Lanaren Deskonposaketa Egitura, emangarriak eta mugarriak. Gainera, 2. atalean arrisku eta kalitate plana aurkeztuko da, arriskuak nola aurreikusi eta hauen aurrean nola jokatuko den azalduz.

2.1 Proiektuaren irismena

Plangintza egoki baten garapenerako, garrantzitsua da garatuko den proiektuaren irismena zuzen definitzea, eta proiektuak behar izango diren prozesuak definitzea, egin behar den lan guztia, behar bezala egiteko.

Azpiatal hau azaltzeko, lau zatitan banatuko da: Betekizunak, Lanaren Deskonposaketa Egitura, Emangarriak eta Mugarriak.

2.1.1 Betekizunak

Proiektu honek kontuan hartu beharreko hainbat betekizun ditu, eta atal honetan, puntuka azalduko dira.

- Kontzeptu bilduma sortzen duen moduluak *pure:variants*-en garatutako SPL-en kodea eta dokumentazioa arakatu beharko du, eta hau egiteko, Iosu Salaberrik sortutako *SPLMiner* softwarea oinarri bezala erabiliko da. Hala ere, *SPLMiner* kodearen egitura eta funtzionamendua proiektu honen helburuak lortzeko moldatu beharko da.

- *pure:variants*-en garatutako *WacLine* [7] SPL proiektuaren kontzeptu nagusiak lortu behar ditu eta formatu batean gorde, bai testu fitxategi edota datu-egitura batean. Erabiliko den SPL proiektua, Onekin taldean garatutako WacLine proiektua izango da.

- Kontzeptu nagusi hauek lortzeko, “Cluster” edo taldekatze eragiketa egiten duen kanpo liburutegi bat erabiliko da. Liburutegi honek terminoen antzekotasunaren arabera sailkatu eta taldekatuko ditu, emaitza moduan kontzeptuak eskainiz. Alabaina, ez da bilatzen kontzeptu bilduma honen zerrendatze optimoa, baizik eta, diseinatzaileari

kontzeptu mapa sortzen lagunduko dion kontzeptu bilduma.

- Kontzeptu bilduma aurkeztuko duen web moduluak, kontzeptu hauek argi eta era ordenatu batean adierazi beharko ditu, garrantzia handiago duen kontzeptua era esanguratsuago batean aurkeztuz. Gainera, kontzeptuak eta bakoitza osatzen duten terminoek duten lotura ere aurkeztuko da. Eta termino bakoitzaren jatorria ere, hau da, termino horiek proiektuaren zein ataletan eta zein fitxategitan agertzen diren adieraziko da, eta fitxategi hauen kodea ere eskuragarri egongo da. Kontzeptu hauek, garrantziaren arabera ordenatuta egongo dira, garrantzitsuena listaren lehen posizioan egonda.

- Web modulu honek ere, kontzeptuak hodei egituran erakusteko aukera ere izango du, kontzeptuak era bisualago batean aurkezteko.

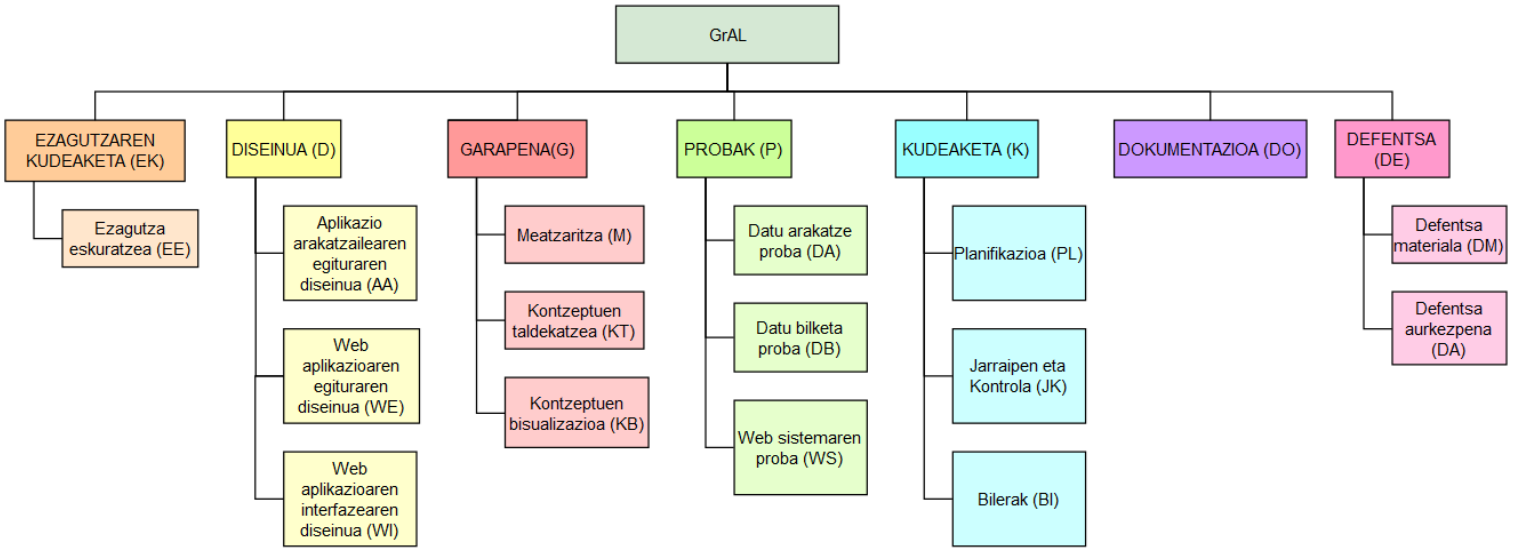
- Web modulu honetan agertuko diren kontzeptuen hizkuntza ingelesa izango da, azkenean softwarea garatzerako orduan kodean erabiltzen diren terminoak ingelesez erabiltzen dira eta. Beraz, hauekin zerikusia daukaten kontzeptuak ere ingelesez emango dira. Ordea, web modulu honetan nabigazioa egiteko, hiru hizkuntza aukeratu ahal izango dira, euskara, ingelesa eta gaztelania, hau da, web modulua eleaniztuna izango da.

- Web modulu honen interfaze grafikoaren detaile guztiak ez dira garatuko. Web modulua funtzional bat garatuko da, eta bertan informazioa modu argi eta ordenatu batean azalduko da, inolako dekorazio gehigarririk gabe.

2.2 Lanaren deskonposaketa

2.2.1 Lanaren Deskonposaketa Egitura

Ondoren lanaren deskonposaketaren egituraren (LDE) atalak xehetasunekin deskribatuko dira, 2.1 irudian beren diagrama erakusten da.



2.1 irudia: LDE diagrama

- **Ezagutzaren kudeaketa (EK):** Proiektuaren garapenean hasi aurretik, ezagutza bat lortu behar da, ezagutzen ez diren kontzeptuak eta teknologiak landuz. Kasu honetan, *Javascript* programazio lengoaiaren egitura ondo aztertu eta ulertu behar da, hau ondo arakatzeko. Beste alde batetik, proiektu oso baten fitxategi eta kodeak arakatzeko erabili behar den teknologia ere ulertu eta jorratu behar da, gero, aplikazio honetan era eraginkor batean aplikatzeko.

  Gainera, arakatze prozesu honen osteko *cluster* eragiketa egiteko erabili behar diren liburutegiak ere nola lan egiten duten ulertu eta ikasi beharko da.

  Beste alde batetik, oinarritzat hartu den *SPLMiner* proiektuaren kodea eta funtzionamendua ulertu eta barneratu beharra dago, gero software hau erabili ahal izateko.

  Ezagutzaren kudeaketarekin bukatzeko, *Spring Framework* erremintaren oinarrizko ezagutzak barneratu behar dira.

- **Diseinua (D):** Aplikazioaren egitura diseinatu beharra dago hau garatu baino lehen. Aurretik esan bezala, gure aplikazio hau bi modulutan banatuko da, eta horregatik lehenik aplikazio arakatzailearen egitura diseinatu behar da eta honen ostean kontzeptu bilduma aurkeztuko duen aplikazioaren egituraren eta interfazearen diseinua jorratu behar dira.

- **Garapena (G):** Garapenaren ataza hiru azpi-atazetan banatzen da. Lehenik, sarrerako proiektu osoa, goitik behera arakatu eta meatzari-lana egingo duen kodearen garapena egingo da. Meatzaritza-lan hau egiteko, *SPLMiner* proiektuaren kodea hartuko oinarritzat. Behin meatze prozesua eginda, terminoak kontzeptu baten inguruan biltzeko kodea garatuko da. Seguruenik prozesu hau aurrera eramateko, aurretik garatutako kanpoko liburutegiren bat erabiliko da. Eta azkenik, aurreko eragiketan lortutako kontzeptu bilduma hau aurkezten duen web aplikazioaren kodea jorratu eta garatuko da.

- **Probak (P):** Atal honetan, aurreko atazan garatutako kodea testatuko da, lortu nahi zen emaitza eta lortzen den emaitza berdinak diren egiaztatzeko. Funtzioak era independente batean funtzionatzeaz gain, azken produktu moduan ere nola funtzionatzen duten probatuko da.

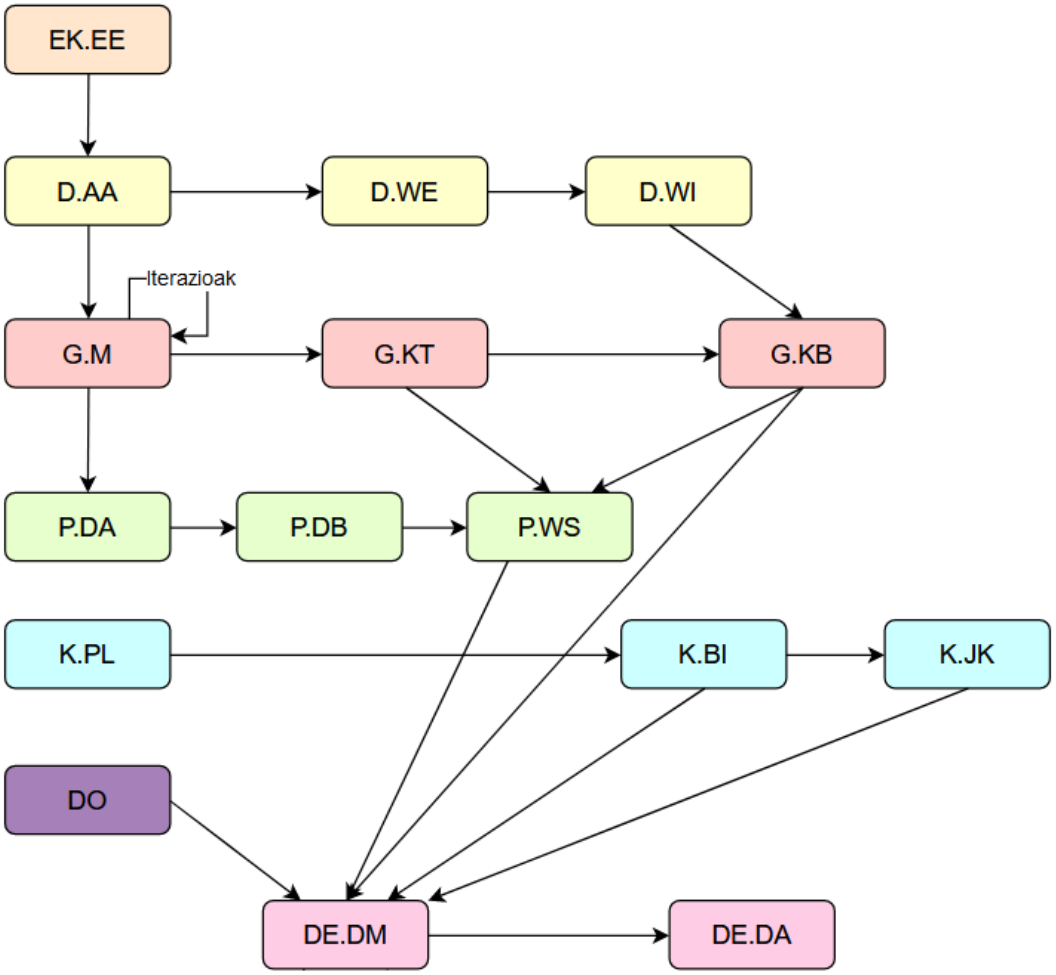
- **Kudeaketa (K):** Proiektua garatzen doan bitartean, emaitza arrakastatsu bat bermatzeko kudeaketa izango da. Horretarako, hasierako plangintza sendo bat ezarriko da proiektuaren oinarri gisa, eta ondoren, bileren eta jarraipen eta kontrolen bitartez, proiektuaren momentu oroko garapen egoera aztertu eta zainduko da. Hasieran ezarritako plangintza betetzen den analizatuko da eta helburua lortzeko pauso egokiak ematen ari garen aztertuko da.

- **Dokumentazioa (DO):** Ataza honetan proiektuaren memoria idatziko da, proiektuaren garapenaren inguruko ezaugarri eta xehetasunak aurkeztuz. Behin memoria idatzita proiektuko zuzendariari bidaliko zaio bere oniritzia jasotzen den arte.

- **Defentsa (DE):** Ataza honetan proiektuaren defentsa burutzeko prestakuntza egingo da.

Aipaturiko, ataza hauek ez dira modu lineal batean gauzatuko, haien artean menpekotasun batzuk existitzen direlako. Haien arteko menpekotasun hau azaltzeko, 2.2 irudia erabiliko da.

Ezin izango da ezer garatu garapen horren diseinua egin baino lehen. Eta edozer garatzen hasteko, garatu beharreko softwarea garatzeko erabiliko diren tresnen funtzionamendua eta hauen egitura ondo bereganatuta egon behar da. Bestalde, proben ataza dago, probak egin ahal izateko, testak garatuta dagoen software baten gainean egin behar izango direlako. Dokumentazioak ere garapen eta proben menpekotasuna izango du, aplikazioaren egitura eta ezaugarriak deskribatzeko hauek definituta egon behar baitira.



2.2 irudia: Atazen arteko menpekotasuna

2.2.2 Atazen iraupenaren balioespena

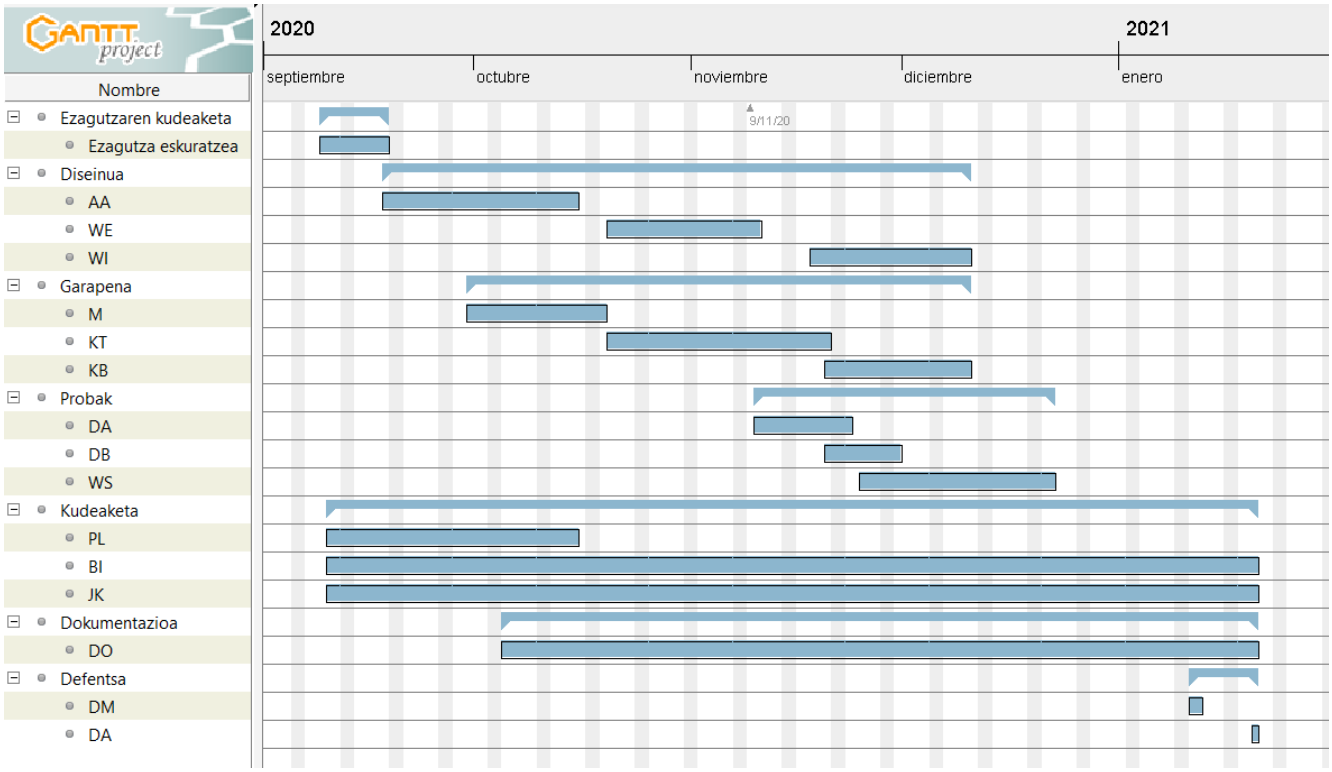
Ataza bakoitzak aurrera eramateko behar izango den denbora balioetsi da, eta 2.1 taulan laburbildu. Denbora hau ordutan emango da, baina ataza bakoitzaren hasiera- eta amaiera-data era aurkeztuko dira. Gainera, egutegian zehar egunetan izango duen iraupena ere azalduko da. Azkenik, taularen oinean, ataza guztien ordu kopuru osoa emango da, eta hau izango da proiektua egiteko balioetsiko diren ordu kopuru osoa.

Ataza	Balioespena (ordu)	Hasiera-data	Amaiera-data	Iraupena (egun)
Ezagutzaren kudeaketa	**30**	2020/09/09	2020/09/20	11
EE	30	2020/09/09	2020/09/20	11
Diseinua	**55**	2020/09/18	2020/12/10	70
AA	25	2020/09/18	2020/10/15	28
WE	15	2020/10/20	2020/11/10	20
WI	15	2020/11/18	2020/12/10	22
Garapena	**90**	2020/09/30	2020/12/10	70
M	30	2020/09/30	2020/10/19	20
KT	35	2020/10/20	2020/11/20	30
KB	25	2020/11/20	2020/12/10	20
Probak	**35**	2020/10/10	2020/12/20	71
DA	15	2020/10/10	2020/10/20	10
DB	10	2020/10/20	2020/10/30	10
WS	10	2020/11/25	2020/12/20	25
Kudeaketa	**35**	2020/09/10	2021/01/20	132
PL	20	2020/09/10	2020/10/15	35
BI	10	2020/09/10	2021/01/20	132
JK	5	2020/09/10	2021/01/20	132
Dokumentazioa	**90**	2020/10/5	2021/01/20	111
DO	90	2020/10/5	2021/01/20	111
Defentsa	**10**	2021/01/11	2021/01/20	3
DM	8	2021/01/11	2021/01/12	2
DA	2	2021/01/20	2021/01/20	1
Totala	**345**			**138**

2.1 taula: Atazen iraupenaren balioespen taula

Aurreikusten den ataza bakoitzaren amaiera-datak atzerapenak jasan ditzake, ataza bukatu baldin ez bada edo arazoak egon baldin badira, errekuperazio-tarteak ezarriko baitira.

Taula honetako informazioa, era bisualago batean adierazteko, 2.3 irudia erabiliko da, Gantt diagrama. Aste bakoitzean, gutxi gorabehera 20 ordu eskainiko zaizkio proiektu honi, eta lan egunak denak izango dira. Astean zehar 20 orduak ez baldin badira bete, asteburuak erabiliko dira sartu beharreko orduak errekuperatzeko. Eguberriei dagokionez, ez da aldaketa handirik egingo, lan plangintza berdinarekin jarraituko da, jai egunak salbu.



2.3 irudia: Gantt diagrama

2.2.3 Emangarriak

Proiektu honek hainbat emangarri ezberdin izango ditu, eta horiek bi talde nagusitan banatzen dira, bat proiektuaren garapenarekin zerikusia dutenak, eta bestea, proiektuaren garapenaren dokumentazioarekin zerikusia dutenak.

- Produktuarekin zerikusia dutenak
  - Meatze, arazketa eta biltze prozesuak egiten dituen moduluaren prototipoa. Emangarri honi, datu kudeaketa izena emango zaio.

  - Datu bilduma aurkeztuko duen web aplikazioaren bistaratze modulua. Emangarri honi, datu aurkezpena izena emango zaio.

- Proiektuaren garapenaren dokumentazioarekin zerikusia dutenak:
  - Proiektuaren helburu eta plangintza dokumentua: proiektuaren hasiera-hasieran idatzitako dokumentua non proiektu honen helburuak eta plangintza adosten diren.

  - Memoria: Proiektuaren eta honen garapenari buruz xehetasunak eta ezaugarriak gordetzen dituen dokumentua. Proiektuaren garapenaren bitartean garatzen joango den emangarria, ia modu paralelo batean.

  - Aurkezpena: Memorian biltzen den informazio garrantzitsuren laburpen bat, proiektu honen defentsan erabiliko dena euskarri gisa.

2.2.4 Mugarriak

Proiektuaren garapen arrakastatsua izateko, proiektu honen garapenaren prozesuan hainbat epe errespetatu behar dira. Hauek, garapen garbi eta seguruago bat lortzea bermatuko digute eta horrela ere garapenaren egoera kontrolpean izango dugu. Hau, hasieran definitzen diren baldintzak dira eta hainbat faktoreengatik aldatu daitezke. Bi mugarri ezberdin aurkezten dira, barne mugarriak eta kanpo mugarriak. Kanpo mugarriak ez ditugu guk ezartzen, guk data tarte batean egin behar dugu lan eta kanpo mugarri hauek data tarte honengatik definituta daude. Eta barne mugarriak aldiz, gure proiektuaren garapena kudeatzeko erabiltzen ditugun eta guk definitzen ditugun mugarriak dira, eta hauek, era batera edo bestera, kudeatzeko aukera existitzen da.

- Barne mugarriak:
    - Meatze prozesua bukatuta (2020/09/20): Sarrera moduan sartzen den proiektuaren arazketa eta meatze lana bukatuta. Hemen ez da emangarririk egongo.

    - Kontaketa prozesua bukatu (2020/10/04): Aldagai, funtzio, klase eta pakete guztien agerpen kopuruaren kontaketa eginda eta hauei gainera, garbiketa bat aplikatu. Hemen ez da emangarririk egongo.

    - Biltze prozesua bukatu (2020/10/25): Kontaketa egin ostean, termino horiek guztiak kontzeptu zabalago batean bitzen dituen softwarea garatuta. Eta bezeroari emaitza (datu kudeatzaile modulua) emangarria aurkeztuko zaio, honen iritzia ezagutzeko.

    - Web aplikazioa bukatu (2020/11/20): Termino multzoa aurkeztuko duen web aplikazioa bukatuta egun honetarako. Eta bezeroari emaitza (datu aurkezpen modulua) emangarria aurkeztuko zaio, honen iritzia ezagutzeko.

- Kanpo mugarriak:
    - Proiektua entregatzeko epea (2020/01/31): Egun honetarako proiektuaren garapena eta honen dokumentazioa entregatuta egon beharko dira.

    - Proiektua aurkezteko epea (2020/02/08-12): Egun honetan proiektuaren defentsa egingo da.

2.3 Arriskuen eta kalitatearen azterketa

2.3.1 Arriskuen analisia eta diseinua

Proiektu guztietan garrantzitsua da proiektua garatzeko prozesuan ager daitezkeen arriskuak kudeatzea. Horregatik, gradu amaierako proiektu honen arriskuak aztertu eta ahal den neurrian aurreikusi eta soluzio posibleak pentsatzea garrantzitsua da, inolako aparteko kasu batean irtenbideren bat izateko, edota nola jokatu behar den jakiteko. Hain zuzen ere, arrisku horiek azkar identifikatuz gero, haien kudeaketa hobetzeko, neurri handiagoan prebenitzeko eta ahalik eta modu eraginkorrenean arintzeko ahalmena handiagoa izango da.
Beraz, atal honetan arrisku posibleak eta bakoitzaren eragina, ondorioak eta prebentzioak erakutsiko dira.

- **Arazo teknikoak erabilitako softwarearekin**: Lan honetan garatu beharreko aplikazioa lehenago beste proiektu batean garatutako kodean oinarritzen da, eta honek kode horrekiko dependentzia dakar. Kode honek arazorik badauka, proiektu honetako aplikazioaren funtzionamenduan eragin zuzena izango du.

    - Probabilitatea: Ez da oso altua, baina eragina zuzena izan dezake. Aurreko kode horrek bere proba kasuak pasa zituen, baina beste funtzionalitate gehigarriak gehitzerako orduan arazoak ager daitezke.

    - Hartu beharreko neurriak: Arazo hauek ekiditeko, soilik aplikaziorako beharrezkoa den softwarea instalatuko da. Eta arazo honen aurrean har daiteken neurria, software horrek eman ahal dituen arazoak gure kabuz konpontzea edo kode hori ordezkatuko duen kodea garatzea izango litzateke. Garapen gehigarri horrek eragin dezakeen atzerapena saihesteko, asteetan zehar errekuperazio-tarteak ezarriko dira, behar izanez gero erabiltzeko.

- **Planifikazio okerra diseinatzea:** Proiektu guztien moduan, honek planifikazio bat exijitzen du, helburuak era arrakastatsu batean lortzeko. Gerta daiteke, planifikazio hau ondo diseinatuta ez egotea eta mugarriak, sartu beharreko orduak, atazak, etab. era okerrean aurreikustea. Honek eragin handia izan dezake proiektuaren garapen orokorrean, entregatu beharreko emangarriak epe barruan ez entregatzea edota proiektua arrakastaz ez bukatzea.

    - Probabilitatea: Arrisku honek probabilitate ertaina du. Oso zaila da planifikazio bikain bat egitea, beti aldatu behar izaten delako gauzaren bat.

    - Hartu beharreko neurriak: Arrisku hau ekiditeko, planifikazioa diseinatzeko orduan, erabaki sendo eta errealistak hartu behar dira, ataza bakoitzarentzako beharrezkoa den denbora eta edozein ezbehar ebazteko denbora aurreikusiz. Lagungarria izan daiteke ere ataza handiak ataza txikiagotan zatitzea, horrela, denbora tarteak eta mugarriak aukeratzeko orduan modu zehatzago batean jokatuko dugu. Eta gainera, sortutako ereduen kontrola eramango da, momentu bakoitzean nola goazen begiratzen joateko.

- **Ikasketaren desbiderapena:** Proiektu hau aurrera eramateko teknologia ezberdinak erabili behar dira, eta teknologia hauetatik batzuk ezagunak dira, baina badaude beste batzuk guztiz ezezagunak direnak. Teknologia berri hauek ezagutu eta ulertu behar dira, eta haiekin lan egiten ikasi. Gerta daiteke, ikasketa honetarako denbora gehiegi behar izatea da, hasiera batean planifikazioa diseinatzerako orduan teknologia honen zailtasun maila ezezaguna baita eta hau gutxietsi daiteke.

    - Probabilitatea: Hau gertatzeko probabilitatea ez da txikia, eta horregatik soluzio posible bat hurrengoa izango litzake.

    - Hartu beharreko neurriak: Ikasketa honetarako denbora gehiago behar izanez gero, beste ataza batzuen hasieratzea beranduagorako utzi eta ikasketa atazarentzako beharrezkoa den denbora eskaintzea, gutxienez oinarrizko funtzionamenduak ulertu eta jaso arte. Eta aurrerago aipatutako errekuperazio-tarteak erabili daitezke oinarrizko ulermena jaso arte.

- **Datuen galera:** Proiektuaren garapenaren igaroan datuak, softwarea, dokumentu lagungarriak eta bai dokumentazio dokumentuak galtzeko arriskua existitzen da. Arrisku hau edozein proiektutan ager daitekeen arriskurik orokorrena da, eta hala ere, askotan ez da aurreikusten.

    - Probabilitatea: Arrisku hau gertatzeko probabilitatea nahiko altua da, eta arrisku honek eragin oso kaltegarriak ekar ditzake proiektuari, momentura arte garatutako guztia galduz, eta hutsetik hasi behar izanez.

    - Hartu beharreko neurriak: Kasu honetan, arrisku hau ekiditeko, dokumentazio dokumentu eta dokumentu lagungarri guztien segurtasun-kopia guztiak D*ropBox*-en gordeko dira. Eta softwarearen segurtasun-kopia *Git* biltegi batean gordeko da eta *Github*-en bitartez egongo da atzigarri uneoro.

- **Teknologia okerra aukeratzea:** Aplikazioaren garapenean, hainbat teknologia eta tresna ezberdin aukeratu beharko dira, eta aukeraketa honetan huts egin daiteke. Teknologia okerra aukeratzeak eragin handiak izan ditzake proiektu honetan.

    - Probabilitatea: Arrisku honen probabilitatea ertaina da. Gutxi gorabehera erabiliko diren teknologien inguruan oinarrizko ideiak finkatuta badaudelako.

    - Hartu beharreko neurriak: Teknologia aukeratzeko orduan, beharrezkoa den denbora eskainiko zaio, hau modu estrategiko batean aztertu eta analizatzeko.

- **Izaera pertsonaleko arazoak:** Proiektua garatzen den bitartean, izaera pertsonaleko hainbat arazo gerta daitezke, eta gehiago bizi dugun pandemia sasoi honetan. Gerta daiteke, garapena denbora batez bertan behera utzi behar izatea osasun baldintzen ondorioz, eta honek zenbait ataza edota proiektu osoaren aurreikusitako epeak berandu betetzea ekar lezake.

    - Probabilitatea: Gaixorik jartzeko aukera handiagoa dago, eta arrisku honek proiektuaren garapenean eragin zuzena ekarriko luke. Beraz, arrisku honen probabilitatea handia da.

    - Hartu beharreko neurriak: Jendearekin harremanetan egotea gertatzen den momentuetan modu arduratsu batean jokatzea. Beti ere segurtasun baldintzak betetzen.

Hauetaz gain, badaude beste izaera pertsonaleko arrisku gehiago, baina arrisku hauen probabilitatea txikiagoa da. Arrisku hauek gertatzeko probabilitatea txikia izan arren, hauek aurreikusteko zailtasuna nahiko altua da eta horregatik horrelako arriskuren bat jasanez gero, planifikazio berrantolatu beharko litzateke.

2.3.2 Kalitate plana

Software produktu batek kalitate egokia duela ziurtatzeko, zenbait jarduera modu sistematikoan egitea beharrezkoa da; horrek proiektuaren hasieratik kalitatea planifikatzea eta horri buruz

ezarritako helburuak (1.2 azpiatala) lortzeko plana egitea eskatzen du. Behin hau guztia definituta, kalitate prozesu zehatzak aplikatu eta aurreikusitako prozesu horiek jarraitzen dutela egiaztatu beharra dago.
Proiektuaren garapena aurrera doan heinean, softwarearen fase desberdinetan zehar ebaluazioak egingo dira garatzen ari den softwareak aurretik proposatutako baldintzak betetzen dituen ala ez jakiteko. Gainera, ebaluazio hauek aldizka egingo dira, garatutako kode berria modu independentean testatuz eta proiektuaren errora gehitu ostean testatuz. Softwarea fase desberdinetan ebaluatzerakoan, neurri zuzentzaileak sortuko dira, proiektu osoa berme handienekin osatzen lagunduko dutenak. Ebaluazio horiek proiektuaren softwarearen historiaren parte diren txostenak sortu beharko dituzte, etorkizuneko proiektuetan berriro arazo berdinak topo ez egiteko.

Softwarearen kalitatea neurtzeko, oinarrizko helburuak lortu diren aztertuko da. Oinarrizko helburu hauek beteta, proiektuaren kalitatea hobetzeko, denbora gelditzen bada, oinarrizko helburu hauek gehiago garatuko dira. Honela, aplikazioari funtzionalitate hobeagoak gehitu zaizkio, emaitza osoago bat lortuz. Hauek, kalitatearen arabera, hiru mailatan sailkatuko ditugu: txarra (ez dira oinarrizko helburuak lortu), ona (oinarrizko helburuak lortu dira) eta oso ona (oinarrizko helburuak lortzeaz gain, garapen edota funtzionalitate hobeagoren bat lortu da). Maila bakoitzak, aurreko mailaren ezaugarriak ere beteko ditu. Hurrengo funtzionalitateak aztertuko dira:

- Meatzaritza-lana
  - Txarra: Ez da kode eta dokumentazio fitxategi guztien terminoen azterketa egiten.
  - Ona: Kode eta dokumentazio fitxategi guztien termino guztiak aztertzen dira.
  - Oso ona: Kodeko edo dokumentazioko terminoak bereizten ditu.

- *Cluster*-a
  - Txarra: Taldekatze prozesua ez da egiten hitzen antzekotasunaren arabera
  - Ona: Taldekatze prozesua terminoen antzekotasunaren arabera egiten da.
  - Oso ona: Taldekatze prozesua esanahiaren arabera egiten da.

- Kontzeptuen zerrendaketa
  - Txarra: Ez da kontzeptuen zerrendaketa argia egiten.
  - Ona: Kontzeptuen zerrendaketa argia egiten da. Eta kontzeptu bakoitzaren terminoen informazio desberdinak eskaintzen dira, izena, agerpen kopurua eta terminoa agertzen den fitxategien zerrenda.
  - Oso ona: Kontzeptu bakoitzak ezaugarri diagramako zer ezaugarrirekin zerikusia daukan azalduko da, eta gainera, termino bakoitza agertzen den fitxategien kodea ere atzigarri egongo da.

- Kontzeptuen hodeia
  - Txarra: Kontzeptu lista ez da hodei itxuran agertuko.
  - Ona: Kontzeptu lista hodei itxuran agertuko da. Bakoitzaren tamaina, jatorrizko proiektuan daukan pisuaren araberakoa izango da.
  - Oso ona: Kontzeptu bakoitzaren gainetik erakuslea pasatzerakoan, kontzeptu honen pisua eta beste kontzeptuekiko izango duen pisuaren ehunekoa azalduko da.

2.4 Informazio eta komunikazio sistemak

2.4.1 Informazio sistema

Gradu amaierako proiektu honen Informazio Sistema *DropBox*-en osatua dago. Modu honetan, fitxategi hauetan aldaketaren bat egiterakoan, edota karpeta honetan edozein fitxategi gehitzen denean, automatikoki hodeira igoko da.

Karpeta nagusia, *TFG* izena duen karpeta da, eta honen barruan hainbat fitxategi eta beste azpikarpeta gehiago aurki ditzakegu:

- *WacLine* [7] eta *InsideSPL* proiektuen kodea: proiektu bakoitzaren izen berdina duten karpetetan daude antolatuta, alegia, *WacLine* proiektuaren kodea "WacLine" deitzen den karpetan gordeta egongo da.

- Irudiak*:* Irudiak deitzen den beste karpeta bat agertzen da, eta bertan LDE diagramaren eta atazen arteko menpekotasunaren irudiak daude, eta hauekin batera ere, *Gant* diagramaren irudia, proiektuen klase eta fluxu diagramak, ezaugarri diagramak eta diseinurako erabiliko diren zenbait irudi.

- *GrAL-ak*: karpeta honetan, aurreko urteetan beste ikasle batzuek garatutakoak diren eta memoria hau sortzeko adibide gisa erabili diren memoriak biltzen dira.

- Beste alde batetik, *TFG* karpeta honetan, *Gant* diagramaren fitxategia aurki dezakegu, .gan formatua duena eta editagarria dena. Honekin batera, editatu daitezkeen testu dokumentuak .docx formatuan gordeko dira, eta behin hauek bukatuta, .pdf formatuko fitxategira esportatuko dira. Eta azkenik, orduen kontrola eramateko orduak.xlsx kalkulu orria aurki daiteke.

Aurreko hauek, garatzen ari garen proiektuaren dokumentazio dokumentuak dira, baina badago beste atal garrantzitsu bat, hots, garatuko den softwarea. Honen informazio sistema *Git* biltegian oinarritzen da. *Git* biltegi lokal bat sortuko da, kodean edozein aldaketa egiterakoan, aldaketak bertan gordetzeko. Honekin, segurtasun kopia izateaz gain, bertsio sistemaren kudeaketa izango dugu, eta edozein momentuan, lehenago gordetako edozein bertsiora jo dezakegu, hortik berriro abiatzeko edo edozein kontsulta egiteko. *Git* biltegiaz gain, *GitHub* plataforman gordeko da gure lokaleko biltegian gordetzen den bertsio kontrola. Git biltegi honen egitura hurrengokoa izango da; gure proiektuaren kodea, beste proiektu batzuen kodearekin batera aurki dezakegu, baina zehazki, "*CMap_creator_assistant*" izeneko karpetan dago. Eta bertan, gradu amaierako proiektu osoaren egitura oso dago, errotik hasita.

Modu honetan, segurtasun kopia izatez gain, biltegi lokalerako atzigarritasuna galduz gero, internet bidez atzigarri egongo da ere, kode eguneratua.

2.4.2 Komunikazio sistema

Proiektu honetan, komunikazio sistema nagusia, unibertsitateak eskaintzen duen web posta izango da. Tresna honen bitartez egingo dira komunikazio orokorrak zuzendarien eta ikaslearen artean. Hala ere, tresna hau ez da beti guztiz erabilgarria eta eraginkorra izango, batzuetan bilerak egin

beharko direlako, kontzeptu teknikoagoei buruz hitz egiteko edota ahozko harremana izateko. Honetarako, bi aukera daude, lehenengoa, aurrez aurreko bilerak egiteko, zuzendariaren bulegoan edo aurretik erreserbatutako klase edota laborategi batean egitea. Eta bigarren aukera, Covid19 pandemiaren ondorioz, gerta daiteke aurrez aurre ezin biltzea eta bilera hauek, online izan behar izatea, kasu honetan ere, unibertsitateak eskaintzen duen *BB Colaborate* tresnaren bitartez.

2.5 Interesatuak

Proiektuan aktiboki parte hartzen dutenak dira proiektuan interesa dutenak, edo proiektua gauzatzearen ondorioz haien interesak uki ditzaketenak. Hauek, pertsonak eta erakundeak dira, hala nola bezeroak, babesleak, erakunde exekutiboa eta proiektuan aktiboki parte hartzen duen publikoa. Halaber, proiektua gauzatzeak edo bukatzeak eragin positiboa edo negatiboa izan dezaketen interesak dituztenak ere, eta beste alde batetik, proiektuan eta haren emangarrietan eragina izan dezaketenak

Proiektu honetan, interesdun handiena proiektu hau garatuko duen pertsona izango da. Bere lana eta esfortzuaren arabera izango du arrakasta, edo ez, proiektuak, eta bera da interesatuena proiektu hau arrakastaz bukatzeko.

Beste alde batetik, proiektuaren zuzendaria den irakaslea, interesdun zuzena da. Izan ere, ikaslea gidatu behar du, honek proiektu arrakastatsu bat lortzeko, eta honek emango dizkio jarraibideak txarto egiten ari den gauzak aldatzeko eta ondo egiten ari den gauzak mantentzeko.

Onekin ikerketa taldea ere, proiektu honen interesduna da, garatuko den softwarea eraginkorra eta erabilgarria baldin bada, taldeko garatzaileek software hau erabiltzeko aukera izango dutelako. Era batean edo bestean, bezeroaren rola jokatzen dute, eta bezeroa beti da interesatu oso garrantzitsua proiektu batean.

Eta azkenik, gradu amaierako proiektu hau zuzendu eta kalifikatuko duen epaimahaia ere interesduna izango da, honek, proiektua aztertu eta kalifikatu egingo du eta.

3. KAPITULUA

Aurrekariak

Memoriaren kapitulu honetan, gradu amaierako proiektu honen testuingurua ulertzeko beharrezkoak diren zenbait gairi buruz hitz egiten da.

3.1 Software Produktu-Lerroak

Software Produktu-Lerroa (SPL) [32] ezaugarri-multzo komuna erabiliz, antzeko software-sistemen bilduma sortzeko, softwarearen ingeniaritzako metodo, tresna eta teknikei erreferentzia egiten die. Ezaugarri horiek, merkatu-segmentu jakin batean edo helburu jakin baten betekizun espezifikoak asetzen dituzte, eta aldez aurretik deskribatutako oinarrizko aktiboen multzo komun batetik abiatuta garatzen dira.

Software Produktu-Lerroak, garapen-paradigma bideragarri gisa azaleratzen ari dira, eta horri esker, enpresek hobekuntzak egin ditzakete merkaturatzeko denboran, produktibitatean, kalitatean eta beste negozio-gidari batzuetan. Software Produktu-Lerroaren ingeniaritzak ere aukera eman dezake merkatuan azkar sartzeko eta erantzun malgua izateko, eta gainera, pertsonalizazio masiborako gaitasuna ere ematen du.

Beraz, modu sinpleago batean esanda, SPL-en definizioa hurrengoa izango litzateke: merkatu eginkizun jakin bat beharrak asetzen dituen eta dagoeneko kudeatutako ezaugarri komunak partekatzen dituzten software sistemen familia.

Horregatik, garrantzitsua da hurrengokoa kontuan hartzea:

- **Domeinua.** SPL-aren irteeran aterako diren produktuak domeinu berekoak izango dira, SPL-ak testuinguru berean arazo zabal bati erantzuna ematen diolako, hainbat xehetasun desberdinekin.

- **Ezaugarriak.** Ezaugarri bat funtzionalitate batekin lotuta dagoen kontzeptua da. Ezaugarri zerrenda domeinua aztertuz ateratzen da, eta ez garatu nahi den produktu bakar bati begira. Horrela domeinuaren hainbat ezaugarri komun izango dira, baina beste batzuk aldakorrak. Aldakortasun mota desberdinekoak izan daitezke (hautazkoak, alternatiboa, etab.) eta beste ezaugarriekiko senidetasun- eta dependentzia-erlazioak izan ditzakete.

- **Produktu edo aldaerak.** Hauek SPL-aren emaitzak dira, zati amankomun eta

hautazkoak dituzten software produktuak. Aldaera bakoitza ezaugarrien hautaketa zehatz baten ondorioz sortzen da, eta ezaugarri hauek aldaketa puntu bezala ezagutzen dira.

- **Aktibo printzipalak.** Aktibo printzipalak SPL baten produktu bat baino gehiagotan erabilitako errekurtsoa edo artefaktua da.

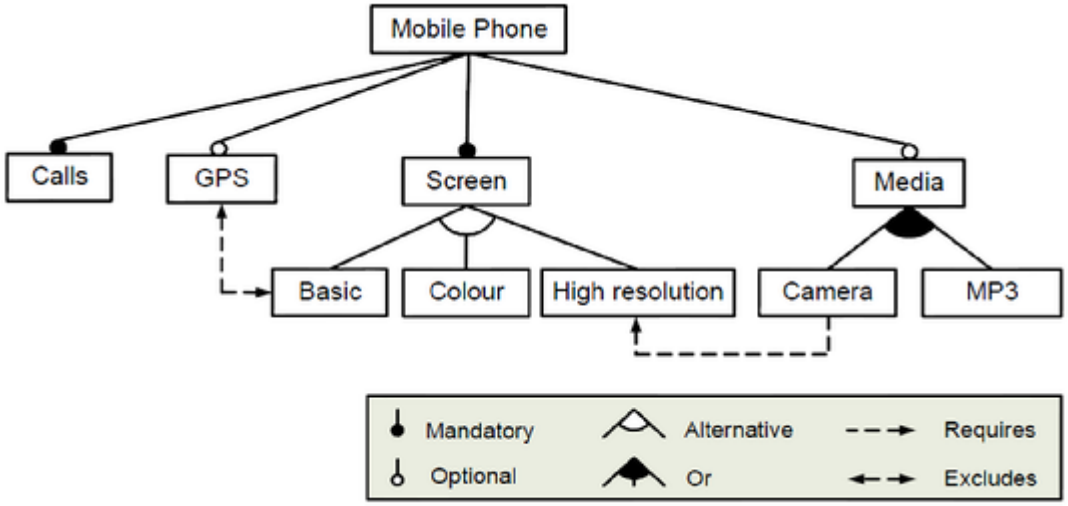
- **Produkzio plana.** SPL-aren produktuak sortzeko estrategia bat existitzen da. Eta estrategia hau, produktuak sortzeko, oinarrizkoa aktibo eta artefaktuen erabilpenaren deskribapena da, eta deskribapen honetan, produktu finalak sortzeko plana nola erabili behar den azaltzen da.

SPL baten adibide bezala, hurrengo 3.1 irudian ager daiteke. Ezaugarri diagraman horretan, Software Produktu-Lerroaren ezaugarri guztiak agertzen dira, eta ezaugarri diagrama horretan oinarrituz, produktu ezberdinak sortuko dira. Kasu honetan, ezaugarri diagramaren domeinua, telefono mugikorrarena izango da. Bertan, produktu honek izango ditzakeen ezaugarriak edo funtzionalitateak aurkezten dira, eta merkatu eginkizun jakin baten beharren arabera, ezaugarriak aukeratuko dira. Kasu honetan, adibide gisa, sor litezkeen hiru produktu desberdin aipatuko ditugu.

- Lehena, hurrengo ezaugarriak izango lituzkeenak: {*Calls*, *GPS*, *Screen*, *Basic*}. Ezaugarri diagraman ikus dezakegunez, telefono mugikorrak *GPS* ezaugarria edukiz gero, honen pantaila sinplea edo basikoa izan beharra dauka.

- Bigarrena, hurrengo ezaugarriak izango lituzkeenak: {*Calls*, *Screen*, *Colour*, *Media*, *MP3*}. Kasu honetan, deiak egiteko gai izango da, sortuko diren produktu guztiak bezala, koloreak dituen pantaila izango du, eta gainera, MP3 multimedia erreproduzitzeko aukera izango du.

- Hirugarrena, hurrengo ezaugarriak izango lituzkeenak: {*Calls*, *Screen*, *High resolution*}. Kasu honetan, produktua nahiko sinplea izango litzateke, izan ere, soilik, derrigorrez eduki behar dituen ezaugarriak dituelako, inolako aparteko ezaugarririk gabe.



3.1 irudia: SPL adibidea

Software Produktu-Lerroen inguruko kontzeptuekin bukatzeko, hurrengo bi kontzeptuak azalduko dira:

- **Domeinu Ingeniaritza.** Software sistema berriak sortzerako orduan, domeinuaren inguruko jakintza berrerabiltzen duen prozesua da. Kontzeptu guztiz nagusia da softwarearen berrerabilpenaren arloan, eta honen helburua, aplikazio partikular baten domeinuan dagoen softwarerako eta etorkizunean egongo denarentzako aplikagarriak diren software-osagaiak identifikatzea, eraikitzea, katalogatzea da. Eta, software domeinu hau, konkordantzia batzuk konpartitzen dituzten sistemak dituen eremua da.

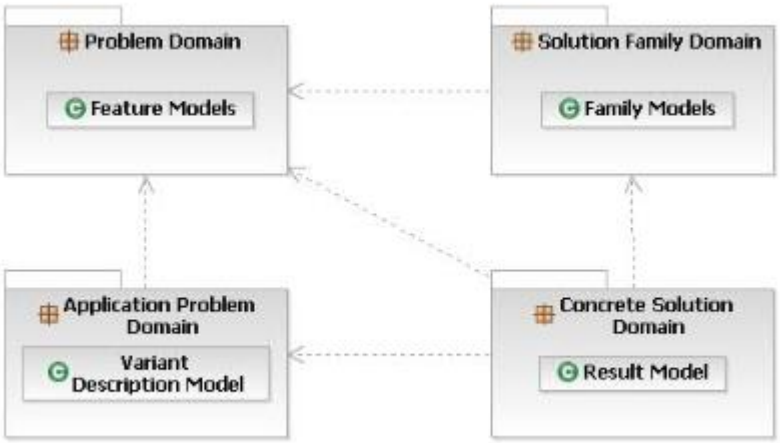
- **Aplikazio Ingeniaritza.** Aplikazioen softwarea garatzeko, planifikazio, diziplina eta metodologia desberdinak jarraitzen dituen teknika da.

3.2 Pure::variants

pure::variants-ek Software Produktu-Lerroak garatzeko prozesuan, fase bakoitzari laguntzeko tresna integratuen multzoa eskaintzen du. pure::variants beste tresna eta datu mota batzuekin integratzen den marko ireki gisa diseinatu da, hala nola, eskakizunak kudeatzeko sistemak, konfigurazio kudeaketa sistemak, kode sortzaileak, konpiladoreak, UML [18] edo DSL [12] deskribapenak, dokumentazioa, iturburu kodea, etab. [21].

Familian oinarritutako softwarearen garapenaren lau ardatzak eta pure::variants-en garapen hau egiteko oinarri gisa erabiltzen diren ereduak 3.2 irudian erakusten dira. Software Produktu Lerroen azpiegitura eraikitzerakoan, arazoaren domeinua hierarkizatutako Ezaugarri Ereduak (*Feature Model)* erabiliz irudikatzen da. Soluzio domeinua, hau da, software familiaren diseinu eta inplementazio zehatza, Familia Eredu (*Family Model)* gisa gauzatzen da.

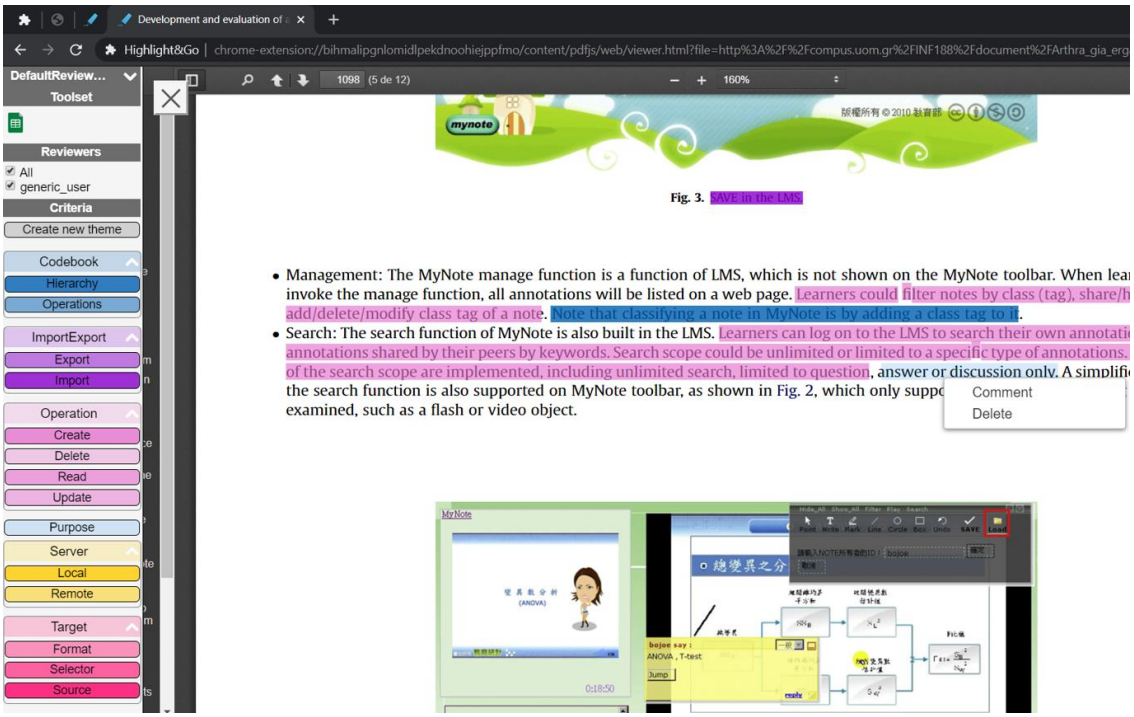
Aplikazio Ingeniaritzarako erabilitako bi modeloak, hau da, produktuaren aldaeren sorkuntza, goian deskribatutako bi ereduekin osagarriak dira. Aukeratutako ezaugarri multzoa eta lotutako balioak biltzen dituen Aldaeraren Deskribapen Ereduak (*Variant Description Model*), domeinuko arazoa, arazo bakarrean adierazten du. Eta Emaitza Aldakorraren Ereduak (*Variant Result Model*), soluzio-familiatik ateratako irtenbide zehatz bakarra deskribatzen du. Bestetik, Ezaugarri Ereduak (*Feature Models*) zuhaitz egitura dauka, eta bertan SPL-aren ezaugarriek zuhaitz honen nodoak osatzen dituzte. Eta azkenik, Familia Ereduan (*Family Model*), *FeatureModel*-eko ezaugarriak inplementatzeko behar diren osagaiak eta fitxategiak biltzen dira. *FamilyModel*, *FeatureModel* eta *VariantDescriptionModel* osagaiak, XML fitxategia dira, eta hauen egitura A eranskinean hobeto azaltzen da, pure::variants-ekin egindako SPL baten adibidearekin.



3.2 irudia: Familian oinarritutako softwarearen garapen ikuspegi orokorra

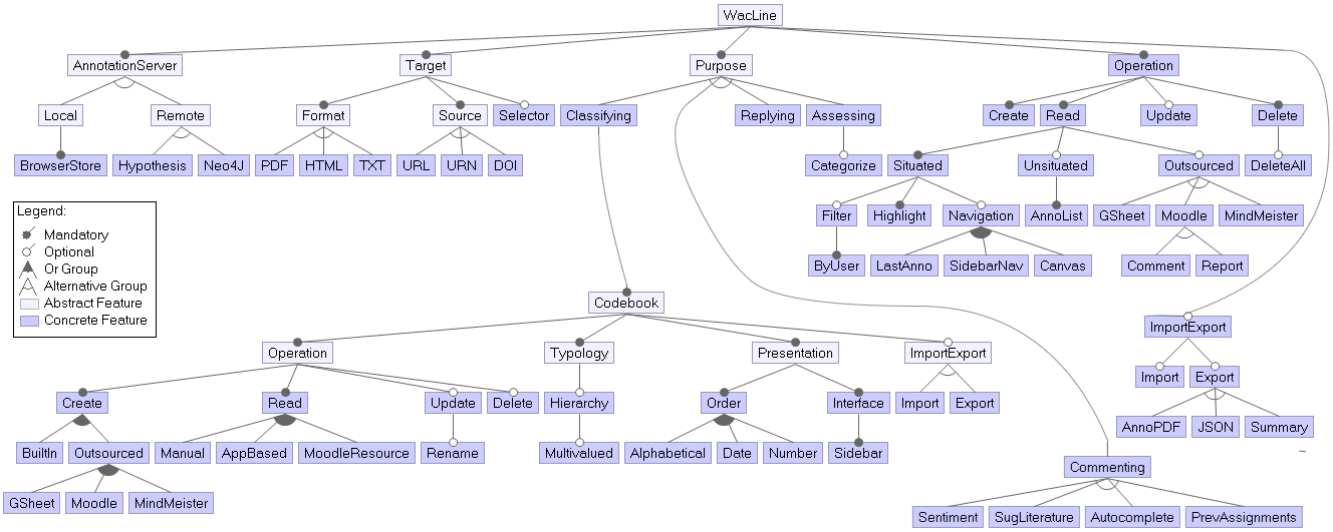
3.3 WacLine

WacLine [7] Donostiako Informatika Fakultateko Onekin ikerkuntza taldeak garatutako Software Produktu-Lerroa da, pure::variants softwarea erabilita, eta web anotazioen eremuko heterogeneotasuna kudeatzeko sortua da. Zehazki, WacLine tresnak pertsonalizatutako web oharpen bezeroak konfiguratzea eta automatikoki sortzea ahalbidetzen du, domeinu zehatzetan oharpen jarduerak egiteko. Sortutako oharpen bezeroak gaur egun *Chromium* [3] moduko web arakatzaileekin bateragarriak diren arakatzaile luzapenak dira (Google Chrome, Opera, …). 3.3 irudian honen adibide bat agertzen da, eta bertan, WacLine-etik sortutako produktu baten irudia erakusten da. *Highlight&Go* izena eman zaio, eta irudi honen ezkerrean, bezeroari eskaintzen zaizkion funtzionalitateak erakusten dira. Erdian, pdf fitxategi bati egindako oharpen batzuk, kolore desberdinetan, aurretik ezarritako irizpide desberdinak aplikatuz. Produktu honek, ikerketa dokumentuen irakurketa eta azterketa erraztea du helburu.



3.3 irudia: *Highlight&Go* produktuaren pantaila argazkia

WacLine SPL-aren ezaugarri eredua, 3.4 erakusten da.



3.4 irudia: WacLine-ren ezaugarri diagrama

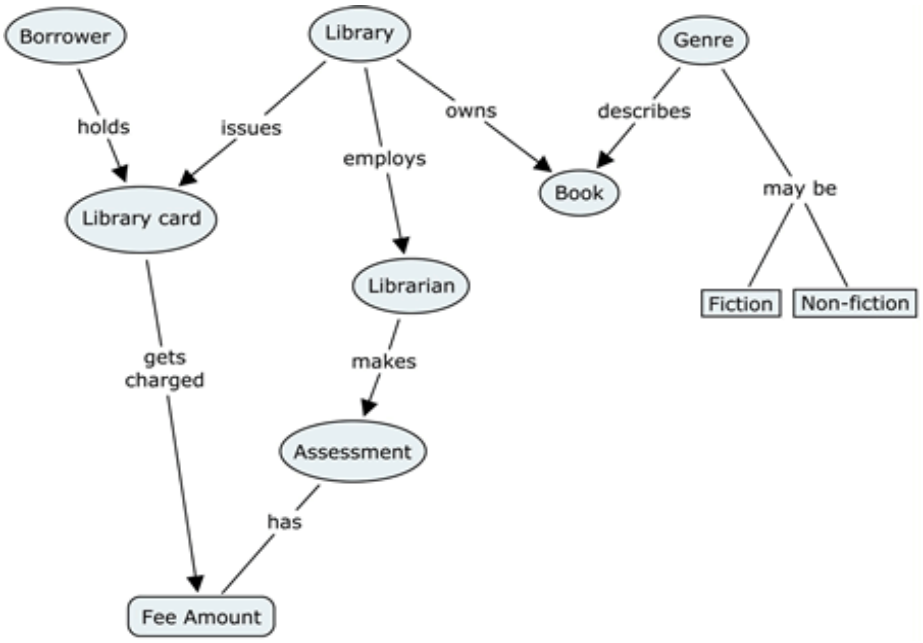
Ezaugarri diagrama honetan ikus daitekeenez, WacLine-eko produktu bakoitzak hainbat ezaugarri derrigorrezko izango ditu (diagraman puntu beltza duten marraz markatutakoak, esaterako, *Target*, *Purpose* eta *AnnotationServer*), hau da, produktu guztiek ezaugarri horiek izango dituzte beti. Beste ezaugarri batzuk, ordea, hautazkoak, haien artean ordezkoak edo haien artean osagarriak dira (puntu zuridun marraz, arku zuriaz edo arku beltzaz markatutakoak, hurrenez hurren); ezaugarri hauek garatzaileak erabakitako konfigurazioaren arabera azalduko dira. Hautazkoak, adibidez, *Delete* eta

Update dira, haien artean ordezkoak, esaterako, *Import* eta *Export* eta azkenik, haien artean osagarriak, adibidez, *Manual*, *AppBased* eta *MoodleResource*.

WacLine proiektua, Gradu Amaierako Proiektu honetan adibide bezala erabilitako SPL-a izan da.

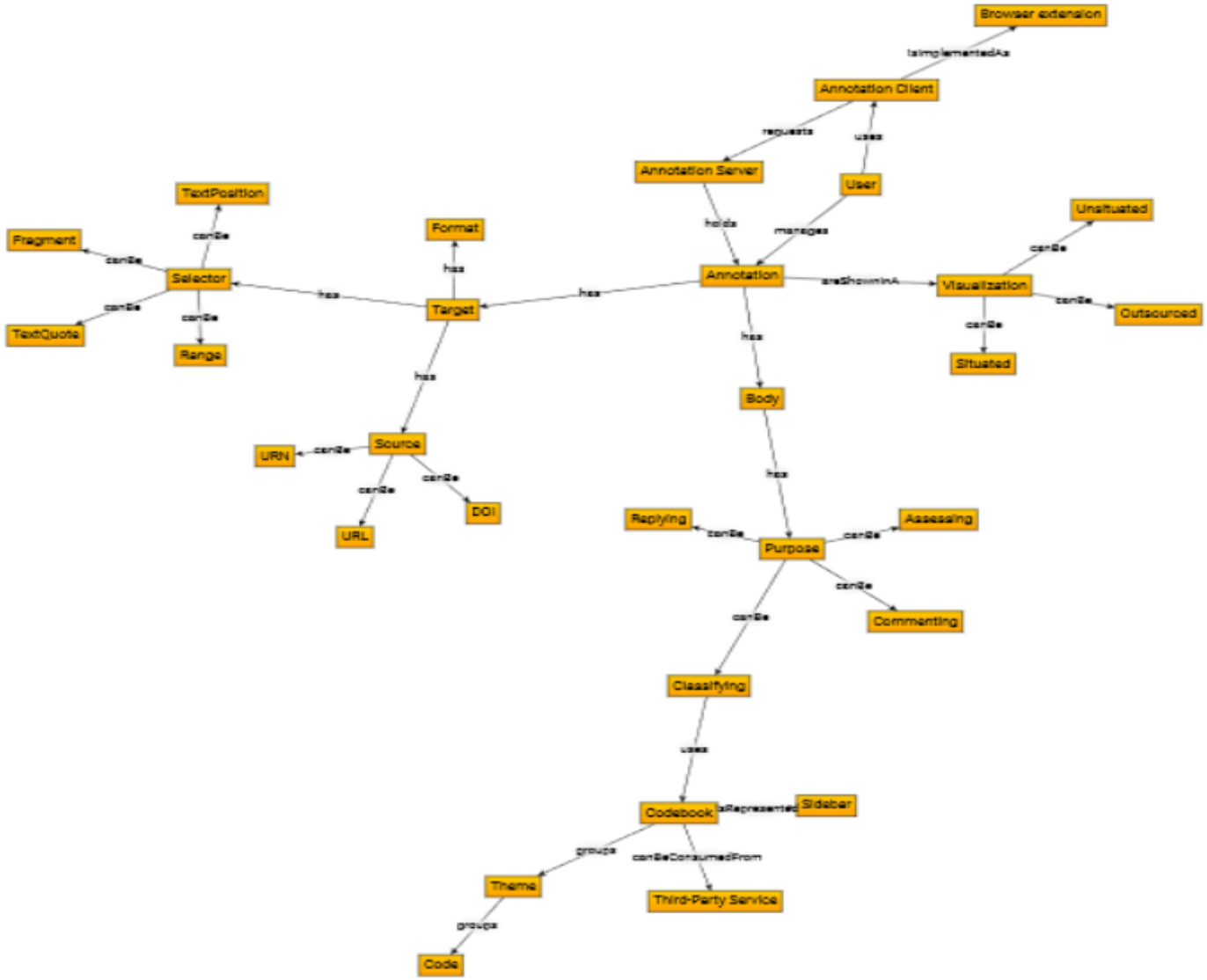
3.4 Kontzeptu mapak

Mapa kontzeptualak gai zehatz bat ulertzen laguntzen duten diagramak dira, ideien eta kontzeptuen arteko harremanak adieraziz. Ideiak hierarkikoki egituratutako nodoetan irudikatu ohi dira eta erlazioak azaltzeko lerroetan hitzen bat gehitzen da. Kontzeptu mapak hiru elementuk osatzen dituzte: kontzeptuak, horiek lotzeko geziak, eta lotura-hitzak. Azkeneko horiek kontzeptuen arteko erlazioak azaltzeko erabiltzen dira. Kontzeptu mapen adibide sinple bat, 3.5 irudian aurkezten da. Bertan, liburutegi baten egitura orokorra azaltzen da. Ikus dezakegunez, liburuak, hauen genero desberdinak, liburutegiko lankideak, liburuak eskatzen dituzten erabiltzaileak, etab. kontzeptu gisa agertzen dira, eta haien artean erlazio baten bodez lotzen dira.



3.5 irudia: Kontzeptu mapa adibidea

Gaur egun, kontzeptu mapak eskuz egiten dira, eta GrAL honen helburua, prozesu horretan laguntzeko tresna garatzea izan da. Garatutako aplikazio honek, WacLine SPL-aren domeinua azaltzen lagunduko duen kontzeptu mapa eraikitzen lagunduko du. 3.6 irudian, eskuz egindako WacLine SPL-aren kontzeptu mapa adierazten da, eta bertan WacLine proiektuaren kontzeptu nagusiak aurkezten dira, esaterako, *Annotation*, *Assessing*, *Commenting*, *Replying*, etab., eta kontzeptu hauek haien arten dituzten loturak ere, adibidez, *has*, *manages*, *requires*…



3.6 irudia: Kontzeptu mapa

4. KAPITULUA

Teknologiak

Aplikazio honen garapena aurrera eramateko software-tresna eta teknologia ezberdinekin lan egin da. Proiektua, Java interpretatutako programazio-lengoaian garatu da. Eta beste alde batetik, web aplikazioa garatzeko JavaScript eta HTML lengoaiak erabili dira, besteak beste. Atal honetan teknologia hauek zehatzago azalduko dira.

4.1 Software-tresnak

Softwarearekin lan egiteko hainbat tresna daude eskuragarri eta hurrengokoak aukeratu dira. Beste alde batetik, software hau osatzen duten diagramak, eskemak, irudiak, etab. sortzeko erabili diren tresnak aipatuko dira. Gainera, tresna bakoitzaren erreferentzia bibliografikoak esteka moduan agertzen dira.

Eclipse

Eclipse [9] kode irekiko software plataforma da. *IBM* enpresak garatu zuen hasiera batean, baina gaur egun *Eclipse* Fundazioak mantentzen du. Hainbat programazio lengoaietan aplikazioak garatzeko erabil daiteke (*Ada*, *C++*, *Java*, *Python*, etab.) eta *Plugin* sistema zabala du ingurunea pertsonalizatzeko. Analisi sintaktikoa egiten duen testu editore bat du eta konpilazioa denbora errealean egiten da, honi esker kodearen idazketa eta zuzenketa izugarri azkartu daiteke.

Eclipsek *Eclipse Public License* lizentzia dauka eta komunitate handi bat du atzetik tresnaren atalak hedatzeaz arduratzen dena. Proiektu honetan jorratu den kode gehiena tresna honek eskaintzen dituen funtzionalitateei esker egin da.

4.1 irudia: Eclipse

Git, GitHub

Git [27] bertsioak kontrolatzeko softwarea da, Linus Torvalds-ek diseinatua. Aplikazioen bertsioen mantentze-lanaren eraginkortasunean eta fidagarritasunean pentsatzen du, bertsio horiek iturburu-kodeen fitxategi asko dituztenean. Fitxategietako aldaketen erregistroa eramatea eta fitxategi partekatuetan pertsona ezberdinek egiten duten lana koordinatzea du helburu. Eta GitHub [25] ordea, edozein garatzaileren aplikazioen kodea ostatatzeko sortu zen, eta Microsoftek erosi zuen 2018ko ekainean. Plataforma horren bidez, garatzaileek beren aplikazio eta tresnen kodea igo dezakete, eta, erabiltzaile gisa, aplikazioa deskargatu ez ezik, haren profilera sartu ere egin daiteke, haren gainean irakurtzeko edo hura garatzen laguntzeko.

4.2 irudia: Git eta Github

Spring Framework

Spring Framework [10] kode irekiko esparrua da, Groovy, Kotlin eta Javan era guztietako aplikazioak sortzen laguntzen duena.

2003an merkaturatu zen, eta ordutik, errendimendu handikoa, arina eta berrerabilgarria den kodea sortzeko, Java [14] enpresa mailako aplikaziorik ezagunena bihurtu da. Izan ere, programazioaren ibilbidean sor daitezkeen arazoak estandarizatzea, arintzea, erabiltzea eta konpontzea du helburu. Spring-ek azpiegitura-euskarria eskaintzen du aplikazio-mailan, eta eredu oso bat eskaintzen du, bai konfiguraziorako, eta bai Javaren bidez garatutako enpresa-aplikazioak programatzeko, plataformaren hedapenean bereizkeriarik egin gabe.

4.3 irudia: Spring Framework

Spring Boot

Spring Boot [11], Spring Framework ezagunean oinarritutako aplikazioen garapena are gehiago sinplifikatzeko sortu zen tresna bat da. Spring Boot-en helburu nagusia Spring-ek gaur egun funtzionatzeko duen konfigurazio konplexuaz erabat ahaztuz, garatzailea soluzioa garatzen hastea da. Gainera, software-tresna honek, web aplikazioak .jar fitxategi gisa konpilatzeko aukera ematen du eta konpilatutako fitxategi hau Java aplikazio arrunt moduan exekuta daiteke.

4.4 irudia: Spring Boot

BootStrap

Bootstrap [20] HTML, CSS eta JavaScript-ekin web-garapenak egiteko kode irekiko tresna bilduma da. Honen bidez, gure web guneari forma eman diezaiokegu CSS eta JavaScript liburutegiak erabiliz. Hainbat osagai ditu: modu-leihoak, menuak, koadroak, botoiak, inprimakia… hau da, gure orria maketatzeko behar ditugun elementuak.

4.5 irudia: BootStrap

Google Drive

Google Drive [31], Google konpainia estatubatuarrak sortutako hodeiko zerbitzua da, eta aukera hauek ematen ditu, besteak beste, fitxategi-mota guztiak hodeian biltegiratzea, hainbat aplikaziorekin online dokumentuak sortu eta editatzea (Google Docs, Spreadsheet [16], …), laneko dokumentu horiek partekatzea, denbora errealean elkarlanean argitaratzea, zure fitxategiak zuzenean nabigatzaile batetik kudeatzea, ordenagailuan deskargatu beharrik gabe, etab.

4.6 irudia: Google Drive

Maven

Maven [1], 2002an sortuta, Java proiektuak kudeatzeko eta eraikitzeko software-tresna da. Tresna honek, Project Object Model (POM) bat erabiltzen du eraiki beharreko software-proiektua, beste modulu batzuetako gelak eta kanpoko osagaiak eta elementuen eraikuntza-ordena deskribatzeko. Zenbait lan egiteko aurrez zehaztutako helburuak ditu, hala nola kodea konpilatzea eta paketatzea.

4.7 irudia: Maven

MySQL

MySQL [16], datu-base erlazionalak kudeatzeko sistema bat da, lizentzia dualaren bidez garatua: lizentzia publiko orokorra eta *Oracle Corporation*-en lizentzia komertziala. Munduko kode ireki ezagunenaren datu-basetzat jotzen da, eta, oro har, web-garapen ingurune guztietarako datu-base orokorren ezagunenetako bat da, Oracle eta Microsoft SQL Server-ekin batera.

4.8 irudia: MySQL

4.2 Programazio lengoaiak

Proiektuan zehar hainbat programazio-lengoaia desberdin erabili egin dira. Bakoitza, arlo desberdinetan aplikatu da, eta hurrengo zerrendan aurkezten dira.

Java

Java [14], klasetan oinarritutakoa eta objektuei zuzendutako programazio lengoaia da. Plataforma ezberdinetan (MAC, Linux, Windows, etab.) exekutatu eta banatu ahal izango diren softwareak diseinatu eta garatzeko aukera ematen du Javak, software hau aldatu beharrik gabe eta makinaren arkitekturan pentsatu beharrik gabe. Eta hau, Java Virtual Machineri esker gertatzen da, aplikazioaren eta gailuaren hardwarearen artean zubia sortzen duen makina birtuala.

Lengoaia honek, liburutegi estandar handia eskaintzen du eta gainera, informazioa automatikoki administratzen duten aplikazioak garatzeko eta exekutatzeko plataforma segurua eskaintzen du. Datuen pribatutasuna babestuz, komunikazio-bide seguruak eskaintzen ditu, eta, sintaxi zorrotza duenez, kodea haustea eragozten du, hau da, ez du kodea hondatzen uzten.

Aurreko arrazoi hauengatik aukeratu izan da lengoaia hau proiektu honen garapenerako. Software lengoaia oso erabilgarri, eragingarri eta ahaltsua da.

JavaScript

JavaScript [5], normalean JS modura laburtua, interpretatutako programazio-lengoaia da. Objektuetara bideratuta dago ere, prototipoetan oinarritua, aginduzkoa eta dinamikoa. Batez ere, bezeroaren aldean erabiltzen da, web-nabigatzaile baten zati gisa ezarria, eta erabiltzailearen interfazean eta web orri dinamikoetan hobekuntzak egiteko aukera ematen du.

JQuery

JQuery [23] JavaScript-en plataforma anitzeko liburutegia da, eta aukera ematen du HTML dokumentuekin elkarreragiteko modua sinplifikatzeko, DOM zuhaitza manipulatzeko, gertaerak maneiatzeko, animazioak garatzeko eta interakzioa gehitzeko.

CSS

CSS [30] (ingelesez, Cascading Style Sheets) kaskadako estilo-orrien lengoaia da, eta HTML markaketa-lengoaian idatzitako elementuak estilizatzeko erabiltzen da. CSSk edukia eta gunearen ikusizko irudikapena bereizi egiten ditu. Estilo-orrien lengoaia hau W3C-k (World Wide Web Consortium) garatu zuen 1996an, HTML ez zelako orria formateatzen laguntzen duten etiketak izateko diseinatu.

HTML

HTML [29], HyperText Markup Language-ren ingelesezko siglak (hipertestu-marken lengoaia) web orriak egiteko markaketa-lengoaia da. Web orrien garapenarentzako markaketa lengoaiari egiten dio erreferentzia, eta softwarearen erreferentzia gisa balio duen estandarra da. Web orri baten edukia definitzeko, oinarrizko egitura eta kodea (HTML kodea) definitzen ditu, hala nola, testua, irudiak, bideoak, jokoak, etab.

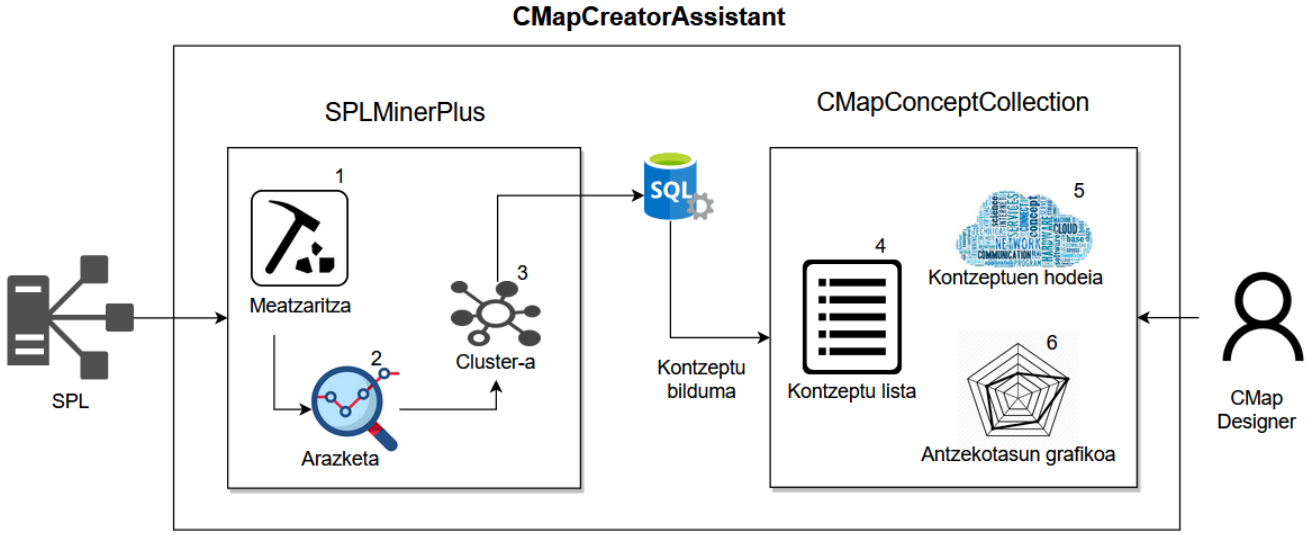
Python

Python [22] programazio-lengoaia interpretatua da, eta kodearen irakurgarritasuna da lengoaia honen filosofiak azpimarratzen duen ezaugarri nagusia. Paradigma anitzeko programazio-lengoaia da, objektuetara, aginduzko programaziora eta, neurri txikiagoan, programazio funtzionalera bideratzen baita. Lengoaia interpretatua, dinamikoa eta plataforma anitzekoa da. Kode irekiko lizentzia du, Python Software Foundation License izenekoa.

5. KAPITULUA

Soluzioaren diseinua

Kapitulu honetan, proiektu honetan garatu den soluzioaren diseinuaren ezaugarriak zehazten dira. Soluzioa izango den aplikazioak aurkezten duen diseinu globala azaltzen da, eta hurrengo bi kapituluetan, soluzioaren garapenaren ezaugarriak xehetasun handiagoekin aurkeztuko dira.



5.1 irudia: Soluzioaren diseinuaren arkitektura

Gradu amaierako proiektu honen emaitza, SPL-aren kontzeptu maparen diseinatzaileari mapa hau sortzen lagunduko dion aplikazio da, eta soluzioa izango den aplikazioren diseinuaren arkitektura 5.1 irudian azaltzen da. Zehazkiago esanda, SPL-aren kodea eta proiektuaren dokumentazioa arakatzen du *SPLMinerPlus* moduluak. Eta ondoren, modulu honen bitartez lortutako kontzeptu bilduma, kontzeptu maparen garatzaileari, modu egokian aurkeztuko dion *CMapConceptCollection* bistaratze modulua garatu da. Beraz, garatu den aplikazioa, *CMapCreatorAssistant*, web aplikazio bat izan da, non, lehen modulu batek, *SPLMinerPlus* moduluak, kontzeptu bilduma bat sortzen duen, eta bigarren modulu batek, *CMapConceptCollection* moduluak, kontzeptu bilduma hau bistaratzen du. Web aplikazioak datuak aurkezteko, datu-base erlazional batekin konektatuta dago. Lehen esan bezala, aplikazio honek bi modulu izango ditu.

Lehenengo moduluan, meatzaritza-lanaren prozesua eramaten da aurrera. Proiektuaren kode eta dokumentazio fitxategi guztiak arakatzen dira, haien testua lerroz lerro aztertuz. Arakatze prozesu

honetan, kodearen aldagai, funtzio, klase, parametro, pakete… hau da, kodean dauden gako-hitz guztiak gordetzen dira, eta baita, gako-hitz hauen agerpen kopuru osoa kode guztian zehar. Gainera biltegiraketan, gako-hitza termino bezala ere izendatu dugu, termino bakoitzaren testuingurua, hau da, terminoak proiektuaren zein tokitan aurkitzen diren ere biltegiratzen da. Meatzaritza-lan honen xehetasun gehiago 6.3.1 azpiatalean aurki daitezke.

Termino guztiak datu-egitura batean bilduta daudenean, termino hauen garbiketa sakon bat egitea komeni da, 5.1 irudiko 2 zenbakia duen prozesua. Izan ere, termino ugari agertzen dira esanahirik ez daukatenak, edota kodean zehar aldagai edo funtzio lagungarri bezala erabili direnak. Garbiketa edo arazketa hau egiteko, irizpide desberdinak erabiltzen dira, 6.3.3 atalean zehazturik daudenak. Eta bukatzeko, terminoen agerpen kopuruaren arabera bilduma ordenatu egiten da.

Behin interesatzen zaizkigun termino guztiak datu-egitura batean bilduta, gerta daiteke termino multzo ordenatu honetan zenbait terminok kontzeptu berdinarekin harremana izatea, hots, termino batzuk kontzeptu berdintsu batzuetakoak izatea. Beraz, termino bilduma hau berriro jorratu egiten da, kontzeptu berdina irudikatzen duten terminoak taldekatzeko, 5.1 irudiko 3 zenbakidun prozesua. Taldekatze edo *cluster* prozesu hau, hitzen arteko antzekotasunaren arabera egiten da, eta horrela, esanahi edo testuinguru berdineko terminoak kontzeptu bakarrean biltzen dira. Hala ere, taldekatze prozesu hau 6.3.4 atalean sakonago jorratzen da.

Eta azkenik, *CMapConceptCollection* web modulu baten bitartez, termino multzo edo talde horiek, eskuragarri jartzen zaizkio kontzeptu mapa diseinatzen duen pertsonari. Kontzeptu mapa aurkezten duen bistaratze ikuspegiaz gain, 5.1 irudiko 5 zenbakidun prozesua, beste bistaratze ikuspegi desberdinak erabiliko dira informazio gehigarriak eskaintzeko, 4 eta 6 zenbakidun prozesuak, baina hau, 7. kapituluan zehaztasun gehiagorekin azalduko da.

6. KAPITULUA

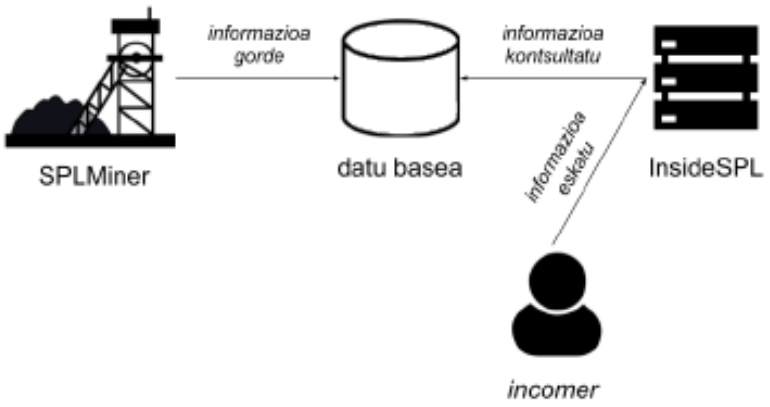
Modulu meatzariaren diseinua eta inplementazioa

Proposatutako soluzioaren garapenean sortutako *SPLMinerPlus* moduluaren inguruan hitz egingo da. Bertan, arazoen aurrean hartu behar izandako erabakiak eta hauen ondorioz lortutako soluzioak izango dituen ezaugarriak aurkeztuko dira. Gainera, SPL arkitektura bistaratzeko proiektuan garatutako softwarea proiektu honetako helburuak lortzeko moldatu egin da, eta moldaketa edo egokitzapen hauen xehetasunak, kapitulu honetan ere azalduko dira.

6.1 Aurreko aplikazioaren kodearen berrerabilpena

Aurretik aipatu bezala, meatzaritzako kodea ez da hutsetik garatu. Lan honetan, Iosu Salaberrik "Software Produktu-Lerroen (SPL) arkitektura bistaratzeko aplikazioaren garapena" proiektuan[1] garatutako *SPLMiner*-a [24] erabili da oinarri gisa. *SPLMiner* honek, *pure::variants* softwarearekin egindako SPL proiektuen kode fitxategiak, ezaugarriak eta ezaugarri hauen arteko erlazioak arakatzen ditu, eta behin horren guztiaren meatzaritza-lana eginda, lortutako emaitzak datu-basean gordetzen ditu.

Iosu Salaberrik garatutako aplikazio osoaren arkitektura 6.1 irudian ikus daiteke. Bi modulu ezberdin bereizten dira, *SPLMiner* eta *InsideSPL*. *InsideSPL* modulua SPL-aren zenbait ezaugarri aurkezteko erabiltzen da, beraz, memoria honetan aurkezten den lanerako ez da interesgarria. Aldiz, *SPLMiner* modulu meatzaria bai.



6.1 irudia: Iosu Salaberriren proiektuaren arkitekturaren diseinua

SPLMiner modulu meatzari honek, *MainClass* izeneko klase nagusia dauka, eta *main* programa

[1] Proiektuaren ADDI helbidea: http://hdl.handle.net/10810/48777

honen hasieran, git biltegi batean aurkituko den SPL-a atzitzeko konfigurazio orokorrak kargatzen dira, helbideak, baimenak, konfigurazio balioak, etab., eta ondoren, SPL-a kargatzen da. Dena konfiguratu eta SPL-a kargatu eta gero, meatzaritza-lanarekin hasten da, meatzariei dei eginez. Programa nagusi honek, meatzaritza-lan desberdinak egiteko aukera ematen du, zehazki hiru: soilik, kontzeptu maparen meatzaritza-lan egiteko aukera, edo, soilik, *FamilyModel*-aren eta *VariantDescriptionModel*-aren meatzaritza-lan egiteko aukera, edo SPL osoaren meatzaritza egiteko aukera, hots, aurreko guztiena gehi *FeatureModel*-aren meatzaritza-lana. SPL-aren atal desberdin hauei buruz, proiektuaren aurrekarien 3.1 atalean hitz egiten da.

Pure::variants-ek hiru XML fitxategi mota sortzen dituenez (.xfm *FeatureModel*-ena, .ccfm *FamilyModel*-ena eta .vdm *VariantModel*-ena), 3.2 atalean azalduta dagoena, meatzaritza-lana egiteko, hiru meatzari desberdin erabiltzen dira (ikusi 6.2 irudia), eta jarraian azalduko dira.

- **FeatureModelMiner**: Meatzari honek, SPL-aren ezaugarrien inguruko informazioaren meatzaritza-lana egingo du, *pure::variants*-ekin sortutako .xfm luzapeneko fitxategiak prozesatuz. Meatzariari deia *mineAll*() funtzioarekin egiten da, eta honek, aurretik sortutako SPL elementuaren informazioa eguneratuko du.

- **FamilyModelMiner**: Meatzari honek, SPL-aren fitxategien egituraren inguruko informazioaren meatzaritza-lana egingo du, *pure::variants*-ekin sortutako .ccfm luzapeneko fitxategiak prozesatuz. Meatzariari deia *mineAll*(*familyModelPaths*) funtzioarekin egiten da, eta honen eragina aurretik sortutako SPL elementuaren informazioa eguneratzea izango da. *familyModelPaths* atributuak meatzariak arakatu beharreko fitxategien kokapenen zerrenda izango da, klase-diagraman ez baitago *FamilyModel* klaserik informazio hori gordetzen duenik.

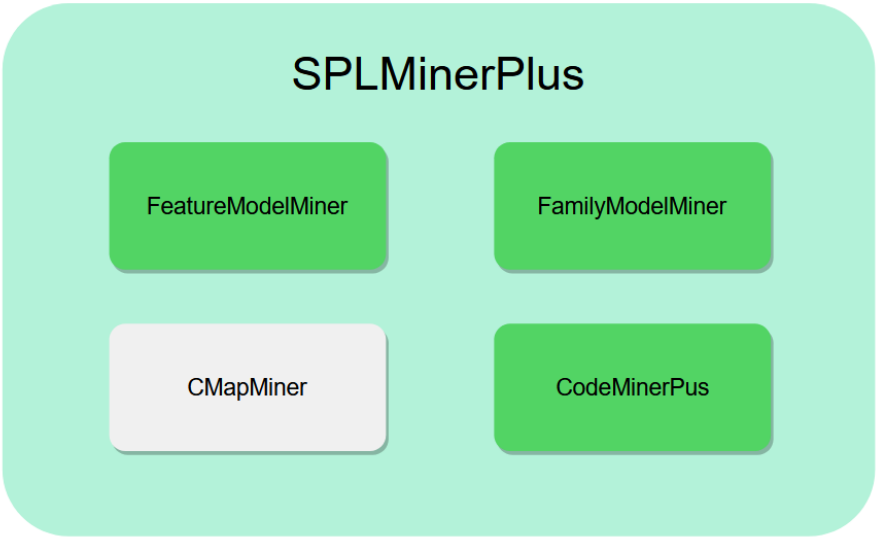
- **CodeMiner**: Meatzari hau aurreko meatzariaren laguntzailea da, eta *CodeElement*-ak mota zehatz batzuetako direnean, zehazki, *CodeFile* motakoak, fitxategiak pure::variants-en anotazioen bila minatzeaz arduratzen da. Meatzariari deia *extractVPsFromFile*(*codeFile*, *fileType*) funtzioarekin egiten da, eta honek *aldaketa puntuak* (ikusi 3.1 atala) bilatuko ditu fitxategian. *codeFile* atributuak, minatu beharreko fitxategiari egiten dio erreferentzia, eta *fileType*-k fitxategiaren motari, izan ere, fitxategi motaren arabera meatzaritza-lan mota desberdinak aplikatuko baitira. Kasu honetan, hau izango da hiru hauetatik meatzari interesgarriena, eta horregatik, 6.3 atalean, meatzari honi buruzko azalpen sakonagoak ematen dira.

  CodeMiner-ak pure::variants-en anotazioak bilatuko ditu, baina anotazio zehatz batzuetaz arduratuko da soilik. Anotazio horiek *SPLMiner* klase nagusian definituta daude eta hurrengoak dira:

```
public static final String[] VARIATION_POINT_NO_XML_STATEMENTS = { "PV:IFCOND",
PVSCL:IFCOND" };

public static final String[] VARIATION_POINT_NO_XML_STATEMENTS_END = {PV:ENDCOND",
"PVSCL:ENDCOND" };
```

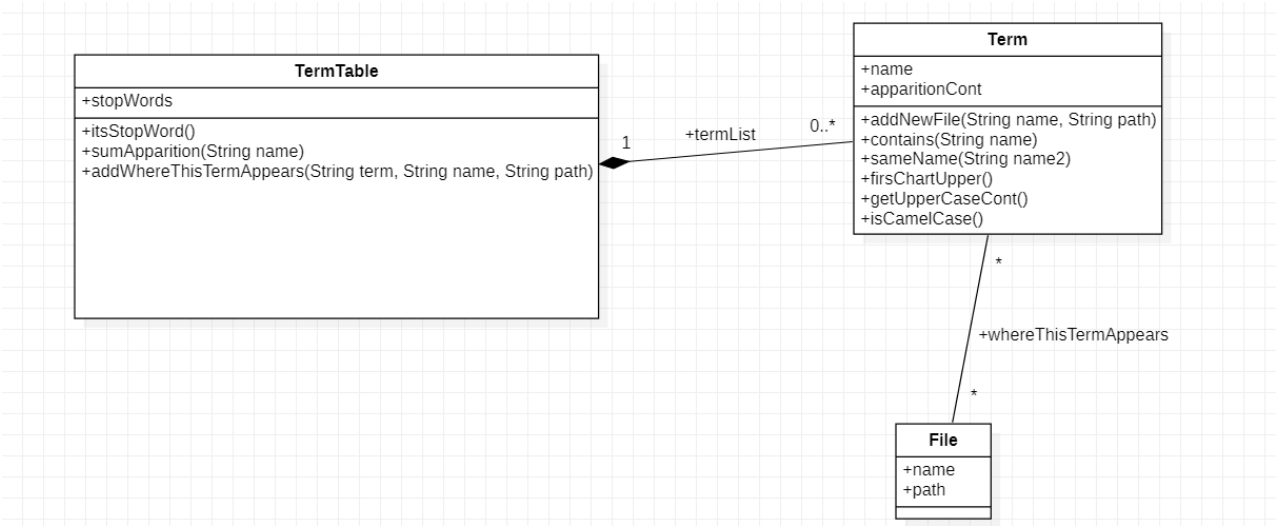
Esan beharra dago, *CMapMiner* izeneko meatzari bat ere existitzen zela, eta honek SPL-aren kontzeptuak azaltzen dituen kontzeptu mapen inguruko informazioaren meatzaritza-lana egiten zuela, *CMap* softwareak sortutako .cxl luzapeneko fitxategiak prozesatuz. Baina ez da proiektu honetan behar izan, eta horregatik, ez da honen funtzionamendua azaltzen.



6.2 irudia: *SPLMinerPlus* meatzariak

6.2 Datu-egitura

SPL-aren diseinuan eta inplementazioan erabilitako terminoak arakatu eta bildu behar izan dira eta horretarako 6.3 irudian aurkezten den klase diagrama deskribatuko da.



6.3 irudia: Datu-egituraren klase diagrama.

6.3 irudiko *TermTable* klaseak SPL-ko termino bilduma errepresentatzen du. *Term* klaseak terminoak errepresentatzen ditu, bakoitzaren izena, termino bakoitzaren agerpen kopuru totala proiektu osoan zehar, eta termino bakoitza agertzen den kode edo dokumentazio fitxategia. *File* klasea, kode eta dokumentazio fitxategiak errepresentatzeko erabiltzen da eta diagraman adierazi bezala, termino bera fitxategi batean baino gehiagotan ager daiteke. Termino bilduma honek, *StopWords* izeneko *String*-en lista bat izango du, eta lista honetan, tratatu edo aztertu nahi ez diren

hitzak agertuko dira. Beraz, arakatu nahi den proiektuaren kodean, *StopWords* listan agertzen den termino bat, bai klase, aldagai, funtzio edo metodo izena azalduz gero, modulu honen kodeak ez du tratatuko.

6.2.1 Datu-basea

Jatorrizko proiektuaren datu-basean SPL-ko fitxategi bakoitzaren izena eta id bat gordetzen zen soilik *Codefile* taulan, eta honen egokitzapen bat egin behar izan da, proiektu honen betekizunak betetzeko, eta fitxategien edukia biltegiratzea izan da.

Kasu honetan, kode eta dokumentazio fitxategien edukia gordetzea ere beharrezkoa da, gero web moduluan erabiltzaileak eskuragarri izateko. Fitxategien edukia gordetzeko, datu-basearen egitura aldatu behar izan da. Izan ere, *Codefile* taularen egituran, atributu berri bat gehitu da. Atributu honek *content* izena du eta fitxategi bakoitzaren eduki guztia biltegiratuko du, bai fitxategi honen edukia edozein lengoaiako kodea bada, eta bai fitxategi honen edukia dokumentaziokoa bada. *content* atributu honetan, 16.777.215 byte gorde daitezke, hots, 16 MB inguru. Hurrengo 6.4 irudian, MySQL-n *describe* adierazpena erabilita lortzen den emaitza:

Field	Type	Null	Key	Default	Extra
ID	varchar(500)	NO	PRI	NULL	
FILENAME	varchar(100)	NO		NULL	
CONTENT	mediumtext	YES		NULL	

6.4 irudia: *Codefile* taularen deskribapena.

Eta 6.5 irudian, adibidez, index.js fitxategiaren egitura datu-basean biltegiratuta erakusten da.

ID: iotrmShxJoCoeKajdLCxS94OoJsb

FILENAME: index.js

CONTENT: (() => {#LINE_BREAK# let fileToLoad = (new URL(document.location)).searchParams.get('file')#LINE_BREAK##LINE_BREAK# function reqListener () {#LINE_BREAK# document.querySelector('pre').innerText = this.responseText#LINE_BREAK# }#LINE_BREAK##LINE_BREAK# let oReq = new XMLHttpRequest()#LINE_BREAK# oReq.addEventListener('load', reqListener)#LINE_BREAK# oReq.open('GET', fileToLoad)#LINE_BREAK# oReq.send()#LINE_BREAK#})()#LINE_BREAK#

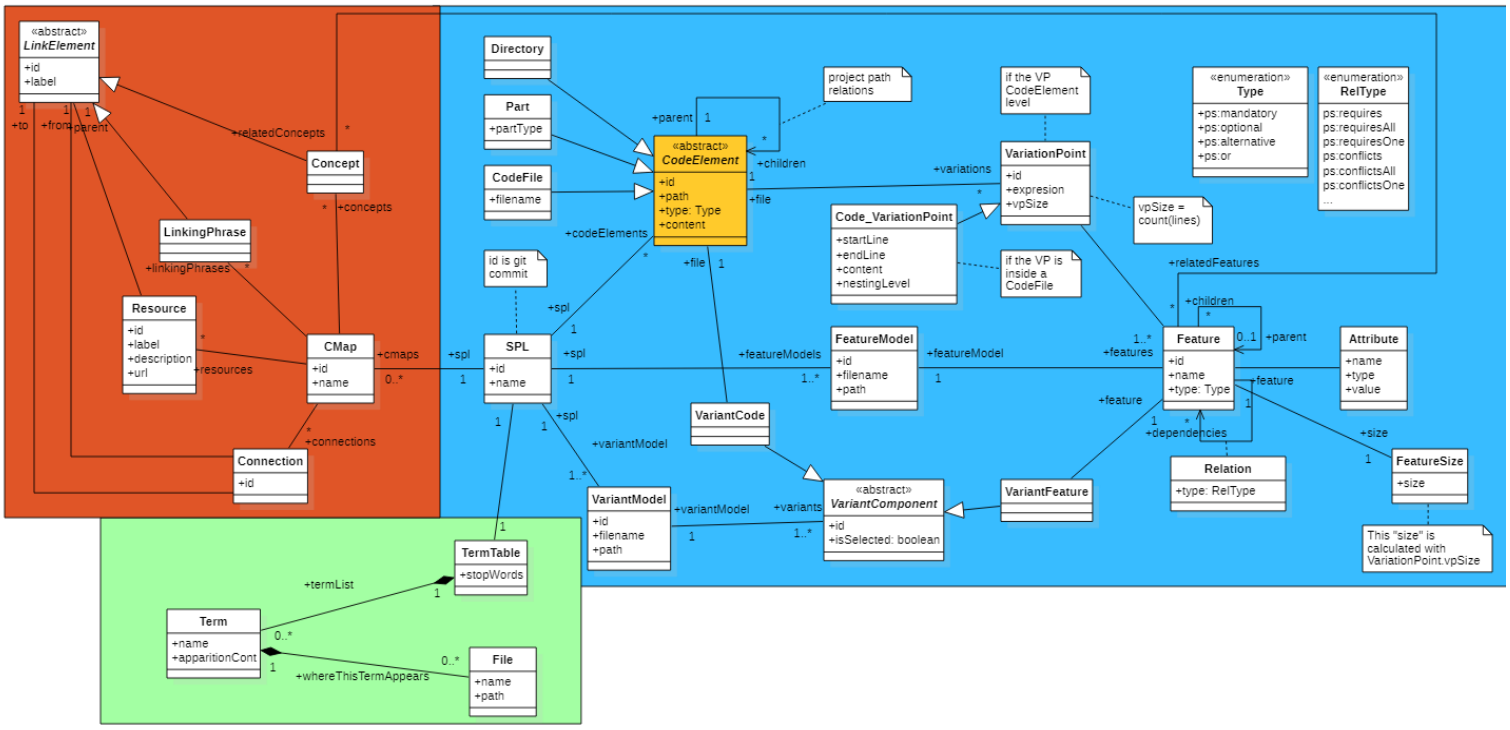
6.5 irudia: *Codefile* taularen tupla baten adibidea.

6.2.2 Klase-diagrama

Iosu Salaberriren “Software Produktu-Lerroen (SPL) arkitektura bistaratzeko aplikazioaren

garapena" proiektuan diseinatutako klase-diagramak bi domeinu ezberdin jasotzen ditu, SPL-aren domeinua eta kontzeptu mapen domeinua. Kasu honetan, kontzeptu mapen domeinua alde batera utziko da, proiektu honetan garatzen den aplikazioak ez duelako kontzeptu mapan meatzaritza-lana egingo. Eta proiektu honen helburuak lortzeko gehitu eta moldatu behar izan diren diagramak 6.6 irudian aurkezten dira.

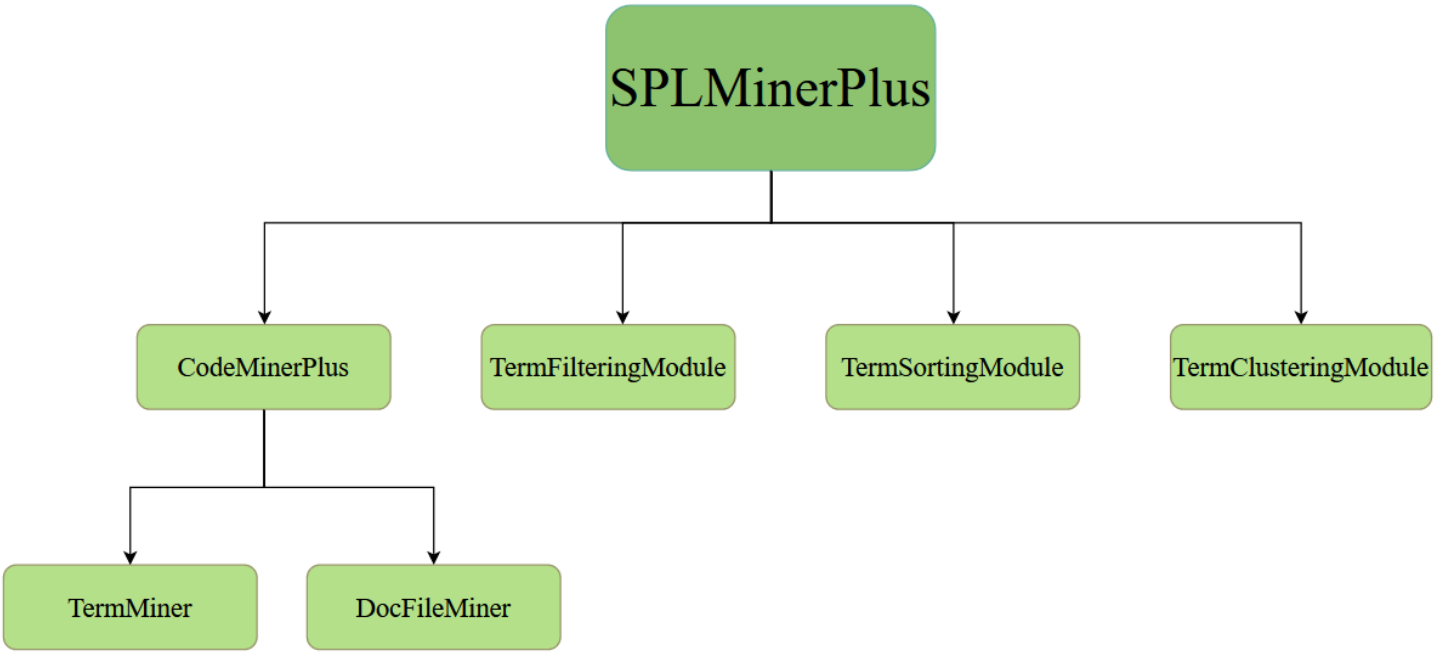
6.6 irudiko klase-diagraman, gorriz, alde batera utzi den domeinua adierazten da. Gero, kolore urdinez osaturiko aldea, guztiz berdin mantendu den zatia izango litzateke. Esan beharra dago, alde urdin honetako *CodeElement* klasea, kolore horia duena, moldatu behar izan dela, atributu bat gehituz. Eta azkenik, kolore berdez dagoen eremuan, klase-diagrama osoari gehitu zaizkion klaseak eta erlazioak azaltzen dira.



6.6 irudia: Klase-diagrama

6.3 *SPLMinerPlus* moduluaren arkitektura

Jatorrizko *SPLMiner* softwarearen programa nagusia (*Main* klasea) ere moldatu behar izan da. Oinarrizko kodean, meatzaritza-lan desberdinak egiteko aukera ematen zen (ikusi 6.1 atala), baina kasu honetan, meatzaritza arrunta eta osoa egitea beharrezkoa denez, aukeraketa hau kendu egin da. Gainera, meatzari horren azpimodulu bat, hots, *CodeMiner* ere moldatu behar izan da. Beraz, moldaketak aplikatu ostean 6.4 irudian aurkezten dena izango litzateke aplikazioaren arkitektura.



6.7 irudia: Aplikazioaren arkitektura

SPL-aren meatzaritza-lana egin ostean, SPL-aren kode eta dokumentazio fitxategi guztien edukia aztertzen hasten da. Lan hau, *CodeMinerPlus* klasean egiten da, eta terminoz termino aztertuz doa, eta termino hauek datu-egituran gordetzen dira. Prozesu hau, hurrengo azpiataletan xehetasun gehiagorekin azaltzen da.

Azkenik, datu-egitura hau, *cluster* eragiketa egiteko prestatzen da, *forCluster.txt* izeneko fitxategi batean inprimatuz eta kode fitxategi guztien informazioa, MySQL datu-basera irauli egiten da. *MainClass* klase honen lana bukatzeko, *cluster* eragiketaz arduratzen den Python kodeari deia egiten zaio eta emaitza, beste *clusterDone.txt* fitxategi batean inprimatzen du, prozesu hau aurreragoko 6.8 atalean sakonago aurkezten da. Azkenik, *clusterDone.txt* fitxategi hau, *CMapConceptCollection* bigarren moduluaren erroan kokatzen du, datu aurkezpen modulu honek atzigarri izan dezan.

6.3.1 *CodeMinerPlus* meatzaria

Aurreko proiektuko *CodeMiner* meatzaria, pure:variants-en anotazioen meatzaritza-lanaz arduratzen denez (ikusi 6.1 atala), fitxategi-kodeak diren fitxategi guztiak arakatzen dira, eta hemen sartzen da proiektu honetako aplikaziorako moldaketa. Puntu honetan, pure::variants softwarearekin egindako SPL proiektuaren kode guztia atzigarri dago, eta honi esker hasten da terminoen azterketa. Azterketa honek suposatu duen erronka, 8 atalean xehetasun gehiagorekin azaltzen da.

CodeMiner meatzarian gehitutako aldaketak hurrengokoak dira, eta horrela sortu da *CodeMinerPlus* meatzari berria:

- **Dokumentazio fitxategien meatzaritza-lana.** Aurreko proiektuan sortutako meatzari moduluek (minerrek), ez dituzte dokumentazio fitxategiak arakatzen eta lan hau egiteko,

meatzari gehigarri bat sortu da. Hau, lehenik eta behin, arakatuko den SPL proiektuaren erroan dagoen *docs* karpetan kokatuko da, bertan dokumentazio fitxategiak egongo direlako. Eta *docs* karpeta honetan, *featureModel* karpeta dagoen aztertuko du. Soilik, *featureModel* izeneko karpeta existitzen bada *featureModel* karpetan sartu eta direktorio honen fitxategi guztien meatzaritza-lana egingo du. Dokumentazio fitxategi hauek SPL-aren egituraren informazioa eskaintzen dute, elementu bakoitzaren deskribapena eskainiz eta hauen propietateak azalduz.

- **Terminoen azterketa**. Meatzari honetan gehitutako kodea, hurrengo adibidean aurkezten den moduan agregatu da. Terminoak, aldagai, klase, metodo eta dokumentazio elementuen izenetatik eratortzen dira, eta horregatik elementu hauek tratatu behar dira. Termino mota ezberdinen azterketa egiteko, modu berdintsuan egin da. Kasu honetan, SPL-aren kodean zehar, klase izenetatik ateratzen diren terminoak aztertzeko kodea aurkezten da.

```
// Class terms
if (line.contains("class")  && line.contains("{")) {

        String[] clss = line.split("class")
        if(clss.length>1) {
                String aux = clss[1];
                String[] lineSecondPart = aux.split(" ");
                if(lineSecondPart.length>1) {
                        String result = lineSecondPart[1];
                        if(result!=null && !result.contains(",") && !result.contains(")")
                && !result.contains("_")  && !termTable.itsStopWord(result)
        && result.length()>3) {
                                if(termTable.appears(result)){
                                        termTable.sumApparition(result);
                                        termTable.addPath(result, cf.getFilename(),
                        cf.getPath());
                                }else {
                                        Term t = new Term(result,10);
                                        t.addNewFile(cf.getFilename(), cf.getPath());
                                        termTable.getTermTable().add(t);

                                }
                        }
}}}
```

Adibidez, *JavaScript* lengoaian idatzita dagoen kodean, aldagaiak erazagutzeko hiru aukera, *var*, *let* eta *const* erabili daitezke. Beraz, kasu bakoitza modu independentean tratatu behar izan da. Gainera, gerta daiteke, aldi berean aldagai bat baino gehiago erazagutzea, hots, aurreko hiru aukeretako bat behin erabiliz, aldagai bat baino gehiago definitzea. Kasu hori ere kontuan hartu behar izan da, eta hurrengo kode zatian azaltzen den moduan tratu dira kasu hauek. Hiru balio hauetako adibide bakarra aurkezten da, beste biak oso antzekoak direlako, soilik, aztertzen ari garen balioaren arabera *String*-a aldatuko da, *let* edo *const* balioengatik.

```
if (line.contains("var") || line.contains(" var") || line.contains("var ")) {

        String[] vr = line.split("var");
        if(vr.length>1) {
           String g = vr[1];
           String[] lineSecondPart = g.split(" ");
           if(g.contains("=")) {
              if(lineSecondPart.length>1) {
                 String result = lineSecondPart [1];
                 if(result!=null && !result.contains(",") && !result.contains("_") &&
                 !result.contains("$") &&!termTable.itsStopWord(result) &&
                 result.length()>3) {
                    if(termTable.appears(result)){
                       termTable.sumApparition(result);
                       termTable.addPath(result, cf.getFilename(), cf.getPath());
                    }else {
                       Term a= new Term(result,1);
                       a.addNewFile(cf.getFilename(), cf.getPath());
                       termTable.getTermTable().add(a);
                    }
                 }

                 //More than one variable defined
                 }else if (!(termTable.itsStopWord(lineSecondPart [1]))){
                    for(int p=1; p< lineSecondPart.length; p++) {
                       String[] result = lineSecondPart [p].split(",");
                       if(result[0].contains(";") && ! result[0].contains("_")) {
                          String[] resultAux = lineSecondPart [p].split(";");
                          if(resultAux!=null && resultAux[0].length()>3) {
                             if(termTable.appears(resultAux[0])){
                                      termTable.sumApparition(resultAux[0]);
                                      termTable.addPath(resultAux[0],cf.getFilename(),cf.getPath());
                                codeWithVariables += resultAux[0]+ "\n";
                                codeWithVariables += resultAux[0]+ "\n";
                             }else {
                                Term a= new Term(resultAux[0],1);
                                a.addNewFile(cf.getFilename(), cf.getPath());
                                termTable.getTermTable().add(a);
                                codeWithVariables += resultAux [0]+ "\n";
                             }
                          }
                       }else {
                          if(result!=null) {
                             if(termTable.appears(result[0])){
                                termTable.sumApparition(result[0]);
                                termTable.addPath(result[0],cf.getFilename(), cf.getPath());
                                codeWithVariables += result[0]+ "\n";
                             }else {
                                Term a= new Term(result[0],1);
                                a.addNewFile(cf.getFilename(), cf.getPath());
                                termTable.getTermTable().add(a);
                                codeWithVariables += result[0]+ "\n";
                             }
                          }
                       }
                    }
                 }
              }
           }}
```

6.3.2 *TermSortingModule*: Termino bildumaren ordenaketa

Behin analisia bukatuta, termino bilduma lortzen da, baina bilduma hau ordenatu beharra dago. Ordenaketa hau, agerpen kopuruaren arabera egin da, hots, agerpen kopuru altuagoa daukan terminoa, termino bilduma honetan lehen posizioan agertzen da. Ordenaketa hau egiteko, Java lengoaiaren *java.util* [17] paketeak eskaintzen duen *Comparator<T>* interfazea [15] erabili da. *TermApparitionComparator* klasea sortu da, aurretik esandako interfazea inplementatzen (*implements*) duena. Konparatzaile honek alderatzen dituen objektuak *Term* (terminoa) dira, eta terminoen *apparitionCont* atributuaren arabera, bilduma ordenatuta itzultzen du, handienetik txikienerako ordenan.

Jarraian, *Comparator<T>* interfazearen inplementazioa:

```
public class TermComparator implements Comparator<Term>{
        private List<Comparator<Term>> listComparators;

    @SafeVarargs
    public TermComparator(Comparator<Term>... comparators) {
       this.listComparators = Arrays.asList(comparators);
    }

@Override
public int compare(Term t1, Term t2) {
        // TODO Auto-generated method stub
        for (Comparator<Term> comparator : listComparators) {
                int result = comparator.compare(t1, t2);
                if (result != 0) {
                        return result;
                }
        }
        return 0;
}
}
```

Eta hemen, konparaketa eragiketa egiten duen klasea:

```
public class TermApparitionComparator implements Comparator<Term> {

  @Override
  public int compare(Term t1, Term t2) {
        return t2.getapparitionCont() - t1.getapparitionCont();
  }
}
```

6.3.3 *TermFilteringModule*: Terminoen arazketa

Terminoen azterketan zehar, SPL-aren kodearen funtzio, aldagai, klase eta metodo guztiak prozesatzen dira. Arakatu nahi den proiektuaren softwarearen meatzaritza egiteko, proiektu honen kode fitxategi guztiak lerroz lerro aztertzen dira, eta lerro bakoitzean terminoren bat agertuz gero, hau tratatu egingo da. Beraz, termino batekin topo egitean, termino honi garbiketa edo iragazpen batzuk aplikatuko zaizkio, atal honetan azalduko direnak, eta iragazpen honen baldintza batzuk betetzen baldin baditu, termino honen tratamenduarekin jarraituko da. Behin arazketa baldintzak pasata, termino hau, termino bilduman (*termList*) jadanik agertzen den konprobatuko da.

Kalibre honetako proiektu batean, honelako milaka eta milaka ager daitezke. Proiektu honen helburu bezala, kontzeptu bilduma izanda, milaka termino hauek 10-20 kontzeptu kopuruko bilduma batera itzuli behar direnez, hau lortzeko, SPL-aren kodean aurkitutako termino guztien gainean arazketa espezifikoa aplikatu da, eta ondoren azalduko da aplikatutako arazketaren eragiketa desberdinak.

- **Funtzio laguntzaileak.** SPL-aren kodean agertzen diren funtzio eta aldagai laguntzaile guztiak ez dira kontuan hartuko eta ondorioz, ez dira tratatuko. Izan ere, kasu hauetan, ez da informazio esanguratsua lantzen, barne kalkuluak edota eragiketak egiteko erabiltzen dira eta ez dute ezer interesgarririk itzuliko.

- **Karaktereak.** Terminoek karaktere zehatz batzuk baldin badituzte, alde batera utziko dira eta ez dira tratatuko. Karaktere zehatz hauek hurrengokoak izango dira: ‘.’, ’$’, ’(’, ’)’, ’_’, ’-’, ’&’, ’{’, ‘}’, ‘+’, ‘.’ eta ‘*’.

- **Terminoaren luzera.** Aztertzen ari garen terminoaren hitzaren luzera 3 baino handiagoa ez bada, alde batera utziko da termino honen tratamendua. Hau egiten da, bat, bi edo hiru luzera duten funtzio edo aldagaiak, normalean lagungarriak direlako. Eta lagungarriak ez diren kasuan, termino honen esangarritasuna ez da oso handia izango, beraz alde batera uztea erabaki da.

- ***stopWords* lista.** Terminoak aurreko arazketa baldintza guztiak pasa baldin baditu, termino hau *stopWords* listan agertzen ote den aztertzen da eta agertzen ez den kasuetan soilik tratatuko da. Bestela, baldintza ez duenez pasatzen, alde batera utziko da.

- **Agerpen kopurua.** Arazketa prozesu osoarekin bukatzeko, *cluster*-a egin baino lehen, azken baldintza aplikatzen da. Baldintza hau, datu-egituran gordeta dauden datu guztiak sinplifikatzeko eta ehunka termino ez izateko ezarri da. Baldintza honek, terminoaren agerpen kopuruarekin zerikusia dauka, eta balio honen arabera egiten da. Adibidez, taldekatze eragiketa egiteko fitxategian, *apparitionCont* atributuan balio zehatz bat baino handiagoa duten terminoak soilik gehituko dira, eta beste guztiak ez dira kontuan hartuko. Baldintza hau, aplikazio hau erabiliko duen erabiltzaileak alda dezake, bakoitzaren irizpidearen arabera jokatu ahal izateko aukera ematen du. Eta datu-egituran dauden balioen arabera, baldintza honen balioa aldatu daiteke, bakoitzaren interes eta helburuen arabera.

6.3.4 *TermClusteringModule*: Taldekatzea

Arazketa lana egin eta behin betiko termino bilduma izanda, taldekatze eragiketa egin behar da, eta horretarako, kanpoko liburutegi bat erabili da, kode irekiko liburutegia. *pyLev* [26] liburutegia

Github plataforman atzigarri dago, eta bertan aurrebaldintzak, postbaldintzak eta kodearen erabilera modua azaltzen da.

Zehazki *Python* programazio lengoaian idatzitako softwarea da, eta honek hitzen taldekatzea, *cluster*-a [26], egiten du. Taldekatze eragiketa hau, hitzen arteko antzekotasunaren arabera egiten du. Antzekotasun hori, *Levenshtein* distantziaren [4] arabera egiten du, eta distantzia hau, hitz batetik abiatuta beste hitz bat lortzeko egin behar diren gutxieneko eragiketen kopurua da. Eragiketa horiek hiru motakoak izan daitezke, karaktere bat txertatzea, ezabatzea edota ordezkatzea. Hemen adibide bat:

> "eman" eta "emango" hitzen arteko distantzia 2 da, gutxienez oinarrizko bi edizio behar baitira batetik abiatuta bestea lortzeko.
>
> 1. eman → emang ('g' bat gehitzea bukaeran)
> 2. emang → emango ('o' bat gehitzea bukaeran)

Taldekatzea gauzatzeko, aurreprozesatze bat egin behar da, terminoak modu zehatz batean sartu behar zaizkiolako programa honi, termino guztiak, listan bata bestearekin hutsune bat utziz egituratu behar dira. Horrela, *pyLev* liburutegiaren kodeari deia egin baino lehen, terminoak modu egokian eta aurretik aplikatutako ordena mantenduz, *forCluster.txt* fitxategian gordetzen dira. Eta ondoren, *pyLev* liburutegiaren kode honek, testu fitxategi honetatik hartuko ditu balioak, taldekatze eragiketa egin dezan. Taldekatzearen emaitza *clusterDone.txt* fitxategian gordetzen da. Kontzeptuak, bi '*' karaktere bitartean aurkezten ditu, eta kontzeptu honekin antzekotasun handien dituzten terminoak listarazten ditu, kontzeptu honen lerro berdinean, termino hauek komaz banaturik.

Egitura honen eta *cluster* eragiketaren adibide sakonago bat C eranskinean ematen da, hala ere, proiektu honetan lortutako kontzeptu baten eta kontzeptu hau biltzen duten terminoen listaren egitura hurrengo adibidean aurkezten da.

```
*Annotation*: APISimulation, AnnotatedTheme, Annotation, AnnotationGroup, AnnotationList, AnnotationUtils, CreateAnnotation, DeleteAnnotation, Operation, ReadAnnotation, ReplyAnnotation,
```

Python kode honen egitura aurkezten da:

```
#%% Liburutegien karga.
import pandas as pd
from sklearn import preprocessing
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np
import sklearn.cluster
import distance

name = open("CMap_creator_assistant\\CMap_creator_assistant\\forCluster.txt", "r")
names = name.readline()
```

```
words = names.split(" ")
words = np.asarray(words)
lev_similarity = -1*np.array([[distance.levenshtein(w1,w2) for w1 in words] for w2 in words])

affprop = sklearn.cluster.AffinityPropagation(affinity="precomputed", damping=0.75)

affprop.fit(lev_similarity)

fitx= open("CMap_creator_assistant\\clusterDone.txt", 'w')

for cluster_id in np.unique(affprop.labels_):
    exemplar = words[affprop.cluster_centers_indices_[cluster_id]]
    cluster = np.unique(words[np.nonzero(affprop.labels_==cluster_id)])
    cluster_str = ", ".join(cluster)
    fitx.write("*%s*: %s \n" % (exemplar, cluster_str))

fitx.close()
```

Azken azkenik, *clusterDone.txt* fitxategia izango da modulu meatzari honen azken emaitza. Bertan egongo dira termino guztiak kontzeptuetan bilduta, eta emaitza hau pasako zaio garatu den bigarren moduluari, *CMapConceptCollection*, honek informazio hau kudeatu dezan.

6.4 *SPLMinerPlus* moduluaren inplementazioa

Aurreko 6.3 atalean *SPLMinerPlus* moduluaren arkitektura azaltzen da, eta atal honetan, *SPLMinerPlus* modulua garatzeko jarraitu den egitura azalduko da. Egitura hori, proiektuaren paketetan zatitzen da, eta pakete bakoitzean klase desberdinak daude. Pakete bakoitza azpiatal batean aurkeztuko da.

6.4.1 *main* paketea

Pakete honetan klase nagusia aurki dezakegu eta 6.1 taulan zehazten diren ezaugarriak ditu.

Klasearen izena	**Edukia**	**Kode lerro kopurua (lehen)**	**Kode lerro kopurua (orain)**
MainClass	Programa nagusia.	270	304

6.1 taula: *main* paketearen egitura

6.4.2 *miners* paketea

Pakete honetan SPL meatzariak aurkitzen dira eta 6.2 taulan zehazten diren ezaugarriak dituzte.

Klasearen izena	Edukia	Kode lerro kopurua (lehen)	Kode lerro kopurua (orain)
FeatureModelMiner	SPL-aren ezaugarrien inguruko informazioaren meatzaritza-lana egiten da	352	352
FamilyModelMiner	SPL-aren fitxategien egituraren inguruko informazioaren meatzaritza-lana egiten da.	408	477
VariantModelMiner	Produktu desberdinen konfigurazioa biltzen duten fitxategi meatzaritza-lana egiten da.	147	147
CodeMinerPlus	SPL-aren kode eta dokumentazioa aztertutako da.	523	1116

6.2 taula: *miners* paketearen egitura

6.4.3 *database* paketea

Proiektuaren pakete honetan, SPL-aren datu guztiak datu-basera irauli egiten dira, lau klasez osatuta dago, eta hurrengo 6.3 taulan hauen ezaugarriak erakusten dira. Pakete honetako klaseak berdin mantendu dira, baina *FamilyModelDB* klasean, datu-baseko *codeFile* taularen konfigurazioa aldatu behar izan da, fitxategi bakoitzaren edukia gordetzeko.

Klasearen izena	Edukia	Kode lerro kopurua (lehen)	Kode lerro kopurua (orain)
MainSql	SPL-aren *insert*-ak egiten dira.	65	65

FeatureModelDB	*FeatureModel*-aren *insert*-ak egiten dira.	116	116
FamilyModelDB	*FamilyModel*-aren *insert*-ak egiten dira.	101	101
VariantModelDB	*VariantModel*-aren *insert*-ak egiten dira	60	60

6.3 taula: *database* paketearen egitura

6.4.4 *comparator* paketea

Pakete honetan, behin termino lista osatua izanda, termino lista hau agerpen kopuruaren arabera ordenatzen da, eta eragiketa horretaz pakete honetan dauden klaseak arduratzen dira. Klase horiei buruzko informazio 6.4 taulan adierazten da, eta *domain.cmap.creator* paketea gertatzen den moduan, soilik orain dituzten kode lerroak adieraziko dira, klase hauek proiektu honetan sortu direlako.

Klasearen izena	Edukia	Kode lerro kopurua (orain)
TermComparator	Klase honek terminoen konparaketa egiten du.	32
TermApparitionComparator	Klase honek terminoen agerpen kopuruak konparatzen ditu.	15

6.4 taula: *comparator* paketearen egitura

6.4.5 *domain.cmap.creator* paketea

Pakete honetan kontzeptu maparen domeinua definitzen da, eta domeinu hau 6.4 taulan aurkezten diren klaseekin eratuta dago. 6.5 taulan, soilik orain duten kode lerro kopurua adieraziko da, klase hauek proiektu honetan sortu direlako.

Klasearen izena	Edukia	Kode lerro kopurua (orain)
TermTable	Klase honek termino lista errepresentatzen du.	89
Term	Klase honek terminoak errepresentatzen ditu.	101
File	Klase honek fitxategiak errepresentatzen ditu.	25

6.5 taula: *domain.cmap.creator* paketearen egitura

7. KAPITULUA

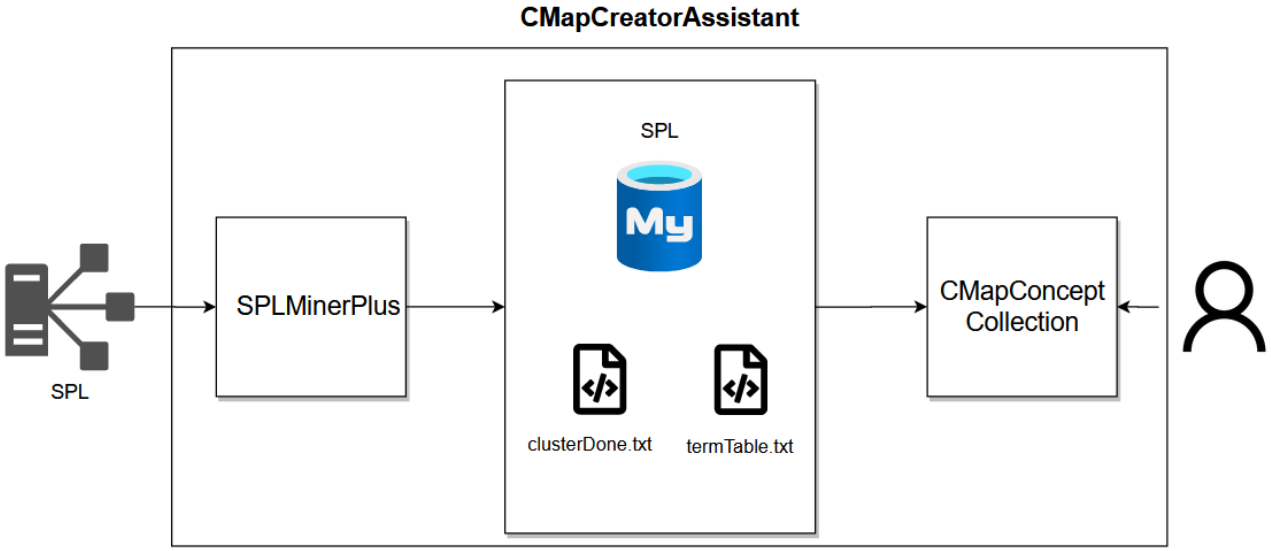
CMapConceptCollection modulua: Kontzeptuen aurkezpena

CMapConceptCollection izena eman zaio terminoen arazketa eginez sortutako kontzeptuak ingeniariari erakutsiko dizkion moduluari. Aurkezpen modulu honen interfazeen garapena baino lehen, interfaze hauen diseinua egiteko prozesua eraman da aurrera, erabiltzaileari ahalik eta esperientzia hoberena eskainiz. Kapitulu honetan, modulu honen diseinu eta garapenerako hartu diren erabaki garrantzitsuenak aurkeztuko dira.

Meatzaritza-lana egiten duen moduluak, aurreko 6. kapituluan azaldu denak, bi emaitza itzuliko ditu. Lehena, terminoen bilduma kontzeptuetan bilduta gehi terminoen informazioa, eta bigarrena, SPL proiektuaren kode fitxategi guztien datuak izango dituen datu-basea. Egitura hau 7.1 irudian aurkezten da.

Lehenengo emaitza, .txt bi fitxategietan egongo da egituratuta eta fitxategi hauek "*clusterDone.txt"* eta "*termTable.txt*" izena izango dute. Beraz, fitxategi hauetako datuak atzitzeko, fitxategi hauek bigarren modulu honetan kargatu behar dira eta ondoren datu hauek kudeatu.

Bigarren emaitza aldiz, MySQL datu-base erlazionalak kudeatzeko sistema batean biltegiratzen da. SPL-aren egitura guztiaren ezaugarriak gordetzen dira bertan, baina aurkezpen modulu honetarako, gehien interesatzen den entitatea *Codefile* (kode edo dokumentazio fitxategia) da. Honek hiru atributu ditu, id bat, fitxategiaren izena eta fitxategi honek izango duen edukia. Horrela SPL-ko fitxategi guztien izenak eta edukiak atzigarri egongo dira une oro. Beraz, modulu honetatik datu hauek atzitzeko, modulua MySQL sistemarekin konektatu behar izan da.



7.1 irudia: Bi moduluen arteko datu-basea eta fitxategiak

7.1 Web interfazearen diseinua

Web modulu honek, SPL-aren kontzeptu bilduma modu garbian aurkezteko hainbat ikuspegi desberdin izango ditu, zehazki, bost.

1. **Hasierako ikuspegia**. Ikuspegi honetan erabiltzaileari sarrera orria aurkezten zaio, eta bertan, arakatutako SPL-aren kodea atzigarri dago.

2. **Kontzeptu eta terminoen ikuspegia**. Ikuspegi honetan arakatu den SPL-aren kontzeptu guztiak listan bistaratzen dira, kontzeptu bakoitzaren ezaugarrien informazio desberdinak aurkeztuz.

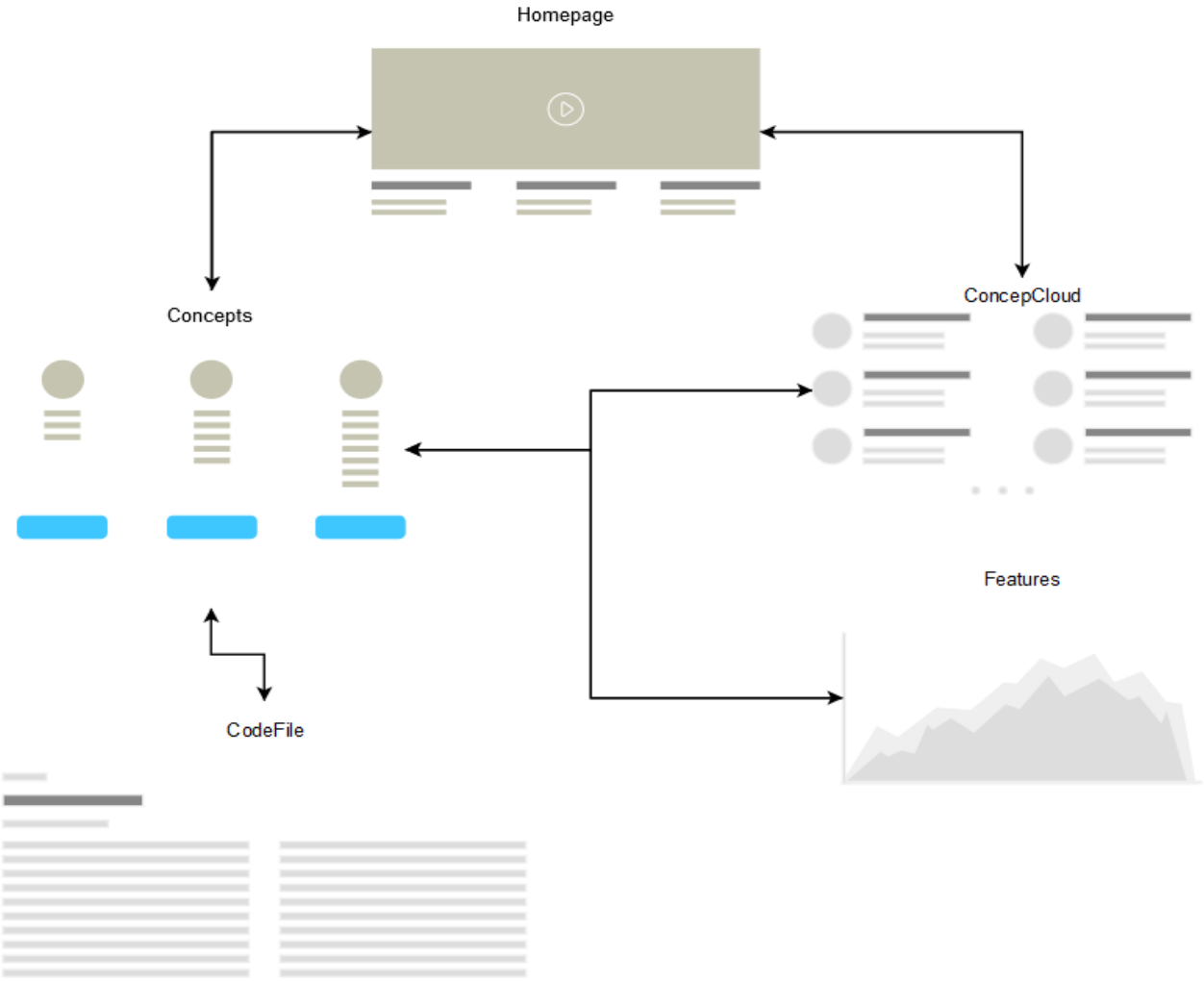
3. **Fitxategien edukiaren ikuspegia.** Ikuspegi hau dinamikoa izango da, kode edo dokumentazio fitxategi desberdinekin lan egiten duelako. Bertan, aukeratutako fitxategiaren edukia aurkeztuko da, beraz, kode fitxategia baldin bada, fitxategi honen kodea aurkeztuko da, aldiz, dokumentazio fitxategia baldin bada, fitxategi honen dokumentazio edukia aurkeztuko da.

4. **Kontzeptu hodeiaren ikuspegia.** Ikuspegi honetan kontzeptuen hodeia aurkeztuko da, eta kontzeptu hauek, kontzeptu bakoitzaren tamaina, honen pisuaren arabera erakutsiko dira. Gainera, erakuslea kontzeptu baten gainetik pasatzerakoan, honen informazioa gehigarria aurkeztuko da.

5. **Terminoaren antzekotasuna ezaugarriekin.** Ikuspegi honetan, grafiko baten bitartez, aukeratutako terminoak, SPL-aren ezaugarri diagramako ezaugarri garrantzitsuenekin daukan antzekotasuna aurkeztuko da. Eta, ikuspegi honetan ere, erakuslea grafikoa honen gainetik pasata, ezaugarrien informazio gehigarriak eskainiko dira.

Ikuspegi hauek, erabiltzaileari aurkeztuko zaizkio eta hobeto ulertzeko, banaka, bakoitzaren egitura eta zer informazio aurkeztuko den hobeto azalduko da. Gainera, web modulu honek eskaintzen dituen ikuspegiak hainbat hizkuntzatan atzigarri daude, eta internalizazio hau E eranskinean azalduko da.

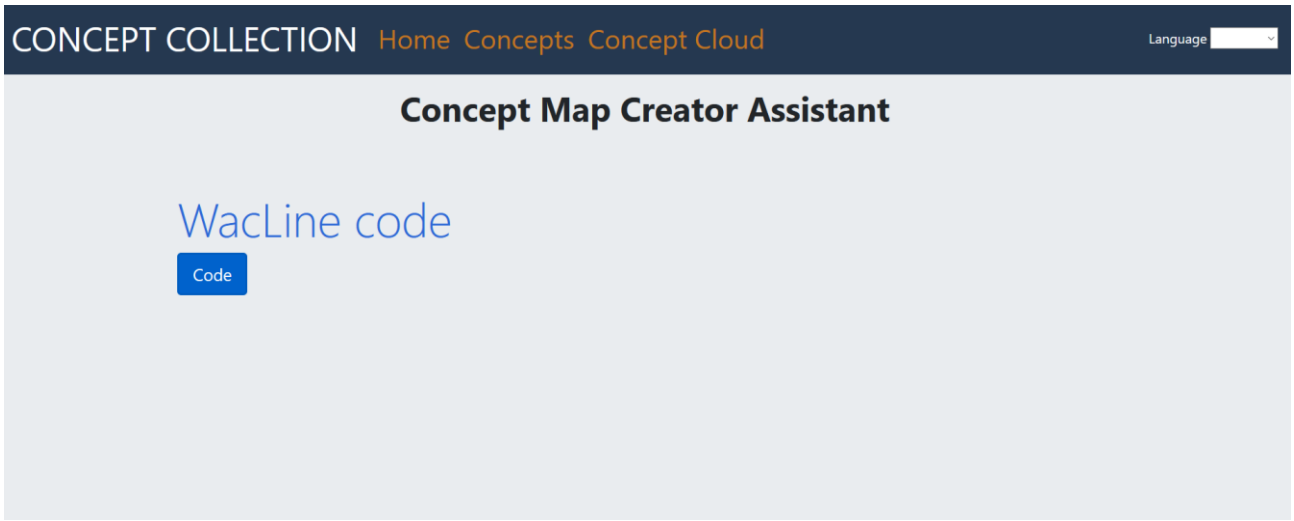
CMapConceptCollection web moduluaren funtzionamendua hobeto ulertzeko, 7.2 irudiko diagraman, ikuspegien arteko nabigazio aukerak erakusten dira, hala ere, eta hurrengo azpiataletan zehatzago azalduko da. *Home* hasierako orritik, kontzeptuak zerrendatuta dauden ikuspegira edota kontzeptuen hodeiaren ikuspegira nabiga daiteke. Eta kontzeptuen zerrendaren ikuspegian egonda, kontzeptu hori eratzen duen terminoa agertzen diren kode fitxategia atzitzeko eta termino bakoitzak SPL-aren ezaugarri garrantzitsuenekin daukan antzekotasuna ikusteko aukera ematen du. Eta kontzeptuen hodeiaren ikuspegitik, kontzeptuen ikuspegira nabiga daiteke, eta alderantziz. Azkenik, kode fitxategiaren ikuspegian egonda, beste ikuspegi guztietara joateko aukera ematen du.



7.2 irudia. Web moduluaren diagrama

7.1.1 Hasierako ikuspegia

Hasierako orri honetan, erabiltzaileak aztertu den SPL-aren kodea atzigarri izango du. Esteka baten bitartez *github* plataformara birbideratuko da, eta bertan kodea aztertzeko aukera izango du, 7.3 irudian ikus daiteken moduan. Gainera, barra nabigatzailea egongo da, web modulu honen ikuspegi guztietan atzigarri izango dena, ikuspegi batetik bestera mugitu ahal izateko.

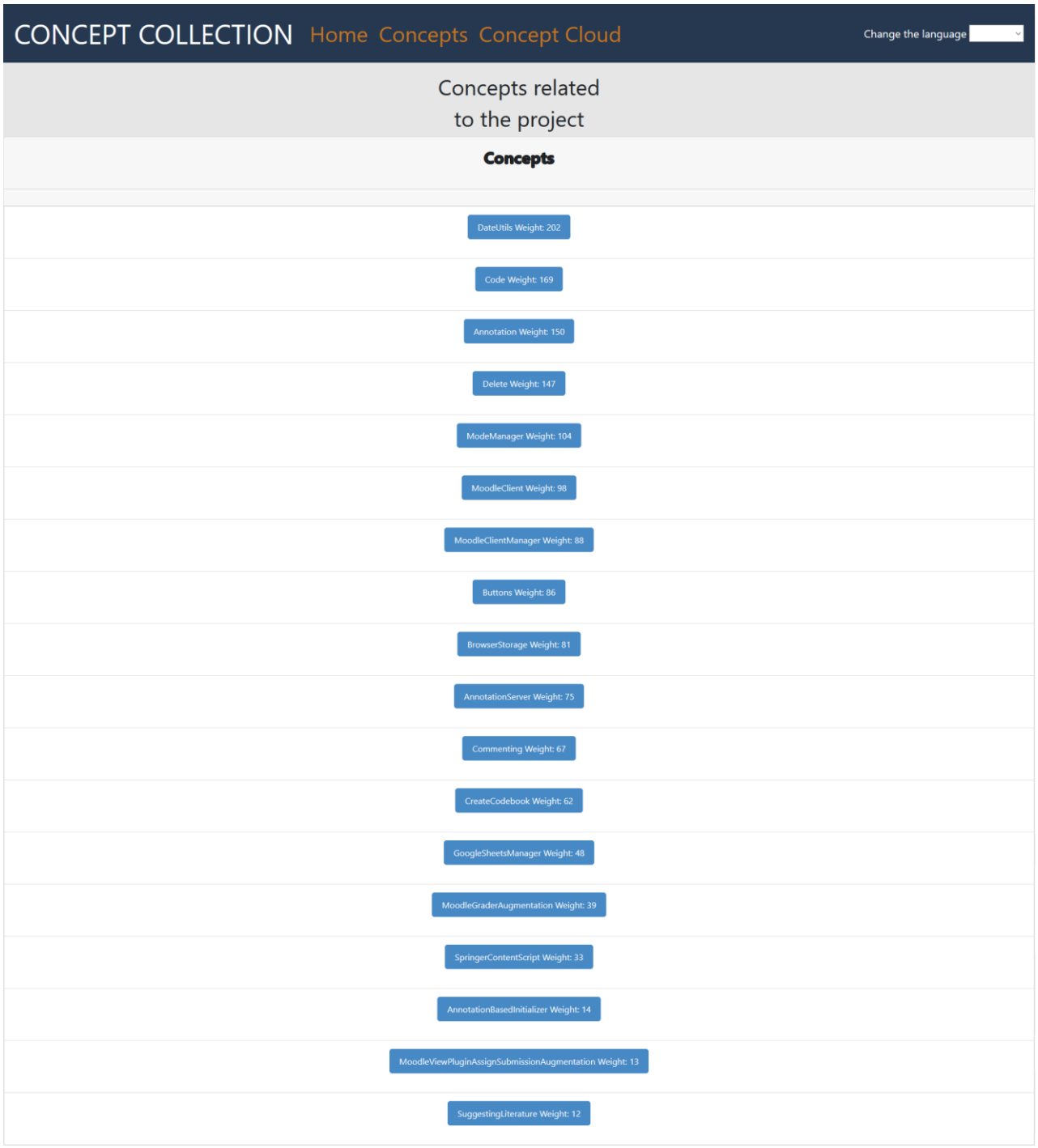


7.3 irudia: Hasierako orria

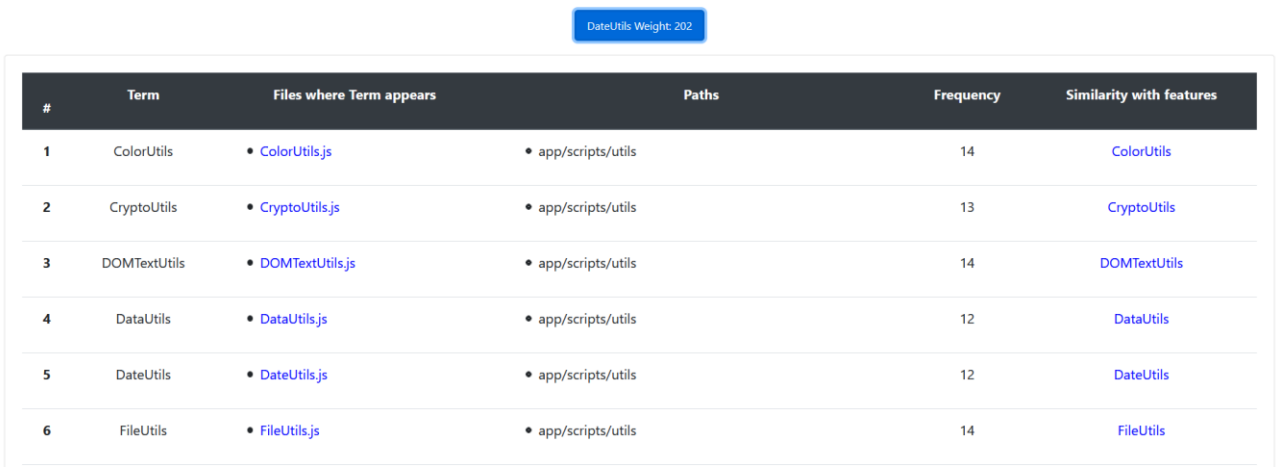
7.1.2 Kontzeptu eta terminoen ikuspegia

Ikuspegi honetan, *SPLMiner* moduluan lortutako kontzeptu guztiak listan bistaratuko zaizkio erabiltzaileari. Kontzeptu zerrenda hau, kontzeptu bakoitzaren pisuaren arabera ordenatuta egongo dira. Hasiera batean, soilik, kontzeptu lista ikusiko du (ikusi 7.4 irudia), non kontzeptu bakoitzaren izena eta pisua erakutsiko diren. Kontzeptu bakoitzean murgiltzeko eta bere inguruko informazio gehiago eskuratzeko, kontzeptu bakoitzak bere burua zabaltzeko aukera emango du. Kontzeptu bat aukeratuz, taula bat erakutsiko da, taula honetan kontzeptua osatzen duten termino guztiak zerrendatuko dira (ikus 7.5 irudia). Termino bakoitzaren honelako datuak aurkeztuko dira:

- Terminoaren izena.
- Termino hau agertzen den fitxategien zerrenda bat. Gainera, fitxategien zerrendan edozein fitxategi aukeratzerakoan, fitxategi honen edukia ikusteko aukera ere ematen da (aurrerago azalduko da).
- Fitxategi bakoitzak SPL proiektuan daukan testuingurua, *path*-a.
- Termino honen agerpen kopuru totala SPL proiektu osoan.
- Terminoak SPL-aren ezaugarri diagramako ezaugarri garrantzitsuenekin daukan antzekotasuna aurkezten duen ikuspegira eramaten duen esteka.



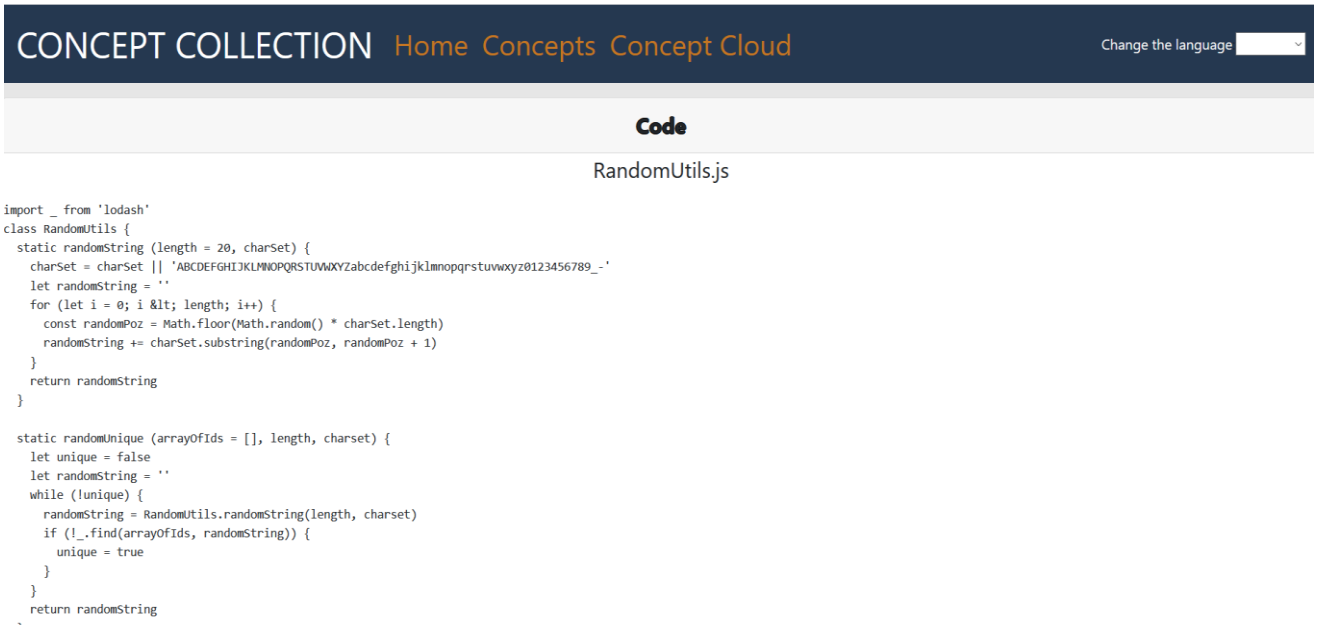
7.4 irudia: Kontzeptu lista



7.5 irudia: Kontzeptuaren egitura

7.1.3 Fitxategiaren edukia aurkezten duen ikuspegia

Ikuspegi honetan, erabiltzaileak aukeratutako terminoa agertzen den fitxategiaren edukia aurkeztuko da. Bertan fitxategiaren izena eta honen edukia eskainiko dira, edozein kontsulta egiteko. Fitxategi hauen edukia, HTML edo beste lengoaia izan daiteke eta 7.6 irudian agertzen den bezala azalduko litzateke.

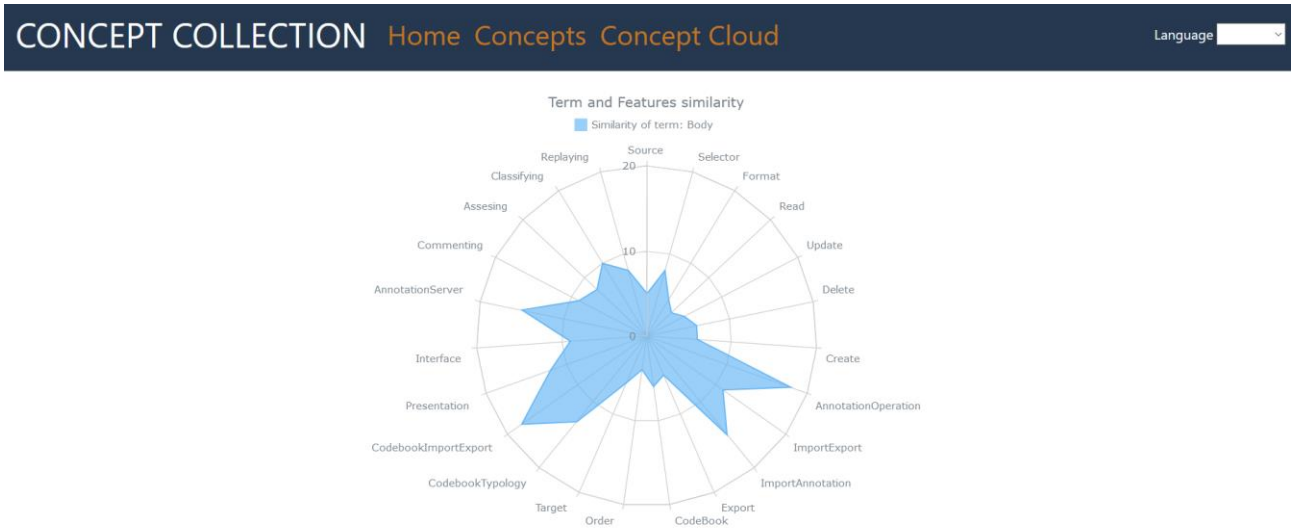


7.6 irudia: Fitxategiaren edukia

7.1.4 Termino eta ezaugarrien arteko antzekotasuna

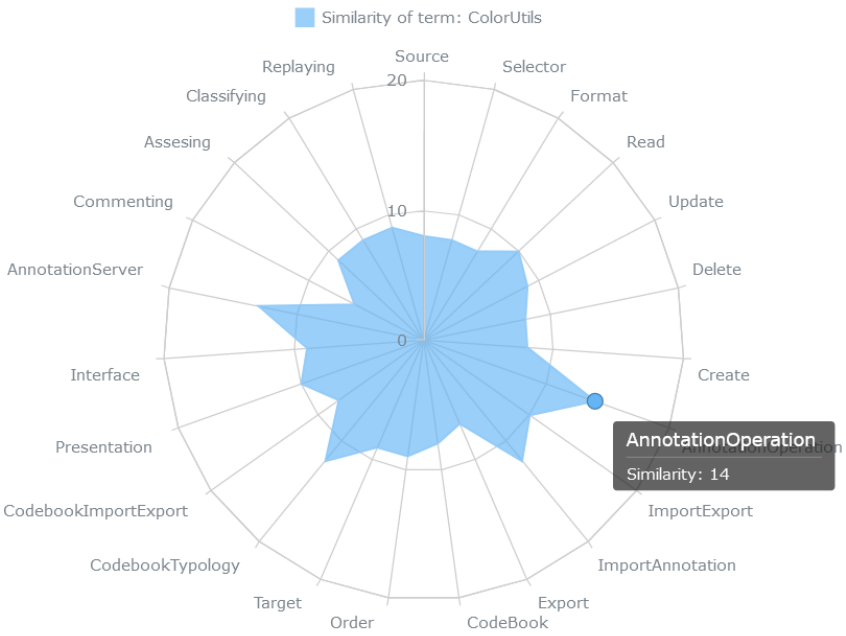
Ikuspegi honetan, erabiltzaileak aukeratutako terminoak SPL-aren ezaugarri garrantzitsuenekin daukan antzekotasuna aurkeztuko da. Antzekotasun hori, *Levenshtein* distantziaren [4] arabera egiten da, eta distantzi hau, hitz batetik abiatuta beste hitz bat lortzeko egin behar diren gutxieneko eragiketen kopurua da. Eragiketa horiek hiru motakoak izan daitezke, karaktere bat txertatzea, ezabatzea edota ordezkatzea, 6.3.4 atalean sakonago azalduta. Eragiketa hau, *SPLMinerPlus* moduluaren 6.3.4 *cluster* atalean erabiltzen den eragiketa berdina da.

Aukeratutako terminoak ezaugarri bakoitzarekin daukan antzekotasuna 7.7 irudian agertzen den grafikoaren arabera aurkeztuko da. Gero eta balio txikiagoa izan, gero eta antzekotasun handiagoa izango du.



7.7 irudia: Terminoaren ezaugarriekiko antzekotasuna

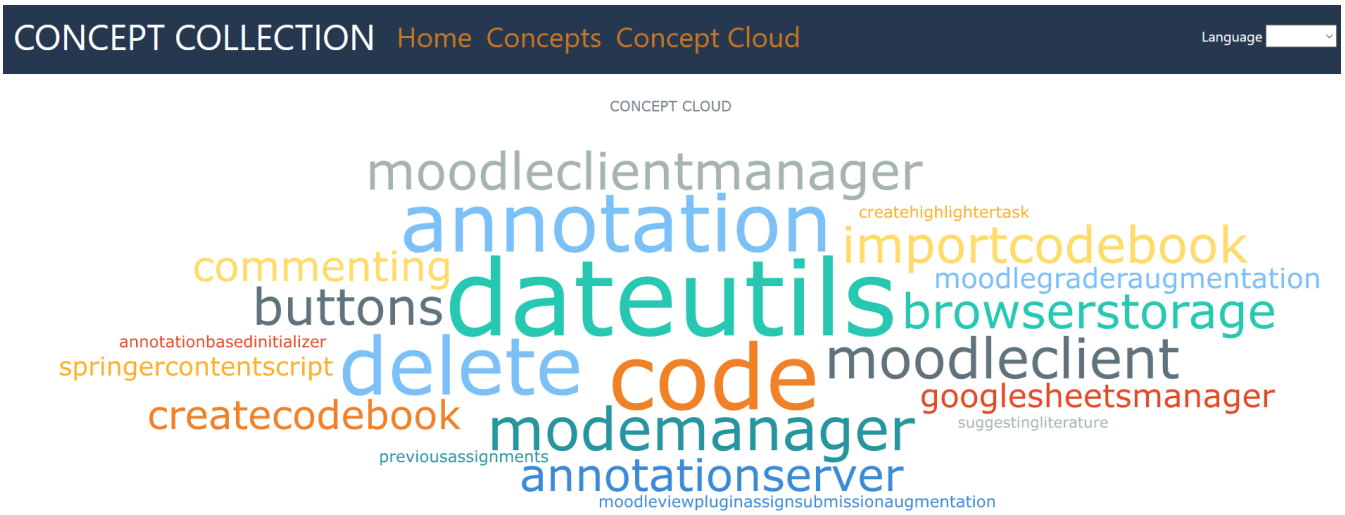
Eta erakuslea grafikoaren gainean jartzerakoan, 7.8 irudian agertzen de bezala, modu zehatzago batean antzekotasunaren balioa erakutsiko du.



7.8 irudia: Terminoaren ezaugarriarekiko antzekotasunaren informazioa

7.1.5 Kontzeptu hodeiaren ikuspegia

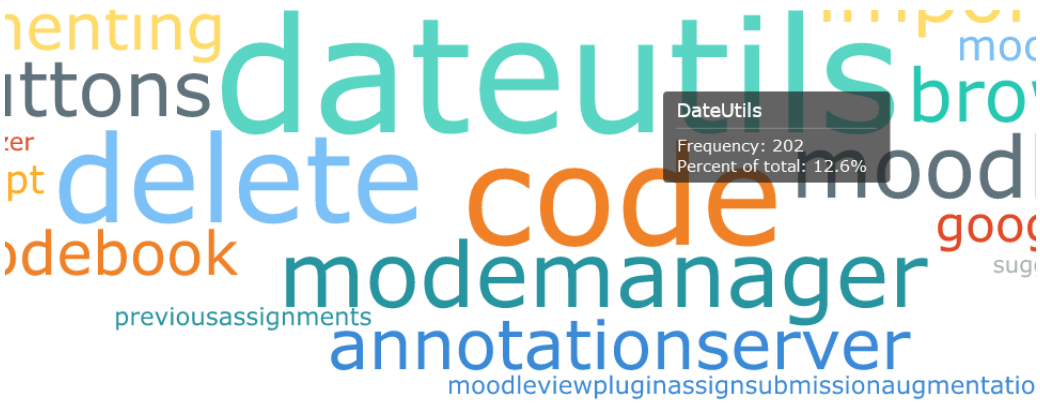
Ikuspegi honetan, erabiltzaileari 7.1.2 azpiatalean aipatutako kontzeptu bilduma aurkeztuko zaio, baina orain, kontzeptu hauek hodei moduko egitura batean bistaratuko dira. Kontzeptu bakoitza bere garrantziaren arabera sailkatuko da, eta pisu handien duen kontzeptuak tamaina handiena izango du eta pisu txikien duen kontzeptua tamaina txikienarekin azalduko da, 7.9 irudia agertzen den bezala.



7.9 irudia: Kontzeptu hodeia

Bestalde, kontzeptu bakoitzaren gainetik sagua pasatzerakoan, 7.10 irudia, kontzeptu honek izango duen pisua eta kontzeptu guztien pisuaren ehunekoaren informazioa ere eskuragarri izango du

erabiltzaileak.



7.10 irudia: Kontzeptuaren informazioa

7.2 *CMapConceptCollection* moduluaren inplementazioa

CMapConceptCollection moduluaren garapena aurrera eramateko, Spring eta SpringBoot tresnak erabili dira. Tresna hauek hurrengo egitura jarraitzen dute, proiektuaren barruan bi direktorio nagusi bereizten dira, *java* eta *resources*, eta hurrengo azpiataletan karpeta nagusi hauen egitura aurkeztuko da. Gainera, web modulu honen konfigurazioa E eranskinean azaltzen da, internazionalizazioarekin batera.

7.2.1 *Java* direktorioa

Direktorio honetan, programa nagusia aurki daiteke, eta honek *launcher* izeneko klasea abiarazten du. *Launcher* klase honetan ikuspegi guztien nabigazioa kudeatzen da. Hurrengo 7.1 taulan, web modulu honen garapenerako erabili diren Java klase bakoitzaren zenbait datu aurkeztuko dira.

Klasearen izena	Edukia	Dependentziak	Azalpenak	Kode lerro kopurua
WebApplication	Programa nagusia.		Bertan web moduluaren exekuzioa gauzatzen da, beste klaseei dei eginez, adibidez, *Launcher* klaseari.	25

Launcher	Ikuspegi guztien kontrola.	WebApplication Codefile Internationalization CMapData	Klase honetan ikuspegi guztien kontrola gauzatzen da. Eta zenbait kalkulu egiten dira ikuspegietan datuak modu egokian aurkezteko.	328
CodeFile	CodeFile klasearen atributuak, eraikitzaileak eta beste zenbait metodo.	CMapData	Klase honetan, datu-baseko *codefile* taularen balioak errepresentatzen dira.	63
CMapData	Datu-basearekin konexioak egiteko metodoak.	CodeFile	Klase honetan, web moduluaren eta datu-basearekiko konexio eragiketak gauzatzen dira.	16
Internationalization	Internazionalizazio rako beharrezko konfigurazioak.		Klase honetan, ikuspegi guztiak hainbat hizkuntzatan aurkezteko behar diren eragiketan gauzatzen dira.	37

7.1 taula: *CMapConceptCollection* moduluaren Java klaseak

7.2.2 *Resources* direktorioa

Direktorio honen barruan beste bi direktorio garrantzitsu nagusitzen dira, *static* eta *templates*. *static* direktorioan ikuspegietan datuak erakusteko fitxategi lagungarriak daude, eta fitxategi hauek hurrengo 7.2 taulan aurkeztuko dira.

Azpidirektorioa	**Fitxategiaren izena**	**Kode lerro kopurua**	**Edukia**
css	anychart-font.min.css	1191	Fitxategi hauetan ikuspegien aurkezpena eta estiloa konfiguratzen da, html elementuen tamaina, koloreak, kokapena, etab.
	anychart-ui.min.css	1727	
	bootstrap.min.css	9893	
	style.css	217	
files	clusterDone.txt	21	Fitxategi hauetan SPLMiner modulu meatzarian lortutako emaitzak biltegiratzen dira, kontzeptuak eta terminoak.
	termTable.txt	119	

js	anychart-base.min.js	2125	Fitxategi hauetan ikuspegietan exekutatzen diren metodo guztiak gordetzen dira.
	anychart-export.min.js	109	
	anychart-radar.min.js	86	
	anychart-tag-cloud.min.js	94	
	anychart-ui.min.js	120	
	sevenshteins.js	91	
	showConcepts.js	24	

7.2 taula: *static* direktorioko fitxategiak

Eta *templates* direktorioan ikuspegien .html luzapeneko fitxategiak daude, eta fitxategi hauen egitura hurrengo 7.3 taulan aurkeztuko da.

Fitxategiaren izena	Kode lerro kopurua	Edukia
base.html	117	Fitxategi honetan ikuspegi guztiek komunean dituzten elementuak definitzen dira, nabigazio barra, hizkuntza aldatzeko zabalgarria, etab.
conceptCloud.html	89	Kontzeptuen hodeia bistaratzeko elementuak definitzen dira.
codeFile.html	46	Dokumentazio edo kode fitxategiaren edukia erakusteko elementuak definitzen dira.
concepts.html	246	Kontzeptuak eta horien datuak aurkezteko elementuak definitzen dira.
features.html	119	Termino bakoitzak SPL-aren ezaugarriekin daukan antzekotasuna erakusteko elementuak definitzen dira.
home.html	37	Hasierako orriko elementuak definitzen dira.

7.3 taula: *templates* direktorioko fitxategiak

8. KAPITULUA

Proiektuaren erronkak eta zailtasunak

Kapitulu honetan proiektuaren garapenean zehar aurkitutako erronkak eta zailtasunak aurkeztuko dira, eta hauek nola ebatzi diren azalduko da. Hausnarketa hau egitea garrantzitsua da, proiektuaren dimentsioa zein izan den hobeto ulertzeko.

- Proiektuaren irismenean, 2.1 atalean, zehaztutako betekizunetan aipatu bezala, sarrera moduan erabiltzen den SPL-aren meatzaritza-lana egiteko, ez da kodea hutsetik garatzen hasi, Iosu Salaberriren *SPLMiner* proiektuaren kodea erabili da. Honek abantaila batzuk baditu, jadanik kodearen zati baten garapena baitago, eta honek, epe luze batera, onurak ekar ditzake, adibidez, denbora eta esfortzua aurreztea. Baina, beste alde batetik, konplikazio ugari ekar ditzake, kode hori ulertu eta erabiltzerako orduan. Izan ere, kasu honetan, ez da kanpo kode moduan gehitu, baizik eta oinarri moduan erabili da. Hau da, kode honetara moldatu behar izan da garapen osoa eta horretarako, kode honen funtzionamendu eta egitura osoa ondo ulertu behar izan da. Hau lortzeko, Gradu Amaierako Lan honen hasieran, 15-20 ordu inguru eskaini zitzaion kodearen funtzionamendu osoa ulertzeko. Hain zuzen ere, ez delako berdina kodeak egiten duena gainetik ulertzea, edo kode hori proiektu propioaren betekizun eta helburuak betetzeko moldatu behar izatea.

- Sarrera moduan erabilitako kode fitxategien meatzaritza-lana egiteko, hasieran kode fitxategiena soilik egin behar zela adostu zelako, *SPLMiner* hau erabiltzea zehaztu zen. Beraz, garapen guztia *SPLMiner* proiektura moldatzea egokitu zen. Baina, lortu nahi ziren emaitzak lortzen ez zirela ikusterakoan, sarrerako SPL-aren dokumentazio fitxategien meatzaritza-lana ere egitea erabaki zen, eta lan hau, ez zuen *SPLMiner*-ek egiten. Horregatik, bi aukera proposatu ziren; *SPLMiner* honen erroa aldatzea eta oinarrian zegoen kodea moldatzea, edo dokumentazio fitxategiak soilik arakatzen zituen beste meatzari bat sortzea. Kasu honetan, bigarren aukera hautatu zen, izan ere, *SPLMiner* horren barru-barruko kodea aldatu behar izango litzateke, eta honek, arrisku handia zekartzan, meatzaritza funtzio gehigarri hau gehitu nahian, jadanik eskaintzen zuen meatzaritza funtzioa izorratzeko arriskua zegoelako. Beraz, *SPLMiner* meatzariarekiko guztiz independentea zen eta proiektu honen hasieran aurreikusten ez zen beste meatzari berri bat garatu behar izan da, eta, bi meatzarien bildura eginez, bai kode eta bai dokumentazio fitxategien meatzaritza-lana lortu behar izan da.

- *SPLMiner* proiektuan sortutako datu-basea ere berrerabili da. Eta 6.2.1 atalean esan bezala, datu-base honi ere moldaketa batzuk egin behar izan zaizkio. Hasteko, taula baten egitura aldatu behar izan da, eta SQL sententzia guztiak aldatu behar izan dira datu berriak gordetzeko. Baina hau ez da izan arazo handiena, arazo handiena, SQL sententzia horiek exekutatzerakoan agertu da. Lehena, datu-basean elementuak gordetzerako orduan, elementu bakoitzak id bakar eta desberdin bat izan behar zuela, eta bazeuden zenbait elementu desberdin id identifikadore berdinak zituztenak. Eta bigarrena, SQL sententziak exekutatzen saiatzerakoan, zenbait elementuren id atributuaren balioak luzeegiak zirela, eta ezin ziren biltegiratu. Taulak sortzerakoan, atributu bakoitzari esleitutako balioari espazioren tamaina zehatz bat esleitzen zaio, eta kasu honetan txikiegiak ziren.

- *CodeMinerPlus* klasea, klase garrantzitsuenetarikoa izan da, eta hau bere moldaketan islatu da. Izan ere, kode fitxategi hau izan da gehien moldatu dena, betekizunak lortzeko egin beharreko eragiketa gehienak bertan egin eta kudeatu direlako. Hasieran, Iosu Salaberriren proiektu bukatuta hartu zenean, kode fitxategi honek 523 kode-lerro zituen eta behin proiektu honetako aplikazio bukatutakoan 1116 lerro ditu. Esan beharra dago, beste zenbait klasetan ere kode garapena aurrera eraman dela, baina kasu hauetan, hutsetik sortutako klaseak izan dira, adibidez, termino lista ordenatzeko, *cluster* eragiketa egiteko, edota terminoen arazketa egiteko, etab.

- Sarrera moduan sartzen den SPL-aren kontzeptu bilduma lortzeko, kode fitxategi guztietako hitz gako guztiak aztertzen dira, baina azterketa honetan zenbait erabaki hartu behar izan dira, eta erabaki hauek hartzeko, proba askoren ondorioz lortutako emaitzak aztertu dira. Aurretik aipatu bezala, kode fitxategien hitz gako guztiak aztertu izan dira, baina hauek aztertzerakoan, emaitza esanguratsua lortzen ez zela arazo bilakatu zen, eta honi soluzioa emateko, hitz gako hauek, bakoitza bere aldetik modu independentean edota beste batzuekin konbinatuz probak egin behar izan dira. Proba hauen azterketak, lan karga handia suposatu du eta 12 ordu eskaini zaio.

  Hasiera batean, termino mota guztiak maila berean aztertzearekin hasi zen, aldagaiak, metodoak eta klaseak modu berdinean aztertuz. Lortu nahi ziren emaitzak lortzen ez zirela ikusita, aldaketaren bat sartzearen beharra ikusi zen. Aldaketa hau, termino mota bakoitza modu desberdinean tratatzearekin zerikusia zuen. Eta hauekin ere probak egin ziren, emaitza hobeagoak lortzeko asmoarekin. Esan beharra dago, aldagaiak definitzeko orduen ere, hiru modu ezberdin existitzen direla, aldagai konstanteak, aldagai lokalak eta aldagai globalak. Hauek ere, termino mota bezala definitzea erabaki zen, hortaz, bost termino motarekin aurkitu ginen, aldagaiak definitzeko hiruak, metodoak eta klaseak.

  Azterketa honek, aldaketa ugari izan ditu soluzioaren garapenean zehar, lortutako emaitzak ez zirelako lortu nahi zirenaren antzekoak. 8.1 taulan agerian jartzen dira terminoen azterketan jasandako aldaketak, eta azterketa hauen probak eranskinetan gehituko dira, zehazki B eranskinean. Azkenean, taulan agertzen den azken saiakera aukera da, hau izan delako emaitza hobeenak lortzen dituen emaitza.

Saiakera	Kontuan hartutako elementuak	Azalpena
1. saiakera	Aldagaiak, metodoak, funtzioak, paketeak	Elementu guztiak maila berean aztertuz.
2. saiakera	Aldagaiak (*var*, *let*, *const*), metodoak, paketeak	Aldagai konstanteak, lokalak eta orokorrak bereiziz guztira 5 termino mota zehaztuta.
3. saiakera	Aldagaiak, (*var*, *let*, *const*), metodoak, funtzioak, klaseak	Paketeak alde batera utzi eta klase izenak kontuan hartuta.
4. saiakera	Aldagaiak, metodoak, funtzioak, klaseak	Klase izenei balio handiagoak aplikatuz.
5. saiakera	Aldagaiak, metodoak, funtzioak, klaseak	Klase izenei balio handiagoak aplikatuz eta arazketa prozesua zorrotzagoa egiten
6. saiakera	Klase izenak era dokumentazio fitxategiak	Dokumentazio fitxategien meatzaritza-lana egitea.

8.1 taula: Terminoen azterketaren saiakerak

- *CMapConceptCollection* moduluan *SPLMiner* modulu meatzarian lortutako kontzeptuak erabiltzaileari aurkezteko, bi modutan egin da. Lehena, datu soilak zerrendatuz, kontzeptu bakoitzaren terminoak eta hauen informazio desberdinak eskainiz, eta bigarrena, kontzeptuen hodei baten bitartez, kontzeptu bakoitzak duen garrantziaren arabera, tamainarekin jolasten.

  Funtzionalitate hauek lortzeko kanpo liburutegiak erabili dira, esaterako *anyChart* [2], eta kanpo liburutegi honen integrazioa aztertu behar izan da. Datu-basean gordeta zeuden datuak, web modulutik atzitu eta egituratu egin behar dira, erabiltzen diren kanpo liburutegiak erabili ahal izateko, adibidez, datuek JSON [13] egitura jarraitu behar zuten, 8.1 irudian agertzen den bezala.

8.1 irudia: Kanpo liburutegiak erabiltzeko JSON egitura

- Gradu Amaierako Proiektu honen hasieran ez zen pentsatu, bigarren moduluak hainbat hizkuntzen integrazio izango zuela. Hala ere, garapenak aurrera egin ahala, garatzaileak garrantzizkoa ikusi zuen hainbat hizkuntza ezberdinen integrazioa gehitzea. Eta hasierako betekizunetan egon ez arren, web modulua internazionalizazioa aplikatzea erabaki zen.

  Prozesu honen garapena 7.2 atalean azaltzen da. Internazionalizazio egitura aplikatzeko, web moduluaren oinarrizko egitura aldatu behar izan da, zenbait konfigurazio gehituz. Konfigurazio hauek gehitzea denbora eta esfortzu zehatza exijitzen du, baina behin konfigurazioa definitua, edozein hizkuntza gehitzeko aukera ematen du, zenbait pausu zehatz jarraituz. Arazo handiena, eskuz sortu beharrean, zenbait funtzioren bitartez sortzen ziren HTML elementuen internazionalizazioa izan da. Elementu hauetan hizkuntza desberdinak aplikatzeko, aurretik aipatutako konfigurazioa ez zelako nahikoa. Konfigurazio hori erabilita, beste zenbait eragiketa gehigarri aplikatu behar izan dira, web moduluaren elementu guztien internazionalizazioa lortzeko.

9. KAPITULUA

Jarraipen eta kontrola

Proiektuaren garapen guztian zehar, honen jarraipen eta kontrola egin da, hasieran diseinatutako planifikazioa jarraitzen eta behar izan diren aldaketak gehitzen, Gradu Amaierako Lan honen bukatze zuzen bat ziurtatzeko. Eta, kapitulu honetan, jarraipen eta kontrol horri esker proiektuak jasandako desbiderapenei buruz hitz egingo da.

9.1 Irismenaren desbiderapena

Gradu Amaierako Lan honen irismenak ez du aldaketa sakonik jasan. Egia da, proiektuaren garapenean zehar zenbait helburu aldatzen joan direla, zenbait betekizun desberdin gehitzen joan direlako, baina 2. kapituluko plangintzan, zehaztutako irismen eta helburu berdinak jarraitu dira, eta gainera, kapitulu honetan diseinatutako LDE diagrama (ikus 2.1 irudia) ez da aldatu.

Hala ere, esan beharra dago, hasierako plangintzan zehaztutako helburuetan, bi modulu garatuko zirela ebatzi zen, *SPLMinerPlus* eta *CMapConceptCollection*. Eta garapenari dagokionez, *SPLMinerPlus* moduluak *CMapConceptCollection* moduluaren pisu berdina edo handiagoa izango zuela zehaztu zen, baina hau garapenean zehar aldatzen joan da. Izan ere, tutorearekin izandako bileretan, *CMapConceptCollection* web moduluaren betekizun berrien proposamenak agertzen joan dira, eta proposamen hauek betetzeko asmoarekin funtzionalitate gehigarriak gehitu zaizkio bigarren modulu honi. Honek, garapen kargari dagokionez, bigarren moduluak lehenak baino lan karga handiagoa suposatu du, eta hau, garapenari eskainitako denboran islatu da, kapitulu honen azken atalean aztertuko dena.

Garapenaren bidean gehitutako funtzionalitate berri hauek hurrengokoak izan dira.

- ***SPLMinerPlus* moduluaren hedapena.** Dokumentazio fitxategien meatzaritza-lana egiteaz arduratzen den meatzariaren garapena. Hasieratik, *SPLMiner* modulua moldatu beharra zegoela argi zegoen, baina dokumentazio fitxategiak aztertzeaz arduratzen zen meatzaria garatzea ez zen aurreikusi.

- ***CMapConceptCollection* moduluaren hedapena.** *SPLMinerPlus* moduluan lortutako kontzeptu bilduma, kontzeptu bakoitzaren garrantziaren arabera hodei itxurako egituran aurkeztea aukera ematen duen ikuspegiaren garapena. Hasierako plangintzan, kontzeptuak eta hauen zenbait informazio aurkeztea definitu zen, lista moduan, baina hedapen gehigarri honekin, kontzeptuen pisua modu grafikoago batean adierazteko aukera ematen da.

- ***CMapConceptCollection* moduluaren hedapena.** *SPLMinerPlus* moduluan lortutako termino guztiak jatorrizko SPL-aren ezaugarri garrantzitsuenekin duten antzekotasuna grafiko baten bitartez aurkezten duen ikuspegiaren garapena.

- ***CMapConceptCollection* moduluaren hedapena.** Web moduluaren internazionalizazioaren aukera.

9.2 Arriskuak eta kalitatea

Atal honetan, proiektuak jasandako arriskuen eta proiektuaren kalitatearen kudeaketa azalduko da.

9.2.1 Arriskuen kudeaketa

Proiektuaren garapena hasi aurretik, gerta zitezkeen zenbait arrisku definitu ziren, 2.3 atalean, hauek gertatuz gero, hauen aurrean nola jokatu aurreikusi ahal izateko.

Egia esan, aurreikusitako arrisku batek soilik eragina izan du proiektu honen garapenean. Arrisku hau, aurreikusi zen lehena izan da, oinarri moduan erabili behar den softwareak zenbait arazo izatea. 8.1 atalean aipatzen den moduan, berrerabili nahi izan den datu-basea ez zen guztiz funtzionala eta zenbait akats zituen. Baina, zenbait erreparazio eginda funtzionala izatea lortu da, beraz ez du atzerapenik suposatu proiektuaren garapenean.

Beste alde batetik, datu-basearekin lan egiterako orduan, MySQL datu-baseak kudeatzeko sistemak eskaintzen duen lan erremintarekin lan egiten, erreminta honek errore potolo bat itzuli zuen, eta errore honi buruz erreminta honen garatzaileei berri emateko eskatzen zuen. Arazo hau hurrengokoa zen, “MySQL Workbench has encountered a problem. No se puede obtener acceso al objeto desechado. Nombre del objeto: ‘HUDForm’. Arazo hau konpontzeko bide bakarra, MySQL sistema guztia konputagailuan berrinstalatzea izan da, eta datu-base guztia berriro egituratu behar izan da, bi ordu inguruko lana. Ordu gehigarri hauek kudeatzeko, planifikazioan aipaturiko (ikusi 2.3.1 atala) errekuperazio-tarteak erabili dira, eta errekuperazio tartea hauek asteburuetan lan egiten burutu dira.

Aurreikusitako beste arriskuek ez dute eragin zuzenik izan proiektuaren garapenean. Izan ere, hasieran diseinatutako planifikazioa jarraitu da, ez da egon teknologien ikasketen desbiderapen garrantzitsurik, ez da datuen galerarik gertatu eta azkenik, ez da izaera pertsonaleko arazorik izan.

Hala ere, aurreikusi gabeko arrisku bat agertu da. Proiektuaren hasieran ez zen aurreikusi proiektua bukatu baino lehen garatzailea lanean hastea, eta honek, modu oso zuzen batean eragin ziezaiokeen proiektuaren garapenari. Garatzaileak lana egiteko aukera izan du eta proiektuaren garapenarekin batera eramatea erabaki da. Erabaki honetara heltzeko, aurretik, proiektuaren egoeraren azterketa, eta aktibitate bi hauek aldi berean egitearen egingarritasunaren azterketa ere egin da. Lana urtarriletik hasita eta goizez izango zela kontuan hartuta, eta Gradu Amaierako Lan honi lehentasuna emanez, bi ekintzak modu paraleloan egiteko erabakia hartu da.

9.2.2 Kalitatearen kudeaketa

Kalitate planari dagokionez, 2.3.2 atalean zenbait funtzionalitateren kalitate plana definitu zen, eta proiektuaren garapenean zehar hauen azterketa egiten joan da. Funtzionalitate bakoitzaren maila aztertzen joan da, eta modu honetan, funtzionalitate baten maila batera heltzean, hurrengora heltzeko behar ziren gehigarrien egingarritasuna aztertzen zen. Esan beharra dago, maila batek aurreko mailetan definitutako funtzionalitateak barneratzen dituela. Behin proiektua bukatuta, hurrengokoa da funtzionalitate hauen azterketa finala (beltzez adierazten da lortu den maila):

- Meatzaritza-lana
  - Txarra: Ez da kode eta dokumentazio fitxategi guztien terminoen azterketa egiten.
  - Ona: Kode eta dokumentazio fitxategi guztien termino guztiak aztertzen dira.
  - **Oso ona: Kodeko edo dokumentazioko terminoak bereizten ditu.**

- *Cluster*-a
  - Txarra: Taldekatze prozesua ez da egiten hitzen antzekotasunaren arabera
  - **Ona: Taldekatze prozesua terminoen antzekotasunaren arabera egiten da.**
  - Oso ona: Taldekatze prozesua esanahiaren arabera egiten da.

- Kontzeptuen zerrendaketa
  - Txarra: Ez da kontzeptuen zerrendaketa argia egiten.
  - Ona: Kontzeptuen zerrendaketa argia egiten da. Eta kontzeptu bakoitzaren terminoen informazio desberdinak eskaintzen dira, izena, agerpen kopurua eta terminoa agertzen den fitxategien zerrenda.
  - **Oso ona: Kontzeptu bakoitzak SPL-aren ezaugarri diagramako zer ezaugarrirekin zerikusia daukan azalduko da, eta gainera, termino bakoitza agertzen den fitxategien kodea ere atzigarri egongo da.**

- Kontzeptuen hodeia
  - Txarra: Kontzeptu lista ez da hodei itxuran agertuko.
  - Ona: Kontzeptu lista hodei itxuran agertuko da. Bakoitzaren tamaina, jatorrizko proiektuan daukan pisuaren araberakoa izango da.
  - **Oso ona: Kontzeptu bakoitzaren gainetik erakuslea pasatzerakoan, kontzeptu honen pisua eta beste kontzeptuekiko izango duen pisuaren ehunekoa azalduko da.**

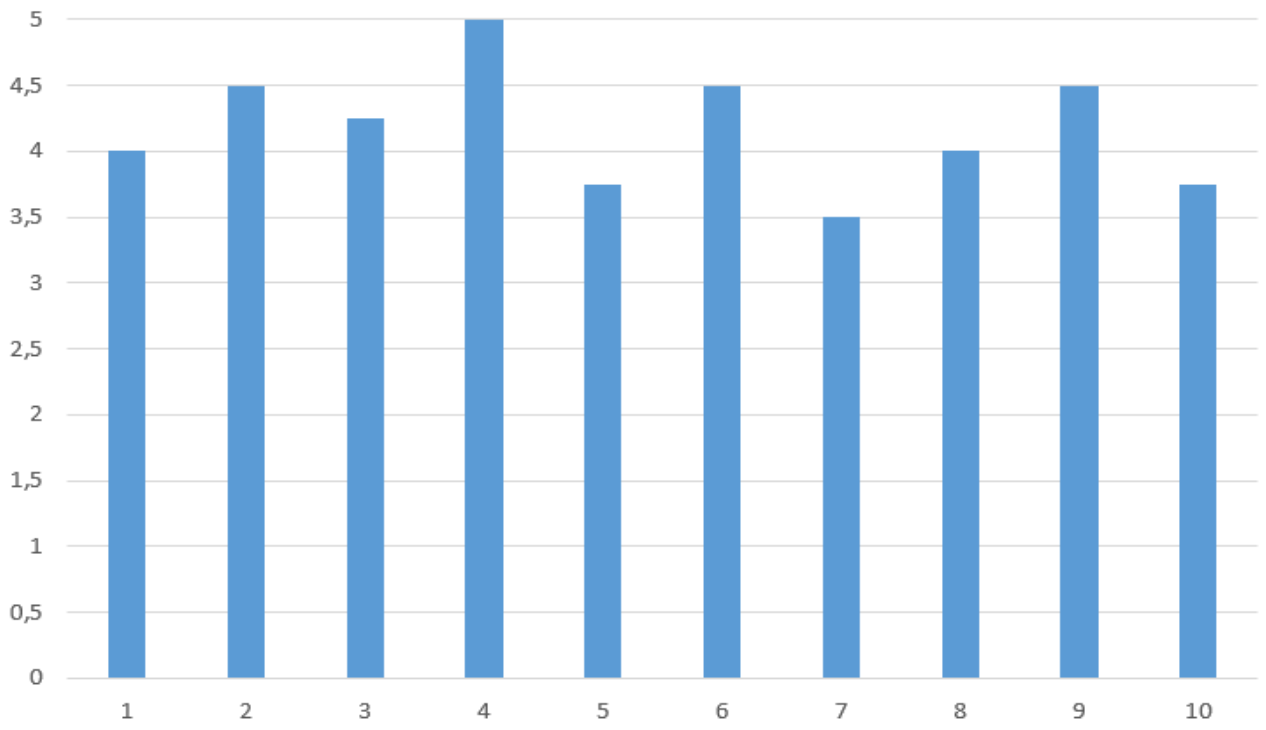
Adibidez, *cluster* eragiketaren maila “Ona” geratu da, maila “Oso ona” lortzeko egingarritasuna, denbora faltaren ondorioz, ez delako bideragarria izan, eta hau, proiektuaren hobekuntza aukeretan, 10.3 atalean, gehitu da.

Gainera, aplikazioren kalitatea neurtzeko, Software Produktu-Lerroak eta *WacLine* SPL-a ondo ezagutzen duten zenbait ikertzaileei aplikazioaren inguruko formulario bat erantzutea eskatu zaie, eta hurrengokoa izan da lortutako emaitza.

Formulario honetan hainbat galdera egin dira, zehazki 11, eta hurrengokoak dira:

1. Lortutako kontzeptuak baliagarriak al dira kontzeptu mapa sortzerako orduan?
2. Kontzeptu bakoitzaren emandako informazioa (termino taula) egokia al da?
3. Kontzeptuen hodeiak eskaintzen duen informazio erabilgarria ikusten duzu?
4. Interesgarria ikusten duzu terminoa agertzen den fitxategiaren edukia eskuragarri edukitzea?
5. Terminoen inguruan emandako informazioa egokia al da?
6. Termino eta SPL ezaugarrien arteko antzekotasuna erakusten duen grafikoa erabilgarria dela uste duzu?
7. Web aplikazioan erabilitako nabigazio barra egokia al da?
8. Web aplikazioan zehar erabilitako menuak egokiak al dira?
9. Nola ikusten duzu internazionalizazioren aukera?
10. Web aplikazioan erabilitako letra mota egokia al da?
11. Orokorrean nola kalifikatu zenuke web aplikazioaren kalitatea?

Lehenengo 10 galderak erantzuteko 1etik 5erako eskala lineala eskaintzen da; 1 kalifikazio txarrena, eta 5 kalifikazioa onena errepresentatuz. Lortutako batezbesteko emaitzeak grafiko batera irauli dira, eta 9.1 irudian aurkezten dira.



9.1 irudia: Formularioaren batezbesteko emaitzak

11. galderak hurrengo formatua jarraitzen du, eta formularioan lortutako erantzun guztietan lehen aukera izan da aukeratua:

- Orokorrean nola kalifikatu zenuke web aplikazioaren kalitatea?
  - **Diseinu erakargarria. Informazio egituratua eta garbia.**
  - Diseinu ez erakargarria. Informazio gehiegi eta ez egituratu.
  - Informazio falta.
  - Web nabigazio motela.

Bestalde, formulario honek azken galdera moduan, *CMapCreatorAssistant* aplikazioaren edozein iruzkin edota komentario uzteko aukera ematen zuen, eta erabiltzaileek bertan utzitako proposamenak erabilgarriak izan dira aplikazioaren funtzionalitateak hobetzeko. Adibidez, horietako proposamen batzuk nabigazio barran koloreen aldaketa edo termino taularen azken zutabeko balioaren aldaketa izan dira, aldaketa hauek aplikazioren nabigazioa hobetuko zutelakoan, aldaketa horiek aplikatu egin dira.

9.3 Plangintzaren desbiderapena

Atal honetan Gradu Amaierako Lanaren plangintzaren desbiderapena aurkezten da. Desbiderapen hau hurrengo 9.1 taularen bitartez azalduko da, non, ataza bakoitzari balioetsi eta dedikatu zaion denbora erreala adieraziko dira. Gainera, taularen azken zutabean, ataza bakoitzak izandako desbiderapena aurkeztuko da.

Ataza	Balioespena	Dedikazio erreala	Desbiderapena
Ezagutzaren kudeaketa	**30**	**40**	**+10**
EE	30	40	+10
Diseinua	**55**	**61**	**+6**
AA	25	18	-7
WE	15	27	+12
WI	15	16	+1
Garapena	**90**	**102,5**	**+12,5**
M	30	30.5	+0.5
KT	35	21	-14
KB	25	51	+26
Probak	**35**	**40**	**+5**
DA	15	12	-3
DB	10	10	+0
WS	10	15	+5
Kudeaketa	**35**	**20,5**	**-14,5**
PL	20	8	-12
BI	10	8,5	-1,5
JK	5	4	-1
Dokumentazioa	**90**	**112**	**+22**
DO	90	107	+22
Defentsa	**10**	**8**	**-2**
DM	8	6	-2
DA	2	2	+0
Totala	**345**	**384**	**+39**

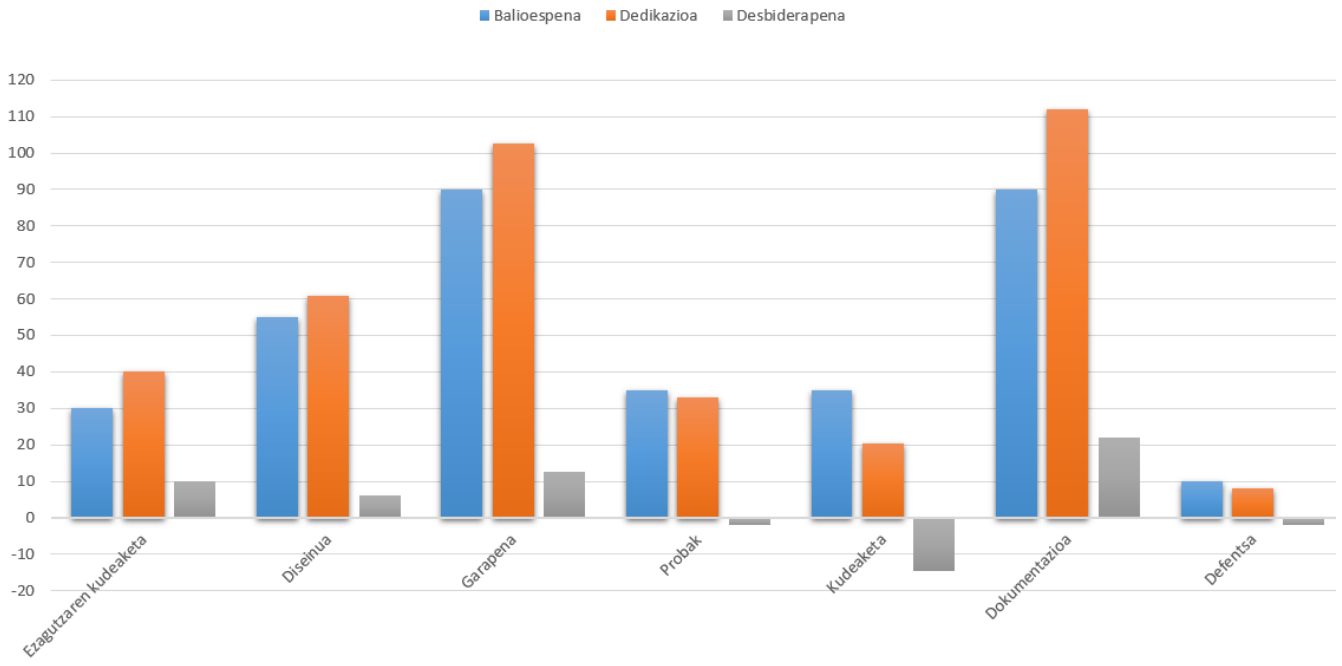
9.1 taula: Atazen iraupenaren desbiderapena

Aurreko 9.1 taulan ikus daitekeen moduan, hasieran diseinatutako plangintzan zehaztutako ordu kopuru totala ez da gehiegi aldatu. Guztira 42 ordu gehiago dedikatu dira, aurreikusitakoaren %11,3 gehiago. Desbiderapen hau, helburuak modu osoago batean betetzeko asmoarekin izan da, eta aurretik esan bezala, Gradu Amaierako Lanaren garapenean zehar funtzionalitate berriak gehitzeko

aukera agertu da eta hau izan da, desbiderapen honen arrazoi nagusia. Funtzionalitate berri hauek, *SPLMiner* moduluan gehitutako dokumentazio meatzaria eta *CMapConceptCollection* moduluan gehitutako ikuspegi gehigarriak izan dira.

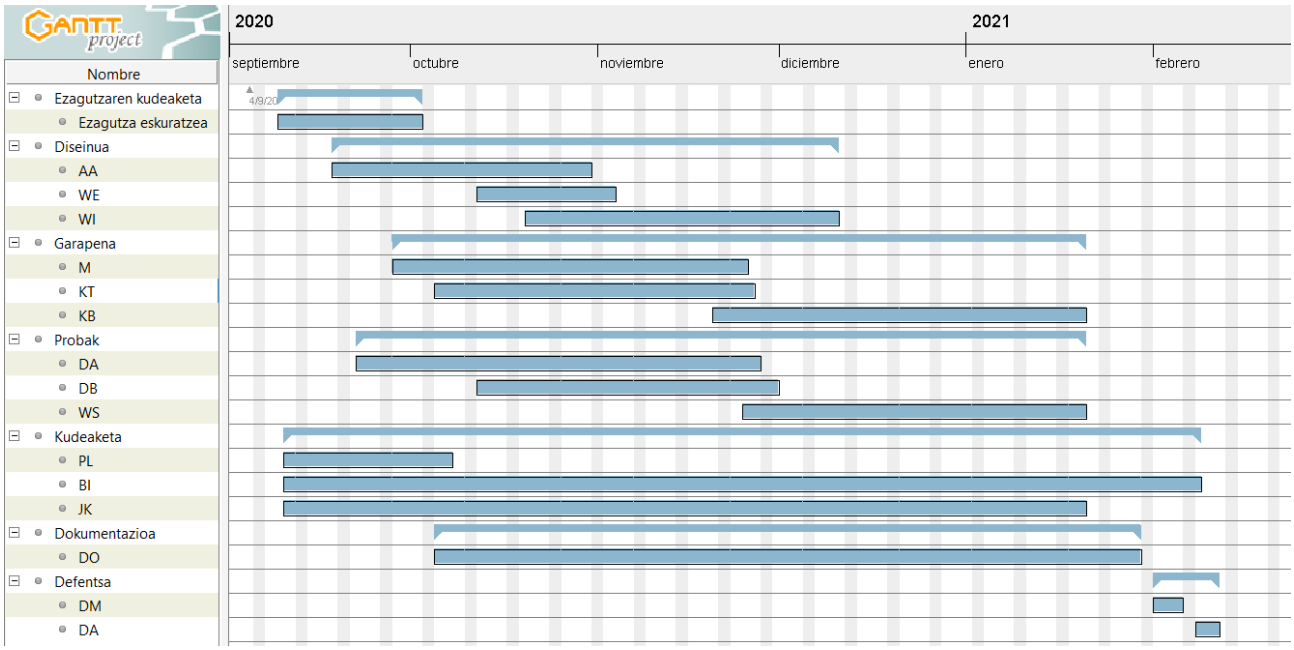
Kapitulu honen hasieran esan bezala, *CMapConceptCollection* web moduluak hasieran pentsatu baino lan karga handiagoa suposatu du proiektu honetan, eta hori, modulu honi eskainitako ordu kopuruetan islatu da. Hain zuzen ere, orduen dedikazioan aurreikusitakoaren %70 gehiago eskaini behar izan zaio. Gainera, aurreikusten ez zen dokumentazio fitxategien meatzaritza-lanaz arduratzen zen meatzariaren garapena egin behar izan da, eta honek meatzaritza arloko %15 gehiagoren dedikazioa suposatu du, hots, 9 ordu. Hala ere, dokumentazioak aurreikusi baino denbora gehiago eraman du, zehazki 17 ordu gehiago. Beste alde batetik, proiektuaren kudeaketak aurreikusi baino denbora gutxiagoa suposatu du, behar bada, ordu gehiegi aurreikusi zirelako proiektuaren kudeaketa ataza honentzako.

Ataza guztien alderaketa, hurrengo 9.1 irudiko grafikoan aurkezten da, proiektuaren garapenean zehar gertatutako desbiderapena modu grafikoagoan ikus dadin.



9.1 irudia: Plangintzaren desbiderapenaren grafikoa

Ataza bakoitzak suposatu duen denbora proiektuaren garapenean zehar, hurrengo Gantt diagraman (ikus 9.2 irudia) aurkezten da. Bertan, ataza bakoitza noiz hasi eta bukatu den erakusten da, eta bakoitzak suposatu duen epea proiektuaren garapenaren bizitza-zikloan. Diagrama horretan ikus daiteke, alde batetik, sekuentzian burutzea espero ziren zenbait ataza, azkenean paraleloan burutu direla, esaterako, meatzaritza eta kontzeptuen taldekatze atazak, edo datu arakatze proba eta datu bilketa proba atazak ere modu paraleloan eraman dira. Eta bestetik, interfazearen lana bukaera aldera utzi behar izan dela, meatzaritza lana ondo eginda zegoela ziurtatu arte.



9.2 irudia: Gantt diagrama

10. KAPITULUA

Ondorioak

Memoriari bukaera emateko, azkenengo kapitulu honetan, lehenik, Gradu Amaierako Lanari buruzko hausnarketa eta ondorioei buruz hitz egingo da, eta bigarrenik, proiektuan zehar ikasi dena eta etorkizunera begira gerta litekeenari buruzko azalpen batzuk emango dira.

10.1 Proiektuari buruzko hausnarketa eta ondorioak

SPLMinerPlus izeneko meatzaria garatuz eta *CMapConceptCollection* izeneko web modulua garatuz, proiektuaren 1.2 atalean zehazturiko helburuak nahiko ondo bete direla esan daiteke. Hala ere, helburu hauek proiektuaren garapenean zehar aldatzen joan dira. Betekizun batzuk alde batera utziz, beste batzuetan zentratzen joan da. Argi zeuden lortu behar ziren zenbait betekizun, hasieran adostu zirelako, baina garapenak aurrera egin ahala, arazoekin topo eginda, helburu berriak sortzen joan dira.

Proiektu honetan garatutako aplikazioari sarrera moduan sartzen zaion SPL guztiz aztertzen da. SPL honen kode fitxategiez gain, dokumentazio fitxategien meatzaritza-lana egiten da, eta fitxategi hauek guztiak aztertzen dira.

SPLMinerPlus meatzariari esker aztertutako termino guztiak ordenatu egiten dira, eta termino hauekin *cluster* (taldekatze) eragiketa egitea lortu da, *pyLev* kanpo liburutegia erabiliz. Eta *cluster* eragiketa honen emaitza, sarrera moduan sartutako SPL-aren kontzeptu nagusiak dira.

Kontzeptu mapa garatuko duen diseinatzaileari, *CMapConceptCollection* web moduluaren bitartez, *SPLMinerPlus* moduluan lortutako kontzeptu datuak ikuspegi desberdinetan eskaintzen dira. Eta gainera, hasieran aurreikusitako betekizun eta funtzionalitate gehiagorekin garatu da, kontzeptu zerrendaren informazio gehigarria eskaintzeko, esaterako, kontzeptuen hodeia, edota, termino bakoitzak SPL-aren ezaugarriekin daukan antzekotasuna aurkeztuz.

Bigarren mailako helburu moduan definitu ziren helburuak ere bete direla esan daiteke. Gradu Amaierako Lan honen garatzaileak, proiektu hau aurrera eramatean, ezagutza teknologiko berriak eskuratu ditu, izan ere, tresna desberdinak erabili dira eta hauetako batzuk guztiz berriak izan dira (ikusi 4. kapitulua). Eta bigarren mailako beste helburu bezala, maila honetako proiektua kudeatzeko gaitasuna erakustea lortu da, bai planifikazio eta antolakuntza aldetik, eta bai garapenaren aldetik ere, proiektuan epearen barnean bukatzea lortu delako, aurretik aipaturiko erabilgarritasun-alderdiak dituen aplikazio prototipoa garatu delako.

Proiektuaren hasieran diseinatutako plangintzan ezarritako mugarriak guztiz errespetatu direla esan daiteke. Egia da, planifikazioan moldaketa minimo batzuk egon direla, 9 kapituluan zehaztu direnak, baina mugarri garrantzitsuen daten barruan lan egin da. Honek, mugarri bakoitzaren hurrengo atazari atzerapenak ez ematea ekarri du, eta horrela, azken momentuan proiektuaren entrega heldu denean, epe barruan arrakastaz egin da entrega.

Beraz, orokorrean, proiektuaren irismenean zehaztutakoa bete egin dela esan daiteke. Ezarri ziren betekizun gehienak garatu dira, eta garapenean sortu diren beste zenbait ere gehitu dira.

10.2 Proiektuan ikasitakoa

Maila honetako proiektua garatu eta kudeatzerakoan, etorkizun profesionalari begira erabilgarriak izan daitezkeen hainbat lezio ikasi dira. Azken finean, Gradu Amaierako Lana garatzerakoan, informatika ingeniaritzako gradu osoan zehar lortutako trebetasunak eta heldutasuna aplikatzea exijitzen da, eta modu honetan ere, trebetasun eta heldutasun maila hauek gora egin dute.

Orain arte, proiektuak hutsetik hasteko ohitura existitu da, eta kasu batzuetan, beste proiektuetako softwarea edota kanpo liburutegiak erabili dira. Baina, kasu honetan, jadanik hasita zegoen proiektu baten kodean oinarritu izan da Gradu Amaierako Lan honetan garatutako softwarea. Eta aurretik garatutako proiektuaren egitura eta funtzionamendua guztia ulertzeak esfortzu gehigarri bat gehitu dio proiektu honi. Hala ere, zenbait lezio berri ikasi dira, besteak beste, kodea iruzkinen bidez dokumentatzea edota softwarearen garapen egituratua egitearen garrantzia. Aurreko bi teknikei esker, denbora eta esfortzua era nabarmenean arindu dira, izan ere, SPLMiner proiekturen kodea nahiko egituratua zegoen eta gainera, ia funtzionalitate guztiek dokumentazio iruzkinak zituzten kodean bertan.

Beste alde batetik, garatzailearen eta zuzendariaren arteko komunikazioa oso garrantzitsua izan da, proiektuaren garapenean aurrera egiteko. Komunikazio gehienak, online bidezko bileren bidez egin dira, edota posta elektronikoaren bidez bidalitako mezuen bitartekoa, hala ere, komunikazioan ez da arazorik egon. Eta komunikazio honek daukan garrantzia ere ikusi da.

Hasieratik amaierara, maila honetako proiektu baten garapena eta kudeaketa. Gradu Amaierako Lan hau arrakastaz eraman da aurrera. Kudeaketa nahiko ondo eraman da eta kudeaketa hau garapenean zehar ikasten joan da, plangintza jarraituz, momentu bakoitzean proiektuaren egoera zein zen aztertuz eta bukatze-data beti presente edukiz. Gainera, kudeaketa hau modu independentean egin da, eta orain arte, ohitura, proiektuak talde lanean egitea izan da, eta kudeaketa talde horietako pertsona guztien araberako zen. Baina kasu honetan, zuzendari baten laguntzaz, garatzailea nondik joan gidatu duena, proiektuaren kudeaketaren plangintza garatzaileak bere aldetik egin du.

Gainera, teknologia berriak bakarrik ikasteko eta erabiltzeko aukeraren aurrean aurkitu da garatzailea. Askotan, zerbitzu berdintsuak eskaintzen dituzten teknologia desberdinen artean aukeraketa egin behar izan da, eta aukeraketa honen garrantzia ere argi ikusi da. Izan ere, aukeraketa hau zuzen ez egiteak, proiektuaren mugarri batzuen atzerapenak ekar ditzake.

10.3 Hobetzeko aukerak

Aplikazioaren garapena bukatu eta gero, hobe daitezken zenbait ezaugarrien funtzionalitateak airean geratu dira.

- *Cluster*-a egiteko orduan, behar bada, beste liburutegiren bat aukeratu, edota kodea hutsetik sortu daiteke. Hitzen antzekotasunaren arabera egin beharrean, hitzen esanahiaren araberakoa izateko, edota, hitzen erroa erabiliz taldekatze prozesua egitea. Adibidez, bezeroa hitzaren inguruan baldin bada, “*bez*”, “*bezero*”, “*bezeroa*”, “*bezeroak*”, “*bez-ak*”, “*bezeroX*”, “*bezero_X*”, “*X_bezero*” etab. Hobekuntza honen garapenarekin hasi izan da, hurrengo GitHub-eko liburutegia erabiliz, CompoundWordSplitter [25], baina ez da gauza handirik lortu.

  Gainera, terminoen arazketa hobeago bat egitea posible izango litzateke. 6.3.3 atalean azaltzen den azterketan kontuan hartzen diren karaktereetako bat duen terminoa tratatuz, hau da, karaktere horietako bat presentatzen duten terminoak zatitu ahal izango lirateke, eta ondoren, termino honen atalak esanahiaren arabera aztertu. Modu honetan, termino guztien azterketa zehatzago bat lortuko litzateke. Hori bai, azterketa zehatzago hau egiteko, ondo egongo litzateke, *cluster*-a hitzen antzekotasunaren arabera egin ordez, hitzen esanahiaren arabera egitea, bestela, hobekuntza honek ez luke zentzu handirik izango.

- Kontzeptu lista eta kontzeptu hauek eraikitzen dituzten terminoak, .txt luzapeneko fitxategi batean gordetzen dira, eta gero fitxategi honetan gordetako datuak bigarren web moduluan kargatzen dira. Modu egokiagoa izango litzateke, informazio hau datu-basean gordetzea, gero web moduluan datu-baseko datuak atzitzeko. Horrela, datuak tratatzeko aukera gehiago edo desberdinak egongo lirateke.

  Terminoen arazketa hobeago bat egitea, 6.3.3 atalean azaltzen den azterketan kontuan hartzen diren karaktereetako bat duen terminoa tratatuz. Hau da, karaktere horietako bat presentatzen duten terminoak zatitu ahal izango lirateke, eta ondoren, termino honen atalak esanahiaren arabera aztertu. Modu honetan, termino guztien azterketa zehatzago bat lortuko litzateke. Hori bai, azterketa zehatzago hau egiteko, ondo egongo litzateke, atal honetako lehen puntuan hitz egiten den taldekatze prozesua erabiltzea, hitzen antzekotasunaren arabera egin ordez, hitzen esanahiaren arabera *cluster* eragiketa egitea, bestela, hobekuntza honek ez luke zentzu handirik izango.

A. ERANSKINA

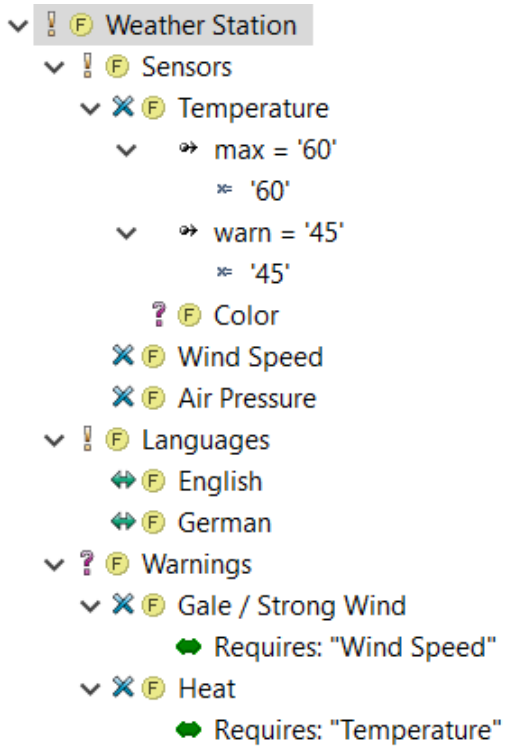
Pure::variants-ekin egindako SPL baten adibidea

Eranskin honetan, pure::variants-en egindako SPL baten adibidea aurkeztuko da. Erabiliko den SPL-a nahiko sinplea izango da, kontzeptuak eta loturak errazago ulertzeko, eta gainera, adibideko SPL-ak ez du zerikusirik izango proiektu honetan azaldu den *WacLine* SPL-arekin.

Kasu honetan, adibide moduan, *WeatherStation* SPL-a jarriko da. SPL honen domeinua, eguraldia aztertzen duen estazioa da. SPL hau tresnaren adibideen atalean eskaintzen dena da.

SPL honek, edozein SPL moduan, oinarrizko hiru fitxategi izango ditu, eta hurrengokoak dira:

FeatureModel → SPL-aren ezaugarriak biltzen dituen XML fitxategia da, eta bertan ezaugarri horien senidetasun-erlazioak, dependentziak, aldakortasun motak, etab. gordetzen dira. Eta hau izango litzateke honen egitura, A.1 irudia *Eclipsen* ikusita:

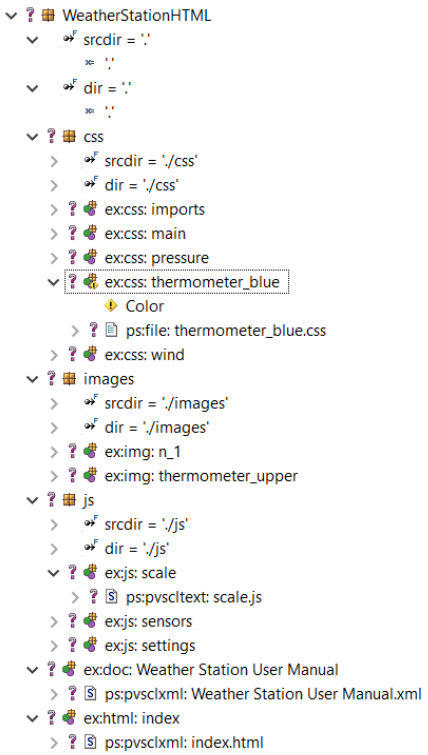


A.1 irudia: SPL-aren *FeatureModel*-a, *Eclipse*-n ikusita.

Eta ondoren, ezaugarri baten errepresentazioa *FeatureModel* fitxategian:

```
<cm:element cm:id="ilpiLoRjAhseJSQP4" cm:name="German" cm:type="ps:feature"
cm:class="ps:feature" cm:default="off">
       <cm:relations cm:id="ib6qOUWrqZOE4mFzU" cm:class="ps:dependencies">
      </cm:relations>
      <cm:relations cm:id="ihK3DDnxcLXLuk3Y2" cm:class="ps:parents">
        <cm:relation cm:id="ivBwL_BhuKMcySQ3J" cm:type="ps:parent">
          <cm:target cm:id="iKby1YUodb-G5xrKm">./ir48hEVg8Y2CeMjeO</cm:target>
        </cm:relation>
      </cm:relations>
      <cm:properties>
       <cm:property cm:id="iyEiEevX1mqjKK4Ln" cm:type="ps:integer" m:name="max">
          <cm:constant cm:id="iD31AzHZr" m:type="ps:integer" >60 </cm:constant>
       </cm:property>
       <cm:property cm:id="i9UhBWBiTXpHz" cm:type="ps:integer" cm:name= "warn">
          <cm:constant cm:id="iH6UzHHuk" m:type="ps:integer" >45 </cm:constant>
       </cm:property>
      </cm:properties>
      <cm:vname>
        <cm:mimedesc cm:id="lmuyc7wwr" cm:encoding="utf-8">Polish</cm:mimedesc>
        <cm:mimedesc cm:id="iYo-wOevT" cm:encoding="utf-8">German</cm:mimedesc>
      </cm:vname>
</cm:element>
```

FamilyModel → Funtzionalitateak inplementatzen dituzten kode fitxategiak, karpeta eta bestelako egiturak gordetzen dituzten XML fitxategia da. Eta hau izango litzateke honen egitura, A.2 irudia *Eclipsen* ikusita.

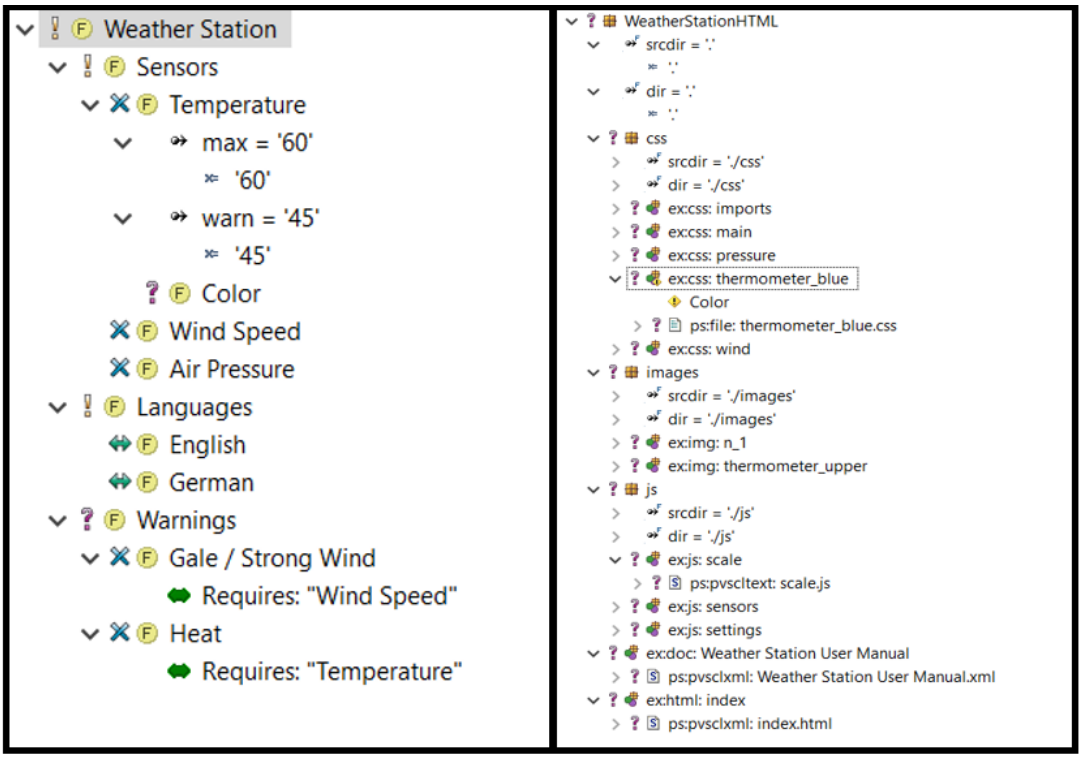


A.2 irudia: SPL-aren *FamilyModel*-a, Eclipse-n ikusita.

Eta ondoren, kode elementu baten errepresentazioa *FamilyModel* fitxategian.

```
<cm:element cm:id="ijpwZFIlDeuk-F88n" cm:name="WeatherStation" cm:type="ps:feature"
cm:class="ps:feature" cm:default="on">
      <cm:relations cm:id="iUct7FAVX7wBrjFtL" cm:class="ps:dependencies">
      </cm:relations>
      <cm:relations cm:id="i0BMoR8b27u8eJPh1" cm:class="ps:children">
        <cm:relation cm:id="i6o_cW77tHelSuxXZ" cm:type="ps:mandatory">
          <cm:target cm:id="ipf7OM2eKL8BNa9v5">./i3o2RSj4Fg7d7nNlj</cm:target>
          <cm:target cm:id="ifNl8acuPP1N84IdM">./ir48hEVg8Y2CeMjeO</cm:target>
        </cm:relation>
        <cm:relation cm:id="idVuxQJyQALq6kJxN" cm:type="ps:optional">
          <cm:target cm:id="iEhEYa5gwqdMQD8Nw">./iBFyOWLYHFpADbYjX</cm:target>
        </cm:relation>
      </cm:relations>
      <cm:properties>
      </cm:properties>
      <cm:vname>
        <cm:mimedesc cm:id="i7Prw9uTUynC6lr78" cm:mimetype="text/plain"
cm:encoding="utf-8">Weather Station</cm:mimedesc>
        <cm:mimedesc cm:id="i1f4VGBG85s5_mDVx" cm:mimetype="text/plain" cm:lang="de"
cm:encoding="utf-8">Wetterstation</cm:mimedesc>
      </cm:vname>
    </cm:element>
```

VariantModel → Produktu desberdinen konfigurazioa gordetzen duen XML fitxategia da. Hautagai dauden ezaugarri eta kode fitxategiak zeintzuk diren azken produktuan azalduko direnak eta zeintzuk ez. Eta hau izango litzateke honen egitura, ezaugarri eta kode elementuen hautaketa A.3 irudia *Eclipsen* ikusita.



A.3 irudia: SPL-aren VariantModel baten bi atalak Eclipse-n ikusita.

Eta ondoren SPL-aren elementu baten (ezaugarri edota kode elementu) errepresentazioa *VariantModelaren* fitxategian:

```
<cm:element cm:id="iftu1QkvqU3yEWAmw" cm:type="ps:file" cm:class="ps:source"
cm:default="on">
      <cm:relations cm:id="iPNR6_WGjsyTwZKRm" cm:class="ps:dependencies">
      </cm:relations>
      <cm:relations cm:id="iO0EH_jIpKcUd38O4" cm:class="ps:parents">
        <cm:relation cm:id="ilCFj5wl7WyuIxAvD" cm:type="ps:parent">
          <cm:target cm:id="iaHzPP6fyGowcPIf4">./i7G3rTS94L1duzd4q</cm:target>
        </cm:relation>
      </cm:relations>
      <cm:properties>
        <cm:property cm:id="iZauFxgE" cm:type="ps:path" cm:fixed="true" cm:name="file">
          <cm:constant cm:id="iZNGXO6lP1o" cm:type="ps:path">n_10.png</cm:constant>
        </cm:property>
        <cm:property cm:id="i_NYrv" cm:type="ps:file" cm:fixed="true" cm:name="type">
          <cm:constant cm:id="iqcUylLNhRb" cm:type="ps:filetype">misc</cm:constant>
        </cm:property>
      </cm:properties>
      <cm:vname>
        <cm:mimedesc cm:id="iuBuYDhsc" cm:encoding="utf-8">n_10.png</cm:mimedesc>
      </cm:vname>
</cm:element>
```

B. ERANSKINA

PROBAK

Eranskin honetan WacLine SPL-aren meatzaritza-lana egin ostean lortutako balioak aurkezten dira. Hainbat konbinazio posible probatu dira emaitza hoberena lortzeko, garapenaren prozesuan estrategia, 8. atalean azalduta dagoena, aldatzen joan delako. '*' karaktere artean dagoen balioa kontzeptua da, eta honi jarraitzen dioten hitzak kontzeptu hau osatzen duten terminoak. Soilik termino mota bakoitzaren eta konbinazio garrantzitsuen kasuak aurkeztuko dira, eta ez dira konbinazio guztiak aurkeztuko asko direlako.

Terminoak taldekatu egiten dira, hau da, "*cluster*" eragiketa aplikatzen zaie, hala ere, ez dira termino guztiak taldekatze prozesuan gehitzen. *cluster* eragiketan gehitzeko baldintza hurrengokoa da, termino bakoitzaren agerpen minimoa 10 izan behar da, honek esan nahi du, 10 bider baino gutxiagotan agertzen bada, ez dela kontuan hartuko.

Hurrengo ataletan aurkezten diren emaitzak, *SPLMinerPlus* moduluan, termino mota desberdinak kontuan hartuz lortutako emaitzak dira. Adibidez, "Klaseak" atalean, termino azterketan klase-izenak soilik kontuan hartuz lortutako emaitzak dira, "Funtzioak eta metodoak" atalean, funtzioak eta metodoak kontuan hartuz lortutako emaitza aurkezten dira, eta horrela atal guztiekin.

Klaseak

Annotation: APISimulation, Annotation, AnnotationList, AnnotationUtils, CreateAnnotation, DeleteAnnotation, ReadAnnotation, ReplyAnnotation, UpdateAnnotation
Code: Body, Canvas, Code, Codebook, Codes, Popup, Resume, Review, Sidebar, Task, Theme, Toolset
AnnotationServer: AnnotatedTheme, AnnotationExporter, AnnotationGroup, AnnotationImporter, AnnotationManagement, AnnotationServer, AnnotationServerManager
MoodleClient: GoogleSheetClient, MoodleClient, MoodleComment, MoodleProvider, MoodleReport, MoodleScraping, MoodleUtils, Neo4JClient
MoodleClientManager: AnnotatedContentManager, ContentScriptManager, HypothesisClientManager, MoodleBackgroundManager, MoodleClientManager, MoodleDownloadManager, Neo4JClientManager
Commenting: Assessing, Classifying, Commenting, CommentingForm, PrimaryStudy, Screenshots
AnnotationBasedInitializer: AnnotationBasedInitializer
MoodleGraderAugmentation: MoodleAugmentation, MoodleGraderAugmentation, MoodleGradingAugmentation
MoodleViewPluginAssignSubmissionAugmentation: MoodleViewPluginAssignSubmissionAugmentation
CreateCodebook: CreateCodebook, CreateHighlighterTask, DeleteCodebook, ReadCodebook, RenameCodebook, UpdateCodebook
Buttons: BackToWorkspace, Background, BuiltIn, Buttons, Hypothesis, Options, TextSummary
GoogleSheetsClientManager: GoogleSheetContentScriptManager, GoogleSheetGenerator,

GoogleSheetsClientManager, GoogleSheetsManager
PreviousAssignments: PreviousAssignments
SuggestingLiterature: SuggestingLiterature
BrowserStorage: BrowserStorage, BrowserStorageClient, BrowserStorageManager, BrowserStorageSearch, ChromeStorage, GroupSelector
ExportCodebook: EmptyCodebook, ExportCodebook, ExportCodebookJSON, ImportCodebook, ImportCodebookJSON, NoCodebook
ModeManager: CodebookManager, GSheetParser, GroupInitializer, HypothesisManager, ModeManager, Neo4JManager, RolesManager, TargetManager, TaskManager
DateUtils: Alerts, ColorUtils, CryptoUtils, DOMTextUtils, DataUtils, DateUtils, DeleteGroup, FileUtils, LanguageUtils, PDFTextUtils, RandomUtils, SheetUtils, URLUtils, UserFilter
SpringerContentScript: ACMContentScript, ScienceDirectContentScript, SpringerContentScript

Funtzioak eta metodoak

constructor: constructor
create: check, create, ready, update
hash: XRef, back, hash, hide, keys, parse
render: enqueue, filter, find, getter, handler, listener, reload, render, values, wrapper
ReadableStream: ReadableStream, WritableStream
transform: transform
invalid: initialize, invalid
isEvalSupported: isEvalSupported
Annotation: Annotation
TypedArray: TypedArray
reset: assert, empty, next, request, reset
resolve: destroy, remove, resolve, resolved
toString: entries, notify, randomString, toString
some: Promise, decode, lookup, open, show, some, toggle

Dokumentazio fitxategiak eta klaseak

Annotation: APISimulation, AnnotatedTheme, Annotation, AnnotationGroup, AnnotationList, AnnotationUtils, CreateAnnotation, DeleteAnnotation, Operation, ReadAnnotation, ReplyAnnotation, UpdateAnnotation
AnnotationServer: AnnotationExporter, AnnotationImporter, AnnotationManagement, AnnotationServer, AnnotationServerManager
Code: Body, Canvas, Code, Codebook, Codes, Export, Import, Popup, Read, Sidebar, Task, Theme, Update
Commenting: Assessing, Classifying, Commenting, CommentingForm, GroupInitializer
MoodleClient: GoogleSheetClient, MoodleClient, MoodleComment, MoodleProvider, MoodleReport, MoodleScraping, Neo4JClient
MoodleClientManager: AnnotatedContentManager, ContentScriptManager, HypothesisClientManager, MoodleBackgroundManager, MoodleClientManager, MoodleDownloadManager, Neo4JClientManager
AnnotationBasedInitializer: AnnotationBasedInitializer
PreviousAssignments: PreviousAssignments
BrowserStorage: BackToWorkspace, BrowserStorage, BrowserStorageClient, BrowserStorageManager, BrowserStorageSearch, ChromeStorage
CreateHighlighterTask: CreateHighlighterTask
MoodleGraderAugmentation: MoodleAugmentation, MoodleGraderAugmentation,

MoodleGradingAugmentation
MoodleViewPluginAssignSubmissionAugmentation: MoodleViewPluginAssignSubmissionAugmentation
CreateCodebook: CreateCodebook, DeleteCodebook, ReadCodebook, RenameCodebook, UpdateCodebook
ImportCodebook: EmptyCodebook, ExportCodebook, ExportCodebookJSON, ImportCodebook, ImportCodebookJSON, ImportExport, NoCodebook
Buttons: Background, BuiltIn, Buttons, Hypothesis, Options, Screenshots, TextSummary
SuggestingLiterature: SuggestingLiterature
GoogleSheetsManager: GoogleSheetContentScriptManager, GoogleSheetGenerator, GoogleSheetsClientManager, GoogleSheetsManager
ModeManager: CodebookManager, GSheetParser, HypothesisManager, ModeManager, Neo4JManager, RolesManager, TargetManager, TaskManager
DateUtils: ColorUtils, CryptoUtils, DOMTextUtils, DataUtils, DateUtils, FileUtils, LanguageUtils, MoodleUtils, PDFTextUtils, PrimaryStudy, RandomUtils, SheetUtils, URLUtils
SpringerContentScript: ACMContentScript, ScienceDirectContentScript, SpringerContentScript
Delete: Alerts, Create, Delete, DeleteGroup, GroupSelector, Resume, Review, Toolset, UserFilter

Aldagaiak (var, let eta conts)

annotation: Annotation, anInstance, annotation, annotations, linearization, randomString, resolution, rotation
stream: getKeys, getWeak, interval, sourceName, stackClean, strBuf, stream, streamId
state: create, hashData, iFirstSave, srcByte, stack, stackSize, startPos, startResult, state, status, store, subtype, validate
offset: offset, offset0, output
code: body, cache, charCode, code, code1, code2, codeBuf, codeId, codebook, codes, core, decode, decoder, encoding, hide, index, isNode, loaded, mode, module, node, second, toUnicode, unicode
promise: Promise, pdfViewer, process, promise, promises, toPrimitive
scale: call, called, newScale, scale, scaleX, scaleY, style, swal
page: flags, image, message, pack, packet, page, pageNum, pageNumber, pageView, path, prev, range, span, tags, type
group: SPECIES, background, global, group, groupCtx, groupId, groups, gulp, lookup, popup, previous, prop, proto
user: XRef, buffer, curr, item, number, other, used, user, users, viewport
file: angle, cipher, fails, file, fileMarker, filename, files, filter, minY, tile, title
reader: ArrayProto, cypher, header, order, reader, ready, refPos, request, review, spreadsheet
font: Config, before, bounds, count, done, floor, font, forOf, format, found
text: dest, htmlText, input, items, left, next, nextNextC, self, step, temp, text, that, tmpCtx
mask: base, cmap, domain, mapping, mark, mask, maxX, task, valid
handler: character, handled, handler, manager, pdfManager
events: Events, event, eventBus, events, exports, points, settings
parser: charset, marker, parameters, params, parser, target
glyph: alpha, glyph, glyphId, glyphName, numGlyphs
transform: transfers, transform, transformStream
list: dict, diff, first, l10n, last, length, limit, list
isRenderable: isRenderable
toObject: IObject, anObject, isObject, object, toIObject, toObject
ITERATOR: ITERATOR, LIBRARY
student: current, document, element, student, studentId
redefine: defined, redefine, redefineAll
Iterators: Iterators, operatorList
listeners: listener, listeners
byteLength: bufferLength, byteLength, outputLength, paddedLength, toLength

context: comment, content, context, contextLabel, contexts, toInteger
fontSize: chunkSize, codeSize, container, controller, elementSize, fontName, fontSize, rowSize
ReadableStream: ReadableStream, WritableStream, readableStream
bits: bbox, bias, bitmap, bits, blob, bytes, capability, digits, eBias, nBits, view, width, widths
fnArray: TypedArray, argsArray, array, fnArray, overlay
writer: after, attribute, criteria, getter, navigator, rowBytes, white, wrapper, writable, writer
action: aFunction, action, button, dictionary, extension, isFunction, notify, option, position, script, version
globalSettings: globalSettings
match: Dispatch, Math, each, match, matchIdx, matches, matrix
beginChunk: beginChunk, endChunk
description: description, descriptor, desiredSize, destination
PROTOTYPE: PROTOTYPE
query: entry, query, queue, qwhere, rubric
setToStringTag: setToStringTag
components: component, components, componentsCount
speciesConstructor: speciesConstructor
eLen: aLen, begin, eLen, eMax, elem, json, level, token, xref
check: callback, check, checkboxes, checksum, chunk, theme
firstDescriptor: firstDescriptor, pullIntoDescriptor
names: canvas, entries, gamma, label, name, names, numComps, numTables, tables, values
property: properties, property, protocol
meta: data, delta, empty, imgData, keys, meta, method
DESCRIPTORS: DESCRIPTORS
newPromiseCapability: newPromiseCapability
intentState: currentWrite, intentState
requestCapability: requestCapability
tileIndex: intIndex, pageIndex, tileIndex, toAbsoluteIndex
textQuoteSelector: textQuoteSelector
high: NAME, args, hash, height, high, html, right, sign
className: className, classof, colorSpace, targetName
loadingTask: loadingTask
readIntoRequest: readIntoRequest, readRequest
decompositionLevelsCount: decompositionLevelsCount
relativeOffset: relativeOffset
cols: cells, cmid, colors, cols, columns, coords, rows
wordSpacing: wordSpacing
isPatternFill: isPatternFill
verticesPerRow: verticesPerRow
defaultWidth: defaultWidth
select: select, selected, selection, splitYBy, support
resource: backpressure, resolve, resource, source
feedbackComment: feedbackComment

C. ERANSKINA

LEVENSHTEIN DISTANTZIA

Lehenik eta behin, modu informalean defini dezakegu bi karaktere kateen arteko *Levenshtein* distantzia, kate batetik hasita bigarren katea lortzeko beharrezkoak diren karaktere bakarreko gutxieneko edizio kopuru gisa (hau da, txertaketak, ezabaketak edo ordezkapenak).

Matematikoki, **a** eta **b** karaktere kateen arteko *Levenshtein* distantzia, $lev_{a,b}(|a|,|b|)$ moduan adieraz daiteke, non **|a|** eta **|b|**, hurrenez hurren, eta C.1 irudian aurkezten den espresioa jarraitzen duen.

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & si\ \min(i,j) = 0, \\ \min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{a_i \neq b_j} \end{cases} & si\ min(i,j) \neq 0 \end{cases}$$

C.1 irudia: *Levenshtein* distantzia

Espresio honetan, $1_{a_i \neq b_i}$-ak 0 itzultzen duen funtzioa adierazten du karaktere kateak, **a** eta **b**, berdinak direnean eta 1 bestela, desberdinak direnean. Bestalde, kontuan izan $lev_{a,b}(|a|,|b|)$ funtzioa **a** katearen lehen **i** karaktereen eta **b** katearen lehen **j** karaktereen arteko distantzia dela.

Adierazpenaren gutxieneko funtzioan, min funtzioa, lerro bakoitzak hiru eraldaketa posibleetako bat neurtzen du. Lehenengo errenkadak **a** katean karaktere bat kentzea adierazten du, bigarren errenkada karaktere bat txertatzeari dagokio eta hirugarren errenkada ordezkapenari dagokio.

Levenshtein distantziaren inplementazioa:

```
function LevenshteinSubminimal(A, B) {
    var a = A.length;
    var b = B.length;
    var f = function(s){return Levenshtein(A, s)};

    if(a >= b)
        return {k: f(B), t: B}

    var m = {k: b+2, t: null}, c1 = a-1, c2 = a+1;
    for(var p = 0; p < b - c1; p++)
        for(var l = c1, c3 = Math.min(c2, b - p); l <= c3; l++) {
            var t = B.substr(p, l), k = f(t);

            if(m.k >= k)
                m = {k: k, t: t};
        }
    return m;
}
```

Levenshtein distantziaren kalkuluaren adibidea

Levenshtein distantziaren eragiketa baten adibidea emateko, zenbait pertsonen izenen lista batean eragiketa hau aplikatuz, terminoan lehen bilketa horrela egingo litzateke. Espainiako 100 izen erabilienak sartu ostean, hurrengo azpi-bilduman agertzen diren izenak azpi-bilduma berdinean biltzen ditu, antzekotasunaren ondorioz:

```
*JUAN:* IVAN, JOAN, JOAQUIN, JUAN, JULIAN
*ANTONIO:* ALFONSO, ANTONIO, SANTIAGO
*LUIS:* HUGO, JULIO, LUCAS, LUIS
*JAVIER:* DAVID, JAIME, JAVIER, XAVIER
*MIGUEL:* MIGUEL
*JOSE:* CESAR, IKER, JESUS, JORDI, JORGE, JOSE, JOSEP, MOHAMED, OSCAR, TOMAS
*ANGEL:* ALEX, ANDRES, ANGEL, DANIEL, RAFAEL
*ALBERTO:* ALBERT, ALBERTO, GUILLERMO, LIBERTO, ROBERTO, UMBERTO
*CARLOS:* CARLOS, MARCOS
*ALVARO:* ALEJANDRO, ALFREDO, ALVARO, EDUARDO, SALVADOR
*DIEGO:* DIEGO, DOMINGO, RODRIGO
*MARIO:* ADRIAN, EMILIO, ENRIQUE, MARC, MARIA, MARIANO, MARIO, MARTIN,
MATEO, PABLO, RICARDO, SERGIO
*IGNACIO:* FERNANDO, GONZALO, IGNACIO
*RAMON:* RAMON, RAUL, RUBEN
*FELIX:* FELIPE, FELIX, PEDRO
*CRISTIAN:* AGUSTIN, CRISTIAN, SEBASTIAN
*VICTOR:* AITOR, HECTOR, NICOLAS, VICENTE, VICTOR
*MANUEL:* GABRIEL, ISMAEL, MANUEL, SAMUEL
*FRANCISCO:* FRANCISCO
```

D. ERANSKINA

Dokumentazio fitxategien meatzaria

Eranskin honetan, garatu den dokumentazio meatzariaren kodea aurkeztuko da. Bertan, *featureModel* izeneko karpeta existitzen bada edo ez konprobatzen da, eta honek esan nahi du, aztertu nahi dugun SPL-ak, SPL honen dokumentazio fitxategi guztiak *featureModel* izeneko karpeta batean egon behar direla, eta hau *WacLine* SPL-aren egitura horrela zegoelako egin da. Baldintza hau programaren aurrebaldintza bat izango litzateke. Baldintza hau beteta, *featureModel* karpeta horren fitxategi eta azpi-direktorio guztien meatzaritza-lana egingo da.

```
public static void mineDocumentation() throws IOException {
	if(containsFM()) {
		File directoryPath = new File(path + "\\featureModel");
		File filesList[] = directoryPath.listFiles();
		BufferedReader bf;
		for(File file : filesList) {
			bf = new BufferedReader(new FileReader(file.getAbsoluteFile()));
			String line;
			while ((line = bf.readLine()) != null) {
				kodea += line + "\n";
				String[] span=null;
				String[] resultWithNumber=null;
				String[] result=null;
				if (line.contains("<span ") && line.contains(".")) {
					span= line.split(">");
					resultWithNumber= span[span.length-1].split("<");
					if(resultWithNumber[0].contains(" ")) {
						result = resultWithNumber[0].split(" ");
					if (result[0].length()<7 &&
				!result[result.length-1].contains(".")) {
						String paux=
						file.getPath().replaceAll("\\\\", "/");
						paux = paux.replace("/"+file.getName(),"");
						String[] pa = paux.split("/git/");
						if(termTable.appears(result[result.length-1])){
						termTable.sumApparition(result[result.length-1])
						termTable.addPath(result[result.length-1],
					file.getName(),pa[1]);
					}else {
						Term a= new Term(result[result.length-1],10);
						a.addNewFile(file.getName(),pa[1]);
						termTable.getTermTable().add(a);
					}}
				}
			}
		}
		bf.close();
}}}
```

Aurreko kodean agertzen den *containsFM*() funtzio laguntzailearen kodea:

```
private static boolean containsFM() {
		File directoryPath = new File(path);
		File filesList[] = directoryPath.listFiles();
		String fm = "featureModel";
		for(File file : filesList) {
			if(file.getAbsolutePath().contains(fm)) {
				return true;
			}
		}
		return false;
	}
```

E. ERANSKINA

CMapConceptCollection moduluaren internazionalizazioa eta konfigurazioa

Eranskin honetan, web *CMapConceptCollection* moduluaren internazionalizazioa eta konfigurazioa aurkezten dira.

Internazionalizazioa

Aurretik, proiektuaren erronken 8. atalean aipatu den bezala, web moduluaren interfazea eleaniztuna izango da, hau da, erabiltzaile eta konputagailuaren arteko elkarrekintza hizkuntza batean baino gehiagotan eman ahal izango da, zehazki, hiru hizkuntza desberdinetan, ingelesa, gaztelania eta euskara.

Hau lortzeko, hurrengo pausoak eraman dira aurrera. Testua bistaratuko den HTML etiketetan, testuaren balio soila jarri ordez, hurrengo adibidean agertzen den atributua gehituko da. Modu honetan, interfazean aukeratuta dagoen hizkuntzaren arabera konfigurazio fitxategi zehatz batzuetatik testu balioak hartuko ditu.

```
<th:a th:text="#{navbar.concepts}"></th:a>
```

Honela, aukeratutako hizkuntzaren arabera, konfigurazio fitxategi desberdinera joango da testuaren bila, eta hiru hizkuntza desberdin daudenez, hiru konfigurazio fitxategi egongo dira. Hauek, hurrengo egitura izango dute, eta adibide bezala “Kontzeptua” testua jarriko da:

- *messages_en.properties* fitxategia, honek ingelesezko testuak izango ditu:

  ```
  navbar.concepts = Concepts
  ```

- *messages_es.properties* fitxategia, honek gaztelaniazko testuak izango ditu:

  ```
  navbar.concepts = Conceptos
  ```

- *messages_eus.properties* fitxategia, honek euskarazko testuak izango ditu:

  ```
  navbar.concepts = Kontzeptuak
  ```

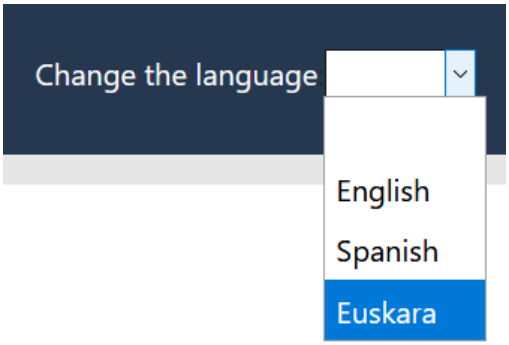
Konfigurazio fitxategi hauek, hizkuntza desberdinen testu guztiak izango ditu eta honela, interfazeko testu guztiak erabiltzaileak aukeratutako hizkuntzan agertuko dira. Gainera, honek hizkuntza berria sortzeko aukera ematen du, beste konfigurazio fitxategi bat sortuz soilik, *.properties* luzapena duena eta bertan testu bakoitzari dagokion etiketan balioak jarriz. Modu honetan, aplikazio hau beste herrialdetako lokalizazio prozesua aurrera eramateko aukera ere ematen du.

Bestalde, zenbait HTML etiketa ez dira zuzenean sortzen, baizik eta, JavaScript eta JQuery kodearen bitartez sortzen dira, eta modu honetan sortutako HTML etiketei ezin zaie aurretik azaldu den soluzioa ezarri, interfaze eleaniztuna eskaintzeko. Beraz, beste soluzio bat sortu behar izan da, etiketa hauek edozein hizkuntzatan agertzeko. Soluzioa, hurrengokoa izan da; web orria kargatzean bistaratu nahi diren etiketen balioak aldagai lokaletan kargatu dira, modu honetan, ez da hasieran azaldu den egitura hori mantentzen. Egitura hori, hurrengo honengatik aldatu da.

```
"<th scope='col'><div><p>"+  term  +"</p></div></th>"
```

Pantailaratu nahi den balioa, hizkuntzaren arabera, *term* balio horretan gordeko da, eta zuzenean <*p*> etiketaren artean joango da.

Erabiltzaileak hizkuntza aukeratzeko, E.1 irudian agertzen den *dropdown* menu bat izango du, nabigazio barraren eskuineko aldean, eta honi klikatuz, aukeran dauden hizkuntzak zerrendatuko dira. Web aplikazioaren edozein ikuspegitan egonda, hizkuntza aldatzeko aukera egongo da, eta *dropdown* menu honek, honako itxura dauka:



E.1 irudia: Hizkuntzak

Konfigurazioa

Moduluaren eta datu-basearen konexioa lortzeko, modulu honen *application.properties* eta *pom.xml* fitxategiak konfiguratu behar dira, beharrezkoak diren konfigurazio ezaugarriak gehituz.
Lehenik, *application.properties* fitxategian, hiru parametro berri erazagutu behar dira eta

hurrengokoak dira:

- *spring.datasource.url* parametroa. Datu-basearen helbidea esleitu behar zaio.
- *spring.datasource.username* parametroa. Datu-base honetara sartzeko erabiltzen den erabiltzailearen erabiltzaile-izena esleitu behar zaio.
- *spring.datasource.password* parametroa. Datu-base honetara sartzeko erabiltzen den erabiltzailearen pasahitza esleitu behar zaio.

Behin parametro hauek konfiguratuta, horrela geratzen da *application.properties* fitxategia. Bertan ere, zerbitzariak konexiorako erabiliko duen portua definituko da:

```
# Database url
spring.datasource.url=jdbc:mysql://localhost:3306/wacline?serverTimezone=UTC


# Username
spring.datasource.username=root

# Password
spring.datasource.password=****

# Port
server.port=9898
```

Eta ondoren, *pom.xml* fitxategian, menpekotasun bat gehitu beharra da datu-baseko konektorerako:

```
<dependency>
        <groupId> mysql </groupId>
        <artifactId> mysql-connector-java </artifactId>
</dependency>
```

Bibliografia

[1] A. S. Foundation,. (2020). *Apache Maven Project*. https://maven.apache.org/ helbidetik eskuratua

[2] AnyChart. (2020). *AnyChart - JS charts*. Interactive JavaScript charts designed to be embedded and integrated into web, desktop, and mobile apps: https://www.anychart.com/products/anychart/overview/ helbidetik eskuratua

[3] Authors, T. C. (2015). *The Chromium Projects*. https://www.chromium.org/ helbidetik eskuratua

[4] Brendan J. Frey, D. D. (2007.eko Otsailak 16). *Clustering by Passing Messages Between Data Points.* https://science.sciencemag.org/content/315/5814/972 helbidetik eskuratua

[5] Eich, B. (1995.eko Abenduak). *Javascript: high-level programing language.* https://www.javascript.com/ helbidetik eskuratua

[6] GitHub. (2006). *GitHub*. https://github.com/github helbidetik eskuratua

[7] H. M. Onekin. (2020). *WacLine.* WacLine: https://github.com/onekin/WacLine helbidetik eskuratua

[8] *Hojas de cálculo*. (2021). https://docs.google.com/spreadsheets helbidetik eskuratua

[9] I. Eclipse Foundation. (2020). *Eclipse Foundation*. https://www.eclipse.org/ helbidetik eskuratua

[10] I. VMware. (2021). *Spring*. Spring makes Java simple: https://spring.io/ helbidetik eskuratua

[11] I. VMware. (2021). *Spring Boot 2.4.2*. Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".: https://spring.io/projects/spring-boot#overview helbidetik eskuratua

[12] JetBrains. (2020). *Lenguaje de dominio específico*. https://www.jetbrains.com/es-es/mps/concepts/domain-specific-languages/ helbidetik eskuratua

[13] *JSON*. (202). https://www.json.org/json-en.html helbidetik eskuratua

[14] O. Corporation. (2014). *Java SE Development Kit 8*. https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html helbidetik eskuratua

[15] O. Corporation. (2020). *Interface Comparator<T>*. https://docs.oracle.com/javase/8/docs/api/java/util/Comparator.html helbidetik eskuratua

[16] O. Corporation. (2021). *Mysql: the world's most popular open source database*. https://www.mysql.com/ helbidetik eskuratua

[17] O. Corporation. (d.g.). *Package java.util.* https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html helbidetik eskuratua

[18] Object Management Group. (2021). *UML*. https://www.uml.org/ helbidetik eskuratua

[19] Ochoa, J. E. (2018). *Analysis and Improvement of a Software Production Process based on the Combination of Model Driven Development and Software Product Lines.* https://riunet.upv.es/handle/10251/107734 helbidetik eskuratua

[20] Otto, J. T. (2011). *Build fast, responsive sites with Bootstrap*. https://getbootstrap.com/ helbidetik eskuratua

[21] pure-systems GmbH. (2020). *pure-system*. https://www.pure-systems.com/ helbidetik eskuratua

[22] Python Software Foundation . (2020). *Python*. https://www.python.org/ helbidetik eskuratua

[23] Resig, J. (2006.eko Abuztuak). *jquery: write less, do more*. https://jquery.com/ helbidetik eskuratua

[24] Salaberri, I. (2020). http://hdl.handle.net/10810/48777 helbidetik eskuratua

[25] TimKam. (2018). *Compound Word Splitter*. https://github.com/TimKam/compound-word-splitter helbidetik eskuratua

[26] ToastDriven. (2014). *toastdriven/pylev.* https://github.com/toastdriven/pylev helbidetik eskuratua

[27] Torvalds, L. (2005). *git –local-branching-on-the-cheap*. https://git-scm.com/ helbidetik eskuratua

[28] Uztarroz, I. E. (2016). Aldaera linguistikoen normalizazioainferentzia fonologikoa etamorfologikoa erabiliz. 221. http://ixa.si.ehu.es/sites/default/files/dokumentuak/8336/nagusia.pdf helbidetik eskuratua

[29] W3C. (2020). *HTML*. https://html.spec.whatwg.org/multipage/ helbidetik eskuratua

[30] W3C. (2021.eko Urtarrilak). *What is CSS?* https://www.w3.org/Style/CSS/Overview.en.html helbidetik eskuratua

[31] *Acceso sencillo y seguro a todo tu contenido* . (2021). https://www.google.com/intl/es_es/drive/ helbidetik eskuratua

[32] BigLever Software. (2013). *Software Product Lines*. https://www.softwareproductlines.com/ helbidetik eskuratua