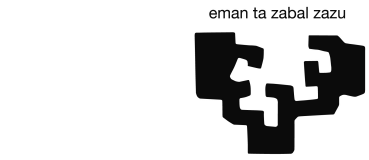


Informatika Ingeniaritzako Gradua

Software Ingeniaritza

Gradu Amaierako Proiektua

Testu historikoak kontsultatzeko plataforma

Egilea

Jon Ander Belamendia Arruabarrena

2016

Laburpena

Gradu Amaierako Proiektu honetan testu historikoak biltzen dituzten webguneez baliatuz, mota ezberdinetako bilaketak egin ahal izatea eskainiko digun web-aplikazio bat garatu dugu. Testu historikoetako hitzak ez-estandarrak izan ohi direnez, aurretik garatutako normalizazio-sistema baten bitartez, hitz horri dagokion hitz estandarra lortzen da eta baita estandar horren lema ere. Normalizazio-sistema hori *Phonetisaurus* tresnan oinarritzen da eta, informazioa berreskuratzeko (*Information Retrieval*), Lucenen oinarritutako *Apache Solr* erabili da.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Sarrera

Dokumentu hau Jon Ander Belamendia Arruabarrenak Euskal Herriko Unibertsitatean egindako Gradu Amaierako Proiektuari dagokio. Proiektu honen zuzendariak izan dira Izaskun Etxeberria Uztarroz eta Iñaki Alegria Loinaz, biak ala biak Donostiako Informatika Fakultateko irakasleak.

Ondorengo kapituluetan azalduko dira, besteak beste, proiektuaren helburuen dokumentua, aurrekariak eta erabilitako teknologia, plataformaren analisia eta diseinua edota garapenaren nondik norakoak.

1.1 Motibazioa

Gaur egun testu ez-estandar ugari daude sarean, sare sozialak direla, erabilpen dialektala dela... Horrez gain, eta hain ezagunak ez izan arren, liburutegi digitaletan agertzen diren testu historikoak daude. Hizkuntzaren aldaera zaharrak erabiltzen dira testu horietan, batzuetan aldaera dialektalak ere badirenak, eta horretan adituak ez diren erabiltzaileek zaila izango dute bilaketak ohiko moduan egitea.

Adibidez, “bekatu” hitza testu historiko batean bilatu nahi izanez gero, bilatzaile arrunt batek ez lituzke emaitza asko itzuliko ziur aski, ez baita garaiko hitza. Aldiz, “bekhatu” edo “bekathu” hitz ez-estandarrak bilatuz gero, emaitza gehiago lortuko lirateke, garai hartan erabiltzen zen hitza baizen. Beraz, ongi legoke bilaketak hitz estandarrekin egin eta hitz horiei lotuta dauden hitz ez-estandarrak ere bilatu ahal izatea.

Interes handia dago testu historikoak prozesatzeko, baina testu horiekin hizkuntzarekin lotutako tresnak erabili nahi direnean, arazo bat dago: emaitzak oso kaskarrak dira, testuak ez direlako estandarrak. Gainera, askotan euskaraz egiten diren bilaketak lemaren arabera egiten direnez, interesgarria litzateke lemen bitartez bilaketak egin ahal izatea, baina arazo berdinarekin aurkitzen gara: testuak ez direla estandarrak.

Horri soluzio bat eman nahian, planteatzen den irtenbide posible bat testuak normalizatzea da, ondoren tresna horiek aplikatu ahal izateko. Hitzak normalizatzeko sistema jadanik garatuta dagoenez, hori integratu ahal izateko web-aplikazio baten prototipo bat sortzea izango da proiektu honen helburu nagusia, *Iturbila* deituko dioguna. Horrez gain, behin hitz estandarrak lorturik, lematizatzailea aplikatzeko aukera ere izango litzateke.

Beraz, proiektu honetan web-aplikazio bat sortu beharko da, zeinaren bidez bilaketa zabalagoak egin ahal izango diren, hau da, hitz estandarren eta lemen bilaketa eginez, hitz horiei lotutako hitz ez-estandarrak ere bilatu ahal izango diren.

2. KAPITULUA

Proiektuaren Helburuen Dokumentua

Memoriaren atal honetan, proiektuak bete beharko lituzkeen helburuak azaltzeaz gain, proiektuaren irismenari buruz hitz egingo da, emangarri nagusiak, baliabideak, proiektua gauzatzeko plana, kalitate-plana, arriskuak eta interesatuak zein diren zehaztuz.

2.1 Helburuak

Gradu Amaierako Proiektua izanik, proiektuaren nondik norakoak azaltzen dituen memoria idaztea eta proiektuaren defentsarako aurkezpena prestatzea derrigorrezkoak izango dira. Horretaz gain, hauek dira lortu nahi ditugun helburu nagusiak:

- Testu historikoak dituzten webgunetara jotzea, batez ere, OCR[1] –Optical Character Recognition– egina dutenekin lotura zuzena izatea.
- Berrerabilpenari lehentasuna eta garrantzia ematea, API[2] eta eduki irekiak erabiliz; webgune horietako testuak eta faksimileak API bitartez atzitzea.
- Testu horietako hitzak normalizatzea; dagozkien euskara estandarreko hitzak eta lemak lortzea.
- Testu historikoetan bilaketa-mota ezberdinak gauzatu ahal izatea: hitz ez-estandar, estandar eta lemen arteko bilaketak egitea.

[1] *OCR* (Optical Character Recognition) testuen digitalizazioko prozesu bat da, faksimile bateko hitzak testu/datu moduan lortzen dituena.

[2] *API* (Application Programming Interface) funtzio multzo bat da, beste software batek abstrakzio-kapa gisa erabiltzeko.

Helburu horiez gain, proiektuaren helburu nagusia aurretik garatutako sistema bati aplikazio erreal bat egitea izango da. Kontuan izan beharko da oinarritzat erabiliko den sistema jadanik garatua dagoela, hau da, sistema horrek erabiliko duen eredua sortua dagoela eta, beraz, bilaketek itzuliko dituzten emaitzen fidagarritasuna horren araberakoa izango dela.

2.2 Irismena

Behin helburuak zehaztuta, atal honetan irismenari buruz hitz egingo dugu, besteak beste, proiektuaren betekizunez, mugarriez, produktuaren irismen-mailez, kalitate-planaz edota arriskuen analisiaz.

2.2.1 Betekizunak

Proiektu honetan web-aplikazio bat sortu beharko da, zeinetan zenbait testu historikotan bilaketa-mota ezberdinak gauzatu ahal izango diren. Betekizunak zein diren zehaztea oso garrantzitsua da proiektuaren beharrak zein diren zehazteko eta, beraz, proiektu arrakastatsu bat lortzeko. Horretarako, hiru azpiataletan banatuko ditugu betekizunak: (1) web-aplikazioaren bilaketaren atalekoak, (2) testuak lortu, normalizatu eta indizeak eguneratzeko atalekoak, eta (3) interfazeak bete beharrekoak.

1. Web-aplikazioaren bilaketaren ataleko betekizunak:
    - Bilaketa-hitza sartzeko eremu bat izango du.
    - Menu zabalgarri bat hitz-mota aukeratzeko: jatorrizkoa, estandarra edo lema.
    - Aukeraren arabera, bilaketa ezberdinak egingo dira:
        - Jatorrizko hitza aukeratuz gero, bilaketa-eremuko hitza soilik bilatuko da, forma horretan.
        - Estandarra aukeratuz gero, bilaketa-eremuan sartutako hitza estandarra dela ulertuko da eta, beraz, hitz estandarraz gain, hitz horri loturik dauden jatorrizko hitzak ere bilatuko dira, hots, hitz ez-estandarrak.
        - Azkenik, lema aukeratuz gero, bilaketa-eremuan sarturiko lemari dagozkion forma ezberdinak bilatuko dira, hots, hitz estandarrak eta baita ez-estandarrak ere.

2. Testuak lortu, normalizatu eta indizeak eguneratzeko atalaren betekizunak:

    - Testu historikoak dituzten webgunetatik lortuko dira testuak eta faksimileak.
    - Helburuetan zehaztu dugunez, berrerabilpenari garrantzia emango diogu, beraz, testuak eta faksimileak lortzeko, APIak erabiliko dira aurrez konfigurazio-fitxategi batean zehazturiko testuak lortzeko.
    - Testuak lortuta, bertako hitzak normalizatu eta lematizatu beharko dira; hitzak estandarrak diren kasuetan ere normalizazio-prozesua egingo da, ondoren lemak lortu ahal izateko.
    - Lorturiko hitzen indizeak gorde beharko dira, ondoren bilaketak egin ahal izateko.

3. Interfazearen betekizunak:

    - Hasierako pantailak bilaketa-eremua, hitz-mota aukeratzeko menu zabalgarria eta bilaketa gauzatzeko botoia izango ditu.
    - Bilaketa gauzatzean, emaitzak modu antolatu eta argi batean bistaratuko dira, aplikazioaren eta erabiltzailearen arteko elkarrekintza ahalik eta hoberena izan dadin. Horretarako, kontuan izan beharko ditugu ondorengo puntuak:
        - Emaitzak obraren arabera sailkatuta egongo dira; testu asko egon daitezkeela aurreikusirik, testuak ezkutatzeko aukeraren bat eskainiko da.
        - Testu bakoitzeko emaitzak orrialdeka banatuko dira.
        - Bilaketaren emaitza aukeratuta, bilaketa-hitza edo dagozkion estandar eta lemak markatuko dira eta, testuarekin batera, faksimilea bistaratuko da.
    - Egun gailu mota ugari daudenez, web-aplikazioak moldagarria izan beharko du.

Betekizunak aztertuta, ikus daiteke garapen-inguruneari dagokion betekizunik ez dagoela, izan ere, proiektu hau prototipo bat izango denez, zerbitzari moduan funtzionatuko duen makina batean exekutatuko baita, hots, lokalean.

Memoriari dagokionez, Informatika Fakultateko webguneko Gradu Amaierako Proiektuak azpiatalean dagoen eredua jarraitzea beharrezkoa izango da, beharren arabera, atalak egokituz.

2.2.2 Mugarriak

Hainbat dira proiektuan zehar egin beharreko atazak. Ataza bakoitza amaitzeko mugarri batzuk jartzea gomendatzen da, proiektuan egon litezkeen atzerapenak kontrolatu ahal izateko.

Gure kasuan, lau mugarri nagusi izango ditugu proiektuan zehar:

- **2015-02-18**: Proiektuaren plangintzarekin amaitzea.
- **2015-03-06**: Ikerkuntza atala amaitzea.
- **2015-03-27**: Proiektuaren analisia eta diseinua amaitzea.
- **2015-05-08**: Web-aplikazioaren garapena amaitzea.
- **2015-06-05**: Memoriarekin amaitzea.

Mugarri horiek orientagarriak soilik izango dira, hau da, proiektuaren martxaren arabera edo sortutako arazoek eragindako atzerapenen arabera moldatu beharko lirateke.

2.2.3 Produktuaren irismen-mailak

Irismena definitzeko orduan, bi maila zehaztuko ditugu: oinarrizko maila eta maila aurreratua.

Oinarrizko maila izango da proiektuak bete beharreko gutxieneko maila eta aurreratua, berriz, oinarrizkoaz gain, aukera gehiago eskaintzen dituena. Behin oinarrizko mailara iritsita, ordura arteko jarraipen eta kontrola egingo genuke eta erabakia hartu ea maila aurreratua lortzeko denborarik izango dugun ala ez, hau da, plangintzan zehazturiko ordu kopurua gainditu bada, oinarrizko mailarekin nahikoa izango genuke.

Hauek dira maila bakoitzak bete beharreko baldintzak:

- Oinarrizko maila: 2.2.1 atalean zehazturiko betekizun guztiak betetzen dituena izango da, funtzio gehiagorik edo hobekuntzarik ez duena.
- Maila aurreratua: betekizunez gain, testuak dituzten webgunetatik ausaz testuak lortu eta etiketatzeko aukera eskaintzen duena, hau da, ausaz hitz ez-estandarrak lortu, dagozkien estandarrak eskuz sartu eta ikasketa prozesua berriro gauzatzen duena.

2.2.4 Irismenaren kudeaketa

Gerta liteke hasiera batean definitutako irismena birplanteatu behar izatea, 2.8 ataleko interesatuek horrela erabaki dutelako edo atzerapenak izan direlako. Kasu horietan, oso garrantzitsua izango da parte interesatuek bilera batean erabakitzea, proiektuaren mugarriak ere aldatzea eragin baitezake.

Beraz, irismenean aldaketaren bat egin behar izanez gero, aurrez bilera batean eztabaidatu eta proiektuaren plangintza berriro aztertuko beharko da.

2.2.5 Lanaren Deskonposaketa Egitura

LDE diagramak proiektuan zehar egin beharreko lana atazetan eta azpiatazetan banatzen du, proiektuaren irismena lortu ahal izateko.

2.1 irudian ikus daitekeen moduan, lana bost atazetan bereizi dugu:

Kudeaketa

Ataza honetan proiektuaren inguruko kudeaketa-lanak sartzen dira; hasieran egin beharreko plangintza, garapenean zehar egin beharreko jarraipena eta kontrola eta egingo diren kudeaketa bilerak.

Ikerkuntza

Batez ere testuingurua aztertuko da eta EKS edo *framework*en erabileraren eta bilaketa-zerbitzarien erabileraren alde onak eta txarrak aztertuko dira. Behin azterketa eginik, teknologia egokienak aukeratuko dira.

Analisia eta diseinua

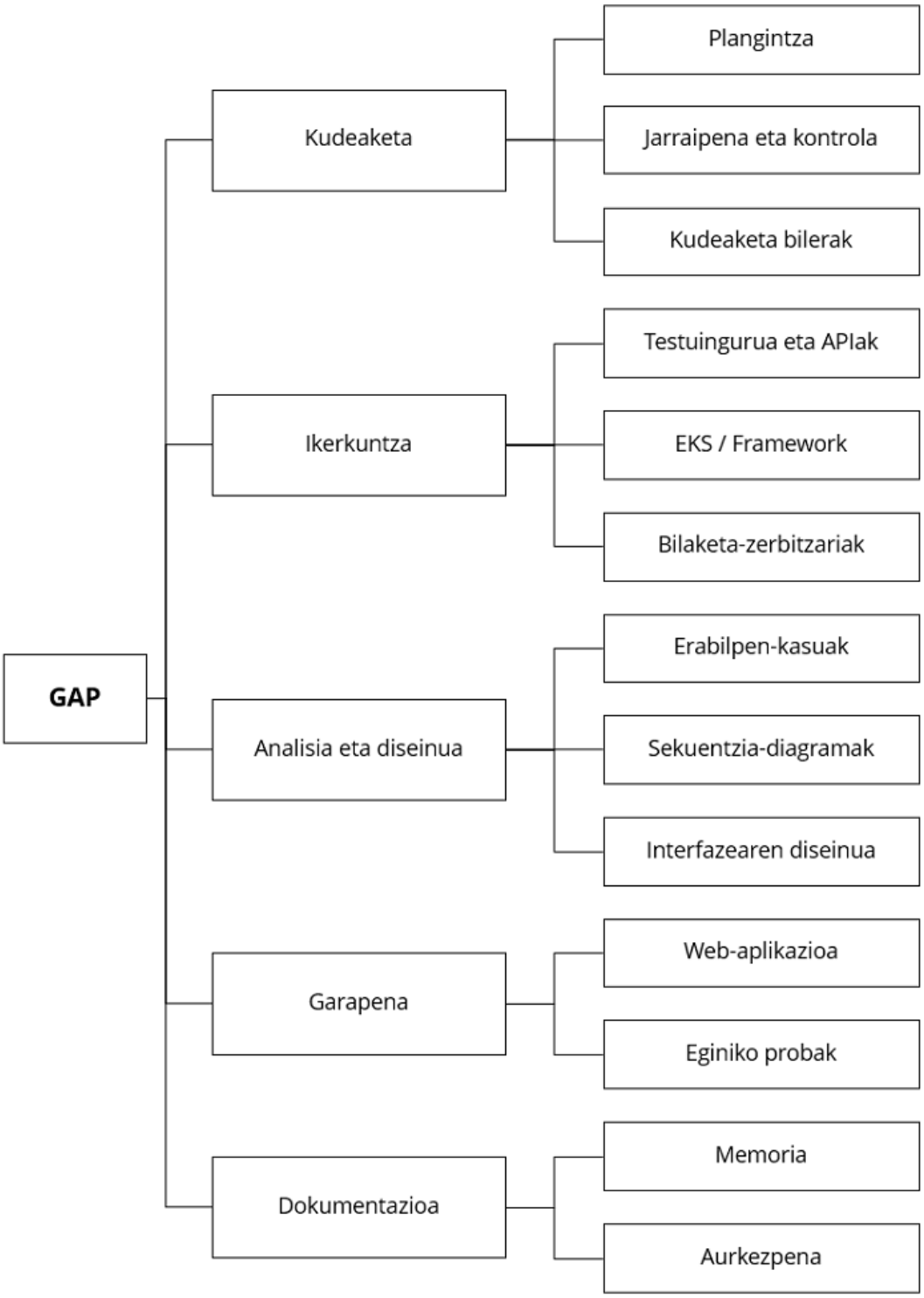
Proiektuaren betekizunak beteko dituzten erabilpen-kasuak deskribatu, horien sekuentzia-diagramak egin eta interfazearen diseinua egingo da.

Garapena

Sekuentzia-diagramei jarraituz eta aukeratutako teknologiez baliatuz, erabilpen-kasuak garatuko dira.

Dokumentazioa

Proiektuaren memoria eta aurkezpena egingo dira.



2.1 Irudia: LDE diagrama.

2.2.6 Atazen definizioa

Ondorengo 2.1 taulan ikus daiteke LDE diagramako ataza bakoitzaren azalpena.

Kodea	Ataza	Azalpena
Kudeaketa		
1.1	Plangintza	Proiektuaren plangintza, irismena, egutegia, arriskuen analisia, kalitate-planak...
1.2	Jarraipena	Proiektuaren egoera aztertzeko bilerak egin zuzendariekin eta, beharren arabera, plangintza aldatu.
1.3	Kontrola	Arriskuak kontrolatu eta erabakitako kalitatea-maila ziurtatu; egin behar den edozein aldaketa zehaztu.
Ikerkuntza		
2.1	Testuingurua eta APIak	Testuak lortzeko erabiliko diren webguneak aztertu eta bakoitzaren APIaren erabilera dokumentatu.
2.2	EKS/Framework erabilera	EKS eta *framework*en abantaila eta desabantailak aztertu eta zein erabili erabaki.
2.3	Bilaketa-zerbitzariak	Bilaketa-zerbitzariren bat erabiltzearen abantailak eta desabantailak aztertu.
Analisia eta diseinua		
3.1	Erabilpen-kasuak	Betekizunak bete ahal izateko erabilpen-kasuak deskribatu, aktorea, ohiko fluxua eta ezohiko fluxua adieraziz.
3.2	Sekuentzia-diagramak	Erabilpen-kasu bakoitzari dagokion sekuentzia-diagrama diseinatu.
3.3	Interfazearen diseinua	Betekizunak betetzeko eta erabiltzaileak bilaketak ahalik eta egokien gauzatzeko interfazea diseinatu.
Garapena		
4.1	Web-aplikazioa	Aplikazioaren erabilpen-kasuen garapena; barne hartzen ditu teknologien instalazioa, konfigurazioa eta bateratzea.
4.2	Eginiko probak	Aplikazioa probatzeko adibideekin eginiko probak.
Dokumentazioa		
5.1	Memoria	Proiektuaren plangintza, aurrekariak, analisia eta diseinua, garapena, jarraipen eta kontrola eta ondorioak biltzen dituen dokumentua idatzi.
5.2	Aurkezpena	Proiektuaren defentsarako aurkezpena prestatu.

2.1 Taula: LDE diagramako atazen definizioa.

2.3 Emangarri nagusiak

Proiektuak hiru emangarri nagusi izango ditu:

- Web-aplikazioaren kodea, ADDI, GitHub edo antzeko plataformara igoko dena.
- Proiektuaren nondik norakoak islatzen dituen memoria.
- Defentsaren eguneko aurkezpena.

2.4 Baliabideak

Ondorengo azpiataletan azalduko dira proiektua garatzeko erabiliko diren software-baliabideak nahiz egin beharko diren eskuraketak.

2.4.1 Software-baliabideak

Proiektu honetan, 2.1 atalean zehaztu moduan, eduki irekiak erabiliko dira. ahal den neurrian. Erabiliko den lan-tresna nagusia Ubuntu 14.04 bertsioa duen konputagailua izango da, izan ere, aurretik garatutako normalizazio-sistemak lan-ingurune horretan darabilen softwarea erabiltzen baitu, adibidez, *Phonetisaurus* edo *Foma*.

Web-aplikazioa garatu ahal izateko, aurretik aukeratu beharko da ea eduki-kudeatzailerik edo *framework*ik erabiliko den. Horren arabera, programazio-ingurunea aldatuko da, baina ez bada eduki-kudeatzailerik erabiltzea erabakitzen, *Bluefish* editorea erabiliko da PHP, HTML, Javascript eta antzeko kodeak idazteko.

Era berean, web-aplikazioa lokalean exekutatu beharko denez, LAMP erabili beharko dugu, hau da, Linux sistema eragilea, Apache web-zerbitzaria, MySQL datu-base kudeatzailea eta PHP programazio-lengoaia.

3.2.3 atalean aztertuko da ea bilaketak gauzatzeko zein aukera dauden, baina bilaketa-zerbitzaririk erabili behar izanez gero, horri dagokion softwarea ere lortu beharko da.

Aipaturiko softwareaz gain, beharren arabera sor liteke beste softwareren bat lortzeko beharra.

2.4.2 Eskuraketak

Proiektu honetan, aurreko talean aipatutako softwarea eskuratzeaz gain, badira proiektuaren arrakasta kolokan jar dezaketen eta, beraz, egin beharko ditugun bi eskuraketa: testuen hitzak normalizatzeko sistema eta hitzak lematizatzeko sistema.

Eskuraketa horiek proiektuaren funtsa badira ere, zuzendariek pasa beharreko ereduak dira, beraz, ez da aparteko prozesurik jarraituko, izan ere, zuzendariek ereduak ahalik eta azkarren pasatzeko konpromezua hartu baitute.

Hala ere, bada kontuan izan beharreko eskuraketa bat: testu historikoak biltzen dituzten webguneetako testu eta faksimileak lortzearena. Proiektuaren helburua ez da bertako informazio guztia eskuratzea eta biltegiratzea, testuak API bitartez lortu, bertako hitzak normalizatu eta lematizatu eta faksimileak ere API bitartez lortzea baizik. Hala ere, arriskuen analisian, 2.7 atalean, hobeto azalduko dugun arren, arriskuak ekidite aldera, testuak biltegiratuko dira, baina faksimileak emaitza bat bistaratzeko orduan soilik lortuko dira jatorrizko webgunetatik, bakoitzaren APIa erabiliz.

Memoria idazteko orduan, Informatika Fakultateko webgunean bi eredu daude Gradu Amaierako Proiektuaren memoria idazteko, LibreOffice eta LaTeXformatukoak. Bietako bat aukeratu behar denez, LaTeXerabiliko da, mota honetako memoriak idazteko egokiagoa delako, besteak beste, atal, paragrafo eta irudien kokapenak automatikoak direlako eta, beraz, azken ordura arte formatua ematen ibili beharrik ez izateko.

2.5 Proiektua gauzatzeko plana

Proiektu bat egiterako garaian, ezinbestekoa izango da plan bat garatzea; horretarako, ondorengo azpiataletan azalduko dugu erabiliko den lan-metodologia, atazen definizioa, egutegia, Gantt diagrama eta lan-karga.

2.5.1 Lan-metodologia

Proiektua ahalik eta egokien garatzeko lan-metodologia bat zehaztu beharko da, denboragalerak ekidin eta egiten den lana eraginkorra izateko. Izan ere, metodologiarik jarraituko ez balitz, ez lirateke akatsak ongi identifikatuko azken momentura arte, beraz, proiektuaren porrota eragin lezake.

Gure kasuan, lana iterazioka egitea erabaki dugu, hau da, teknologiak pixkana gehitzen eta funtzionalitateak pausoz pauso garatzen joatea, horrela, proiektuaren oinarriari eragin liezaiokeen akatsen bat gertatuko balitz, garaiz erreakzionatu eta egin beharreko aldaketak egin ahal izateko denbora izateko.

Behin ikerkuntza atala buruturik, proiektuaren garapenean hiru iterazio planteatu dira:

- Lehen iterazio batean, aurretik aukeratutako webgunetako testuak lortzeko API deiak zehaztuko dira.
- Bigarren iterazioan, bilaketa-zerbitzariko indizeak eskuz sortuko dira eta aplikazioaren pauso garrantzitsuenak ere garatuko dira.
- Hirugarren, eta azken, iterazioan, hasieran lortutako testuen hitz ez-estandarrak normalizatu eta lematizatuko dira. Hitz horiek bilaketa-zerbitzariak modu automatiko batean indexatuko ditu eta web-aplikazioaren bitartez bilaketak egin ahal izango dira.

Iterazio bakoitzaren aurretik diseinua egitea beharrezkoa izango da, Software Ingeniaritza irakasgaian ikas genuen moduan, diseinu bat lantzeak garatzeko garaian lana asko errazten baitu.

2.5.2 Egutegia

Informatika Ingeniaritzako Gradua amaitzeko egin beharreko lana da Gradu Amaierako Proiektua. Gutxi gorabehera 300 orduko dedikazioa eskatuko digun proiektua izanik eta hain proiektu handia kudeatzen dugun lehen aldia izanik, plangintzarekiko desbideraperak egon daitezkeela aurreikusten dugu. Hala ere, hori ere bada proiektua gauzatzearen helburuetako bat, hau da, dauden desbideraketak aztertu eta plangintza birmoldatzea proiektuaren arrakasta lortzeko.

300 ordu horiek banatzeko, ataza bakoitzari dagokion ordu-kopuruaren estimazioa egin behar da baina, Proiektuen Kudeaketa irakasgaian ikasi moduan, komeni da ordu gutxiago planifikatzea, atzerapenik izanez gero, ordu-kopuru horretan amaitu ahal izateko.

Hori horrela, 2.2 taulan ikus daiteke ataza bakoitzari eskainiko zaion dedikazioaren estimazioa. Aipatu behar da ez dela zehaztasun handirik eskaintzen, hau da, ez dela ataza bakoitzean sartuko azpiataza bakoitzaren xehetasunik eskaintzen, izan ere, lehen proiektu

handia izanik, ez baitakigu zehazki atal bakoitzari zenbat denbora eskaini beharko diogun. Gainera, xehetasun handietan sartzeak denbora gehiago eskatuko liguke eta, gure kasuan behar duguna plangintza orokor bat denez, ez genioke probetxurik aterako.

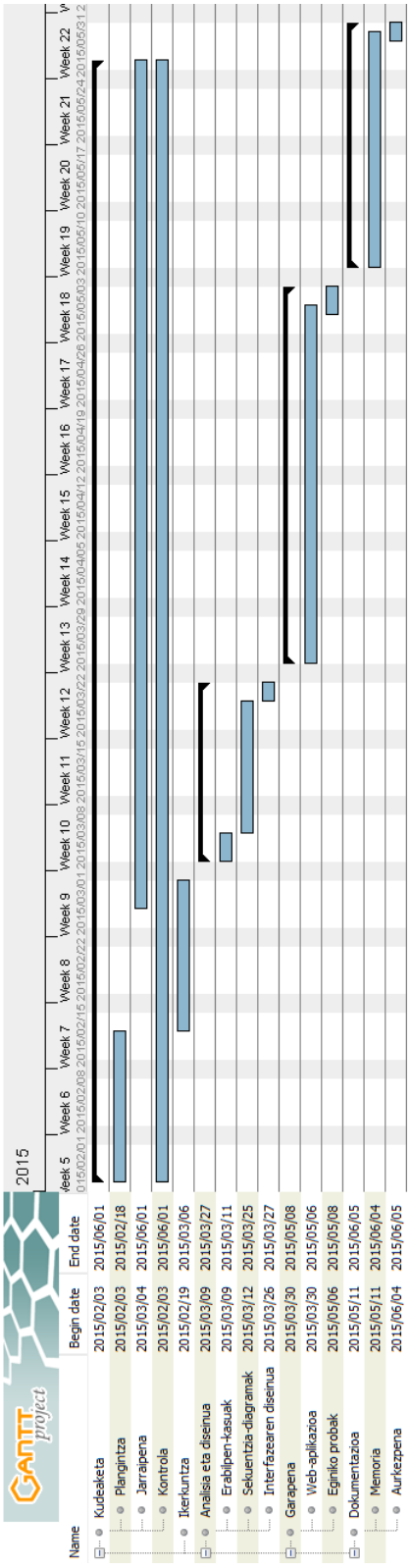
Kodea	**Ataza**	**Estimazioa**
Kudeaketa		
1.1	Plangintza	15 ordu
1.2	Jarraipena	10 ordu
1.3	Kontrola	5 ordu
		Kudeaketa: **30 ordu**
Ikerkuntza		
2.1	Testuingurua eta APIak	10 ordu
2.2	EKS/Framework erabilera	10 ordu
2.3	Bilaketa-zerbitzariak	10 ordu
		Ikerkuntza: **30 ordu**
Analisia eta diseinua		
3.1	Erabilpen-kasuak	10 ordu
3.2	Sekuentzia-diagramak	40 ordu
3.3	Interfazearen diseinua	10 ordu
		Analisia eta diseinua: **60 ordu**
Garapena		
4.1	Web-aplikazioa	95 ordu
4.2	Eginiko probak	5 ordu
		Garapena: **100 ordu**
Dokumentazioa		
5.1	Memoria	55 ordu
5.2	Aurkezpena	5 ordu
		Dokumentazioa: **60 ordu**
		Guztira: **280 ordu**

2.2 Taula: Ataza bakoitzaren estimazioa.

2.5.3 Gantt diagrama

Aurreko 2.5.2 azpiatalean ataza bakoitzari egin diogun estimazioa grafikoki ikusteko modurik onena Gantt diagrama erabiltzea da. Diagrama horren bitartez, ataza bakoitzaren estimazioa zein den hobeto ikus daiteke proiektuak irauten duen denboran zehar.

2.2 irudian dago hasierako estimazioen Gantt diagrama.



2.2 Irudia: Hasierako estimazioen Gantt diagrama.

2.5.4 Lan-karga

2.2 irudiko Gantt diagraman ikus daitekeenez, 2015-02-03tik 2015-06-05ra planifikatu dugu proiektua, hots, lau hilabete, 76 lanegun. Egun bakoitzeko dedikazioa aztertuz gero, ia 4 ordu dira egunean sartu beharko genituzkenak.

Kontuan izan behar dugu, proiektuaz gain, irakasgaiak ere aurrera atera behar direla eta azterketak ere prestatu behar direla, beraz, hasierako hilabeteetan dedikazioa txikiagoa izango da, eta gerta liteke amaierakoan ere dedikazioa jaistea lanean hasiz gero. Hala ere, lanek uzten duten neurrian, proiektuarekin aurrera jarraitzen saiatuko gara.

Jakina da dedikatzeko denbora gutxi dugulako geldialdiak egin behar izanez gero, utzitako tokitik berriro hastea zailagoa gerta litekeela, batez ere kode baten garapenean, beraz, hori gertatuz gero, planteatu beharko dugu ea dedikazio horrekin aurrera jarraituko dugun edo behin-behinean proiektua alde batera utziko dugun, betiere zuzendariekin hitzegin ondoren.

2.6 Kalitate-plana

Proiektuaren arrakasta neurtzeko modurik garbiena, zalantzarik gabe, prototipoa probatzea da, hau da, eginiko bilaketak esperotako emaitzak modu antolatu batean itzultzea. Hala ere, garatutako produktuak kalitate-maila minimo bat betetzen duela ziurtatzeko adierazle batzuk zehaztuko ditugu eta kalitatearen kontrola ere gauzatuko dugu.

2.6.1 Kalitatearen adierazleak

- Web-aplikazioaren azkartasuna.
  - Kalitate elementua: web-aplikazioak arintasunez funtzionatzea.
  - Azalpena: erabiltzaileek erabili beharreko aplikazioa denez, modu ‘azkar’ batean funtzionatu beharko du, bilaketen emaitzak azkar itzuliz.
  - Neurtzeko modua: bilaketa-emaitzak itzultzeko denbora neurtuz.
  - Maximo onargarria: 6 segundo.
  - Lortu nahi den neurria: 1-2 segundo.
- Interfazearen kalitatea.
  - Kalitate elementua: web-aplikazioaren interfazearen egitura.

  - Azalpena: aplikazioa gailu ezberdinetarako moldagarria izatea, emaitzak zenbateraino antolatuta bistaratzen diren eta zenbateraino erabilerraza den.
  - Neurtzeko modua: aplikazioa zenbat gailu ezberdinetara moldatzen den, bilaketako emaitza guztiak pantaila berean azaltzea, *scroll* egin beharrik gabe eta ahalik eta klik gutxienekin.
  - Minimo onargarria: ordenagailuetarako soilik moldagarria izatea eta emaitzak orritan banatu gabe erakustea, baina bai testuka banatuta.
  - Lortu nahi den neurria: egungo gailu guztietara moldatzea, emaitzak testuka eta testu bakoitzeko emaitzak orrika banatuta egotea eta gehienez 5 klik-ekin emaitza zehatz bat bistaratzea.

- Memoriaren kalitatea.

  - Kalitate elementua: eduki argiak eta osatuak izatea.
  - Azalpena: proiektuaren prozesu guztia dokumentatzea, ahalik eta modu argienean eta edukiak modu antolatu eta osatu batean.
  - Neurtzeko modua: atalak ongi antolatuta eta garatuta dauden, ea irudi edota diagrama lagungarririk duen eta proiektuan eman diren pauso guztiak biltzen dituen.
  - Minimo onargarria: gutxienez 75 orrialde izatea eta edukiak antolatuta izatea.
  - Lortu nahi den neurria: 100 orrialdetik gora izatea, prozesu guztia dokumentaturik egotea modu argian, irudi eta diagrama lagungarriekin.

Aurreko kalitate-adierazleen gain, kontuan izan beharko ditugu, besteak beste, sistemaren fidagarritasuna –oro har ongi funtzionatzea, sistemak hutsik ez egitea– edota kodefitxategien antolaketa nahiz argitasuna –txukun eta era ordenatuan idatzita egotea, beharrezko iruzkin eta argibideekin–.

2.6.2 Kalitatearen kontrola

Aurreko atalean zehaztutako kalitate-adierazleen kontrola egin beharko dugu. Kontrol zehatza amaieran egingo da, A eranskineko egiaztapen-zerrenda betez, baina proiektuaren bizi-zikloan zehar ere ziurtapen batzuk egin beharko ditugu, adierazle horiek betetzeko helburuarekin.

2.7 Arriskuen analisia

Proiektua aurrera eramateko garaian, hainbat oztopo, arrisku topa ditzakegu bidean. Arrisku hauek aurrez identifikatzeari esker, hauek arazo bihurtzea ekidin dezakegu. Hala ere, esan beharra dago, bi motako arriskuak daudela: batetik, ezagunak direnak, identifikatu eta analizatzen diren arriskuak, horien erantzunak planifikatu daitezkeenak eta, bestetik, ezezagunak diren arriskuak, modu proaktibo batean kudeatu ezin direnak.

Ondorengo puntuetan identifikatuko ditugu zein diren proiektu honetako arrisku ezagunak, gertatzeko duten probabilitatea eta izan dezaketen eragina:

- Testuak lorteko webguneek funtzionatzeari uztea.
  - Arriskuaren deskribapena: gerta liteke testuak lortu nahi diren webguneek funtzionatzeari uztea. Kontuan izan beharko da proiektuaren garapena zein puntutan aurkitzen den; hasieran gertatuz gero, beste webguneren bat bilatu beharko litzateke, baina amaiera aldean gertatuz gero, web-aplikazioaren zati handi bat aldatu behar izatea eragingo luke. Azken horrek proiektuaren plangintza moldatu behar izatea ere eskatuko liguke.
  - Probabilitatea: txikia.
  - Eragina: oso handia.
- Web-aplikazioaren garapenerako tresnak ezin eskuratu.
  - Arriskuaren deskribapena: web-aplikazioaren garapenerako eskuratu beharreko bi tresna-mota bereizten dira: aukerakoak eta behar-beharrezkoak. Ezin bada aukerakoren bat eskuratu, beste aukeraren bat aztertu beharko da, baina eskuratu ezin dena behar-beharrezkoa bada, proiektuaren plangintza erabat aldatu liteke.
  - Probabilitatea: ertaina.
  - Eragina: aukerako tresna bada, txikia; behar-beharrezkoa bada, oso handia.
- Garatutako kodea galtzea.
  - Arriskuaren deskribapena: kodea biltegiratzen den gailua hondatzen bada, ordura arteko guztia galduko genuke. Hori gerta ez dakigun, GitHub erreposito-rioan jasotzeaz gain, aldian behin segurtasun-kopiak egingo ditugu.

  - Probabilitatea: txikia.
  - Eragina: handia.

- Proiektuaren memoria galtzea.
  - Arriskuaren deskribapena: memoriaren dokumentua dagoen gailua hondatuz gero, idatzitako guztia galduko genuke. Hori ez gertatzeko, erabiltzaileari fitxategiak hodeian gordetzeko eta sinkronizatzeko aukera ematen dion Dropbox aplikazioa erabiliko dugu eta segurtasun-kopiak ere egingo ditugu.
  - Probabilitatea: ertaina.
  - Eragina: hasieran gertatuz gero, txikia; amaiera oso handia.

- Proiektuan atzerapenak izatea.
  - Arriskuaren deskribapena: arrazoi ezberdinengatik gerta liteke atzerapenak egotea, baina beti atzerapen hori zerk eragin duen aztertu beharko da. Oro har, bi motako arrazoiak izan daitezke: teknikoak, web-aplikazioaren garapenean sorturiko arazoek eragindakoak edo pertsonalak, gaixotasunek edo arazo pertsonalek eragindakoak.
  - Probabilitatea: proiektua beste irakasgai batzuekin batera egin behar denez eta irakasgaiak amaitzean lanean hasteko asmoa dagoenez, atzerapenak izateko probabilitatea handia da.
  - Eragina: hasierako plangintzan eragin handia izan dezake, batez ere mugarrietan, baina plangintza beharren arabera moldatuz gero, eragina txikitu daiteke.

2.8 Interesatuak

Alde batetik, proiektua arrakastaz amaitzeko lehen interesatua ikaslea bera izango da, hau da, proiektua aurrera eramateko erantzukizuna hartuko duena.

Bestetik, proiektuko zuzendariak izango lirateke interesatuak: Izaskun Etxeberria eta Iñaki Alegria. Hauen zeregina izango da proiektuak bete beharreko atalak ikasleari azaltzea eta proiektuan zehar sor daitezkeen zalantzak argitzea. Bi zuzendariak “Konputagailuen Arkitektura eta Teknologia” sailekoak diren arren, proiektua Software Ingeniaritza espezialitatekoa denez, ahal duten neurrian lagunduko dute.

Software Ingeniaritza espezialitateko proiektua izanik, espezialitate horretako irakasleez osatutako epaimahaia izango da azken interesatua.

2.8.1 Komunikazio-plana

Zuzendariekin egingo den komunikazioari dagokionez, e-posta bidez edo aurrez aurreko bileren bitartez egingo da. Bilera-datak erabakitzeko, ikasleak mezu bat bidaliko die zuzendariei edo zuzendaritako baten fakultateko bulegotik pasako da.

Epaimahaiarekin, aldiz, ez da komunikaziorik izango defentsa-eguna iritsi arte.

3. KAPITULUA

Aurrekariak eta Erabilitako Teknologia

Atal honetan proiektuaren helburuak bete ahal izateko eginiko azterketa azaltzen da. Azterketa hori gauzatzeko, kontuan izan dira orain arte sortuta dauden webgune, tresna eta abarrak, izan ere, 2.1 atalean aipatu dugun moduan, proiektuaren helburuetako bat aurrekoek eginikoa berrerabiltzea baita.

Berrerabilpenari dagokionez, testu historikoak biltzen dituzten guneak erabili behar direnez, hauek atzitzeko aukerarik dagoen aztertu beharko dugu, hots, ea APIrik baduten eta zein den erabilera.

Proiektuaren puntu garrantzitsuenetako bat da Etxeberria et al. (2016) lanean garatutako sistema aplikazio erreal batean kokatzea, hau da, testu historikoetako hitz ez-estandarrak izanik, hitz horri dagokion estandarra eta lema lortzea, hitz estandarren edota lemen bilaketak eginez, ez-estandarrak ere lortu ahal izateko.

Beraz, ondorengo azpiataletan proiektuaren aurrekariak, teknologia posibleen azterketak eta erabilitako teknologien azalpenak ematen dira.

3.1 Aurrekariak

Proiektua kokatzeko, aurrekari gisa, testuingurua, APIen erabilera eta testuen normalizazioa azpiatalak landuko ditugu.

3.1.1 Testuingurua

Testu historikoak biltzen dituzten guneen bitartez informazioa lortu behar dugunez, euskarazko testuak dituztenak soilik aztertu ditugu eta hauek izan dira, hain zuzen ere, hiru aukeratuak.

Wikisource[1]

Wikisource, edo euskaraz, Wikiiturriak, jatorrizko testu zaharrak nahiz lizentzia librea duen edozein testu biltzen duen liburutegi askea da, Creative Commons-en aitortu eta partekatu-berdin lizentziapean.

2003 urteko azaroan sortu zen proiektua *Project Sourceberg* izenpean, baina urte bereko abenduan bertan izena aldatu zitzaion, eta gaur egun ezagutzen dugun Wikisource izatera pasatu zen. Hebreerazko Wikisourceren bertsioa izan zen azpidomeinu independente bat lortzen lehena, 2004 urteko abuztuan, eta orduz gero beste zenbait azpidomeinu sortu dira: 63 hain zuzen ere. Hortaz, esan liteke hizkuntza bakoitzak azpidomeinu bat duela, baina hori ez da beti horrela; izan ere, oso artikulu gutxi dituzten hizkuntzek ez baitute azpidomeinu propioa, kategoria baizik. Azken kasu hau da euskararen egungo egoera. Wikisourcen 4 milioitik gora artikulu dauden egun, eta euskarazkoak 52 obra besterik ez dira.

Iturriak[2]

Iturriak i2basque[3]-ren tresna bat da, liburutegi, ikertzaile eta hezkuntza komunitatearentzat zenbait testu historiko erabilgarri jartzen dituena, herritarren laguntzaz dokumentu horien edukiak transkribatu ahal izateko. Honek ere Wikisourceren lizentzia berdina du.

Ondorengo helburuak ditu:

- Euskal Herriko Ondare Dokumentala zaintzea, hedatzea eta erabilera sustatzea.
- Testuak kultura- eta hezkuntza-zentroen sareetan jartzea, dokumentu historikoen kontsulta arruntez haratago, zerbitzu gehiago eskainiz.

[1] https://wikisource.org/wiki/Main_Page/Euskara
[2] http://www.iturriak.eus/index.php/Portada
[3] i2basque Euskadi Informazio Gizartean Planaren barruko programa da, Hezkuntza, Unibertsitate eta Ikerketa Sailaren ekimenez sortua.

- Dokumentu horien erabilera sustatzea eta erabiltzaile kopurua handitzea, herritarrei dokumentazio historikoaren garrantzia eta memoria pertsonala nahiz kolektiboaren balioa helarazteko.

Proiektu hau 2014ko martxoaren 3an ireki zen eta egun 29 obra ditu. Obra horietatik 5 soilik dira euskarazkoak, eskuizkribu bat eta 4 liburu, baina ez daude guztiz osaturik; hain zuzen ere, *Escualdun Cocinera (1864)* da osotasunez euskaraz traskribatutako bakarra.

Armiarma - Klasikoen gordailua[4]

Euskal literaturako testu klasikoen gordailu bat da, euskarazko literatura, euskaraz sortutakoa eta euskarari ekarritakoa biltzen dituena. Azken urteetan geldirik izan den Euskal Testuen Gordailua eta Susa literatur argitaletxearekin elkarlanean sortua izan da. Azken honek Armiarma literatur atariaren oinarrizko zutabe ikusten zuen euskal literaturako testu klasikoak sarean jartzea eta, beraz, asmo biak batzetik sortu zuten 2005. urtean, biekin elkarlanean.

Alde batetik, testu klasikoak biltzen ditu, baina bestetik, badira beste zenbait teorian klasikoak ez direnak edota testu literarioak ere ez direnak; adibidez, testu epigrafikoak, ahozko literatura edo herri literatura. Hala ere, gure lanerako, literaturako idazlanak soilik dira interesgarriak.

Testuen gordailu honen helburua, aurreko kasuen moduan, obra horiek irakurtzea, erabiltzea eta ezagutzea dira, eta ez literatura eta filologiaren esparruko jendeak soilik, beste askorentzat ere interesgarria izan baitaiteke eskuragarri dauden obra osoak atzitu ahal izatea. Lizentziari dagokionez, Jabego Intelektualaren Legearen arabera jatorrizko idazlanak bestelako eskubiderik ez baleuka, nahi bezala erreproduzitu daitezke.

3.1.2 APIen azterketa

Aurreko hiru aukerak aztertu ostean, ikusi dugu Wikisource eta Iturriak guneek badutela APIa, baina ez Armiarmaren Klasikoen Gordailuak. Beraz, Wikisource eta Iturriak plataformek API berdina dutela baliatuz, bi guneak erabiltzea erabaki dugu, hots, bi guneetako testuak lortuko ditugu.

Hala ere, egin beharreko eskaerak ezberdinak izango dira hizkuntzaren edota kokatuta dauden tokiaren arabera; adibidez, Wikisourcen ingelesezko eskaera bat egin nahi bada,

[4] http://klasikoak.armiarma.eus

honakoa izango da abiapuntua: http://wikisource.org/w/api.php. Aldiz, azpidomeinua duten kasuetan, nahikoa izango da hasieran dagokion domeinua, '.' ikurra eta aurreko esteka jartzea; adibidez, Herbehereetako Wikisourcen kontsulta bat egin nahi badugu, eta azpidomeinua nl dela jakinik, ondorengotik abiatu beharko genuke kontsulta: http://nl.wikisource.org/w/api.php. Esan bezala, hizkuntza batzuek azpidomeinu propioa dute, baina beste batzuek ez. Azpidomeinua ez dutenentzat, euskara kasu, ingelesezko estekatik abiatuko litzateke kontsulta. Horrela, behin amaiera puntua finkaturik, horren ostean '?' ikurra jarri eta zenbait parametro pasa beharko lirateke nahi dugun kontsulta egiteko.

Iturriak-en kasuan ez dago azpidomeinurik, beraz, euskarazko eta gaztelaniazko testuak domeinuaren arabera bereizten dira: *.eus* domeinua euskarazko testuetarako eta *.es* domeinua gaztelaniazkoentzat.

Egin beharreko eskaera zehazteko, ondorengo parametroak pasatu behar zaizkio:

- **action:** zein ekintza egin nahi den adierazteko. Kontsultak egiteko, action=query izango litzateke.
- **format:** eskaeraren erantzuna zein formatutan lortu nahi den adierazteko. Egun erabiltzen diren formatuak JSON eta XML izango lirateke, hortaz, format=json eta format=xml lirateke.
- **titles:** eskaeraren titulua adierazteko; eskaera bat baino gehiago egin nahi badira, honela konbina daitezke: titles=PageA|PageB|PageC.
- **prop:** orriaren aurreko bertsioren bat nahi bada; azkena lortzek prop=revisions jarri behar da.
- **rvprop:** orriaren azken bertsioaren edukia nahi dugula adierazteko: rvprop=content.
- **rawcontinue:** dirudienez, edukiaren formatua alda daiteke eta, eskaerak orain arte moduan egin ahal izateko, rawcontinue jarri behar da.

Adibide moduan, XML formatuan Wikisourceko Axularren *Gero* obraren bigarren orrialdea lortzeko, ondorengo API deia egin beharko genuke:

```
https://wikisource.org/w/api.php?format=xml&action=query&titles=Page:
AxularGero.djvu/2&prop=revisions&rvprop=content&rawcontinue
```

```
<api>
   <query>
      <pages>
         <page _idx="130954" title="Page:AxularGero.djvu/2">
            <revisions>
               <rev contentformat="text/x-wiki"
                  contentmodel="proofread-page">
                  <div class="pagetext"> {center|GERO BI PARTETAN
                  partitua eta bereicia LEHENBICIKOAN EMAITENDA,
                  ADITCERA, cenbat calte eguiten duen, luçamendutan
                  ibiltceac, egitecoen gueroco utzteac. Bigarrenean...
               </rev>
            </revisions>
         </page>
      </pages>
   </query>
</api>
```

3.1 Irudia: Aurreko API deiaren emaitza, XML formatuan.

Aurreko eskaeraren emaitza 3.1 irudian ikus daiteke, testuari dagokion atala laburturik.

Aldiz, eskaera JSON bidezkoa bada, XMLren etiketa berdinak dituen JSON itzuliko luke, 3.2 irudikoa, hain zuzen ere.

Gure kasuan, `revisions` etiketaren edukia soilik erabiliko dugu, eskaturiko testuaren edukia izango duena.

Beraz, Wikisource nahiz Iturriak plataformetatik beharrezko testuak lortzeko ez genuke arazorik izango.

3.1.3 Testuen normalizazioa

Proiektu honetan aztertuko diren testuak nagusiki testu ez-estandarrak izango dira, hots, testu historikoak izanik, euskararen estandarizazio-prozesua baino lehen idatzitako testuak (XVI. mendetik XX. mende erdira artekoak). Hizkuntza prozesatzeko tresna gehienak hizkuntza estandarretan soilik idatzitako testuak prozesatzeko garatuak izan direnez eta proiektuan zehar bilaketak egiteko testu historikoak erabiliko ditugunez, normalizazio-prozesu bat egitea behar-beharrezkoa izango da. Adibide moduan, 1.1 atalean esan genuen moduan, egungo “bekatu” hitza testu historikoetan bilatzeko, jatorrizko hitzak, hots, forma ez-estandarrak bilatu beharko dira: “bekhatu” eta “bekathu”, adibidez.

```
{
  "query": {
    "pages": {
      "130954": {
        "pageid": 130954,
        "title": "Page:AxularGero.djvu/2",
        "revisions": [
         {
           "contentformat": "text/x-wiki",
           "*": "<div class="pagetext"> {center|GERO BI PARTETAN
                 partitua eta bereicia LEHENBICIKOAN EMAITENDA,
                 ADITCERA, cenbat calte eguiten duen, luçamendutan
                 ibiltceac, egitecoen gueroco utzteac. Bigarrenean..."
          }
        ]
      }
    }
  }
}
```


3.2 Irudia: Aurreko API deiaren emaitza, JSON formatuan.

Normalizazio-prozesu hori gauzatzeko, hiru teknika erabili ohi izan dira:

- Erregeletan oinarritutako metodoak hasieran gehien erabili izan diren metodoak dira. Metodo horietan, testuetako aldaeretan gertatzen diren fenomenoak aztertzen dira, eta eskuz idazten dira erregela fonologikoz osatutako gramatikak. Erregela horiek aldaerei aplikatuz gero, forma estandarrak aurkitzeko aukera dago.

- Teknika ez-gainbegiratuak deitutakoak, metodo erabat automatikoak direnak, heuristikoetan edo etiketatu gabeko corpusetan oinarritutakoak. Teknika horien artean edizio-distantzia edo distantzia fonetikoa dira erabilienak, aldaeratik gertuen dauden forma estandarrak lortzeko. Halako metodoak, *baseline* moduan erabiltzen dira maiz, proposatzen diren sistema berriak horiekin konparatzeko.

- Ikasketa automatikoa oinarri duten teknikak ere aplikatu izan dira ataza ebazteko. Teknika horien bitartez, sistemari hainbat adibide ematen zaizkio ikas dezan, eta gero eskatu egiten zaio ikasitakoa aplikatzea adibide berrien aurrean. Ikasketarako adibideak lortzeko eskuzko lana behar denez, teknika gainbegiratuak dira.

Egun eskuz idatzitako erregelak erabiltzen dituzten metodoak erabiltzen diren arren, ezin da dena erregelen bitartez normalizatu eta, beraz, beste tresna batzuekin probatu beharra

dago. Horixe izan da, hain zuzen ere, Etxeberria et al. (2016) lanak hartu duen bidea, hots, ikasketa automatikoarena, izan ere, eskuzko erregeletako sistema bat garatzeak denbora asko eta azterketa sakon bat eskatuko bailuke.

Garatutako normalizazio-sistema *Phonetisaurus* tresnan oinarritzen da. *Phonetisaurus* Josef Novak-ek garatutako tresna-multzoa da, 2012. urtean FSMNLP2012 (*Finite-State Methods and Natural Language Processing*) *workshop*ean aurkeztua, eta grafema-fonema (G2P, Grapheme-to-Phoneme) bihurketa egiten duten sistemak eraikitzeko balio du (Novak et al. (2012)). 3.3.1 atalean tresnaren inguruko xehetasun gehiago emango dira.

3.2 Teknologia posibleak

Proiektua garatzeko teknologia posibleak azaltzen dira atal honetan, izan ere, erabiliko den teknologiaren aukeraketa okerrak eragin izugarria izan baitezake, bai teknologiaren erabiltzeko zailtasunean bai proiektuaren dedikazioan. Horregatik, proiektuaren beharrak ikusirik, teknologiaren azterketa egin da, batik bat, Edukiak Kudeatzeko Sistemak erabili ala ez jakiteko, edo horien partez *Framework*ak erabiltzearen alde onak zein txarrak ikertzeko, eta testuetako bilaketak gauzatzeko aukera ezberdinak aztertzeko.

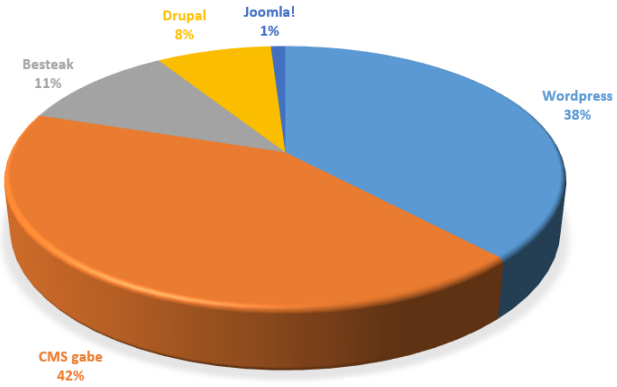
Azterketa horiek egiterako garaian, kontuan izan dira graduan zehar izandako irakasgaiak, adibidez, Edukiak Kudeatzeko Sistemei buruz ikasitakoa “Softwarea Garatzeko Tresna Aurreratuak” irakasgaian, edo “Proiektuen Kudeaketa” irakasgaian ikasitako EKS horietako baten erabilpena edota azterketak egiteko gaitasuna.

3.2.1 Edukiak Kudeatzeko Sistemak

Web-aplikazio bat egiterako garaian, puntu garrantzitsuetako bat aplikazioak izan beharko dituen funtzionaliteak aztertzea izango dira, hots, ea funtzionalitate horiek ohikoak diren –edukiak sortu, editatu, kudeatu, publikatu– edo, bereziak direnez, garatu behar izango diren.

Funtzionalitateak ohikoak diren kasuetarako, Edukiak Kudeatzeko Sistemak (EKS, ingelesezko CMS) erabiltzea gomendatzen da. EKS horien artean erabilienak Wordpress, Joomla eta Drupal dira. 3.3 irudian ikus daiteke EKS bakoitzaren erabilera datuak 2016. urtean.

Sistema mota hauek erabiliz, oso modu errazean gehitu daitezke testu, irudi eta antze-



3.3 Irudia: EKSen datuak, 2016. urtean. Iturria: BuildWith.

koak, edukietatik nabigatu, erabiltzaileak kudeatu, menuak administratu, eta abar. Gainera, funtzionalitateak hedatzeko aukera eskaintzen dute, *plugin*en bitartez. Eta hau dena, programatzen jakin beharrik izan gabe.

Hala ere, 3.3 irudian ikus daitekeen moduan, aplikazioen ehuneko handia da EKSrik erabili gabe garatua izan dena. Horren arrazoia, batik bat, aplikazio oso pertsonalizatuak sortu behar izateagatik da eta aplikazioaren beharrak asetuko dituzten *plugin*ak sortu gabe daudelako. Hori bai, kasu honetan behar-beharrezkoa da programatzen jakitea.

Proiektu honen kasuan, aplikazioa erabat pertsonalizatua da, hainbat tresna bateratu behar baitira, beraz, EKS bat erabiliko bagenu, lan gehiago suposatuko liguke bere kodean sartzea eta moldatzen hastea. Gainera, webguneak orri nagusi bakarra izango du, bilaketarena hain zuzen ere, eta prototipo bat izango denez, EKSrik ez erabiltzea erabaki dugu. Horrek ez du esan nahi garapenaren kode zati guztia idatzi beharko dugunik, izan ere, egun badaude lana asko arinduko diguten zenbait tresna, besteak beste, *framework*ak, bilaketa-zerbitzariak, eta abar.

3.2.2 *Framework*en erabilera

3.2.1 atalean erabaki dugun moduan, ez dugu EKSrik erabiliko, beraz, aplikazioa garatzerako orduan *framework* bat erabiltzea aztertu nahi dugu.

Framework bat HTML, CSS eta JavaScript fitxategien bilduma bat da, webgune bat garatzeko diseinu-patroiak dituena. Horri esker, kode errepikakorra behin eta berriz idatzi behar ez izatea lortzen dugu eta, beraz, horretan denbora galtzea saihesten digu. Gainera, interfaze garbiak eta tamaina eta mota guztietako gailuetara moldagarriak dira.

Hala ere, azter ditzagun *framework*ak erabiltzearen abantailak:

- **Azkartasuna**. Lanaren zati handi bat egina dagoenez, behin *framework*aren funtzionamendua ikasirik, webguneak azkarrago garatzea ahalbidetzen du.
- **Erraztasuna**. Egunerokotasunean erabiltzen diren funtzionalitateetako asko garatuta daudenez, ez dago gauzak nola egin pentsatzen ibili beharrik, oso erosoa izan litekeena.
- **Osagai gehigarriak**. *Framework* sendoek osagai gehigarri asko dituzte, konplexuagoak izan litezkeen funtzionalitateak garatu beharrik ez izateko.
- **Laguntza**. Sor litezkeen zalantzak argitzeko erraztasunak eskaintzen dituzte, komunitateen bitartez.

Eta desabantailak:

- **Ikasketa**. *Framework* bakoitzak modu ezberdinean funtzionatzen duenez, nola erabili ikastea behar-beharrezkoa izango da.
- **Gehiegizko kodea**. Garatzaileen behar gehienak asetzeko sortuak direnez, webgune oso sinpleetarako kode-zati asko egon liteke soberan, beraz, erabiltzaileak webgunea atzitzeko denbora gehiago behar izatea eskatuko luke.
- **Azken itxura**. Aukera asko modu lehenetsian daudenez, webgune baten eta beste baten itxura oso antzekoak izan litezke, ez bada CSSa moldatzen behintzat; hortaz, *Framework*ak diseinatzaileentzat baino, garatzaileentzat sortuak izan direla esan liteke.

Behin abantailak eta desabantailak azterturik, *Framework* bat erabiltzea erabaki dugu, izan ere, desabantailei erreparatuz gero, oso gutxi baitira. Gainera, gure kasuan, proiektu honen helburuetako bat teknologia berriak ikastea da eta kode asko edota beste webgune baten itxura antzekoa izateari ez diogu aparteko garrantzirik emango.

Orain zein aukeratu erabaki beharko dugu. Egungo *framework* ezagunenak *Bootstrap*, *Foundation* eta *Skeleton* izan daitezke, baina badira beste batzuk ere, hala nola, *Pure*, *Ink*, *Base*, *Metro UI CSS*... Hiru ezagunenen arteko konparaketa[5] aztertu ostean, *Bootstrap* aukeratu dugu, egunean egoteaz gain, 2.2.1 atalean zehaztutako betekizunetako bat webaplikazioa gailuetara moldagarria izatea denez, egokiena izan litekeelako.

[5] http://responsive.vermilion.com/compare.php

3.2.3 Bilaketa-zerbitzariak

Bilaketen funtzionalitatea garatzeko, hainbat aukera egon litezke, besteak beste, hitzak datu-base batean jaso eta kontsulta egin, edo testuak fitxategietan gorde eta bertan bilatzea. Aukera horien desabantailetako bat eraginkortasuna da, hitz bat bilatzeko, datu-base edo fixategi guztian bilatu behar baita.

Eraginkortasun hori hobetzeko, gaur egun badaude beste modu batera funtzionatzen duten tresnak: bilaketa-zerbitzariak. Hauen funtzionamendua honakoa da: fitxategi bat jasotzen denean, indizeetan hitzak jartzen ditu, hortaz, hitz bat bilatzean, zuzenean dagokion fitxategia edo, gure kasuan, testu zatia itzuliko digu.

Gaur egun hainbat bilaketa-zerbitzari daude, besteak beste, *Constellio*[6], *SearchBlox*[7] edo Amazonen *CloudSearch*[8]. Hala ere, kode irekikoak diren eta bi enpresa handik erabiltzen dutenak aztertuko ditugu: *Twitter*rek denbora errealeko bilaketak gauzatzeko erabiltzen duen *Apache Solr*[9] eta PayPal jabe duen *eBay* enkanteko webguneak erabiltzen duen *Elasticsearch*[10]. Biak ala biak *Lucene* proiektuan oinarrituta daude, hots, informazioa berreskuratzeko kode irekiko APIan, *Apache License 2.0*[11] lizentziapean, eta funtzionalki oso-oso antzekoak dira. Ondorengo puntuak kontuan izan behar ditugu bietatik egokiena aukeratu ahal izateko:

- *Elasticsearch*ek JSON formatua soilik onartzen duen bitartean, *Solr*ek XML, CSV eta JSON onartzen ditu.
- *Elasticsearch*ek nahiko dokumentazio urria du, beraz, azkarrago lan egiten dela iruditu arren, denbora asko gal liteke.
- Bilaketa-aplikazioa sortzeko *Solr*ek dituen ongi eratutako bi konfigurazio fitxategiak irakurri eta moldatu behar direnez, pausoz pauso eta ulertuz garatu beharra dago.

Aurreko puntuetan ikus daitezke ezberdintasunak oso urriak direla eta, beraz, biekin garatu ahal izango litzatekeela inolako arazorik gabe. Baina kontuan izanik *Solr*ekin pausoz pauso eta pauso bakoitza ongi ulertuz garatzen joan behar dela, eta zuzendariek hasiera batetik gomendatua izan dela, *Apache Solr* izan da aukeratua.

[6] http://constellio.com
[7] http://www.searchblox.com
[8] https://aws.amazon.com/es/cloudsearch
[9] http://lucene.apache.org/solr
[10] https://www.elastic.co
[11] https://www.apache.org/licenses/LICENSE-2.0.html

3.3 Erabilitako tresna eta teknologiak

3.2 atalean teknologia posibleak aztertu ostean, atal honetan erabilitako tresna eta teknologiak sakonago aztertuko ditugu, bakoitzaren ezaugarri eta oinarrizko erabilerak azalduz. Teknologia posibleez gain, hitzak normalizatu eta lematizatzeko transduktoreak erabili beharko ditugunez, beste bi tresnen erabilerak ere aztertu beharko ditugu: *Phonetisaurus* eta *Foma*.

3.3.1 Phonetisaurus[12]

3.1.3 atalean esan dugun moduan, Etxeberria et al. (2016) lanean eraikitako normalizazio-sistemak ikasketa automatikoan du oinarria eta *Phonetisaurus* tresnaren bitartez garatua dago. Tresna hori grafema-fonema (G2P, Grapheme-to-Phoneme) bihurketa egiten duten sistemak eraikitzeko pentsatua dago. Hala ere, normalizazio-kasu horretan, grafemen arteko bihurketa egiteko erabili da: aldaerak sarrera gisa eta horiei dagozkien hitz estandarrak irteera gisa. Hori lortzeko, tresnari ikasteko emango zaizkion datuak aldatuko dira, ondoren aipatuko ditugun urratsak berdinak izango baitira.

*Phonetisaurus*ek WFST –Weighted Finite-State Technology– teknologia erabiltzen du, hau da, transduktore haztatuekin lan egiten du. Teknologia hori oso malgua da eta hizkuntzaren azterketa eta prozesamenduko hainbat aplikaziotan erabili da (hizketaren ezagutza, itzulpen automatikoa, etiketatzea eta abar). Tresna hau erabiltzeko, beharrezkoa da OpenFST[13] instalatua izatea.

Novak et al. (2015) lanean tresnari buruzko xehetasun asko ematen diren arren, erabiltzailearen ikuspuntutik tresnaren funtzionamendua lau urratsetan laburbiltzen da. Lehenengo hiruak ikasteko urratsak dira eta laugarrena ikasitakoa aplikatzeko urratsa da.

1. **Datuen prestaketa**. Ikasketa automatikoan oinarritzen den tresna denez, lehendabizi ikasteko datuak prestatu behar dira, hots, ikasteko adibideak eman behar dira formatu jakin batean.

2. **Datuen lerrokatzea**. Ikastegiko hiztegian hainbat bikote daude, zeinetan lehen osagaia grafemaz osatutako adierazpena den eta bigarrena, berriz, fonemaz osatutakoa (gure kasuan, azken hau ere grafemaz osatua izango da). Urrats honetan bi alde

[12] https://github.com/AdolfVonKleist/Phonetisaurus
[13] http://www.openfst.org/twiki/bin/view/FST/WebHome

horien arteko lerrokatzea egiten da eredu estatistiko jakin baten arabera (*multigram* eredua).

3. **Ereduaren entrenamendua**. Lerrokatzea eginda, hiztegiaren ikuspegia aldatzen da, lerrokatzez osatutako hiztegi berri bat baita, eta horrekin *n*-gram motako eredu bat entrena daiteke. Eredua entrenatzeko, hizkuntza modelatzeko hainbat aplikazio erabil daitezke (OpenGrm Library, MITLM, SRILM eta abar) eta bukaeran, *Phonetisaurus*ek WFST transduktore gisa adierazten du erabilitako aplikazioak sortu duen eredua.

4. **Deskodeketa**. WFST transduktoreari sarrera gisa hitz bat ematen badiogu, hau da, grafema-kate bat, hitz horri dagokion probabilitate handieneko fonema-katea lortzen da azken urrats honetan. Aukera dago probabilitate handieneko *m* kateak eskatzeko.

Etxeberria et al. (2016) lanean normalizazio-sistema eraikita dago jadanik, hau da, ikasteko urratsak bete dira eta normalizazioko proposamenak egiten dituen transduktorea sortu da. Proiektu honetan transduktore hori erabili nahi dugu deskodeketa egiteko, hots, testuetan ageri diren hitz ez-estandarrak normalizatzeko, eta urrats hori betetzeko beharrezkoa izango dugu *Phonetisaurus*.

Deskodeketa urratsean, sarrera berriei dagokien hipotesi onena lortzen da eta hori egiteaz arduratzen den programa `phonetisaurus-g2p` da, zeinak hainbat parametro behar dituen:

- **Model**. Eredua inplementatzen duen WFST transduktorea.
- **Input**. Deskodetu nahi diren hitzak dituen sarrera-fitxategia zehazteko.
- **Nbest**. Hipotesi hoberenen gehienezko kopurua.
- **Beam**. Bilaketa sakonera; defektuzko balioa 500 da, baina 5000 balioarekin emaitza hobeak lortzen ditu, nahiz eta denbora gehiago pasa.

Hori horrela, deskodeketa-prozesua gauzatzeko, ondorengo deia egin beharko litzateke:

```
$ phonetisaurus-g2p --model=fi.fst --input=test --isfile --words
--nbest=5 --beam=5000 >test.out
```

Dei honen bitartez, parametro gisa pasatako fitxategiko hitz ez-estandarren 5 hipotesi lortuko dira gehienez, baina emaitza horiek hitz estandarrak diren baieztatzea falta da. Horretarako, eta hitz estandarren lemak lortzeko, hurrengo azpiataleko *Foma* tresna erabili beharko da.

3.3.2 Foma[14]

Aurreko atalean esan dugun moduan, *Phonetisaurus*en bitartez lorturiko emaitzak hitz estandarrak diren baieztatzea eta horien lemak lortzea falta da. Baieztapen hori egiteko nahiz lemak lortu ahal izateko, jada sortuak izan diren bi transduktore erabiliko dira eta, horiek aplikatu ahal izateko, *Foma*ren *flookup* komandoa erabili beharko dugu.

Foma Mans Huldenek sortu eta mantendutako kode irekiko tresna multzo bat da, egoera finituko automatak eta transduktoreak erabiltzeko. Tresna horrek barne-hartzen ditu konpilatzailea, programazio-lengoaia eta C liburutegia. Hainbat aplikaziotan erabil daiteke, baina hizkuntzaren tratamendu automatikoan analisi morfologikoak egiteko erabili izan da gehien.

Gure kasuan, hiru transduktore erabiliko ditugu:

- `phonetis.fst` WFST transduktore haztatua, probabilitate handieneko grafema-kateak lortzeko.
- `xuxen_UTF_NOALD_NORARE_foma.fst` FST transduktorea, pasatako hitza estandarra den ala ez jakiteko.
- `xuxen_stem_manex.fst` FST transduktorea, hitz estandarren lemak lortzeko.

Hasteko, *Phonetisaurus*i deia egingo zaio, probabilitate handieneko grafema-kateak lortzeko, hau da, hitz ez-estandar bati dagokion estandarra izateko probabilitatea lortzeko:

```
phonetisaurus-g2p --model=phonetis.fst --input=not-standard-words \
   --isfile --words --nbest=5  --beam=5000 >test-out
```

Aukera ezberdinen probabilitateak izanik, hitza estandarra den jakiteko, hots, automatak onartzen duen ala ez jakiteko, ondorengo deia egin beharko dugu:

[14] http://foma.googlecode.com

```
cat test-out | flookup xuxen_UTF_NOALD_NORARE_foma.fst >standards
```

Hitz estandarrak jakinik, *flookup* bera, beste transduktorea eta Perl programa bat erabiliz, estandar horien lemak lortuko ditugu dei honen bitartez:

```
cat standards | perl stemmer/luzeena.pl >lemmas.txt
```

Azken finean, Perl programak, hitz bakoitzeko, ondorengo deia egiten dio *flookup*i:

```
flookup -i -x -b stemmer/xuxen_stem_manex.fst
```

Itzultzen dituen emaitzen artean, hainbat aukera dauden kasuetan, lemarik luzeena hartuko dugu lema gisa.

Azkenik, lorturiko emaitzak bateratuz, hitz ez-estandar, estandar eta lema hirukoteen zerrenda lortuko genuke, *Apache Solr* erabiliz, indexatu ahal izateko.

3.3.3 Apache Solr[15]

Apache Solr bilaketa-plataforma bat da, *Apache Lucene*n oinarritua eta bilaketa-zerbitzari gisa funtzionatzen duena. Bere ezaugarri nagusien artean, testuetan bilaketak egiteko aukera, bolumen handiko trafikoarentzat optimizatua egotea, emaitzak nabarmentzea eta bere eskalagarritasuna izango lirateke, bilaketa banatuak eta indizeen bikoiztea ahalbidetzen dutenak. Horrez gain, bilaketen emaitzak multzokatzeko aukera eskaintzen du eta, gure kasuan, emaitzak taldekatu nahi direnez, arreta berezia eskaini beharko diogu ezaugarri horri.

Lehenik eta behin, bilaketa banatuari loturiko hainbat termino azalduko ditugu, ondorengoa hobeto ulertzeko asmoz.

- **Klusterra (*Cluster*):** *Solr*en nodo multzoa da, unitate bakar gisa kudeatzen dena.
- **Nodoa (*Node*):** *Solr* exekutatzen den bitartean, JVM[16] instantzia bat da.

[15] http://lucene.apache.org/solr
[16] Java Virtual Machine, euskaraz, Javaren Makina Birtuala.

- **Partizioa (*Partition*):** Dokumentu bilduma osoaren azpimultzo bat da.
- **Informazio-zatia, fragmentua (*Shard*):** Partizio bat hainbat nodotan jaso daiteke eta nodo guzti hauen multzoak osatzen du *shard* bat.
- **Liderra (*Leader*):** *Shard* bakoitzak lider gisa identifikaturiko nodo bat du, zeinari partizio bati dagozkion dokumentuen idazketa guztiak bideratuko zaizkion.
- **Bilduma (*Collection*):** *Shard*/*Replica* bat edo gehiagok osatzen duen multzoa da.
- **Erreplika Faktorea (*Replication Factor*):** Klusterrak mantentzen duen dokumentuen gutxieneko kopia kopurua da.
- **Transakzio Erregistroa (*Transaction Log*):** Nodo bakoitzak idazketa operaziotarako erabiltzen duen erregistroa.

Ikus daiteke aukera asko eskaintzen dituela, baina, gure kasuan, bi nodo soilik erabiliko ditugu, bilduma bakarra eta bakoitza bere *shard*ekin. Hala ere, aplikazioaren eskalagarritasuna hobetzeko aukera asko eskaintzen ditu, beraz, etorkizun batean ongi etor daitezke.

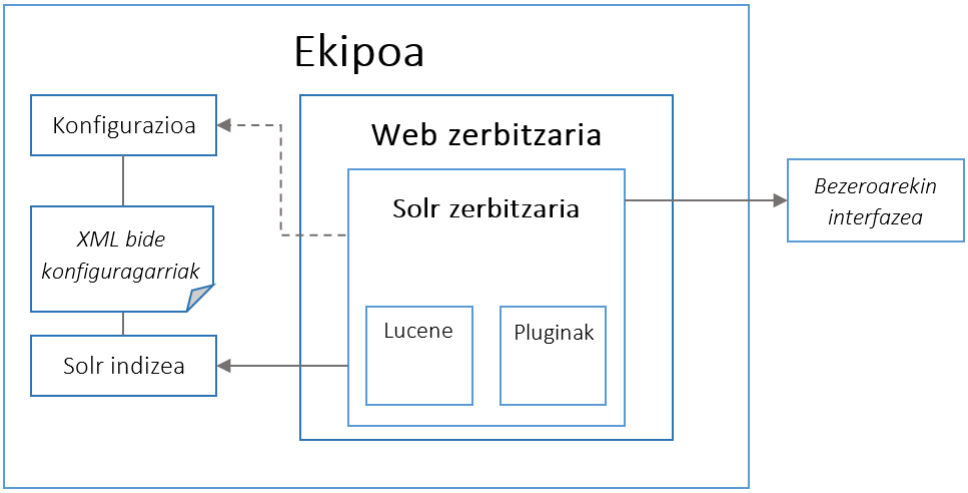
Ezaugarri nagusiak

Esan bezala, *Apache Solr* plataforma *Apache Lucene*ren liburutegian oinarrituta dago; *Lucene* bilaketa motorra eta indexatzailea da eta *Solr*ek ondorengo funtzionalitateak gehitzen dizkio:

- HTTP eta HTTPS protokoloen bidezko atzipena *Lucene*ra.
- Bilaketetan abiadura azkarragoa lortzeko *cache* memoria.
- Webaren administrazio interfazea.
- Eskalagarritasun altua.
- XML fitxategien bidez, datuen eta zerbitzariaren eskemaren konfigurazioa.
- Emaitzen multzokatzea.
- Zerbitzarien banaketa.

Plataforma hau Java lengoaia erabiliz idatzita dago eta Tomcat, Jetty eta beste Servlets batzuen bilaketa-zerbitzari independiente moduan erabiltzen da.

Antolaketari dagokionez, 3.4 irudian ikus daitekeen moduan, web-zerbitzariaren barnean dago *Solr* zerbitzaria eta, aurretik aipatu dugunez, azken honek *Lucene* bilaketa motorra eta indexatzailea, eta pluginak ditu. Era berean, *Lucene*k jasoko ditu *Solr* indizeak eta XML formatuko fitxategien bitartez, bide ezberdinak konfiguratuko dira azken konfigurazioa lortzeko, hots, *Solr* zerbitzariaren konfigurazioa lortzeko.



3.4 Irudia: *Apache Solr*en antolaketa.

Alderantzizko indizea

Solr gai da erantzun-denbora oso laburrak lortzeko, izan ere, bilatu nahi den hitza ez baita jatorrizko testuan bilatzen, indize batean baizik. Halaxe ikasi dugu "Datu-baseen Diseinua" irakasgaiko diseinu fisikoari dagokion atalean: "erregistroen atzipena azkartzen duten atzipen-egiturak dira indizeak".

Hori horrela, dokumentu edo testu bat indexatzen denean, hitz bakoitzeko gako bat sortuko da, gakoa eta agertzen den dokumentua bilduko dituena; indize mota honi alderantzizko indizea (*inverted index*) deritzo.

Alderantzizko indizeak hobeto ulertze aldera, adibide moduan, ondorengo bi esaldiak hartuko ditugu:

Urrutiko intxaurrak hamalau, gerturatu eta lau.
Hortzik ez eta intxaurrak ukan.

Esaldi horiek indexatuko balira, *Solr*ek lehenengo tokenizatu egingo luke eta banaturiko hitz bakoitzeko, baina errepikatu gabe, gako bat sortuko luke, zein esalditan agertzen den jasoaz. Gako horiekin hitzen lista bat sortuko litzateke, 3.1 taulan ikus den moduan.

Hitza	Esaldi_1	Esaldi_2
urrutiko	✓	
intxaurrak	✓	✓
lau	✓	
gerturatu	✓	
eta	✓	✓
hamalau	✓	
hortzik		✓
ez		✓
ukan		✓

3.1 Taula: Hitzen agerpenak esaldietan.

Behin hitzak indizeetan gorderik, hitz baten bilaketa egiteko orduan indizeetan bilatuko litzateke, eta ez jatorrizko esaldietan. Adibidez, aurreko esaldietan “intxaurrak"eta “hortzik"hitzak bilatuz gero, 3.2 taulan ikus daiteke hitz bakoitzaren agerpena esaldietan.

Hitza	Esaldi_1	Esaldi_2
intxaurrak	✓	✓
hortzik		✓

3.2 Taula: “Intxaurrak"eta “hortzik"hitzen agerpenak esaldietan.

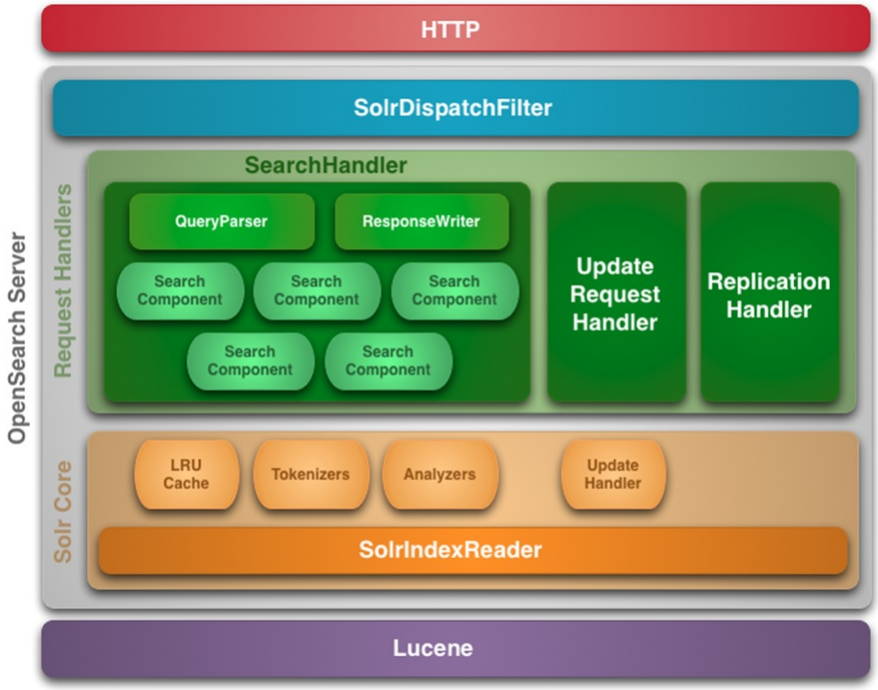
Kontuan izan beharko da, baita ere, letra larriz edo xehez hasten diren hitz berdinak banatzea nahi dugun edo, “Intxaurrak” hitza bilatuz gero, “intxaurrak” hitzaren emaitzak ere agertzea nahi den.

Adibideak alde batera utzita, esaldiak indexatu beharrean, *Solr*ek sintaxi edo eskema jakin bat betetzen duten dokumentuak indexatzen ditu, lehen bezala, dokumentuko hitz bakoitza tokenizatuz eta hitz bakoitza zein dokumentutan agertzen den jasoaz.

Arkitektura

3.5 irudian ikus daiteke zein den *Apache Solr*en arkitektura. Arkitektura horren barne izango dira ondorengo osagaiak:

- **RequestHandlers**. Eskatutako kontsulta prozesatzen dutenak; kontsulta hori prozesatzeko, beste osagai batzuk dituzte:
  - **Query Component**. Bilaketarako osagaia.
  - **Grouping Component**. Emaitzak multzokatzeko.
  - **Highlight Component**. Emaitzak nabarmentzeko.
  - **Debug Component**. Emaitzen arazketa egiteko.
- **SearchComponents**. Emaitzari eskaeraren datuak gehitzen dizkioten osagaiak.
- **ResponseWriters**. Emaitza dagokion formatura eraldatzen dutenak.
- **SolrCore**. Dokumentu eta konfigurazio multzoa.
- **UpdateHandler**. Dokumentuen eguneratze logika.
- **Cache**. *Caching* estrategia.

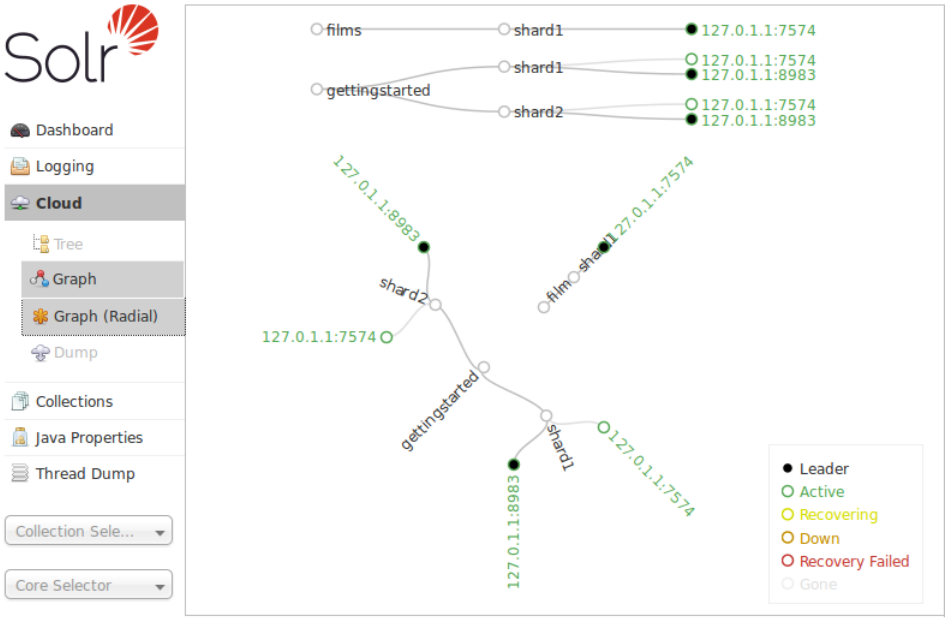


3.5 Irudia: *Apache Solr*en arkitektura. Iturria: Open&Search.

Funtzionamendua

Funtzionamenduari dagokionez, behin *Apache Solr* instalaturik, beharrezko bildumak sortu beharko dira, horiek konfiguratu, datuak kargatu eta eskaerei erantzuna emateko prest egongo litzateke.

Adibide moduan, bi bilduma sortuko ditugu: “gettingstarted"eta “films"izenekoak. Lehenengo bildumak bi *shard* eta bi *replica* izango ditu eta, bigarrenak, *shard*/*replica* bakarra. Hori horrela, 3.6 irudian ikus dezakegu sorturikoaren errepresentazio grafikoa.



3.6 Irudia: *Apache Solr*eko adibide baten errepresentazio grafikoa. Iturria: Confluence.

Bildumak konfiguratzeko bi modu daude:

- **Dinamikoa**. Ez da eskemarik zehazten eta, beraz, eremuak dinamikoki sortzen dira.
- **Eskema bidezkoa**. *schema.xml* fitxategian zehazten dira indexatuko diren fitxategiek izan behar duten gakoa, eremuak eta hauen ezaugarriak:

“Informazioaren Kudeaketa Aurreratua” irakasgaian ikusi genuen moduan, eskema bat definitzeak denbora gehiago eskatzen duen arren, datuak formatu jakin batean jasota daudela ziurtatzen da, hots, modu antolatu batean jasota daudela eta, beraz, garatze- eta mantzentze-kostuak murrizten dira. Aldiz, modu dinamikoan, eskemarik ez dagoenez, jasotzen diren datuak ez daude kontrolatuta, zaborra pilatzea eta informazioaren egitura eza

eragingo lukeena. Eskema bat erabiltzearen alde onak ikusita, eskema bidezkoa erabiltzea erabaki dugu, 3.7 irudiko modukoa, beraz, hurrengo kapituluan eskemaren diseinua egin beharko dugu.

```
<uniqueKey>film</uniqueKey>
<field name="film" type="string" indexed="true" required="true"/>
<field name="price" type="float" default="0.0" indexed="true"/>
{...}
```

3.7 Irudia: Eskema baten eremuak zehazten dituen adibide bat.

Datuak indexatzeko garaian, eskema bidez konfiguratua bada, XML fitxategiak sintaxi bat jarraitu behar du: `<add>` etiketa barnean bilduko dira dokumentuak, `<doc>` elementuak izango direnak, eta, era berean, eskeman definitutako eremuak betetzen dituzten balioekin. 3.8 irudian ikus daiteke adibide bat.

```
<add>
  <doc>
    <field name="film">Filmaren izena</field>
    <field name="price">2.5</field>
    {...}
  </doc>
  {...}
</add>
```

3.8 Irudia: Datuak indexatzeko eskema bat jarraitzen duen XML fitxategiaren adibidea.

Dokumentu hauek indexatzeko, API bitartez egin daiteke, *Solr*ek duen *script* baten bitartez edo liburutegien bitartez. Halaber, datuak lortzeko ere API bitartez edo liburutegi baten bitartez egin daiteke. Aurrerago, garapena egiteko garaian zehaztuko dugu zein metodo erabiliko dugun.

3.3.4 Bootstrap[17]

3.2.2 atalean ikusi dugun moduan, *Bootstrap framework*a erabiltzea erabaki dugu, egunean egoteaz gain, batez ere, nabigatzaileekin eta mugikorrekin duen bateragarritasunagatik.

[17] http://getbootstrap.com

Bootstrap Twitterrek 2011. urtean garatu zuen kode irekiko *framework* bat da, *Twitter Bootstrap* izenez ere ezagutua, web-aplikazioen diseinurako sortua. Tipografia, botoiak, nabigazio menuak eta beste hainbat elementudun diseinuzko ereduak ditu, HTML eta CSS estilo-orrietan oinarritua, JavaScript bidezko hautazko gehigarriekin.

Hasiera batean antzematen ez den arren, asko eta asko dira Bootstrap erabiliz sortutako webguneak. Gainera, oso ongi dabiltza gailu ezberdinetan, adibidez, mahaigaineko ordenagailuetan, eramangarrietan, tabletetan edo mugikorretan. Hau horrela, diseinatzaileek diseinu esklusiboak sor ditzakete tamaina ezberdinetarako pantailetarako, gailua zein den kontuan izan gabe.

Egun baliabide eta proiektu asko daude *framework* honen inguruan eta, hain ezaguna izanik, etengabe aurrerapen eta hobekuntza ugari dituzten bertsio berriak sortzen dabiltza.

Bootstrap erabiltzea erabaki dugunez, azter ditzagun erabiltzearen abantailak:

- **Mobile First**. *Framework* guztia *mobile first* filosofia jarraituz garatuta dago, hau da, web moldagarrien *–responsive web–* garapenerako gomendatua. Betekizunetan eskatzen zen ezaugarria izanik, kontuan izan beharreko ezaugarria.

- **Deskarga pertsonalizagarria**. Ez dago *framework* osoa jaitsi beharrik funtziona dezan, are gehiago, oinarrizko funtzionalitateak exekutatzeko ez du JavaScript ere behar.

- **LESSerako optimizatua**. LESSen –CSS preprozesagailu potente bat– erabilerarako optimizatua.

- ***Grid* moldagarria**. *Bootstrap*en edukiak antolatzeko modua %100 moldagarria da eta bista guztien diseinua pentsatzeko aukera eskaintzen du, erabiltzailearen esperientzia hobetzeko.

- **Nabigatzaileen arteko bateragarritasuna**. Egungo nabigatzaile guztiekin bateragarria izateko garatua dago.

- ***Font icons*en erabilera**. CSS bidezko ikonoak alde batera utzi eta *font icons* ikonoak erabiltzen dira, webaren errendimendua optimizatuz.

- **Dokumentazio bikaina**. Online duen dokumentazioa oso kalitate onekoa da, adibide oso sinple eta argiekin, eta kode berrerabilgarriekin.

Eta baita desabantailak ere:

- **Ez dago SASSentzat prestatua**. LESS preprozesagailua beharrean SASS CSS preprozesagailua erabiliz gero, hobe *Foundation* erabiltzea, ez baitago honentzat prestatua.
- **Elementu batentzat klase asko**. Azken xehetasuneraino pertsonalizatzeko aukera eskaintzen duen arren, kasu batzuetan elementu bati ezarri beharreko klase kopurua gehiegizkoa izan liteke, beraz, oso ongi jakin behar da klase hauek erabiltzen.
- **JavaScript eta jQuery beharrezkoa funtzionalitate batzuentzat**. Beharrezkoa izango da JavaScript eta jQuery liburutegiak deskargatzea, adibidez, nabigatzebarra edo irudien *slider*a erabiltzeko.

Erabileraren abantailak eta desabantailak ikusita, argi dugu aukera oso ona izango litzatekeela gure webgunea garatzeko, ez baitugu SASS erabiliko eta, beste funtzionalitate batzuk garatzeko ere JS eta jQuery liburutegiak erabiliko baititugu.

Oinarrizko erabilerari dagokionez, hiru aukera eskaintzen ditu *Bootstrap*ek:

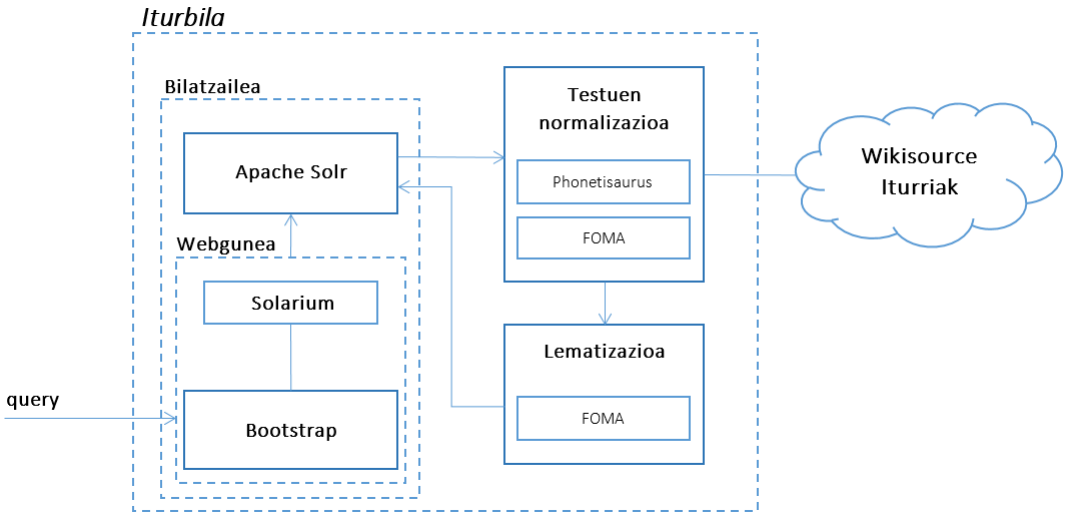
- Jada konpilatutako Javascript eta CSS kodea jaistea. Modurik errazena.
- Iturburu-kodea jaistea, jatorrizko LESS eta Javascript fitxategiak dituena. Alde txarra, LESS konpilatzailea eta konfigurazio-lana eskatzen dituela.
- Iturburu-kodea SASS formatuan jaistea. SASS erabiliko ez dugunez, aukera baztertua.

Ikusirik azken aukera baztertu dugula eta, hasiera batean behintzat, LESS konpilatzailerik ez dugula, lehenengo aukeraz baliatuko gara, hots, modurik errazenaz. Hortaz, konpilatutako kodea jaistearekin nahikoa izango dugu.

4. KAPITULUA

Analisia eta Diseinua

Aurreko 3.3 atalean zenbait elementu aztertu ditugu. Elementu horiekin aplikazio bat garatu nahi dugu, zeinetan administratzaile batek Wikisource eta Iturriak webgunetako testuak lortu ondoren, bertako hitz ez-estandarrak normalizatuko dituen *Phonetisaurus* eta *Foma* erabiliz, *Foma* bitartez lemak lortuko dituen eta lorturiko hitz estandar nahiz lemak indizeetan jasoko dituen, *Apache Solr*en bidez. Horrez gain, erabiltzaileak bilaketak egiteko aukera izango du *Bootstrap* erabiliz sortutako webgunetik, *Apache Solr*eko indizeen bitartez, *Solarium* liburutegia erabiliz. Elementuen antolaketa hobeto ulertze aldera, ikus 4.1 irudia, sistemaren antolaketa orokorra irudikatzen duena.



4.1 Irudia: Sistemaren antolaketa orokorra.

Behin sistemaren eskema orokorra zein izango den finkaturik, atal honetan produktuaren nondik norakoaren buruzko azalpen sakonagoa egingo da, sistema osoaren funtzionamendua hobeto ulertzeko asmoz. Horretarako, erabilpen-kasuen eskema orokor bat azalduko da 4.1 atalean eta, ondoren, erabilpen-kasu bakoitzaren xehetasunak emango dira azpiatal bakoitzean, bete beharreko aurrebaldintzak eta fluxu normala nahiz ezohiko fluxua azalduz.

Ondorengo 4.2 azpiatalean, *Apache Solr*en diseinuan jarriko da arreta. Bertan, nodoen sorrera eta zehazturiko eskemen diseinuak azalduko dira.

Azkenik, 4.3 azpiatalean, web-aplikazioaren interfaze grafikoaren diseinuaren hurbilpen bat egingo da, 2.2.1 atalean zehaztu diren betekizunak betetzen dituena.

4.1 Erabilpen-kasuen analisia eta diseinua

Erabilpen-kasuak zehazterako garaian, bi erabiltzaile-mota egongo direla erabaki dugu: administratzailea eta erabiltzaile arrunta. Era berean, erabiltzaile bakoitzak erabilpen-kasu nagusi bat izango du.

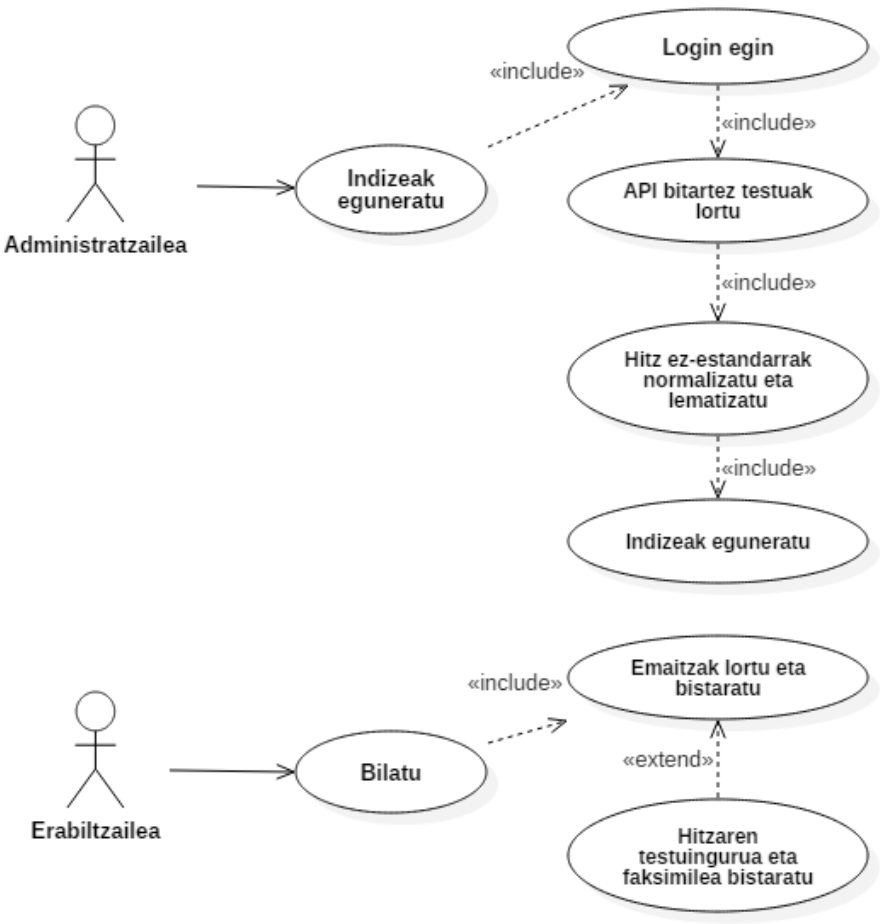
4.1.1 azpiatalean ikusiko dugun moduan, administratzailearen kasuan, indizeak eguneratu ahal izango ditu. Horretarako, login egin beharko du, API bitartez testuak lortu, hitz ez-estandarrak normalizatu eta lematizatu eta, azkenik, indizeak *Solr*en eguneratu.

Aldiz, 4.1.2 azpiatalean azalduko dugunez, erabiltzaile arruntak bilaketak egiteko aukera izango du. Behin emaitzak lortu eta bistaratuta, bilaketa-emaitza baten testuingurua eta faksimilea ere bistaratu ahal izango ditu.

4.2 irudiak erakusten du orain arte azaldutako erabilpen kasuen eskema orokorra.

4.1.1 Indizeak eguneratu - Administratzailea

Hasteko, administratzaileak, indizeak eguneratu ahal izateko, identifikatu egin beharko du eta, sarrera ongi balioztatuz gero, API bitartez testuak lortuko ditu. Ondoren, testu horietako hitz ez-estandarrak normalizatu eta horien lemak lortuko dira eta, azkenik, testu eta hitz horien indizeak eguneratuko dira *Solr*en. Horiek izango dira, hain zuzen ere, ondorengo azpiatalatan deskribatuko diren erabilpen-kasuak.



4.2 Irudia: Erabilpen-kasuen diagrama.

Login egin

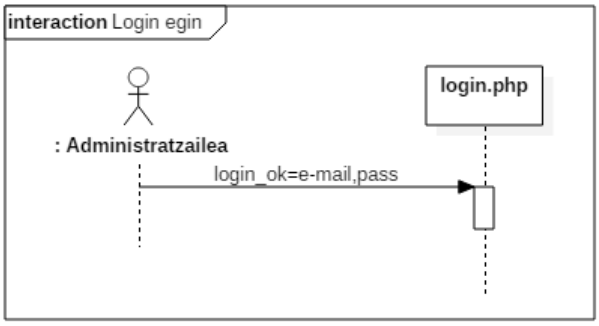
Administratzaileak login egiteko aukera izango du, indizeak eguneratu ahal izateko.

Deskribapena:

- Aktorea: Administratzailea.
- Aurrebaldintzak: Administratzailearen e-posta helbidea eta pasahitza konfigurazio fitxategi batean zehazturik egotea.
- Fluxu normala:
  - Administratzaileak aurretik zehazturiko e-posta helbidea eta dagokion pasahitza sartu beharko ditu.
  - E-posta eta pasahitza zuzenak badira, API bitartez testuak lortzera pasatuko da.

- Ezohiko fluxua: Administratzaileak okerreko e-posta edota pasahitza sartuz gero, errore mezua agertuko da eta berriro saiatzeko aukera izango du.

4.3 irudian dago "Login egin" EKren sekuentzia-diagrama.



4.3 Irudia: "Login egin" EKren sekuentzia-diagrama.

API bitartez testuak lortu

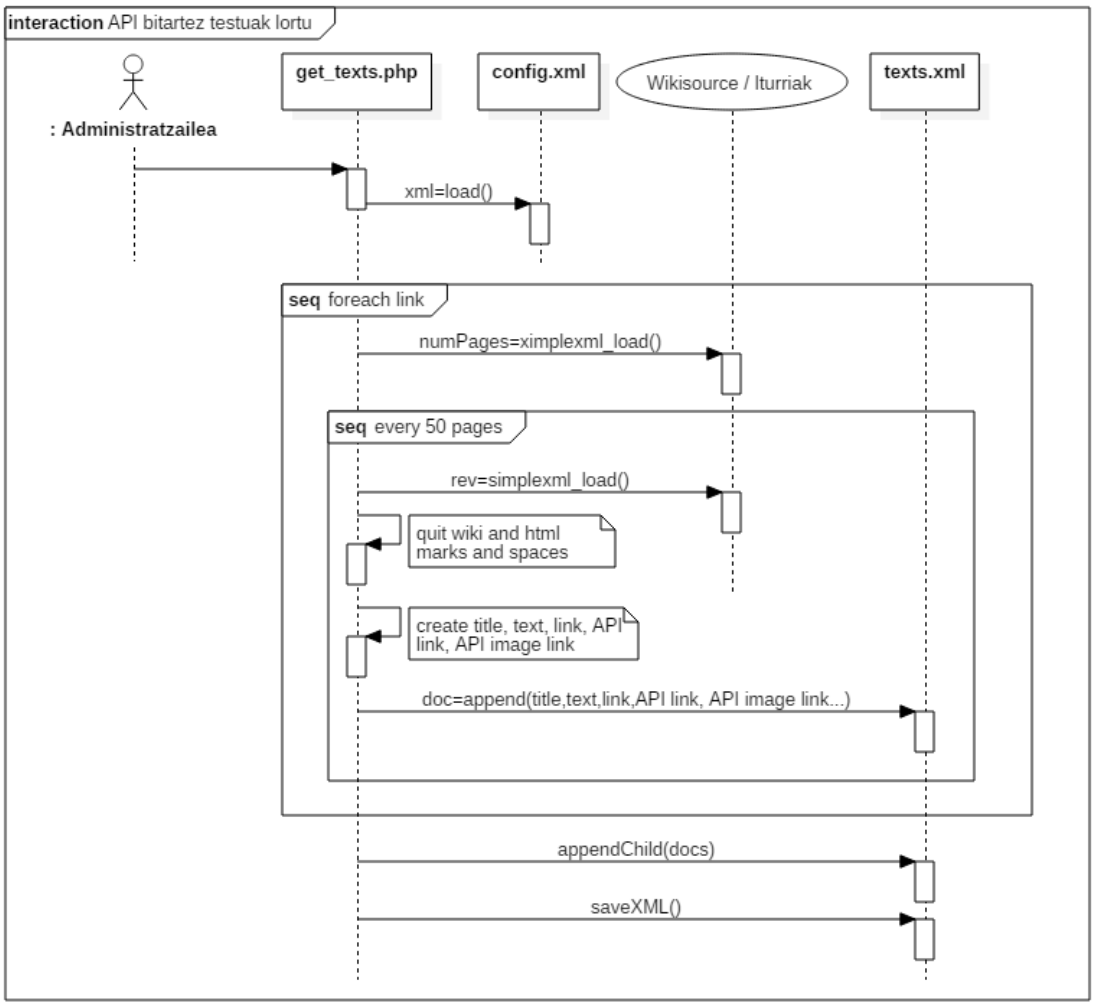
Wikisource eta Iturriak webgunetako APIak erabiliz, testuen izenburu eta estekak biltzen dituen XML fitxategiko testuak lortu.

Deskribapena:

- Aktorea: Administratzailea.
- Aurrebaldintzak:
  - Login egina egotea.
  - XML fitxategian gutxienez izenburu eta esteka bat egotea.
  - Estekak zuzenak eta Wikisource eta Iturriak gunetakoak izatea.
- Fluxu normala:
  - Obra bakoitzeko, hau da, fitxategiko esteka bakoitzeko, ondorengoa egin.
    - Bakoitzari dagokion APIaren esteka lortu eta deia egin, orrialde kopurua lortzeko.
    - Behin orrialde kopurua jakinik, esteka luzeegia ez izateko, 50 orrialderi dagozkien deiak egin eta emaitzak garbitu, izan ditzakeen zuriune, Wiki eta HTML markak kenduz.
    - Titulua, testu garbia, jatorrizko webgunera esteka, APIaren esteka eta irudiaren estekak *texts.xml* izeneko XML dokumentu batean gordetzen joan, testu guztiak jaso arte.

  - Testu guztiak jasorik daudenean, testuko hitz ez-estandarrak normalizatzera eta lematizatzera pasatu.
- Ezohiko fluxua: Erroreren bat gertatuz gero, errore mezua agertuko da eta exekuzioa geldituko da.

4.4 irudian ikus daiteke “API bitartez testuak lortu” EKren sekuentzia-diagrama.



4.4 Irudia: “API bitartez testuak lortu” EKren sekuentzia-diagrama.

Hitz ez-estandarrak normalizatu eta lematizatu

Behin testu guztiak izanik, testuko hitz ez-estandar bakoitzari dagokion hitz estandarra eta lema lortu.

Deskribapena:

- Aktorea: Administratzailea.
- Aurrebaldintzak:
  - Login egina egotea.
  - Testu guztiak biltzen dituen XML fitxategia ongi sortua egotea.
- Fluxu normala:
  - Testu guztiak biltzen dituen XML fitxategiko hitz guztiak tokenizatu.
  - Aurretik indexatutako hitzak, errepikatutako hitzak eta letra ez den karakteren bat duten hitzak kendu, eta gutxienez bi letra dituzten hitzak soilik hartu.
  - `deia.sh` izeneko *script* baten bitartez, lortu hitz ez-estandar bakoitzari dagokion hitz estandarra.
  - Behin hitz estandarrez osatutako fitxategia izanik, luzeena.pl Perl fitxategiari pasa hitzak hitz estandarren lemak lortzeko.
  - Hitz ez-estandarra, hitz estandarra eta lema hirukoteak, *indexes.xml* izeneko XML fitxategi batean jaso.
- Ezohiko fluxua:
  - Hitz ez-estandarrei dagokien hitz estandarrak ez badira lortzen, hauek ez dira kontuan hartuko.
  - Hitz estandar batek hainbat lema itzultzen dituen kasuan, karaktere gehien dituena aukeratuko da.
  - Erroreren bat gertatuz gero, errore mezua agertuko da eta exekuzioa geldituko da.

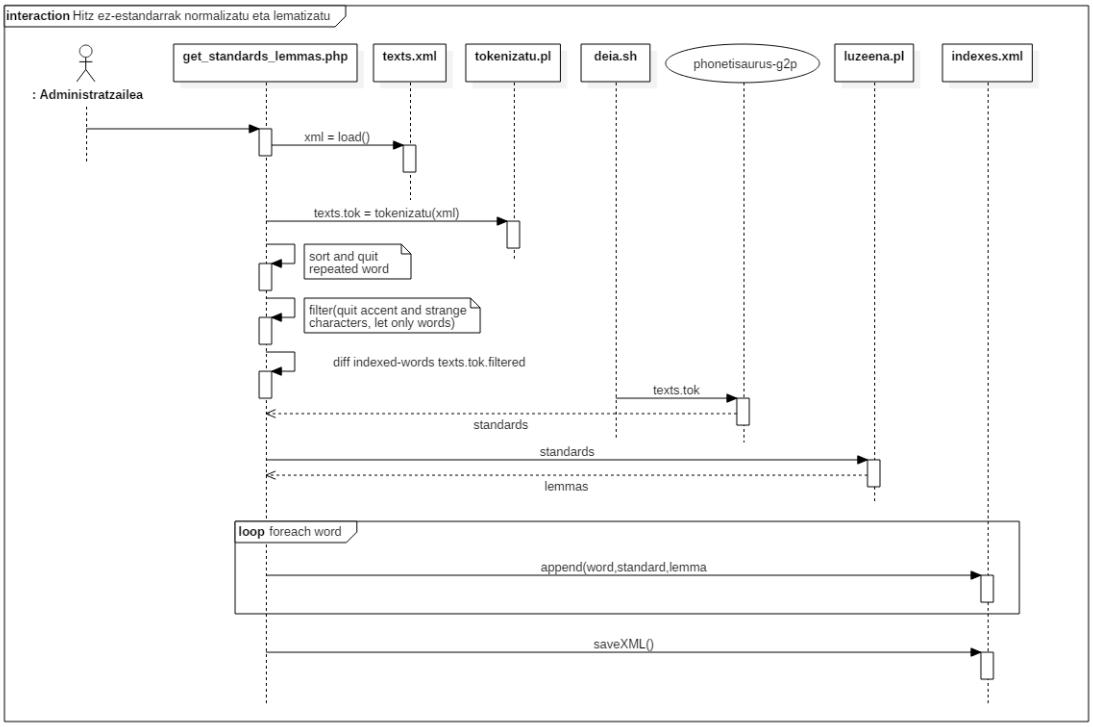
"Hitz ez-estandarrak normalizatu eta lematizatu" EKren sekuentzia-diagrama 4.5 irudian ikus daiteke.

Indizeak eguneratu

API bitartez lortutako testuak eta testuko hitz bakoitzari dagokion hitz estandarrak eta lemak Solr-en jaso.

Deskribapena:

- Aktorea: Administratzailea.
- Aurrebaldintzak:
  - Login egina egotea.
  - Testu guztiak biltzen dituen XML fitxategia ongi sortua egotea.

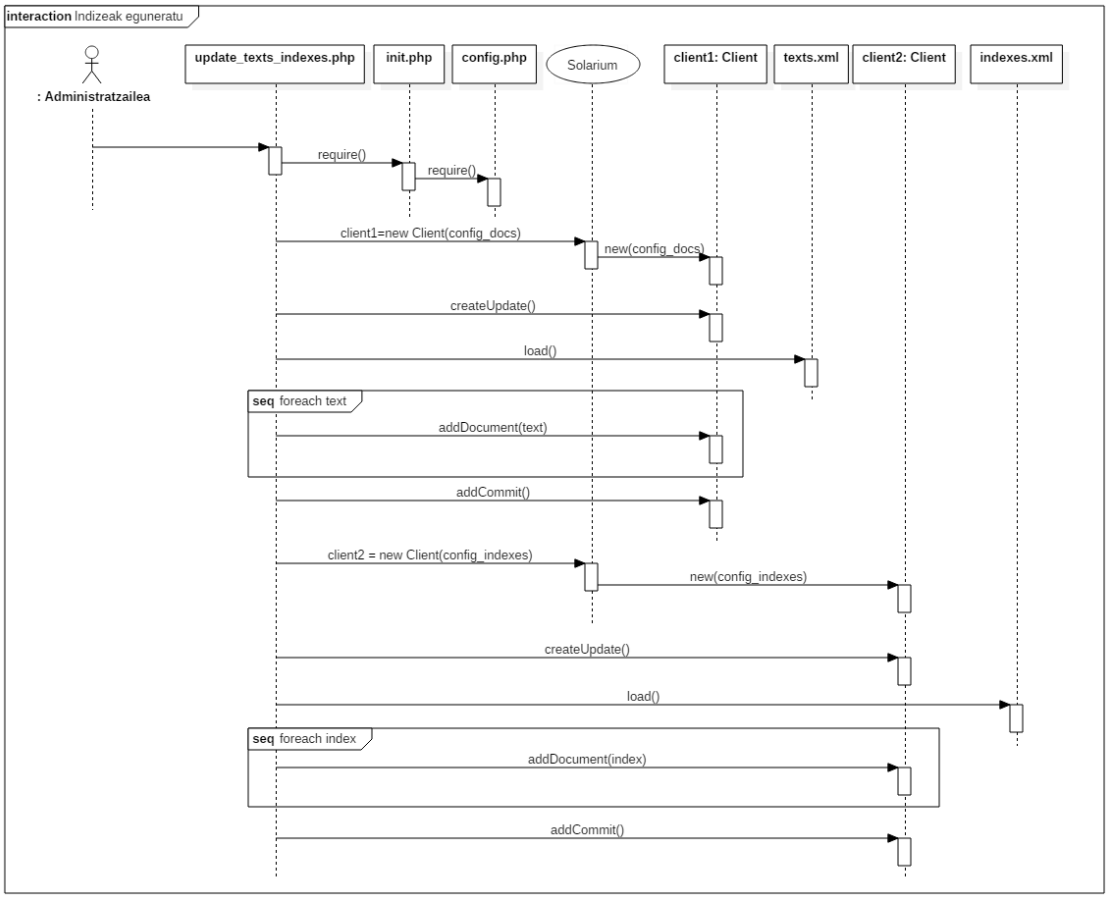


4.5 Irudia: "Ez-estandarrak normalizatu eta lematizatu" EKren sekuentzia-diagrama.

  - Hitz ez-estandar, estandar eta lema hirukoteak dituen XMl fitxategia ongi sortua egotea.

- Fluxu normala:
  - *Solarium* bitartez Solr-ekin konexioa ireki.
  - *texts.xml* fitxategiko parametroak –id, titulua, testua, esteka, APIaren esteka eta irudiaren esteka– *Solarium*en objektu batera pasa eta gordetzeko agindua eman (*commit*).
  - *indexes.xml* fitxategiko hitz hirukoteak –ez-estandarra, estandarra eta lema– *Solarium*en objektu batera pasa eta konexioa itxi gordetzeko agindua (*commit*) eman ondoren.
- Ezohiko fluxua:
  - Testuekin arazoren bat izanez gero, errore mezua agertu eta indizeak eguneratzera pasatu.
  - Indizeekin arazoren bat izanez gero, errore mezua agertu eta exekuzioa amaitu.

4.6 irudian ikus daiteke "Indizeak eguneratu" EKren sekuentzia-diagrama.



4.6 Irudia: "Indizeak eguneratu" EKren sekuentzia-diagrama.

4.1.2 Bilaketa - Erabiltzailea

Azken finean, garatu nahi den proiektuaren funtsa erabiltzaileek bilaketak gauzatu ahal izatea da. Horretarako, bilaketaren erabilpen-kasua bi ataletan banatzen da: bilaketa bera, hots, emaitzak lortu eta bistaratzea, eta, behin emaitzak lorturik, emaitza baten testuingurua eta faksimilea bistaratzea.

Emaitzak lortu eta bistaratu

Erabiltzaileak bilaketa gauzatu nahi duen hitza eta hitz-mota aukeratu ostean, hitz horri dagozkion jatorrizko hitz guztien bistaratu dizkio, modu antolatu batean.

Deskribapena:

- Aktorea: Erabiltzailea.

- Aurrebaldintzak: Ez daude.
- Fluxu normala:
  - Erabiltzaileak testu-kutxan testuetan bilatu nahi duen hitza idatzi eta jatorrizko hitza, estandarra edo lema den aukeratu.
  - *Solarium* bitartez *Solr*ekin konexioa ireki.
  - Estandarra edo lema aukeratzen den kasuetan, hitz horri dagozkion jatorrizko hitz guztiak lortu *Solr*i deia eginez; jatorrizkoen kasuan, urrats hau ez egin.
  - Behin jatorrizko hitzak lorturik, hitz horiek guztiak testuetan bilatu *Solr*en bitartez.
  - Lorturiko emaitzak bistaratu: testuka sailkatuta, orrien bitartez antolatuta eta bilaketako hitzari dagozkion jatorrizko hitz guztiak markatuta.
- Ezohiko fluxua:
  - Testu-kutxa hutsik utziz gero edo hizkiak ez diren karaktereak sartuz gero, errore mezua agertu eta berriro saiatzeko aukera eman.
  - Aukeratutako hitz eta hitz-motari dagokion emaitzarik ez badago, emaitzarik ez dagoela ohartarazi eta berriro saiatzeko aukera eman.
  - Beste edozein errore gertatuz gero, exekuzioa gelditu eta hasierako orrira eraman.

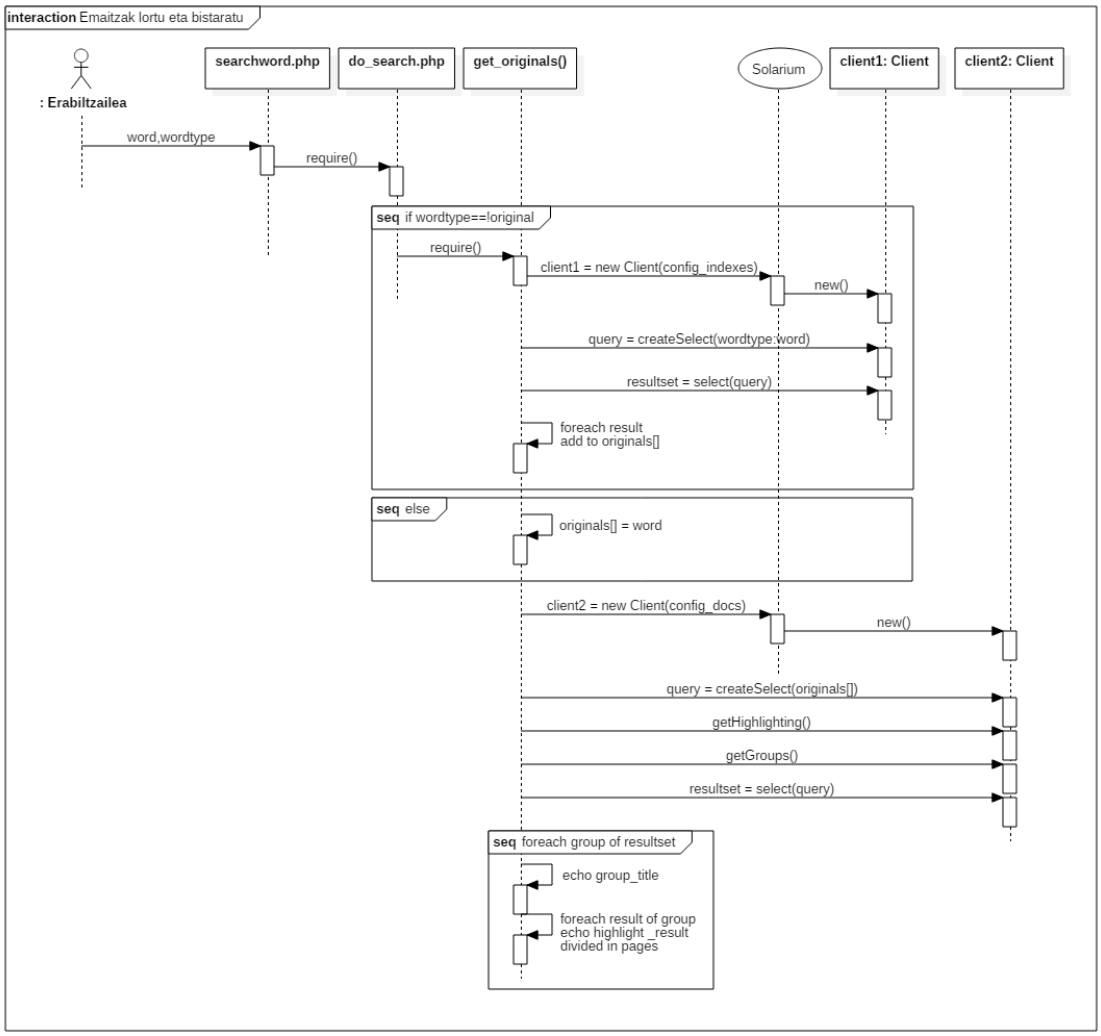
"Emaitzak lortu eta bistaratu" EKren sekuentzia-diagrama 4.7 irudian ikus daiteke.

Hitzaren testuingurua eta faksimilea bistaratu

Behin emaitzak izanik, aukeratutako bilaketako hitzari dagokion orrialdea bistaratuko dio erabiltzaileari: testu moduan, jatorrizko hitz guztiak markatuta, eta dagokion faksimilearekin batera.

Deskribapena:

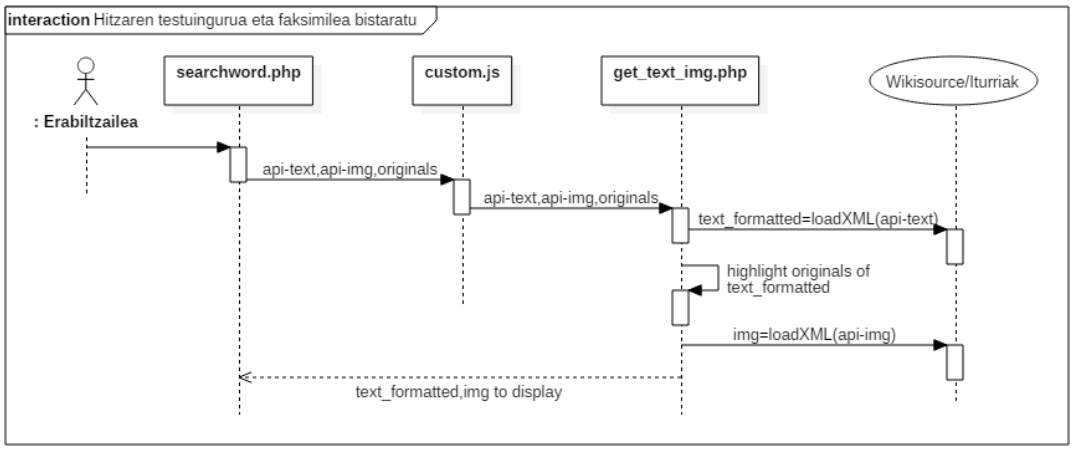
- Aktorea: Erabiltzailea.
- Aurrebaldintzak: Gutxienez bilaketa emaitza bat egotea.
- Fluxu normala:
  - Erabiltzaileak emaitza baten gainean klikatu.
  - API bitartez formatudun testua eta faksimilea lortu.
  - Gaineko leiho batean, lortutako testua, faksimilea eta testuaren jatorrirako esteka bistaratu.
  - "x" botoiari sakatuz gero, leihoa ez bistaratu, aurretik lortutako emaitzak bistaratuz.



4.7 Irudia: "Emaitzak lortu eta bistaratu" EKren sekuentzia-diagrama.

- Ezohiko fluxua:
  - Emaitzak ez badu faksimilerik, irudirik ez duen irudia agertu.
  - Konexio arazorik izanez gero, formaturik gabeko testu soila eta irudirik ez duen irudia agertu.
  - Beste edozein errore gertatuz gero, exekuzioa geldituko da eta aurretik lortutako emaitzak bistaratu.

4.8 irudian ikus daiteke "Hitzaren testuingurua" EKren sekuentzia-diagrama.



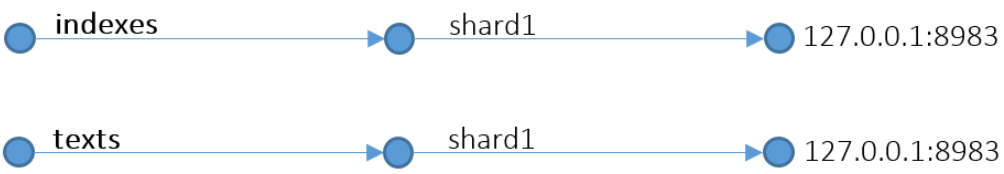
4.8 Irudia: "Hitzaren testuingurua eta faksimilea bistaratu" EKren sekuentzia-diagrama.

4.2 Apache Solr

3.3.3 atalean *Apache Solr*en inguruko xehetasun orokorrak azaldu ditugunez, azpiatal honetan proiektua aurrera eramateko diseinuan jarriko dugu arreta.

Behin *Apache Solr* instalatuta izanik, zein eta zenbat nodo sortu behar diren erabaki behar dugu. Kasu honetan, 4.9 irudian ikus daitekeenez, bi nodo sortuko dira:

- **Testuen nodoa**. Testuak jasoko dituen nodoa izango da, API eta estekekin batera.
- **Indizeen/hitzen nodoa**. Hitz bakoitzaren jatorrizko hitza, dagokion hitz estandarra eta lema jasotzeaz arduratuko den nodoa izango da.



4.9 Irudia: Sorturiko nodoen errepresentazio grafikoa.

Ondorengo urratsa nodo bakoitzaren konfigurazioa zehaztea izango litzateke. Horretarako, datuak modu antolatu batean jasotzeko asmoz eskemak erabiliko ditugula erabaki dugunez, eta bi nodo ditugunez, eskemen diseinuak zehaztu beharko ditugu.

4.10 irudian testuen eskema ageri da, indexatu nahi den testu bakoitzak jarraitu behar dituen eremuekin:

- **Id**. Dokumentu bakoitza identifikatzeko beharrezkoa, gako errepikaezina.
- **Title**. Testuaren izenburua.
- **Text**. Testua bera, formaturik gabe.
- **Link**. Testuaren jatorrira esteka.
- **ApiLink**. API esteka, testuingurua bistaratzean, testuaren formatua lortzeko.
- **ApiImg**. Irudia lortzeko API esteka, faksimilea bistaratzeko beharrezkoa.

```
<schema name="texts-schema" version="1.5">
   <uniqueKey>id</uniqueKey>
   <field name="id" type="string" indexed="true" stored="true"
      required="true" multiValued="false"/>
   <field name="title" type="string" indexed="true" stored="true"
      required="true" />
   <field name="text" type="text_general" indexed="true" stored="true"
      required="true" />
   <field name="link" type="string" indexed="true" stored="true"
      required="true" />
   <field name="apilink" type="string" indexed="true" stored="true"
      required="true" />
   <field name="apiimg" type="string" indexed="true" stored="true" />
</schema>
```

4.10 Irudia: Testuen indizeak definitzen dituen eskema.

4.11 irudian indizeen edo hitzen eskema ageri da, ondorengo eremuekin:

- **Original**. Jatorrizko hitza; errepikaezina izango denez, gakoa.
- **Standard**. Jatorrizko hitzari dagokion hitz estandarra.
- **Lemma**. Hitz estandarrari dagokion lema.

```
<schema name="indexes-schema" version="1.5">
   <uniqueKey>original</uniqueKey>
   <field name="original" type="string" indexed="true" stored="true"
      required="true" multiValued="false" />
   <field name="standard" type="text_general" indexed="true" stored="true"
      required="true" />
   <field name="lemma" type="text_general" indexed="true" stored="true"
      required="true" />
</schema>
```

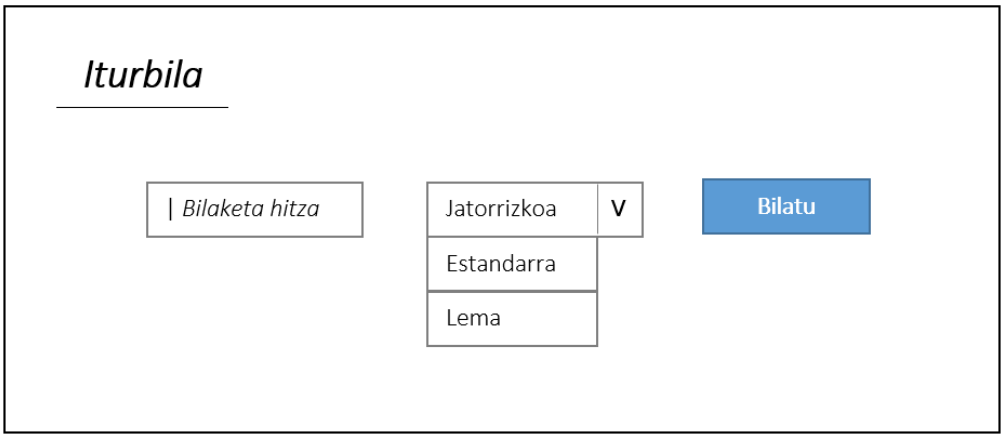
4.11 Irudia: Hitzen indizeak definitzen dituen eskema.

4.3 Web-aplikazioaren interfazea

2.2.1 atalean interfaze grafikoaren betekizunak zehaztu dira eta, beraz, horiek ahalik eta hobekien betetzen dituen interfaze bat diseinatuko dugu.

Bilaketa gauzatzeko orrian, 4.12 irudian ikus daitezkeen hiru elementu nagusi egongo dira:

- Bilaketa hitza sartzeko testu-kutxa bat.
- Zabaltzen den menu bat, bilaketa-hitza jatorrizkoa, estandarra edo lema den zehazteko.
- Bilaketa botoi bat, bilatzeko agindua emateko.

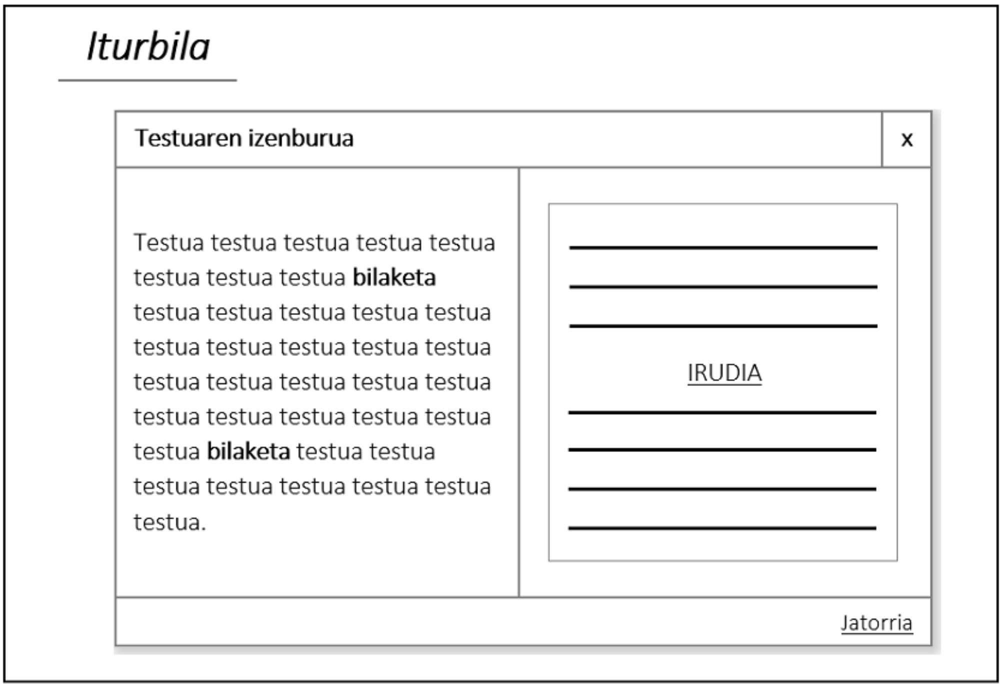


4.12 Irudia: Hasierako pantailaren diseinua.

Behin bilaketa agindua emanik, emaitzak bistaratzeko unea da; horretarako, 4.13 irudiko eredua jarraituko da. Hau da, bilaketa hitzak nabarmenduko dira, emaitzak testuka sailkatuko dira eta testu bakoitzeko emaitzak orrika banatuko dira. Konfigurazio fitxategian zehaztutakoaren arabera, orri bakoitzean agertzen diren emaitza kopurua aldagarria izango da. Testuak ezkutatzeko aukera eta testu bateko emaitza guztiak bistaratzeko aukera ere eskainiko dira. Aukera horiek guztiak eskaintzen dira, erabiltzaileak bistaratu nahi duena ahalik eta errazen egin dezan, bilaketa erosoago bat ahalbidetuz.

4.13 Irudia: Emaitzak bistaratzeko pantailaren diseinua.

Emaitza baten gainean klikatuz gero, leiho bat ezarriko da emaitzen gainean. Leiho horretan agertuko da emaitzaren testuingurua testu gisa, orrialdeari dagokion faksimile irudia eta jatorrirako esteka. Hain zuzen ere, 4.14 irudian ageri den moduan.



4.14 Irudia: Emaitzaren testuingurua eta faksimilea bistaratzeko pantailaren diseinua.

5. KAPITULUA

Garapena

Atal honetan, proiektuaren garapenaz hitz egingo dugu; besteak beste, *Apache Solr*en instalazioaz, nodoen sorreraz, administrazio atalaz, eskemak zehazteaz eta *Apache Solr* eta *Solarium* liburutegiaren bateratzeaz. Gainera, erabilpen-kasu bakoitzaren garapena nola gauzatu den azalduko dugu, interfaze grafikoaren irudi batzuk ikusiko ditugu eta erabilitako teknologiak ere zerrendatuko ditugu.

5.1 Apache Solr

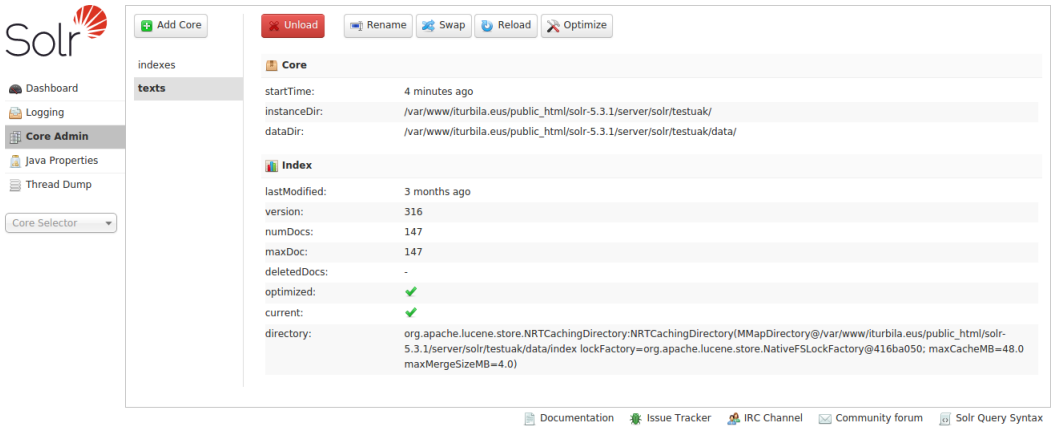
*Apache Solr*en inguruko instalazioa zein garapena egiteko ezinbestekoa izan da *Apache Solr Reference Guide*[2] izeneko wiki-aren laguntza.

4.2 atalean aipatu dugun moduan, lehenengo pausoa *Solr* instalatzea eta martxan jartzea da. Horretarako, *Java Runtime Environment* instalatu ostean, *Solr*en webgune ofizialetik fitxategi trinkotua jaitsiko dugu eta, bertako fitxategiak erauzi ondoren, hasieratu egin beharko dugu, ondorengo komandoa exekutatuz.

```
$ solr-5.3.1/bin/solr start -noprompt
```

Betekizunetan esan dugunez, 2.2.1 atalean, prototipo bat izanik, lokalean exekutatuko da, beraz, ordenagailua hasieratzen den bakoitzean martxan jartzen ibili behar ez izateko, aurreko komandoaren lerroa `/etc/rc.local` fitxategiari gehituko diogu.

Hurrengo pausoa nodoak sortzea izango da. 4.2 atalean erabaki dugunez, bi nodo sortu beharko ditugu: testuen eta indizeen edo hitzen nodoa. Hori egiteko modurik errazena, administrazio-atalean sartuz egitea da, `http://localhost:8983/solr/#/` helbidean sartuz eta, *Core Admin* azpimenua aukeratu ondoren, *Add core* aukeratuz. 5.1 irudian ikus daiteke administrazio-atalaren egoera, jada bi nodoak sortuta.



5.1 Irudia: Administrazio-atalean, *texts* eta *indexes* nodoak sortuta.

Nodoak sortu ondoren, hurrengo pausoa eskemak konfiguratzea izango da. Horretarako, 4.2 atalean diseinatu ditugun bi eskemak, bakoitzari dagokion fitxategira gehitu beharko ditugu:

- Testuen eskema (4.10 irudiko eskema) ⇒ `solr-5.3.1/server/solr/testuak/conf/schema.xml`
- Indizeen eskema (4.11 irudiko eskema) ⇒ `solr-5.3.1/server/solr/indizeak/conf/schema.xml`

Behin bi eskemak konfiguraturik, aldaketak gauzatzeko *Apache Solr* berrabiarazi beharko dugu edo, `/etc/rc.local` fixategian hasieratzeko deia gehitu dugunez, sistema berrabiarazi.

Konfiguratu ostean, sortuko dugun web-aplikazioaren bitartez *Apache Solr*eko indizeak eguneratzea eta bilaketak egin ahal izatea errazteko, 5.2 atalean azalduko dugun *Solarium* liburutegia erabiliko dugu.

5.2 Apache Solr eta Solarium liburutegia

Hasiera batean *Solr PHP Client*[1] erabiltzearen aukera aztertu zen arren, oso oinarrizko eskaerak egiteko soilik balio zuenez, *Solarium*[2] erabiltzea erabaki da, aukera gehiago eskaintzen dituelako eta, batez ere, *Solr*ek eskaintzen dizkigun ondorengo bi aukerekin lan egiteko aukera ematen digulako: emaitzak multzokatzeko aukera eta bilaketa-hitzari dagokion esaldi zatia nabarmentzeko aukera.

Solarium liburutegiaren erabilerarako, ezinbestekoa izan da daukan dokumentazioa[1] aztertzea, bertan baitaude *Solarium*ek eskaintzen aukera guztiak ahalik eta egokien erabiltzeko adibideak.

Hain zuzen ere, bertako atal batean ematen dira web-aplikazio baten eta *Apache Solr*en arteko lotura egiteko azalpenak. Gure kasuan, 5.2 irudiko konfigurazio-datuekin, PHP aldagai bat sortuko dugu eskema bakoitzeko.

```
$config_indexes = array(
   'endpoint' => array(
      'localhost' => array(
         'host' => '127.0.0.1',
         'port' => 8983,
         'core' => 'indexes',
         'path' => '/solr/',
      )
   )
);
```

```
$config_docs = array(
   'endpoint' => array(
      'localhost' => array(
         'host' => '127.0.0.1',
         'port' => 8983,
         'core' => 'texts',
         'path' => '/solr/',
      )
   )
);
```

5.2 Irudia: Web-aplikazioa eta *Solr* bateratzeko konfigurazio-datuak.

Konfigurazio-aldagaiak sortu ostean, aldagai horietako bakoitzarekin *Solarium* bezero bat sortuko dugu:

```
$client_indexes = new Solarium\Client($config_indexes);
$client_docs = new Solarium\Client($config_docs);
```

Ondorengo ataletan azalduko da indizeak eguneratzeko edo bilaketa-eskaerak egiteko jarraituko diren pausoak.

[1] https://github.com/PTCInc/solr-php-client
[2] https://github.com/solariumphp/solarium

5.3 Erabilpen-kasuen garapena

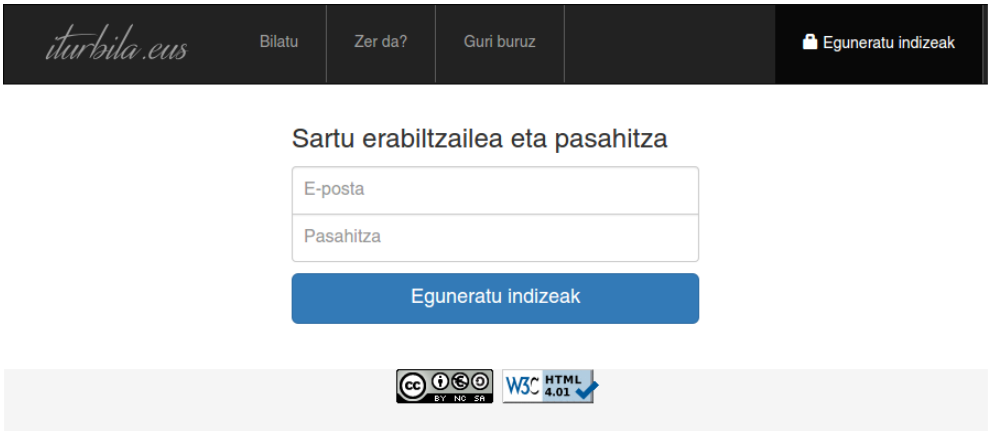
Analisia eta diseinua egin ondoren, 4.1 azpiatalean deskribatutako erabilpen-kasuak garatu beharko ditugu. Garapen hori egiteko jarraitu diren oinarrizko pausoen azalpenak emango ditugu atal honetan.

Aplikazioaren egiturari dagokionez, aurreko kapituluko 4.1 irudiko eskema orokorra izango da aplikazioaren egitura zehazteko erabiliko dugun eskema: aplikazio nagusia *Bootstrap* erabiliz sortuko da, bilaketak egiteko *Apache Solr*, bien arteko lotura *Solarium* liburutegiaren bitartez gauzatuko da eta, azkenik, testuak normalizatzeko eta hitz estandarrak lematizatzeko *Foma* eta *Phonetisaurus* erabiliko dira.

Era berean, web-sistemak irakasgaian ikasitako gaiak ere praktikan jarriko ditugu, batez ere, PHP kodea garatzeko orduan edo sortutako HTML kodea nabigatzaile guztiek ahalik eta beretsuen interpretatzeko. Horretarako, garapena gauzatzeko garaian, W3C estandarra jarraitu eta balidatuko[3] dugu.

5.3.1 Login egin

Login pantaila egin ahal izateko, 5.3 irudian ikus daitekeenez, *Bootstrap*ek eskaintzen duen eredu bat hartu da, administratzailearen e-posta eta pasahitza sartzeko aukera ematen duena. Sartzen diren datuak `config.php` konfigurazio-fitxategian zehaztutakoekin bat badatoz, testuak API bitartez lortzera pasatuko da.



5.3 Irudia: Login egiteko pantaila.

[3] https://validator.w3.org

5.3.2 Testuak API bitartez lortu

3.1.2 atalean Iturriak eta Wikisource guneen APIen azterketa egin ostean, aplikazioaren bitartez testuak lortu beharko ditugu. Testuak lortu ahal izateko, lehenik eta behin, zein testu lortu nahi diren zehaztu beharko da, indexatzen diren testuak kontrolpean izateko. Horretarako, *texts_links.xml* izeneko XML fitxategi bat sortu dugu, 5.4 irudikoa, testu bakoitzeko izenburua eta jatorrizko indizerako esteka biltzen dituena.

```
<texts>
  <text>
    <title>Axular Gero</title>
    <link>https://wikisource.org/wiki/Index:AxularGero.djvu</link>
  </text>
  <text>
    <title>Escualdun Cocinera</title>
    <link>http://www.iturriak.eus/index.php/Index:AM_018972OCR.djvu</link>
  </text>
  {...}
</texts>
```

5.4 Irudia: *Texts_links.xml* konfigurazio fitxategia, testuen izenburu eta estekekin.

API deiak egiteko, Wikisourceren gomendioa da ahalik eta eskaera gehien bateratzea dei bakarrean, eta horixe egingo dugu, beraz, API deia egin ahal izateko, orrialde kopurua lortu beharko dugu. Adibidez, Axularren *Gero* obraren orrialde kopurua lortzeko, estekaren abiapuntuaren jarraian, ondorengo parametroak pasa beharko genituzke:

```
?format=xml&action=query&titles=File:AxularGero&prop=imageinfo&iiprop=size
```

API eskaerak gauzatzeko, PHPren `simplexml_load_file()` funtzioa erabiliko dugu, eskaeraren XML formatuko emaitzak erabili ahal izateko.

Behin orrialde kopurua izanik, testuen eskaera egin behar da, eskaerak bateratuz. Hala ere, ikusi dugu 50 izenetik gorako eskaerek arazoak ematen dituztela estekaren luzera dela-eta, beraz, eskaerak 50naka egingo ditugu, orrialde guztiak lortu arte.

Ikus dezagun, adibide moduan, Axularren *Gero* obraren lehenengo hiru orrialdeak lortzeko egin beharreko API deia:

```
?format=xml&action=query&titles=Page:AxularGero/1|Page:AxularGero/2
|Page:AxularGero/3&prop=revisions&rvprop=content&rawcontinue
```

Itzultzen duen emaitzak HTML, Wiki eta antzeko markak izan ditzake, kendu behar izango ditugunak ordezkatze-metodo eta espresio erregularren bitartez. Marka horiek kendu ahal izateko, PHPren `str_replace` eta `preg_replace` funtzioak erabiliko dira.

Testuak lortu ondoren, testu bakoitzari dagokion faksimilearen lortzeko API deia sortuko dugu, emaitzak bistaratzen direnean, faksimilea ikusi nahi izanez gero, erabiliko litzatekeena. Honakoa izango da API deia, `page1` parametroaren 1 zenbakiaren ordez, lortu nahi den orrialdearen zenbakia jarriz:

```
?format=xml&action=query&titles=File:AxularGero&prop=imageinfo&
iiurlparam=page1-1024px&iiprop=url
```

Azkenik, testuaren izenburuarekin, testu soilarekin, estekarekin, API estekarekin eta irudiaren API estekarekin `texts.xml` izeneko XML fitxategia sortuko dugu. Fitxategi hori, bertako hitz ez-estandarrak normalizatu eta lematizatu ostean, *Solr*en bitartez indizeak eguneratu ahal izateko erabiliko dugu.

5.3.3 Hitz ez-estandarrak normalizatu

Hitz ez-estandarrak normalizatu aurretik, beharrezkoa izango da `texts.xml` fitxategiko testu soilak lortzea, `only_texts` deituko dioguna, ondoren, `tokenizatu.pl` Perl programaren bitartez, hitz guztiak tokenizatzeko:

```
perl standard/scriptak/tokenizatu.pl only_texts > only_texts.tok;
```

Behin hitzak tokenizaturik, Linux komandoen bitartez, errepikatutako hitzak kenduko ditugu, baita karaktere arraroak dituztenak ere eta, luzerari begira, gutxienez bi letrakoak hartuko ditugu. *Phonetisaurus*i lana kentzeko asmoz eta, ondorioz, exekuzioa azkarrago egiteko asmoz, aurretik indexatutako hitzak ez dira berriro aztertuko. Hori egin ahal izateko, orain arte lortutako hitzen eta ordura arte indexatutako hitzak biltzen dituen `indexed_words` fitxategiaren alderaketa egingo da, hitz berriak soilik biltzen dituen `to-index` izeneko fitxategia sortuz.

Hitz berri horiek normalizatu ahal izateko, nahikoa izango da `deia.sh` *script*ari parametro gisa `to-index` fitxategia pasatzea:

```
sh standard/deia.sh to-index
```

Script hori exekutatzean, pasatako fitxategiaz gain, `phonetis` katalogoa, hipotesi kopurua eta `phonetis.fst` transduktore haztatua pasako zaizkio `2testphon.sh` *script*ari:

```
sh standard/scriptak/2testphon.sh to-index phonetis 5 \
   datuak/phonetis/phonetis.fst
```

Parametro horiei esker, azken *script* hori izango da benetan *Phonetisaurus*i deia egingo diona:

```
phonetisaurus-g2p --model=datuak/phonetis/phonetis.fst --input=to-index \
   --isfile --words --nbest=5  --beam=5000 >test-out
```

Azkenik, *Phonetisaurus*ek itzultzen dituen hipotesien artean estandarrak soilik aukeratuko ditugu, *flookup* tresnaren eta beste transduktore baten bitartez:

```
cat test-out | flookup xuxen_UTF_NOALD_NORARE_foma.fst >standards
```

Hori horrela, `to-index` fitxategiko hitzak normalizaturik, hitz estandar horien lemak lortu beharko ditugu.

5.3.4 Lematizatu

Hitz estandarren lemak lortu ahal izateko, `luzeena.pl` izeneko Perl programari deituko diogu:

```
cat standards | perl stemmer/luzeena.pl >lemmas.txt
```

Era berean, Perl programa horrek, hitz bakoitzeko, aurreko atalean erabilitako komando bera erailtzen du, *flookup*, baina transduktore ezberdin batekin:

```
flookup -i -x -b stemmer/xuxen_stem_manex.fst
```

5.3.5 *Solr*eko indizeak eguneratu

*Solr*eko indizeak eguneratzeko garaian, bi eguneraketa-mota egingo direla kontuan izan behar dugu: testuena eta hitzena.

Eskema bakoitzerako *Solarium* bezero bat sortuta dugunez, testuen bezeroari eguneraketa bat egin nahi dugula adieraziko diogu `createUpdate()` funtzioaren bitartez:

```
$update = $client_docs->createUpdate();
```

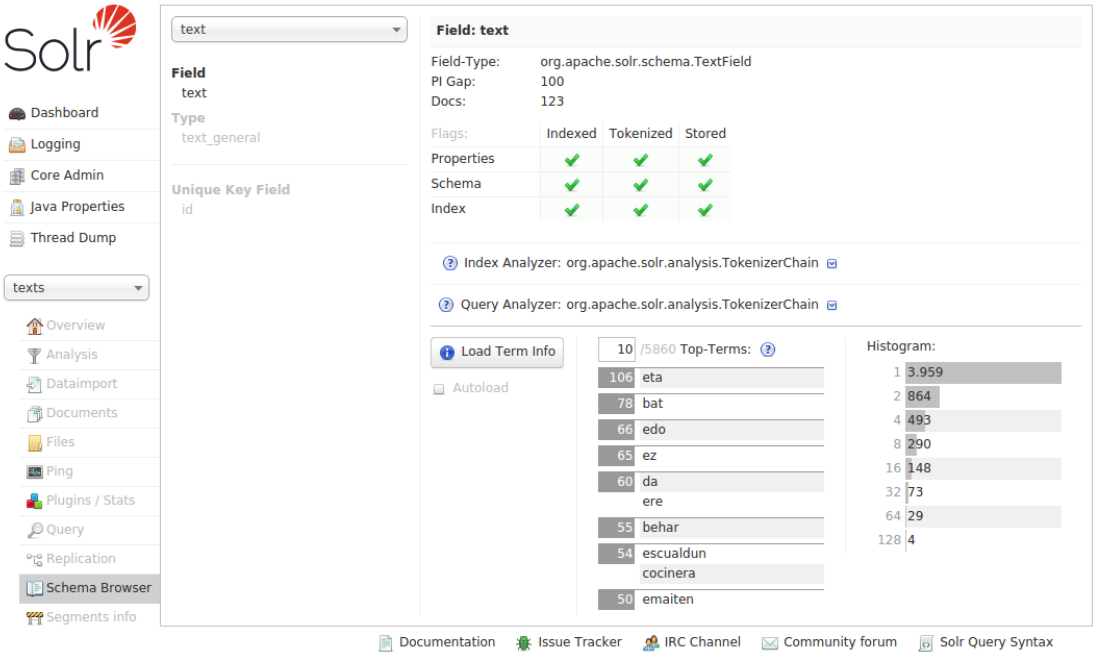
Ondoren, `doc1` aldagaia sortuko dugu eta, testu guztiak biltzen dituen *texts.xml* fitxategiaz baliatuz, etiketa bakoitza dagokion `fields` aldagaian jasoko dugu, ondoren array batean jasoz:

```
$doc1 = $update->createDocument();
$fields = $d->getElementsByTagName('field');

$doc1->id = $fields->item(0)->nodeValue;
$doc1->title = $fields->item(1)->nodeValue;
$doc1->text = $fields->item(2)->nodeValue;
$doc1->link = $fields->item(3)->nodeValue;
$doc1->apilink = $fields->item(4)->nodeValue;
$doc1->apiimg = $fields->item(5)->nodeValue;

$array[] = $doc1;
```

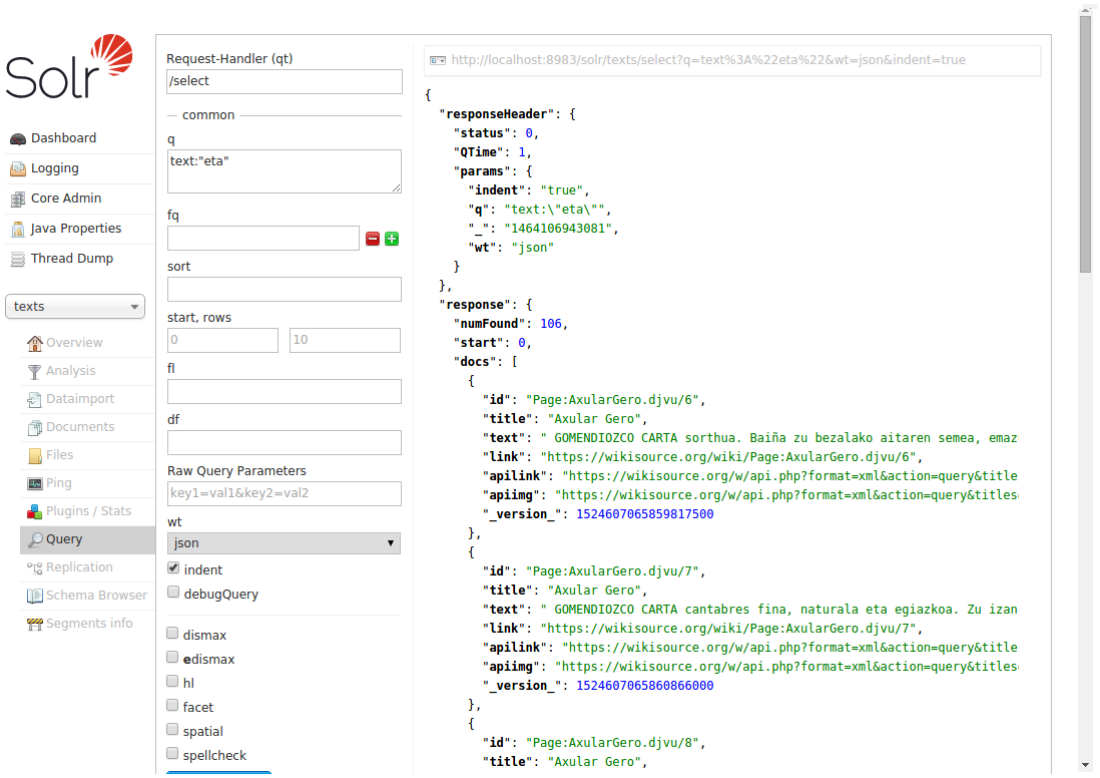
Azkenik, aurretik sortutako `update` aldagaiari dokumentuak gehitu eta eguneraketak gauzatzeko agindua eman ostean, `client_docs` eguneratuko dugu, aurretik emandako datuekin:

```
$update->addDocuments($array);
$update->addCommit();
$client_docs->update($update);
```

5.5 irudian ikus dezakegu *texts* nodoa ongi eguneratu dela, pasatutako testuetako hitzak indizeetan jaso baititu. Are gehiago, indexatutako testuak zein diren ere kontsulta dezakegu *Solr*en administrazio-atalaren bitartez, 5.6 irudian ikus daitekeen moduan.



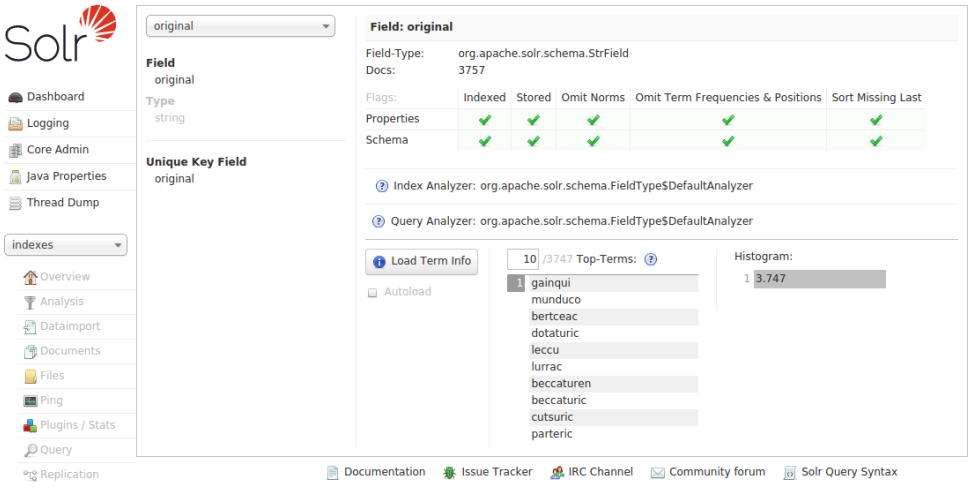
5.5 Irudia: Testuetako hitzak ongi eguneratu direla ikusteko administrazio-atala.



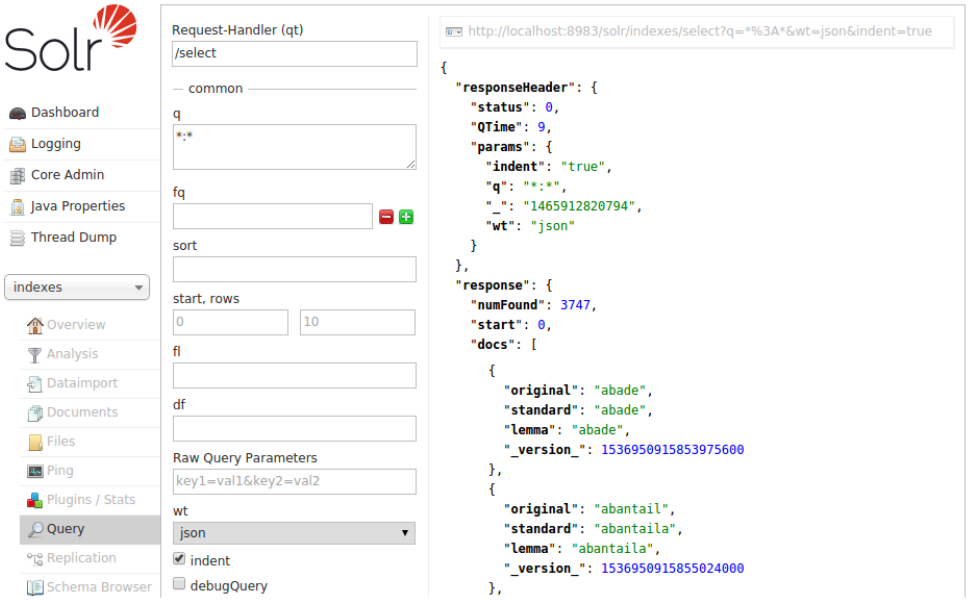
5.6 Irudia: Testuak kontsultatzeko administrazio-atala.

Behin testuak eguneraturik, era berean gauzatuko da hitzen eguneraketa ere, hau da, `client_indexes` indizeen bezeroari eguneraketa bat egin nahi dugula adierazi, *indexes.xml* fitxategiko eremu bakoitzarekin `doc2` aldagaia sortu eta `client_indexes` indizeen bezeroa eguneratuz.

Testuekin egin dugun moduan, 5.7 irudian ikus dezakegu *indexes* nodoa ongi eguneratu dela eta, 5.8 irudian, horien kontsulta ere egin dezakegula.



5.7 Irudia: Hitzak ongi eguneratu direla ikusteko administrazio-atala.



5.8 Irudia: Hitzak kontsultatzeko administrazio-atala.

5.3.6 Hitzen bilaketak egin

Testuetan bilaketak egin ahal izateko, erabiltzaileak bilatu nahi duen hitza sartu eta hitz-mota (jatorrizkoa, estandarra edo lema) aukeratu beharko ditu.

Hitz-mota estandarra edo lema aukeratuz gero, lehenengo pausoa indizeen *Solarium* bezero bat sortu eta hitz horri dagozkion jatorrizko hitzak –hitz ez-estandarrak– lortzea izango da:

```
$client_indexes = new Solarium\Client($config_indexes);
$query = $client_indexes->createSelect();
$query->setQuery($word_type.":".$q);
$resultset = $client->select($query);
```

Behin jatorrizko hitzak lorturik edo hitz-mota jatorrizkoa aukeratuz gero, jatorrizko hitz horiek testuetan bilatu beharko dira. Horretarako, testuen bezero bat sortu eta eskaera zein den esan beharko zaio:

```
$client_docs = new Solarium\Client($config_docs);
$query = $client->createSelect();
$query->setQuery($query_originals);
```

Hala ere, ondoren emaitzak bistaratzea errazte aldera, emaitzak testuka multzokatzeko aukera aktibatuko ditugu:

```
$hl = $query->getHighlighting();
$hl->setFields('text');
```

Baita bilaketa-hitzei dagokien esaldi zatia markatzeko aukera ere:

```
$groupComponent = $query->getGrouping();
$groupComponent->addField('title');
```

Aukera guztiak ongi zehaztu ondoren, `selecta` gauzatuko dugu:

```
$resultset = $client->select($query);
```

Deia eginik, `resultset` aldagaian ditugu emaitzak guztiak, hortaz, emaitzak bistaratzera pasatuko gara.

5.3.7 Emaitzak bistaratu

Emaitzak taldekatuta eskatu ditugunez, *Bootstrap*en `panel-group`[4] bat sortu beharko dugu, non testu bakoitzeko `panel`[5] bat sortuz, testuko emaitzak ezkutatu ahal izango diren, hau da, erabiltzaileak interesatzen zaizkion testuen emaitzak soilik bistaratu ahal izan ditzan.

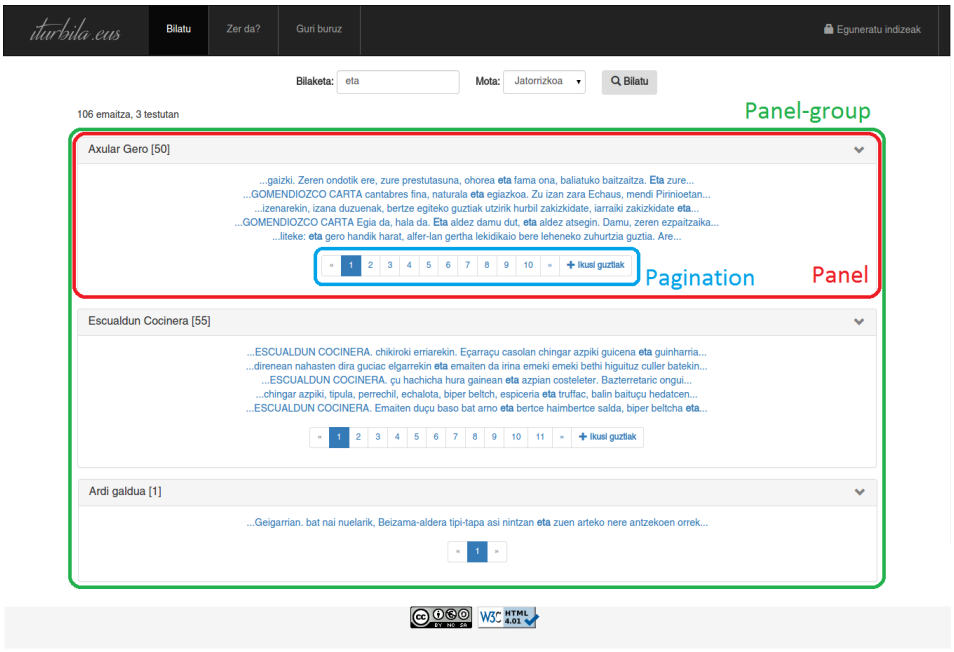
Ondoren, `panel` bakoitzeko emaitzak erakusteko, `resultset` objektuko `highlight` emaitzak hartuko dira, baina, guztiak bistaratu beharrean, emaitzak orrialdetan banatuko dira *Bootstrap*en `pagination`[6] modulua erabiliz; `config.php` konfigurazio-fitxategian zehazten da, hain zuzen ere, gaindi ezin daitekeen bistaratu beharreko emaitza kopurua. Hala ere, testu baten emaitza guztiak bistaratzeko aukera ere eskaintzen da, *JavaScript*en *jQuery* liburutegia erabiliz, ondorengo kode zatiaren bitartez:

```
$(".more-results").on( "click", function() {
   var title = $(this).attr('data-title');
   $("div[data-titlenum^='"+title+"']").show();
   $(this).parent().parent().parent().find('.paginator').hide();
   $(this).hide();
});
```

5.9 irudiak erakusten ditu `panel`, `panel-group` eta `pagination` osagaiak zein diren.

Hasiera batean emaitza gisa bistaratuko ez den arren, emaitzak aztertzen joan ahala, emaitza bakoitzeko `modal`[7] bat sortu beharko dugu, hau da, emaitzen gainean ezarriko den leiho bat. Horrez gain, leiho horretan bistaratu behar diren datuak lortu eta bilatutako

[4] http://getbootstrap.com/javascript/#collapse-example-accordion
[5] http://getbootstrap.com/components/#panels
[6] http://getbootstrap.com/components/#pagination
[7] http://getbootstrap.com/javascript/#modals



5.9 Irudia: Hitz baten emaitzak bistaratzea, *Bootstrap*en `panel`, `panel-group` eta `pagination` osagaiak identifikatuz.

hitzaren jatorrizko hitzak markartzeko, `data-originals`, `data-api` eta `data-img` parametroak sartuko ditugu HTML kodean.

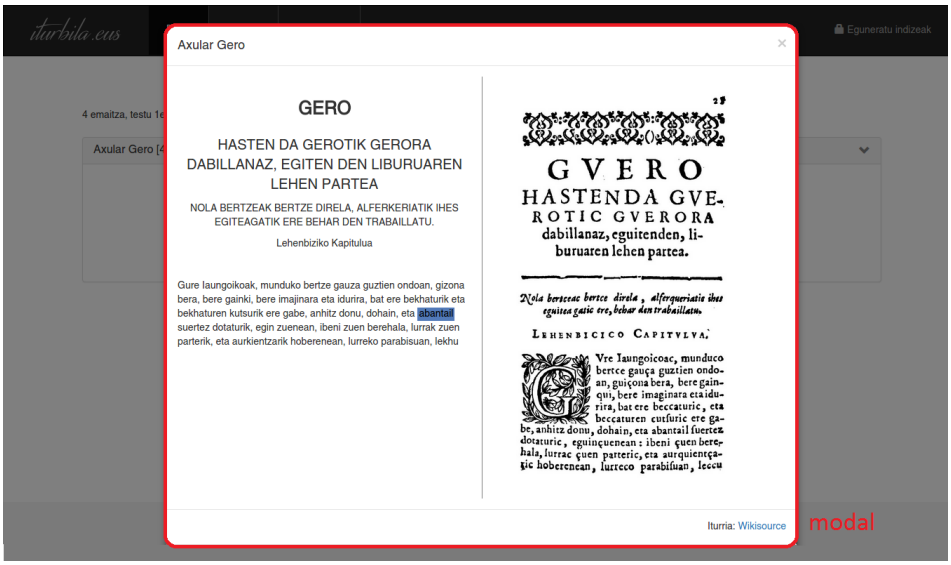
Emaitza baten gainean klikatzean, emaitza horri dagokion testua eta faksimilea bistaratuko dira.

5.3.8 Hitzaren testuingurua eta jatorrizko faksimilea bistaratu

Aurreko atalean esan dugun moduan, emaitzak bistaratzeko `modal` osagaiak eta osagai horiek behar dituzten datuak lortzeko `data-api` eta `data-img` parametroak ditugunez, *jQuery* liburutegia erabiliz, `get_text_img.php` fitxategiari beharrezko datuak lortzeko eskatuko diogu.

Horretarako, bi API dei egin beharko ditu: bata, formatudun testua lortzeko, *Solr*en jasotako testuak ez baitu formaturik eta, bestea, faksimilea lortzeko. Hori horrela, formatudun testuan jatorrizko hitzak markatuko dira, `data-originals` parametroari esker eta irudia ere bistaratuko da, `modal` osagaia osatuz, 5.10 irudian ikus daitekeen moduan.

Aukeraturiko emaitzak ez badu faksimilerik, irudirik ez duela adierazten duen irudia bistaratuko da. Era berean, formatudun testua lortzeko arazoren bat izan bada, bilatutako jatorrizko hitzak markatutako testu soila bistaratuko da, hots, formaturik gabekoa.



5.10 Irudia: Emaitza baten testuingurua eta faksimilea bistaratzea, `modal` osagaiaren bidez.

Behin garapena amaituta, garatutako kodearen azken bertsioa GitHub[8] berri batera igo da, baina kontuan izan beharko da, martxan jarri ahal izateko, *Phonetisaurus* eta *Flookup* instalatuta izatea eta Solarium eta *Apache Solr* deskargaturik izateaz gain, kofiguratzea.

5.4 Web diseinu moldagarria (Responsive web)

Web-aplikazioa garatzeko *Bootstrap* erabili denez eta, 3.3.4 atalean ikusi genuen moduan, *Bootstrap mobile first* filosofia erabiliz garatua dagoenez, webgunea moldagarria izango da. Bada, ikus dezagun, adibidez, mugikor batean nola ikusiko litzatekeen.

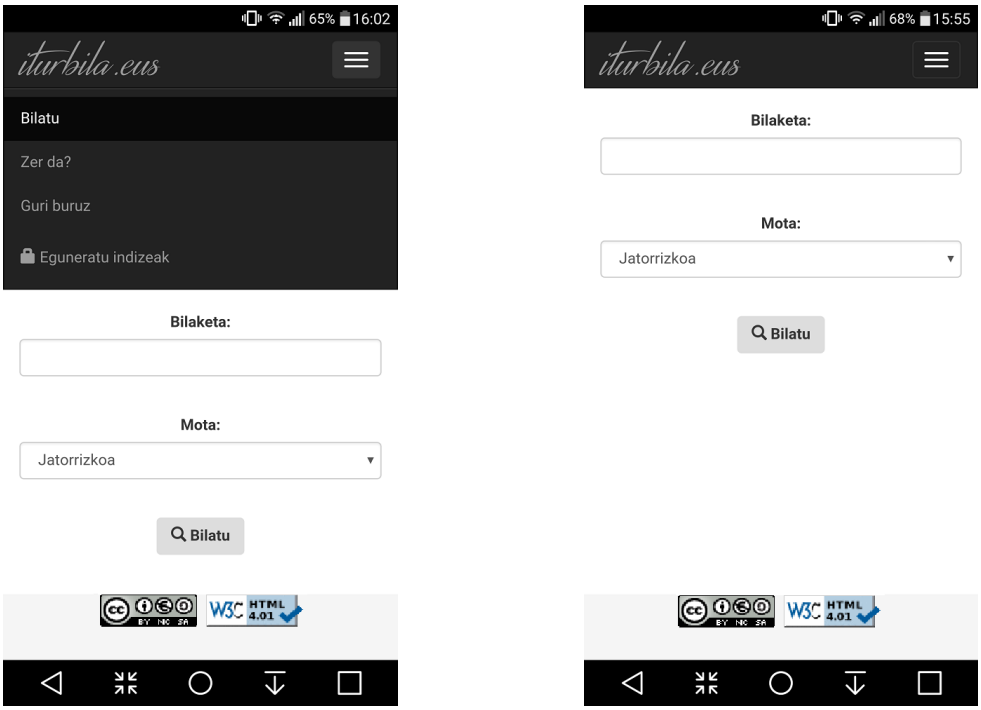
Menu xume bat egin den arren, 5.11 irudian ikus daiteke menua erabat moldatua dagoela mugikorrerako bertsiorako.

Bilaketa-eremuak ere pantailaren neurrira egokituta daude, 5.11 irudian dagoen moduan. Adibidez, “jainko” hitza bilatuko dugu eta hitz-mota “lema” aukeratuko dugu.

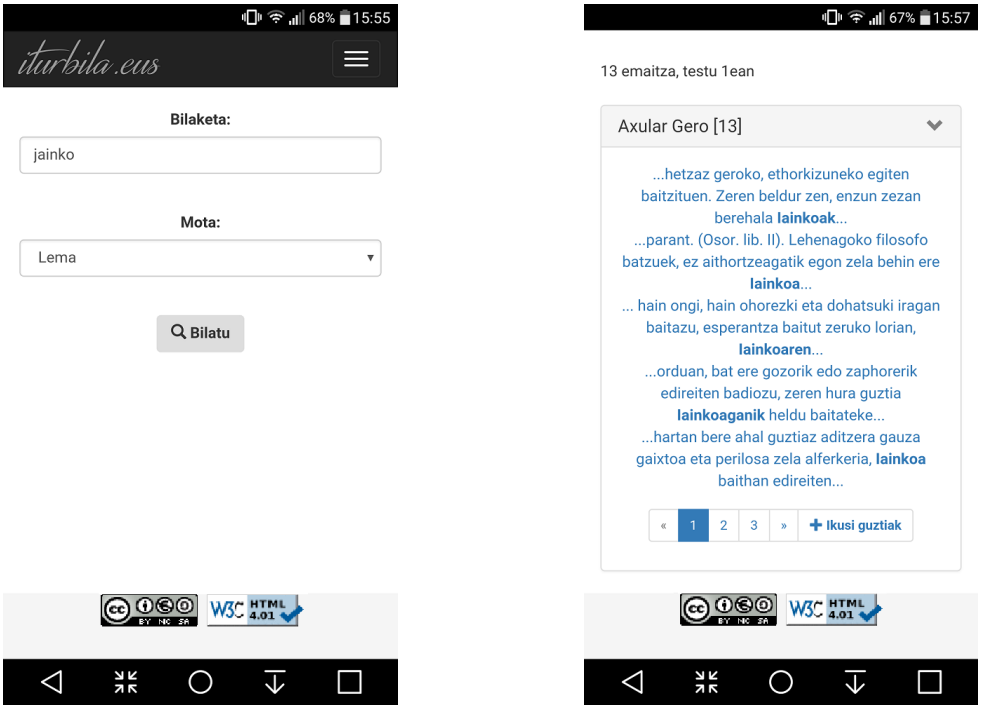
Itzultzen dituen emaitzen bistaratzea 5.12 irudian ikus dezakegu. Hala ere, mugikorra horizontalean jarriz gero, berriro egokitzen da, hau da, 5.13 irudian dagoen moduan bistaratzen da.

Ez da testuinguru eta faksimilea bistaratzearen irudirik jarri, testuaren azpian agertzen denez faksimilea, ez delako pantaila batean sartzen, baina hau ere egokitzen da.

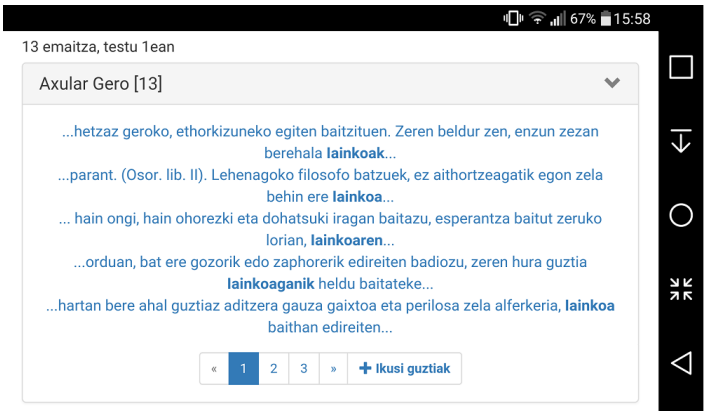
[8] https://github.com/jabelamendia/iturbila



5.11 Irudia: Aplikazioaren menua ezkerrean; eskuinean, bilaketa-eremuak.



5.12 Irudia: Ezkerrean, “jainko” hitza eta hitz-mota “lema” aukeratuta; eskuinean, bilaketaren emaitza bertikalean.



5.13 Irudia: "Jainko" hitza eta hitz-mota "lema" aukeratuta, bilaketaren emaitza horizontalean.

5.5 Eginiko probak

Behin garapena amaiturik, aplikazioa probatzeko benetako proba batzuk egin dira. Probak egin ahal izateko ondorengo obren zati batzuk erabili dira: Gero (1643), Escualdun Cocinera (1864) eta Ardi galdua (1918). Testuak lortu ahal izateko, obren estekak gehitu ditugu konfigurazio-fitxategian, API bitartez testuak lortu, bertako hitzak normalizatu eta lematizatu ahal izateko. Fitxategi hori erabiliz eta API deiak eginez, guztira 147 testu lortu ditugu eta 3.747 indize ezberdin, hau da, 3.747 hitzen estandar eta lemak lortu ditugu.

Probak egiterako garaian ez dugu kontuan izan normalizazio-sistemak itzultzen dituen emaitzak hobeak ala okerragoak diren, hori ez baita proiektu honen helburua eta, gainera, normalizatzeko erabiltzen den ereduaren menpekoa baita. Kasu honetan, erabiltzaile arrunt baten ikuspuntutik probatu da web-aplikazioaren erabilera.

Orain arte, aditua ez den erabiltzaile batek testu historikoetan, adibidez, "jainkoa" hitza bilatu nahi zuenean, ziur aski ez zituen nahi beste emaitza lortzen, hitza estandarra izanik eta testu historikoetako hitz gehienak ez-estandarrak izanik, aukera gutxi zeudelako.

Gure aplikazioan, "jatorrizkoa" bada bilatu nahi den hitz-mota, "jainkoa" hitzak ez du emaitzarik itzultzen, baina gehitu diogun normalizazio-sistemari esker, "estandarra" aukeratuta 3 agerpen itzultzen ditu. Zehazki indizeetan jasorik dagoen "iainkoa" hitzaren bilaketa egiten du.

Baina zer gertatzen da, "jainkoa" hitzaz gain, "jainkoak", "jainkoaren", "jainkoagandik" eta antzerakoak ere bilatu nahiko bagenitu? Banan-banan bilatzen joan beharko litzateke? Bada, ez, lematizatzaile bat ere gehitu baitiogu. Beraz, "jainko" lemaren bilaketa eginez,

ondorengo hitzen bilaketa gauzatuko litzateke: “iainko”, “iainkoak” eta “iainkoaganik”. Guztira 13 emaitza lortu ditugu bilaketa-mota hau eginez.

Antzerako gauza gertatzen da gainontzeko hitzekin ere; beste bi adibide ematearren, “etorkizun” eta “aipamen” hitzen bilaketa-mota ezberdinak egingo ditugu:

- “Etorkizun” hitza.
  - Jatorrizkoa: ez du emaitzarik itzultzen.
  - Estandarra: ez du emaitzarik itzultzen; emaitzak lortzeko asmoz, probatu dira “etorkizun”, “etorkizuna” eta “etorkizunaren”, baina ez da emaitzarik lortu.
  - Lema: 4 emaitza bistaratu dira: “ethorkizunera”, “ethorkizunerat” eta “ethorkizunari” hitzenak, hain zuzen ere.
- “Aipamen” hitza.
  - Jatorrizkoa: ez du emaitzarik itzultzen.
  - Estandarra: ez du emaitzarik itzultzen.
  - Lema: 2 emaitza, “aiphamenean” eta “aiphamenik” jatorrizko hitzenak.

Probak egin diren bitartean, aplikazioak ez du errorerik eman eta esperotako emaitzak itzul ditu. Beraz, aplikazioa ongi ibiltzeaz gain, ondoriozta dezakegu adituak ez diren erabiltzaileentzat laguntza handia dela, bilaketa arruntak eginez ez baitzen emaitzarik lortzen, baina orain bai estandar edo lemen bilaketa eginez.

5.6 Erabilitako teknologia eta tresnak

Asko izan dira proiektu hau garatzeko erabilitako teknologia eta tresnak. Laburbiltze aldera, proiektuan zehar erabilitakoak zerrendatuko ditugu, bakoitzaren erabilera nagusia azalduz:

- **Bootstrap**. Web-aplikazioko bistaratzeak gauzatzeko.
- **HTML / PHP**. Web-aplikazioaren egitura eta garapena egiteko.
- **CSS3**. Aplikazioaren itxura moldatzeko.

- **Javascript / jQuery liburutegia**. Webguneak dinamiko egiten dituen lengoaia izanik, ondorengoak garatzeko:
  - Orrialde bakoitzean agertu beharreko emaitzak kudeatzeko, `paginator` erabiltzeko.
  - Orrialdetan banatutako emaitza guztiak bistaratzeko.
  - Testuaren formatua eta faksimilea bigarren plano batean kargatzeko, `modal` osagaia osatuz.
- **Apache Solr**. Testuetako indizeak jasotzeko, ondoren bilaketak gauzatzeko.
- **Solarium liburutegia**. Web-aplikazioaren eta *Apache Solr*en lortura egiteko.
- **Phonetisaurus**. Testuetako hitzak normalizatzeko.
- **Foma / Flookup**. Hitzak estandarrak diren jakiteko eta, estandarrak diren kasuetan, lema lortzeko.

Azkenik, aipatu behar dugu memoria garatzeko LaTeX erabili dugula, *Overleaf*[9] inguruenean eta, bertsioen kontrolerako, *GitHub*[10] erabili dugula.

[9] https://www.overleaf.com
[10] https://github.com

6. KAPITULUA

Jarraipena eta Kontrola

Atal honetan proiektua egiteko denbora erreala aurretik planifikatutako estimazioarekin alderatuko dugu, izan diren desbideraketak lortuz eta horien zergatia azalduz. Kontuan izan behar da 280 ordu planifikatu zirela, eta planifikatzeke geratzen ziren 20 ordu atzerapenetarako erabiliko zela, beraz, plangintzako estimaziotik 20 ordutan pasatzea normala litzateke.

Desbideraketez gain, Gantt diagrama berria egingo dugu eta mugarri berriak zein izan diren ere azalduko dugu, izan ere, aurrerago azalduko ditugun bi geldialdi garrantzitsu izan ditugunez, asko aldatu baitira.

Era berean, hasieran definitutako zein irismen-maila bete den eta kalitate-adierazleak ere bete diren ala ez aztertuko dugu, eskuraketa berririk egin den kasuan zein izan den eta, azkenik, izandako arriskuei aurre egiteko erabilitako kudeaketa azalduko ditugu.

6.1 Desbideraketak

Plangintzan aipatu genuen moduan, lehen proiektu handia izanik, plangintzan akatsak izango zirela aurreikusi zen eta, hein handi batean, halaxe izan da. Hasieran eginiko estimazioaren eta pasatako denboraren alderaketa ikus daiteke 6.1 taulan.

Oro har, izan diren desbideraketarik handienak garapenean eta memorian eman dira, planifikatutakoa baino denbora gehiago eskaini baitiegu. Hala ere, izan dira denbora aurreztu ditugun atalak ere, ikerkuntza eta analisia eta diseinuarena, hain zuzen ere.

Kodea	Ataza	Estim.	Erreala	Desb.	
Kudeaketa		**30 ordu**	**35 ordu**	**+5 ordu**	**+%17**
1.1	Plangintza	15 ordu	20 ordu	+5 ordu	+%33
1.2	Jarraipena	10 ordu	9 ordu	-1 ordu	-%10
1.3	Kontrola	5 ordu	6 ordu	+1 ordu	+%20
Ikerkuntza		**30 ordu**	**27 ordu**	**-3 ordu**	**-%10**
2.1	Testuingurua eta APIak	10 ordu	13 ordu	+3 ordu	+%30
2.2	EKS/Framework erabilera	10 ordu	8 ordu	-2 ordu	-%20
2.3	Bilaketa-zerbitzariak	10 ordu	6 ordu	-4 ordu	-%40
Analisia eta diseinua		**60 ordu**	**51 ordu**	**-9 ordu**	**-%15**
3.1	Erabilpen-kasuak	10 ordu	7 ordu	-3 ordu	-%30
3.2	Sekuentzia-diagramak	40 ordu	36 ordu	-4 ordu	-%10
3.3	Interfazearen diseinua	10 ordu	8 ordu	-2 ordu	-%20
Garapena		**100 ordu**	**137 ordu**	**+37 ordu**	**+%37**
4.1	Web-aplikazioa	95 ordu	131 ordu	+37 ordu	+%38
	Apache Solr	-	*23 ordu*		
	Solarium liburutegia	-	*7 ordu*		
	Hitzen normalizazioa	-	*16 ordu*		
	Hitzen lematizazioa	-	*9 ordu*		
	EKen garapena	-	*76 ordu*		
4.2	Eginiko probak	5 ordu	6 ordu	+1 ordu	+%20
Dokumentazioa		**60 ordu**	**89 ordu**	**+29 ordu**	**+%48**
5.1	Memoria	55 ordu	83 ordu	+28 ordu	+%51
5.2	Aurkezpena	5 ordu	6 ordu	+1 ordu	+%20
Guztira		**280 ordu**	**339 ordu**	**+59 ordu**	**+%21**

6.1 Taula: Hasierako estimazioen eta denbora errealaren desbiderapenak.

Plangintzarekiko desbiderapen orokorra 59 ordukoa (%21) den arren, kontuan izan behar dugu 280 ordu planifikatu genituela. Proiektuaren 300 orduko dedikazioa kontuan izanez gero, 39 ordukoa (%13) da desbiderapena, beraz, kontuan izanik kudeatzen dugun lehen proiektu handia dela, nahiko emaitza ona da.

Ondorengo lerroetan ataza bakoitzaren desbideraketaren zergatia azaltzen saiatuko gara.

Kudeaketa

Kudeaketaren atalean estimatutakoa baino 5 ordu gehiago sartu behar izan dira, planifikazioaren atal garatu bat nahi zelako, besteak beste, betekizun zehatzak azalduz, atazak deskribatuz, proiektua gauzatzeko plana zehaztuz edota kalitate- eta arrisku-planak eginez.

Ikerkuntza

Ikerkuntzaren atalean APIen erabilera ezagutu eta dokumentatzeko aurreikusitako denbora baino gehiago behar izan da, baina bilaketa-zerbitzariak erabiltzearen alde onak ikusita, laster hartu da bat erabiltzearen erabakia.

Analisia eta diseinua

Software Ingeniaritza irakasgaian ikasi genituen garatzen hasi aurretik analisi eta diseinu on bat egitearen abantailak. Horregatik aurreikusi genuen hainbeste denbora dedikatuko geniola, baina azkenean gutxiago behar izan dugu.

Garapena

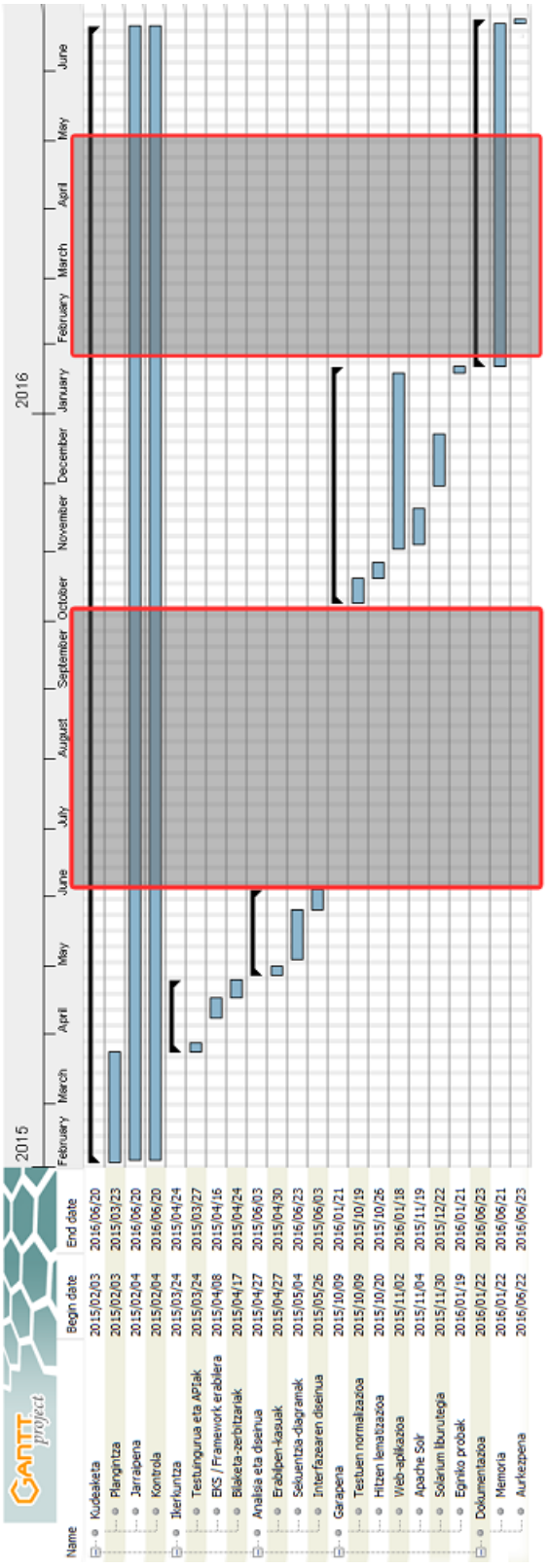
Hauxe izan da ordu aldetik gehien eskaini diogun atala, baita ordu gehien desbideratu garena ere. Desbiderapen nagusia *Apache Solr*en instalazio eta konfigurazioan izan dugu, izan ere, martxan jarri eta eskemak sortzeko zailtasunak izan baititugu. Gainerakoan, plangintzan xehetasunik eman ez genuen arren, ikus daiteke zehazki azpiatza bakoitzari eskainitako dedikazioa. Aipatu behar da "EKen garapena" azpiatazean sartzen direla web-aplikazioaren egitura orokorra eta *Bootstrap* ezartzea.

Dokumentazioa

Azken ataza hau izan da okerren planifikatu duguna, estimatutako orduen %48 gehiago sartu behar izan baitugu. Izan liteke memoria osatu bat egin nahi izan denez, irudi eta diagrama erabilgarriekin, horregatik luzatu izana. Esan beharra dago LaTeX erabiliz memoria gutxi egin izan ditugula eta, beraz, taulak eta antzekoak egiteko sarean laguntza ere bilatu behar izan dugula.

6.2 Gantt diagrama

Aurretik aipatu ditugun desbideraketen ondorioz Gantt diagrama berri bat sortu behar izan da, 6.1 irudikoa. Bertan ikus daiteke proiektuan zehar egin behar izan ditugun bi geldialdiak; zorionez, lehenengo geldialdirako analisia eta diseinua eginak genituen baina garapenarekin hasi gabe geunden, beraz, ez genion berriro atal berdinari ekin beharrik.



6.1 Irudia: Amaierako Gantt diagrama.

6.3 Mugarriak

Hasiera batean 2.2.2 azpiatalean mugarriak zehaztu genituen arren, nagusiki bi aldiz aldatu behar izan ditugu:

- Lehenengoan, irakasgai eta azterketekin bateratzeko zailtasunak izan zirelako eta, horiek amaitzean, lan-kontuak zirelako. Erabaki zen lana amaitzean, urrian, mugarri berriak ezartzea, urtarrileko deialdian entregatzeko asmoz.
- Memoriaren desbideraketa zela-eta, urtarrileko deialdian ezin izan zen aurkeztu eta, arazo pertsonalak zirela medio, ezin izan zitzaion dedikaziorik eskaini maiatza arte. Beraz, proiektua erabat amaitzeko, memoria bereziki, azken mugarriak ezarri ziren.

Hauek dira proiektua amaitzeko ezarri ziren mugarriak:

- **2016-06-05**: Proiektuaren memoria erabat amaitzea.
- **2016-06-23**: Memoria ADDIra igo, web-aplikazioaren garapena GitHub-ekin sinkronizatu eta defentsa-eguneko aurkezpena amaitu.

6.4 Irismen-maila

Plangintzako 2.2.3 atalean zehaztu ziren bi irismen-maila: oinarrizkoa eta aurreratua. Orain arteko jarraipen eta kontrola egin ostean, eta ikusirik atzerapenak izan ditugula, oinarrizko-mailan geratzea erabaki dugu. Beraz, etorkizuneko lan bat izan liteke maila aurreratua betetzeko eskatzen zen etiketatzeko aukera garatzea.

6.5 Kalitate-plana

Proiektua amaitu ondoren, atal honetan, hasiera batean A eranskinean definitu genuen egiaztapen-zerrenda beteko dugu.

6.2 taulan ikus dezakegunez, hasiera batean definitu genituen oinarrizko-maila betetzeko kalitate-zerrenda betetzen du.

Proiektuaren kalitate-zerrendaren egiaztapena	Bai/Ez
Proiektuaren kudeaketa	
Proiektuaren betekizun guztiak bete dira?	Bai
Proiektuaren irismena bat dator egindako garapenarekin?	Bai
Desbiderapenak %15 baino txikiagoak izan dira?	Bai
Arriskuak kontrolatuta egin dira?	Bai
Eskuraketak ongi egin dira?	Bai
Komunikazio-plana esan bezala egin da?	Bai
Proiektuaren interesatuak momentuko egoeraren berri izan dute?	Bai
Web-aplikazioa	
Bilaketa-eskaerek 6 segundo baino gutxiago irauten dute?	Bai
Emaitza bat bistaratzeko 5 klik baino gutxiago egin behar dira?	Bai
Emaitzak testuka banatzen dira?	Bai
Testu bateko emaitzak orrialdetan banatzen dira?	Bai
Interfazea moldagarria da gutxienez bi gailu ezberdinetarako?	Bai
Memoria	
Memoriak 75 orrialdetik gora ditu?	Bai
Edukia modu antolatu batean dago?	Bai
Irudi edota diagrama lagungarririk du?	Bai

6.2 Taula: Proiektuaren kalitate-zerrendaren egiaztapena.

Gainera, web-aplikazioaren azkartasunaren maximo onargarria 6 segundokoa zen arren, bilaketa gehienek 2 segundo baino gutxiago hartzen dute. Kontuan izan behar da egin diren probak hiru obra ez osatuekin egin direla, hau da, 147 orrialderekin; orrialde horietatik 3.747 indize ezberdin lortu dira, hots, 3.747 hitz ez-estandarren hitz estandarrak eta lemak lortu direla. Hala ere, bilaketa-zerbitzari bat erabili denez, ez da aparteko arazorik espero, baina kontuan izan beharko da erantzun-denborak handitu daitezkeela.

Interfazearen kalitateari dagokionez, emaitza guztiak antolatuta bistaratzen dira, testuetan banatuta eta testu bakoitzeko emaitzak orrialdeka banatuta.

Azkenik, memoriaren kalitate on bat lortu da, izan ere, azaltzeko gauza asko zeuden arren, irudi eta diagramen bitartez ulergarriagoa eta eramangarriagoa izatea lortu baitugu.

6.6 Eskuraketak

Hasiera batean planifikatu ziren eskuraketez gain, badira beste hiru software-baliabide eskuratu behar izan ditugunak.

- EKS edo *framework* bat erabili edo ez aztertu ostean erabaki genuena, *Bootstrap frameworka*.
- Bilaketa-zerbitzari bat erabiltzea erabaki genuenekoa, *Apache Solr*.
- *Apache Solr* eta web-aplikazioaren arteko lotura gauzatzeko, *Solarium* liburutegia.

6.7 Arriskuak

Plangintzako 2.7 atalean egin genuen arriskuen analisiari esker, ez dugu ezustekorik izan; web-aplikazioa garatzeko tresnak arazorik gabe eskuratu ahal izan ditugu eta aplikazioaren kodea galdu ez dugun arren, *GitHub* errepositorioan egon da sinkronizaturik.

Aipatu behar da memoriaren dokumentua galtzea ekiditeko ez dela hasieran esan bezala Dropbox erabili, baizik eta zuzenean hodeian garatu dela, *Overleaf* ingurunearen bitartez, nahiz eta aldian behin segurtasun-kopiak ere egin diren.

Hala ere, proiektuan atzerapena izan dugun arren, atzerapen hori zerk eragin duen aztertu dugu; oro har, arazoak teknikoak eta pertsonalak izan dira, baina mugarriak mugituz eta plangintza aldatuz, proiektua bide onetik eraman dugu, honen arrakasta lortuz.

7. KAPITULUA

Ondorioak

Gogora dezagun hasiera batean bete behar genuen helburu nagusia: hitzak normalizatzeko garatutako normalizazio-sistema integratu ahal izateko web-aplikazio baten prototipo bat sortzea. Helburu nagusiaz gain zehaztu ziren beste helburuak ere, berrerabilpenari lehentasuna eta garrantzia ematea esaterako, bete dira, nahiz eta plangintzarekin alderatuz desbideraketak egon diren. Plangintzan azaldu genuenez, proiektuak bete beharreko gutxieneko irismen-maila oinarrizko maila zen eta horixe izan da, hain zuzen ere, bete den irismen-maila.

Gradu Amaierako Proiektu hau Software Ingeniaritza espezialitatekoa izanik, arreta berezia eskaini diogu web-aplikazioa garatzeko erabilitako tresnen analisiari. Lehenik eta behin, testu historikoak dituzten plataformak aztertzeaz gain, bilaketa-zerbitzariren bat eta EKS edo *framework*en bat erabiltzeko aukerak ere aztertu ziren; azterketaren ostean, *Apache Solr* eta *Bootstrap* erabiltzea erabaki genuen. Bien arteko lotura egiteko, hau da, *Bootstrap* erabiliz garatutako web-aplikaziotik *Apache Solr* bilaketa-zerbitzariari eskaerak egiteko eta datuak indexatzeko *Solarium* liburutegia erabili dugu.

Behin erabiliko zen teknologia aukeraturik, API bitartez lortutako testuetako hitzak normalizatu behar izan genituen. Normalizazioa egin ahal izateko, *Phonetisaurus*en oinarritutako eta jada garatutako normalizazio-sistema bat erabili da eta, arazo puntualen bat izan arren, zuzendarietako bati esker, sistema horren garatzailetako bat izanik, arazoa konpontzea lortu genuen. Era berean, lematizatzaile bat gehitzeko aukera aztertu ondoren, transduktore bat erabiliz eta *Foma*ren *flookup* komandoaren bidez garatu dugu funtzionalitate hori, lemak erabiliz bilaketak egin ahal izateko.

Jatorrizko hitza, hitz estandarra eta lema hirukotea lorturik, bilaketa-zerbitzarian indexatu aurretik, eskemak zehaztu izan behar genituen. Hori izan zen *Apache Solr* erabiltzearen zailtasunetako bat, nodo bakoitzari zegokion eskema nahi genuen moduan konfiguratu ahal izatea.

Behin bilaketa-zerbitzaria konfiguratuta eta datuak izanik, pixkanaka erabilpen-kasu guztiak garatzen joan ginen, momentuan sor zitezkeen zalantzak foroetan kontsultatuz argituz.

Beraz, esan bezala, proiektuaren helburuak bete dira baina etorkizunean egin litezkeen zenbait lan eta proposamen egingo ditugu ondorengo atalean, ikasitako lezioak zein izan diren azaldu ostean.

7.1 Ikasitako lezioak

Proiektua aurrera eramateko graduan zehar eskuratutako gaitasunak ezinbestekoak izan dira, ez soilik irakasgaietan ikasitako programazio-lengoaiagatik edota analisi eta diseinuak egiten jakiteagatik, proiektuak aurrera eramateko moduagatik eta sor daitezkeen arazoei aurre nola egin jakiteko baizik.

Proiektuaren bizi-zikloan zehar motibazio-etapa ezberdinak izan dira; adibidez, hasiera batean, oraindik proiektua ongi finkatu gabe edo zehazki nola egin ez dakizunean, motibazioa gutxitu daiteke. Aldiz, garapena aurrera doan heinean, hau da, sortutako aplikazioa itxura hartzen hasten denean eta ordura arteko lana emaitzak ematen hasten denean, motibazioa handitzen da eta, are gehiago, egiten den lana eraginkorragoa da.

Hainbat lezio ikasi diren arren, hona hemen hurrengorako kontuan izango ditugunak:

- **Lanean konstantzia bat mantentzea**. Proiektu honetan bi geldialdi garrantzitsu egin behar izan dira, lau hilabete eta hiru hilabetekoak, hurrenez hurren. Geldialdi horiek lan erritmoa geratu dutenez, berriro proiektuari ekitea zaila izan da, beraz, hurrengo baterako kontuan izan beharko dugu dedikazio minimo bat mantentzea, ahal den neurrian behintzat.

- **Kodea berrerabiltzea**. Proiektuaren helburuetako bat zen berrerabilpenari garrantzia ematea; horri esker, lana errazteko tresna edota liburutegi asko jada sorturik daudela konturatu gara, nahiz eta nagusiki *Bootstrap* eta *Solarium* soilik erabili ditugun. Beraz, zalantzarik gabe, hurrengo proiektu bat hasi bezain laster, aztertu

beharko dugu ea garatu beharreko funzionalitatea jada inplementaturik dagoen edo garapenean laguntzeko liburutegi edota tresnarik dagoen, horrek ere dedikazioa gutxiagotzea eragin baitezake.

- **Memoriaren estimazioa doitzea**. Plangintzaren desbideraketarik handienetako bat izan da memoriaren estimazioarena, beraz, hurrengo proiektuetarako kontuan izango da lezio hau, memoriari dedikazio gehiago esleitzeko.

7.2 Etorkizunerako lanak eta aukerak

Hasiera batean aipatu genuen moduan, garatu den web-aplikazioa aurretik garatua izan den sistema baten aplikazio errealaren prototipo bat da, beraz, hobekuntza asko izan ditzake.

W3C estandarran jarraitu den arren, denbora dela medio, bada egin beharko litzatekeen baina azkenean egin ez den kontrol bat, softwarearen kodearen kalitate-kontrola. Beraz, komeniko litzateke beste edozein hobekuntza egiterako, kalitate-kontrol bat egitea. Hala ere, 6.5 atalean ikusi dugun moduan, aplikazioaren erantzun-denborak onak izan dira eta egin diren probatan ez du errorerik eman.

6.4 atalean esan dugun moduan, oinarrizko irismen-mailan geratu gara, beraz, etorkizunerako lan bat izan liteke irismen-maila aurreratua lortzea. Horretarako, plataformetatik testuak ausaz lortu beharko lirateke eta eskuz hitz ez-estandar bakoitzari dagokion estandarra emango litzaioke, ondoren ikasketa-prozesu berri bat gauzatzeko.

Beste aukera bat izan liteke hizkuntza desberdinetara moldatzea; hori egin ahal izateko, lorturiko testuen hizkuntzaren arabera normalizazio-sistema ezberdina aplikatu beharko litzateke. Euskalkiekin ere antzerako gauza bat egin liteke, baina agian testuen metadatuen laguntza beharko genuke, zein mendetakoa edo urtetakoa den jakinez gero, euskalkia identifikatu ahal izateko eta euskalki horri dagokion normalizazio-sistema aplikatzeko.

Azkenik, garatu den web-aplikazioa erabiltzaileentzat atzigarri uztea ere kontuan izan beharreko hobekuntza litzateke; horretarako, garapen-ingurunearen analisi bat egin beharko litzateke. Analisi hori egitean, ezinbestekoa izango da *Apache Solr*, *Phonetisaurus* eta *Foma* tresnekin bateragarriak izatea, tresna horiek instalatu behar izango baitira aplikazioaren funtzionamendu egoki bat ziurtatzeko.

Beraz, prototipo bat izanik, ikus daiteke nahi adina hobekuntza egin litezkeela.

Ea etorkizun batean proiektuari jarraipena ematen zaion!

Eranskinak

A. ERANSKINA

Proiektuaren kalitate-zerrenda

Proiektuaren kalitate-zerrenda	Bai/Ez
Proiektuaren kudeaketa	
Proiektuaren betekizun guztiak bete dira?	
Proiektuaren irismena bat dator egindako garapenarekin?	
Desbiderapenak %15 baino txikiagoak izan dira?	
Arriskuak kontrolatuta egin dira?	
Eskuraketak ongi egin dira?	
Komunikazio-plana esan bezala egin da?	
Proiektuaren interesatuak momentuko egoeraren berri izan dute?	
Web-aplikazioa	
Bilaketa-eskaerek 6 segundo baino gutxiago irauten dute?	
Emaitza bat bistaratzeko 5 klik baino gutxiago egin behar dira?	
Emaitzak testuka banatzen dira?	
Testu bateko emaitzak orrialdetan banatzen dira?	
Interfazea moldagarria da gutxienez bi gailu ezberdinetarako?	
Memoria	
Memoriak 75 orrialdetik gora ditu?	
Edukia modu antolatu batean dago?	
Irudi edota diagrama lagungarririk du?	

A.1 Taula: Proiektuaren kalitate-zerrenda.

B. ERANSKINA

Bilera-aktak

Atal honek proiektuan zehar egin diren bileretako aktak biltzen ditu.

1. Bilera-akta

Data: 2015-02-03

Kideak: Izaskun Etxeberria, Iñaki Alegria eta Jon Ander Belamendia

Lekua: Informatika Fakultatea, Donostia

Ordua: 15:00

Hitz egindakoa:

- Proiektuaren nondik norakoak zehaztea:
  - Proiektuaren ikuspegi orokorra.
  - Proiektuaren helburuak azaldu.

Egin beharrekoa:

- Plangintzaren atazak zehazten hasi.
- WIkisource, Iturriak eta Armiarmaren Klasikoen Gordailua aztertu eta bat aukeratu garatuko den aplikazioarekin erabili ahal izateko.

2. Bilera-akta

Data: 2015-03-23

Kideak: Izaskun Etxeberria, Iñaki Alegria eta Jon Ander Belamendia

Lekua: Informatika Fakultatea, Donostia

Ordua: 15:00

Hitz egindakoa:

- Plangintza egina dago.
- Wikisource eta Iturriak plataformek API berdina dutenez, biak erabiliko direla erabaki da.
- Aplikazioaren erabilpen-kasuen azalpena.

Egin beharrekoa:

- Aztertu ea plataforma horiek bilatzailerik duten eta hitzak markatzeko aukera eskaintzen duten.
- APIaren erabilera ikusteko oso oinarrizko dei batzuk probatu, ea beharrezko informazioa lortzeko arazorik dagoen.

3. Bilera-akta

Data: 2016-04-28

Kideak: Izaskun Etxeberria, Iñaki Alegria eta Jon Ander Belamendia

Lekua: Informatika Fakultatea, Donostia

Ordua: 17:00

Hitz egindakoa:

- Plataformek ez dute baliogarri izango zaigun bilatzailerik eta hitzak markatzeko aukerarik ere ez dute eskaintzen.
- Oinarrizko deiak erakutsi dira eta beharrezko testuak lor ditzakegu.

Egin beharrekoa:

- Bilaketa-zerbitzariren bat erabiltzeko aukera aztertu.
- Faksimileak lortzeko, API bidez ezin badira lortu, *web scraping* metodoa aztertu.

4. Bilera-akta

Data: 2015-06-30

Kideak: Izaskun Etxeberria, Iñaki Alegria eta Jon Ander Belamendia

Lekua: Informatika Fakultatea, Donostia

Ordua: 16:00

Hitz egindakoa:

- Plangintza, analisia eta diseinua atalak amaitu dira.
- Ikaslea lanean hasia denez eta, beraz, proiektuari dedikatzeko denborarik ez duenez, proiektuarekin urriatik aurrera jarraitzea erabaki da. Erabaki honek proiektuaren garapenean eragin handia duenez, ikasleak zuzendariak aldez aurretik jakinaren gainean jarri ditu.
- Behin lana amaiturik, proiektuari jarraipena emango zaio urriatik aurrera, mugarri berriak zehaztu beharko dira eta plangintza berrikusi eta, beharrezkoa bada, aldatu beharko da.

Egin beharrekoa:

- Urriatik aurrera, plangintza berrikusi beharko da.

5. Bilera-akta

Data: 2015-11-03

Kideak: Izaskun Etxeberria, Iñaki Alegria eta Jon Ander Belamendia

Lekua: Informatika Fakultatea, Donostia

Ordua: 15:00

Hitz egindakoa:

- Ikasleak plangintza berrikusi eta mugarri berriak ezarri ditu; hurrengo deialdia urtarrilean izanik eta hiru hilabete eskas soilik geratzen direnez, saiakera egingo da proiektua urtarrilean entregatu ahal izateko.
- Aurretik e-posta bitartez kontaktua izan denez, zuzendariek pasatutako Foma eta Phonetisaurus tresnak aztertu eta instalatu dira, hitzak normalizatu eta lematizatu ahal izateko.
- Aurretik bilaketa-zerbitzarien azterketa egina denez, zuzendariek proposatu dute Apache Solr erabiltzea.

Egin beharrekoa:

- Apache Solren eskemak zehazteko pausoak aztertu, zein eskema jarraitu behar den jakiteko.
- Testu bat eskuz lortu eta bertako hitzak normalizatzen eta lematizatzen saiatu.

6. Bilera-akta

Data: 2015-11-20

Kideak: Izaskun Etxeberria, Iñaki Alegria eta Jon Ander Belamendia

Lekua: Informatika Fakultatea, Donostia

Ordua: 15:00

Hitz egindakoa:

- Web-aplikazioaren analisia eta diseinua egin dira.
- Apache Solr ulertzeko eta, batez ere, eskemak non eta nola sortu zehaztea kosta egin da, baina lortu da.
- *Web-scrapping* erabili gabe lortu da faksimileak lortzea, APIen bitartez.
- Hitz ez-estandarrei dagozkien hitz estandarrak lortu dira.
- Web-aplikazioa garatzeko aukera ezberdinak aztertu ostean, ikasleak Bootstrap frameworka erabiltzea proposatu du eta zuzendarien baieztapena lortu da.
- Garatuko den web-aplikazioaren eta Apache Solr bilaketa-zerbitzariaren arteko konexioa egiteko eta bilaketak gauzatzeko, ikasleak Solarium liburutegia erabiliko duela jakinarazi du.

Egin beharrekoa:

- Interfazearen diseinua egin ondoren, web-aplikazioaren egitura orokorra zehaztu Bootstrap erabiliz.
- Solarium liburutegia erabiliz, aplikazioaren eta Apache Solren arteko lotura burutu eta probak egin.

7. Bilera-akta

Data: 2015-12-18

Kideak: Izaskun Etxeberria, Iñaki Alegria eta Jon Ander Belamendia

E-postaz eginiko komunikazioa.

Hitz egindakoa:

- Ikasleak Apache Solren eta web-aplikazioaren arteko lotura burutu du.
- API dei asko ez egitearren eta, Wikisourcek gomendatu moduan, deiak bateratu dira, kontsulta bakar batean 50 orrialde lortu ahal izateko.

Egin beharrekoa:

- APIak erabiliz testuak automatikoki lortu eta, bertako hitzak normalizatu eta lematizatu ondoren, Apache Solreko indizetan jaso.
- Orain artekoaren jarraipen eta kontrolari berrikusketa bat egin, desbiderapenak zein diren eta irismen-maila aurreratua lor daitekeen jakiteko.

8. Bilera-akta

Data: 2016-01-12

Kideak: Izaskun Etxeberria, Iñaki Alegria eta Jon Ander Belamendia

E-postaz eginiko komunikazioa.

Hitz egindakoa:

- Jarraipen eta kontrola egin ostean, eta memoriaren atal dezente falta direnez garatzeko, ekaineko deialdian aurkeztu beharko da.

Egin beharrekoa:

- Memoria amaitzen joan eta zuzendariei pasa, irakurtzen joan daitezen.

9. Bilera-akta

Data: 2016-02-09

Kideak: Izaskun Etxeberria, Iñaki Alegria eta Jon Ander Belamendia

E-postaz eginiko komunikazioa.

Hitz egindakoa:

- Arazo pertsonalak direla-eta, ikasleak ezingo du proiektuarekin jarraitu.

Egin beharrekoa:

- Ahal bezain laster, proiektuari ekingo dio berriro, memoria amaitu ahal izateko.

10. Bilera-akta

Data: 2016-05-12

Kideak: Izaskun Etxeberria, Iñaki Alegria eta Jon Ander Belamendia

E-postaz eginiko komunikazioa.

Hitz egindakoa:

- Ikasleak proiektuari ekingo dio berriro, memoria ahalik eta azkarren amaitzeko, ekainean entregatu ahal izateko.

Egin beharrekoa:

- Memoria amaitzen joan eta zuzendariei pasa, irakurtzen joan daitezen.

11. Bilera-akta

Data: 2016-06-02

Kideak: Izaskun Etxeberria, Iñaki Alegria eta Jon Ander Belamendia

Lekua: Informatika Fakultatea, Donostia

Ordua: 16:00

Hitz egindakoa:

- Zuzendariek memoriaren zati baten lehen irakurketa egin dute eta hobekuntzak proposatu dituzte.

Egin beharrekoa:

- Memoriarekin amaitu, hurrengo zuzenketa egin ahal izateko.
- Proposatutako hobekuntzekin memoria hobetu.

12. Bilera-akta

Data: 2016-06-09

Kideak: Izaskun Etxeberria, Iñaki Alegria eta Jon Ander Belamendia

Lekua: Informatika Fakultatea, Donostia

Ordua: 16:00

Hitz egindakoa:

- Zuzendariek memoriaren bigarren irakurketa egin dute eta hobekuntzak proposatu dituzte.
- Defentsa-eskaera egiteko onarpena eman dute zuzendariek.

Egin beharrekoa:

- Proposatutako hobekuntzekin memoria hobetu.

13. Bilera-akta

Data: 2016-06-14

Kideak: Izaskun Etxeberria, Iñaki Alegria eta Jon Ander Belamendia

Lekua: Informatika Fakultatea, Donostia

Ordua: 16:00

Hitz egindakoa:

- Zuzendariek memoriaren azken irakurketa egin dute eta ontzat eman dute, baina formatu kontu batzuk moldatu behar dira.

Egin beharrekoa:

- Memoriaren formatu kontuak moldatu.
- Defentsarako aurkezpena prestatu.

14. Bilera-akta

Data: 2016-06-21

Kideak: Izaskun Etxeberria, Iñaki Alegria eta Jon Ander Belamendia

Lekua: Informatika Fakultatea, Donostia

Ordua: 16:00

Hitz egindakoa:

- Memoriaren azken bertsioa ontzat eman dute, beraz, ADDIra igotzeko prest dago.

Egin beharrekoa:

- Memoria ADDIra igo.
- Defentsarako aurkezpena prestatu.

Bibliografia

[1] Solarium dokumentazioa. http://solarium.readthedocs.io/en/stable, .

[2] Apache Solr Reference Guide. https://cwiki.apache.org/confluence/display/solr/Apache+Solr+Reference+Guide, .

[3] Izaskun Etxeberria, Iñaki Alegria, Larraitz Uria, and Mans Hulden. Evaluating the noisy channel model for the normalization of historical texts: Basque, spanish and slovene. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, may 2016.

[4] Josef R. Novak, Nobuaki Minematsu, and Keikichi Hirose. WFST-Based Grapheme-to-Phoneme Conversion: Open Source tools for Alignment, Model-Building and Decoding. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pages 45–49, Donostia–San Sebastian, July 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W12-6208.

[5] Josef R. Novak, Nobuaki Minematsu, and Keikichi Hirose. Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the WFST framework. *Natural Language Engineering*, pages 1–32, 2015.