eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Ingeniaritzako Gradua

Software Ingeniaritza

Gradu Amaierako Proiektua

Balidazio-eskemetan oinarritutako interfaze-sorkuntza dinamikoa

Egilea

Jagoba Gascón Sánchez

informatika fakultatea
facultad de informática

2013

Laburpena

Gradu-amaierako proiektu hau anotazio linguistiko konplexuen testuinguruan kokatzen da. Proiektuaren helburuak hiru zatitan bereizi dira. Lehena, Armiarma web-aplikazioa eta *LibiXaML* liburutegia egokitzea bilaketa konplexuagoak onartu ahal izateko; horretarako Relax NG eskemen erabilpena errazten duen JavaScript liburutegia garatu da. Ondoren, *LibiXaML* liburutegiak bilaketarako erabiltzen dituen XPath-en sorkuntza automatizatzea, Relax NG eskematik abiatuz. Eta amaitzeko, analisi-sorkuntzako prototipoa hobetzea, hasieran sortutako liburutegia berrerabiliz eta Relax NG eskemak erabiliz.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

Kode-zatien aurkibidea

1. KAPITULUA

Sarrera

Dokumentu hau 2013. urtean Jagoba Gascón Sánchez-ek, UPV/EHU unibertsitateko Donostiako Informatika Fakultatean Informatika Ingeniaritzako Graduan ikasle denak, egindako Gradu Amaierako Proiektuari dagokio. Proiektuaren zuzendariak Aitor Soroa Etxabe eta Xabier Artola Zubillaga, UPV/EHUko Donostiako Informatika Fakultateko irakasleak, izan dira.

Proiektua IXA taldearen barruan garatu da. IXA taldea Euskal Herriko Unibertsitateko Donostiako Informatika Fakultatean kokatzen den ikerketa talde bat da. Informatikariz eta hizkuntzalariz osatuta dago eta Hizkuntza Naturalaren Prozesamenduaren arloa jorratzen du.

Proiektu honetan Armiarma izeneko anotazio linguistiko konplexuak kudeatzeko web-aplikazio batean hobekuntzak eta funtzionalitateak gehitu dira. Aplikazio hau Euskaltzaindiaren Lexikoaren Behatokia egitasmoan kokatzen da, eta bere helburu nagusia corpus[1] ezberdinekin osatutako anotazio-amaraun konplexu baten gainean hizkuntza-kontsultak eta eguneraketak egiteko aukera ematea da.

Proiektuaren helburua web-aplikazioan funtzionalitateak gehitzea izan da. Funtzionalitate horiek garatzeko Relax NG[2] eskema bat izan da oinarri eta abiapuntua. Proiektuak

[1] Corpusak testu-dokumentu bilduma handiak dira, hizkuntzaren analisia egiteko erabiliak.
[2] REgular LAnguage for XML Next Generation. XMLrako eskema-lengoaia da.

hiru atal izan ditu:

Lehena, interfazearen egokitzea. Aplikazioak bilaketa-sistema eskasa zuen, kodean bertan idatzita zeuden bilaketa-eremu gutxi batzuk baino ez zituen onartzen, eta ondorioz, bilaketa-sisteman aldaketak egin nahi izanez gero kodea moldatu behar zen. Proiektu honetan dinamikoki sortutako interfazea garatu dugu, bilaketa-sistema aberatsa eta eguneratua mantenduko duen interfazea lortuz.

Bigarrena, eskema batetik abiatuz kontsultan erabiltzen diren XPath[3]-en sorkuntza automatikoa garatzea. XPath horiek denboran zehar aldaketak izateko arriskua zela-eta, automatikoki lortzeko bidea bilatu behar zen.

Azkenik, erabiltzaileei analisi berrien sorkuntza-sistema eskaintzea. Aplikazioaren helburuetako bat analisi linguistiko berriak sortzea zen; hasieran, ordea, funtzionalitate hori ez zegoen inplementatua.

[3] *XML Path Language*. XML dokumentu bat prozesatzeko espresioak eskaintzen dituen lengoaia.

2. KAPITULUA

Testuingurua eta aurrekariak

Proiektu guztia aurrera eramatea ezinezkoa izango zen aurretik egindako lana erabili ezean. Aurretik IXA taldeak lan asko egin du lengoaia naturalaren prozesamendurako tresnak garatzen gehienbat. Bestalde, proiektua prototipo batetik abiatuz egin da, ikasle ohiek eta IXA taldeak garatutakoa alegia. Atal honetan aurretik egindako lan hori guztia aztertuko dugu.

2.1 Testuingurua

Proiektuaren testuingurua osatzen duten tresnak azalduko dira atal honetan, IXA taldeak analisi linguistikoak errepresentatzeko erabiltzen duen anotazio-amarauna eta analisi linguistikoak aztertuko ditugu. Baita proiektu honen oinarri izan diren Relax NG analisien eskemak ere.

2.1.1 Anotazio-amarauna

Anotazio-amarauna IXA taldean analisi linguistikoak errepresentatzeko erabiltzen den eta XML teknologian oinarritzen den datu-eredua da, lengoaia-prozesatzaile ezberdinek sortutako informazioa errepresentatu, irakurri eta gordetzeko erabilia.

Testu bat hainbat prozesutatik igaro ondoren corpus-unitate bat sortzen da. Honetan jatorrizko testuaz gain prozesuen ondorioz sortutako anotazio-amarauna izango dugu. Prozesu bakoitzak sarrera bat eta irteera bat izango ditu, eta bakoitzaren irteera hurrengoaren sarrera izango denez, ezinbestekoa da tresnekiko independentea den datu-eredua erabiltzea. Hori da anotazio-amaraunaren helburua, lengoaiaren tratamendua egiteko prozesuan erabiliko diren erremintengandik independentea den eredu estandarra definitzea.

Amaraunaren datu-ereduak anotazio-eskema bat jarraitzen du. Eskemaren arabera sarrerako testua eta berari lotutako informazio linguistikoa bereizita adierazten dira, bien arteko erlazioa ere beste egitura batean gordetzen da. Sarrerako testua aingurez osatua dago. Testuaren elementu bakoitzari erreferentzia egingo dion aingura egongo da dokumentu hauetan, eta aingura bakoitza bere informazio linguistikoa duen egiturari lotua egongo da esteka baten bitartez. Datu-egitura honi esker bi aingura ezberdinek analisi linguistiko bera izan dezakete, datuen erredundantzia murriztuz.

Esan bezala, hainbat prozesuren ondoren anotazio linguistikoak lortzen ditugu; anotazio linguistikoak elkarri lotutako XML dokumentuen amaraun batean datza. Amaraun hori hiru dokumentu mota bidez errepresentatzen da:

- **Aingurak:** Corpuseko hitz fisikoak nahiz aurreko prozesuetan lortutako anotazioak izan daitezke. Normalean, testuetako hitz fisikoetatik lortutako interpretazio linguistikoak erabiltzen dira aingura moduan.

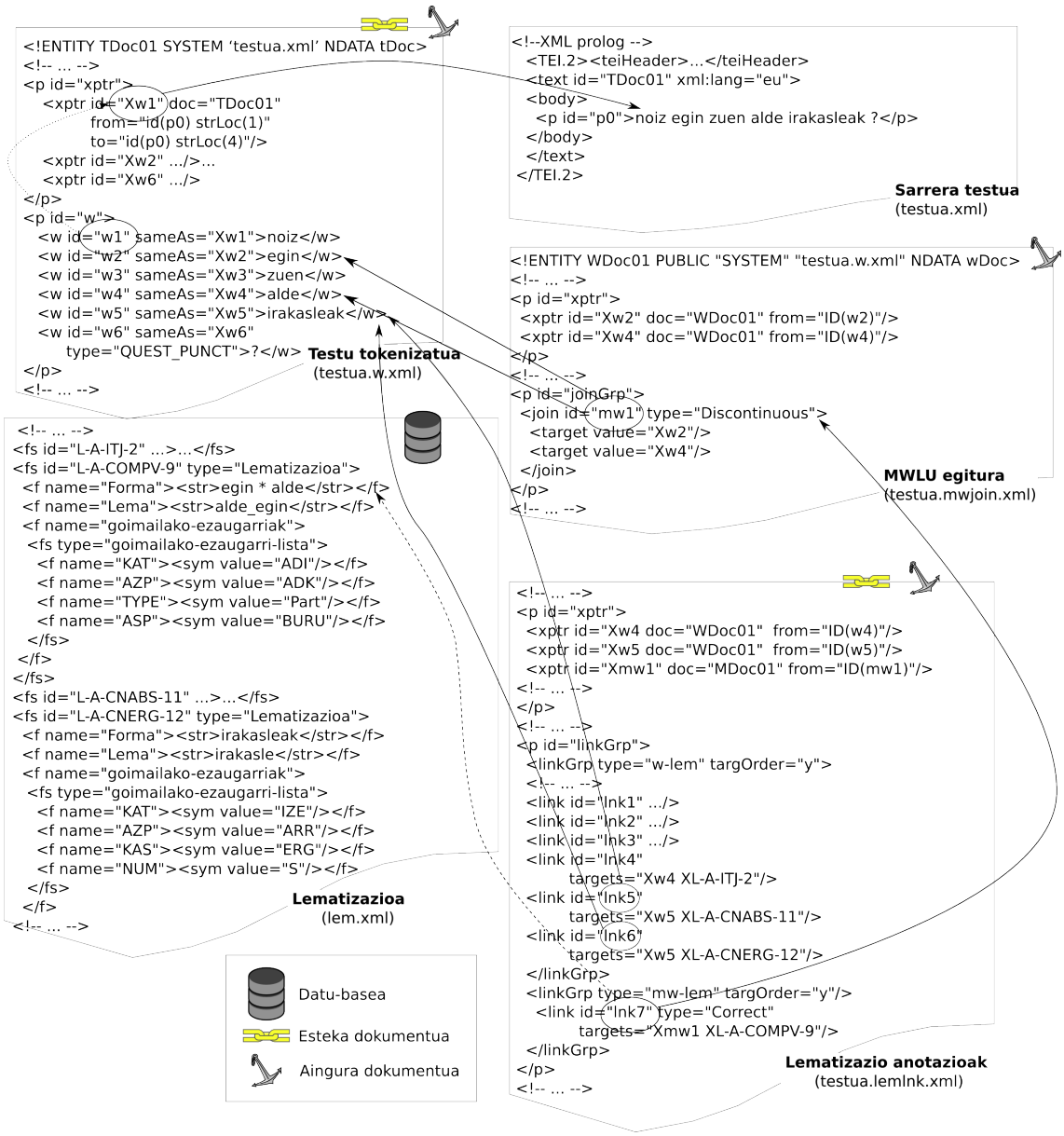
- **Anotazio-estekak:** Esteka bakoitzak interpretazio linguistiko bat aingura batekin lotzen du. Interpretazio-anbiguotasuna, hau da, hitz batek interpretazio ezberdinak izatea, esteka ezberdinak aingura berean lotuz errepresentatzen da, interpretazio linguistiko asko dituelarik. Desanbiguazioa, ordea, esteka horietako bat zuzena dela adieraztean datza, besteak alde batera utziz.

- **Informazio linguistikoa:** Analisi-prozesuan lortutako analisi linguistikoak dira. TEI-P4[1] gidalerroak jarraitzen dituzte analisiek. Dokumentu hauetan, aurrerago azalduko ditugun, XML egitura duten FS egiturak gordetzen dira.

[1] *TEI (Text Encoging Initiative) gidalerroak:* kasu honetan, anotazio linguistikoak kodetzeko XML formatu bat definitzen dute.

Hurrengo irudian, lematizazio-anotazio linguistikoak lortzeko pausoak azaltzen dira: lehenik, sarrera-testua tokenizatzen da, sarrera-testuko hitzak eta puntuazio-markak identifikatuz; ondoren, MWLU egitura delakoan, hitz anitzez osatutako tokenak identifikatzen dira: gure adibidean *egin* eta *alde* hitzek *alde egin* aditza osatzen dute. Azkenik, testuaren lematizazio-analisia egiten da.

2.1 Irudia: Anotazio linguistikoak: amarauna.



2.1.2 Analisi linguistikoak

Feature Structures

FSak (Feature Structure) helburu ezberdina duten datu-egiturak dira; hauetan ezaugarriak banaka nahiz multzoka azaltzen dira, TEIk definitutako XML egitura erabiliz. Hona hemen adibide bat:

2.1 Kode-zatia: Analisi linguistikoak.

```
<fs id="L-A-IZE-ARR-1994" type="lematizazioa">
    <f name="forma">
        <str>idatzitako</str>
    </f>
    <f name="lema-osatua">
        <str>idatzi</str>
    </f>
    <f name="ezaugarri-morfologikoak">
        <fs type="goimailako-ezaugarri-lista">
            <f name="AZP">
                <sym value="ARR"/>
            </f>
            <f name="KAS">
                <sym value="ABS"/>
            </f>
            <f name="KAT">
                <sym value="IZE"/>
            </f>
            <f name="ADZ">
                <fs type="Gako">
                    <f name="Sarrera">
                        <str>adz_sarrera</str>
                    </f>
                    <f name="Homografo-Id">
                        <nbr value="123"></nbr>
                    </f>
                </fs>
            </f>
        </fs>
    </f>
</fs>

<fs id="M-A-IZE-ARR-230" type="analisia">
    <f name="forma">
        <str>aizkolariak</str>
    </f>
    <f name="lema-osatua">
        <str>aizkolari</str>
    </f>
    <f name="goimailako-ezaugarriak">
        <fs type="goimailako-ezaugarri-lista">
            <f name="KAT">
                <sym value="IZE"/>
            </f>
            <f name="AZP">
                <sym value="ARR"/>
            </f>
            <f name="BIZ">
                <plus/>
            </f>
```

```
            <f name="FSL" org="list">
                <sym value="@OBJ"></sym>
                <sym value="@PJ"></sym>
            </f>
        </fs>
    </f>
</fs>
```

Adibidean mota ezberdineko bi FS azaltzen dira. Ikus daitekeen bezala bakoitza analisi linguistiko ezberdin bati dagokio, lehena *"idatzitako"* hitzari dagokion lematizazio bat da, eta bigarrena *"aizkolariak"* hitzaren analisi morfosintaktikoaren emaitza bat.

FS bakoitzak bi atributu ditu: identifikadorea, bakarra FS guztien artean, eta mota. FS bakoitzak F (*feature* edo ezaugarri) zerrenda bat du bere barruan. Bi kasuetan, lema[2], forma[3] eta ezaugarri morfologikoak azaltzen dira. Ezaugarri morfologikoak ezaugarrien zerrenda bat da; honetan, kategoria (KAT), azpikategoria (AZP), kasua (KAS), mugatasuna (MUG) eta biziduna (BIZ) ezaugarriak eta balioak ikus daitezke.

Analisi linguistikoen eskemak: Relax NG

Proiektu honetan Relax NG (RNG moduan ere ezaguna) eskemek garrantzi handia izan dute. Relax NG (REgular LAnguage for XML Next Generation) XML baten egitura definitzeko eta balidatzeko balio duen eskema-lengoaia da. Lengoaia sinplea eskaintzen du eta, XML sintaxiaz gain, XML ez den beste sintaxi konpaktu bat ere bai.

Analisi linguistikoen egitura definitzeko erabiltzen da Relax NG eskema, hau da, analisia ondo eratua[4] eta baliozkoa[5] dela ziurtatzeko. Hona hemen Relax NG eskema sinple baten adibidea:

2.2 Kode-zatia: Liburu baten Relax NG eskema

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <start>
        <element name="f">
            <attribute name="name">
```

[2]Unitate semantiko bat osatzen duten karaktere multzoa da, hitzaren esanahia gordetzen du. *Idatzitako*, *idatzita* edo *idazten* moduko hitzek lema bera dute: *idatzi*.

[3]Testuan agertzen den hitza bera; lema batek forma ezberdin asko izan ditzake.

[4]XML sintaxia jarraitzen duen dokumentua da.

[5]Ondo eratua egoteaz gain, eskema baten erregelak jarraitzen dituen eskema.

```
                <value>ADOIN</value>
            </attribute>
            <element name="str">
                    <data type="string"/>
                </element>
        </element>
    </start>
</grammar>
```

Relax NG eskemeak, beste lengoaia batzuek ez bezala[6], definitu nahi den egituraren hasiera markatzen du `start` nodoaren bitartez. Hala ere, egitura guztiak ez du honen barruan egon behar, elementu bat definitu eta erreferentziaz deitu baitaiteke. Hona hemen aurreko eskemaren baliokidea:

2.3 Kode-zatia: Liburu baten Relax NG eskema baliokidea

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
    <start>
        <element name="f">
            <attribute name="name">
                <value>ADOIN</value>
            </attribute>
            <ref name="mota.str"/>
        </element>
    </start>

    <define name="mota.str">
        <element name="str">
            <data type="string"/>
        </element>
    </define>
</grammar>
```

Proiektuan eragin handia izango duten Relax NG eskemen hainbat ezaugarri deskribatuko ditugu: Kardinalitatea errepresentatzeko hainbat nodo eskaintzen ditu: `zeroOrMore`, `oneOrMore` eta `optional`, hau da, zero edo gehiago, bat edo gehiago eta zero edo bat; ez dago nodo kopuru jakin bat nahi dugula esateko aukerarik; bestalde, elementu multzo batetik bat aukeratzeko `choice` nodoa erabiltzen da.

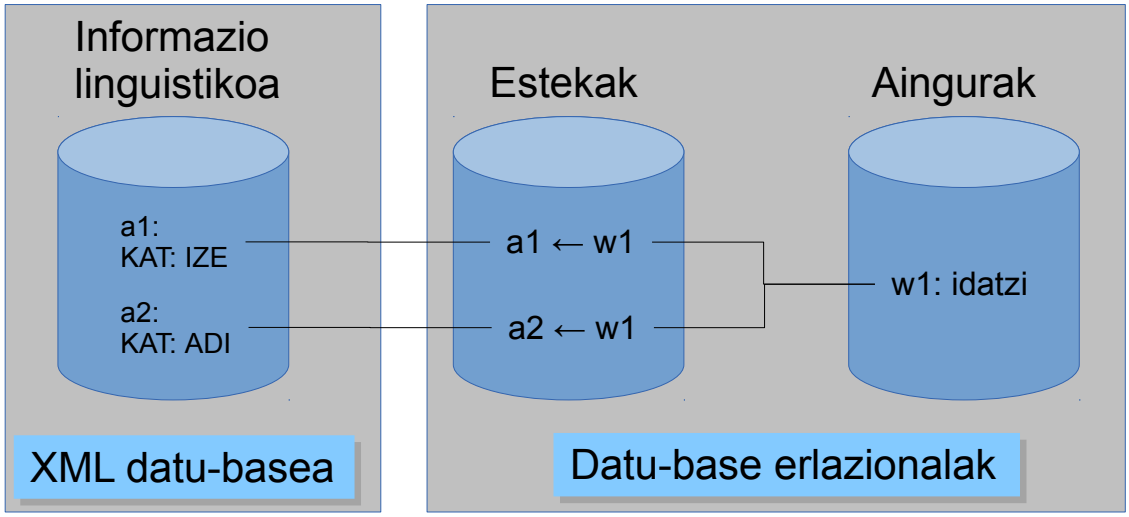
2.1.3 Informazioaren biltegiratzea

Esan dugu informazio guztia hiru fitxategi motatan banatzen dela: tokenak, analisi linguistikoak eta bien arteko lotura egiten duten estekak. Sistema hau ez da oso erabilgarria,

[6] XMLSchema-k, adibidez, elementu bakoitza indibidualki balidatzen du.

ordea, testu multzo handiak kontsultatu nahi denean; horregatik datu-baseetan gordetzea erabaki zuten, datu-base erlazional batean esteka eta aingurak, eta XML datu-base batean analisi linguistikoak.

2.2 Irudia: Hitzen eta analisien arteko erlazioa datu-baseetan.



2.2 Aurrekariak

Esan bezala proiektua hastean prototipo[7] bat garatuta zegoen, Armiarma kontsulta-tresna. Atal honetan prototipo hori, eta *LibiXaML* liburutegia aztertuko ditugu.

2.2.1 Armiarma kontsulta-tresna

Armiarma kontsulta-tresna webgune batek eta C++ lengoaiaz idatzitako modulu batek osatzen dute. Armiarma anotazio linguistikoak kudeatzeko web-aplikazioa da. Funtzionalitate nagusia eta proiektu honetan eragin handiena izango duena bilaketa da; horregatik horretan gehiago sakonduko dugu.

Kontsultak lau atal nagusi ditu: lehenak bi parametro ditu, bat segmentazio, morfosintaxi edo lematizazio mailan lan egiteko, eta bestea analisi linguistiko zuzenak edo guztiak bilatzeko. Beste hiru zatietan bilaketa-eremuak definitzen dira: lehen zatian bilatu nahi dugun hitza definitzen da eta beste bietan berorren testuingurua definitzen da.

[7] Mikel Astiz, Joseba Alberdi eta Zuhaitz Beloki, 2009 - 2011

Hitza bera eta hainbat ezaugarri erabiltzen dira bilaketan: lema edo forma den, kategoria, azpikategoria eta beste ezaugarri batzuk. Bilaketaren emaitza KWIC[8] moduan ematen da, eta nahi izanez gero hitzaren analisi linguistikoak ikus daitezke.

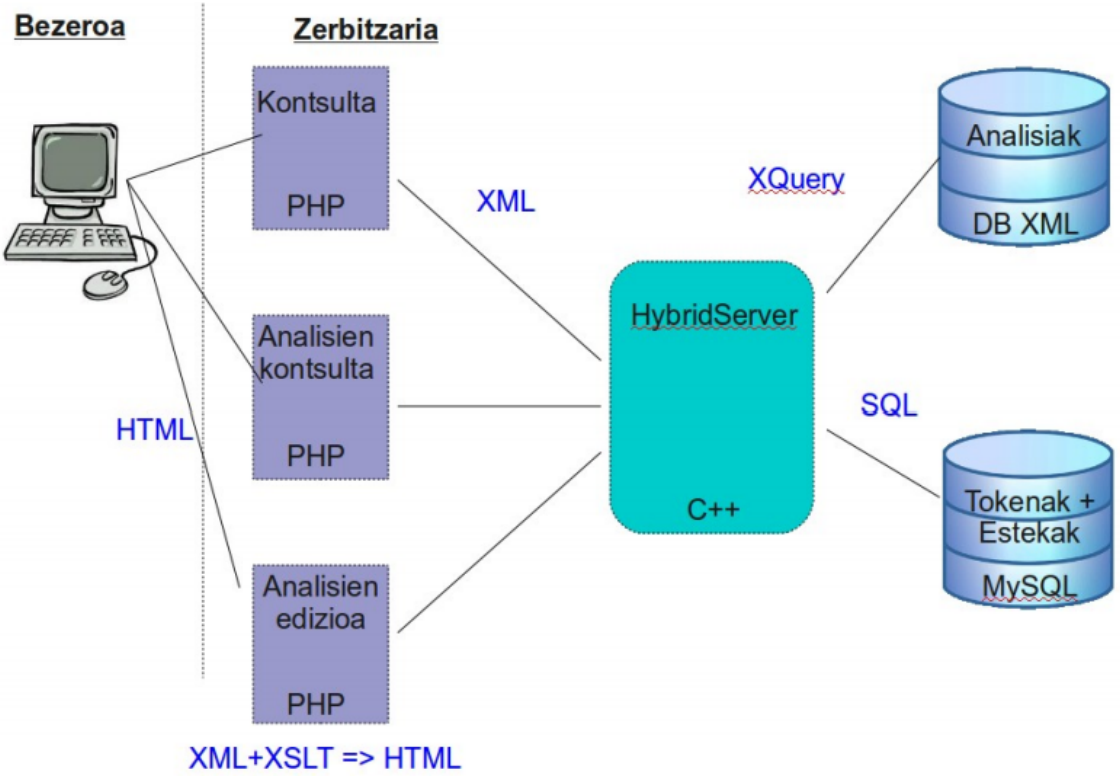
2.3 Irudia: KWIC moduko emaitzaren eredua.

- eta kaleratutako adierazpenean . Idatzian , euskarak bateratzailea izan behar
horiek , ez baitziren inon idatziak . Legeak zioen langileak frogatu
bere bizipenen liburua ari da idazten . CD batean gorde du

Arkitektura

Aplikazioak *front end/back end* motako arkitektura du. *Front end* delako azaldu berri dugun Armiarmaren interfazea da; *back end* zatia, ordea, *HybridServer* izeneko C++ lengoaiaz idatzitako moduluak eta aurrerago azalduko dugun *LibiXaML* liburutegiak osatzen dute.

2.4 Irudia: Aplikazioaren arkitektura. Iturria: Zuhaitz Belokiren KBPa, 2011.



[8] Key Word In Context. Bilaketa-baldintzak betetzen dituen elementu bakoitza, bere testuinguruan kokatuta, itzultzean datza.

HybridServer moduluak hiru zerbitzu eskaintzen ditu: kontsulta, analisien eskaera eta desanbiguazioa. Armiarmak web-aplikaziotik bilaketa-parametroa jaso eta XML motako erantzuna emango du, non bilaketa-hitza eta haren ingurukoak egongo diren. *Front end*-ak hau jaso eta formatu emango dion XSLT[9] *script* bat aplikatzen dio pantailan datuak erakutsi baino lehen.

2.5 Irudia: Bilaketa baten emaitzaren adibidea.

Aplikatutako transformazioak, formatu emateaz gain, hitz gakoetan estekak sortzen ditu; esteka batean klik egitean, hitz horren analisiak eskatzen zaizkio *HybridServer* moduluari. Halere ez du merezi honetan sakontzea proiektuan zehar ez delako aldatuko. Lehen aipatu dugu Armiarma aplikazioak hiru funtzionalitate dituela: kontsulta, analisien ikuskatzea eta analisien sorkuntza. Lehen biak azalduta, hirugarrena azaltzeko unea da.

Analisien sorkuntzaren prototipo txiki bat baino ez dago web-aplikazioan. Prototipo honetan, Relax NG eskemak sekuentzialki irakurtzen dira eta ezaugarriak aurkitu ahala erabiltzaileari balioa sartzeko eskatzen zaio, betiere kontuan izanik ezaugarriak aukerazkoak (*optional*) edo beharrezkoak diren, edo testu hutsa edo zerrenda batetik balio bat jasotzen duten.

[9] *Extensible Stylesheet Language Transformations:* W3C erakundearen XML dokumentuak eraldatzeko estandarra da.

2.2.2 *LibiXaML*

LibiXaML anotazio-amarauna osatzen duten fitxategi guztiak maneiatzeko liburutegia da. C++ lengoaiaz idatzia, fitxategi nahiz datu-baseekin lan egiteko prestatua dago. Aipatu dugu anotazio-amarauna bi datu-base motatan banatzen dela, datu-base erlazionala eta XML datu-basea. *LibiXaML* liburutegiak, oraingoz, XML datu-baseekin bakarrik egin dezake lan, baina etorkizunean erlazionalekin ere lan egin ahal izatea espero da.

LibiXaML liburutegia ehun klasetik gora osatzen dute baina guri gehien interesatzen zaiguna analisi linguistikoak adierazteko erabiltzen den FS klasea da. Klase honek, beste gauza askoren artean, bilaketa-parametroak jaso eta XPath kontsulta bat sortzen du; hori oso garrantzitsua izango da proiektu honetan, interfazearen egokitzea egiterakoan aldaketak egin beharko zaizkiolako.

3. KAPITULUA

Proiektuaren helburu-dokumentua

Atal honetan proiektuaren planifikazioa azalduko dugu. Proiektuaren irismena, denboraren kudeaketa, kalitate-plana eta komunikazio-plana azalduko ditugu, besteak beste.

Proiektu honek hiru zati izango ditu. Hirurak linealki gauzatuko dira proiektuak irauten duen bitartean; beraz, zailtasun handia izango dugu bigarren eta hirugarren zatien planifikazio zehatza egiteko, izan ere, zati bakoitzak ikasketa, diseinu eta garapen independentea izango du. Horregatik, planifikazioa hirutan banatzea erabaki da, eta zati bakoitza amaitzean hurrengoaren planifikazioa egin beharko da.

3.1 Proiektuaren irismena

Atal honetan proiektu honen irismena zehaztuko dugu, betekizunak eta murriztapenak definituz. Lanaren deskonposaketa-diagrama eta lan-paketeen deskribapena ere azalduko ditugu atal honetan.

Proiektuaren helburua Armiarma web-aplikazioa hobetzea da. Horretarako honen gainean hainbat aldaketa egingo dira, amaitu gabe dauden funtzionalitateak gehituz, eta daudenak hobetuz.

Proiektua hiru zatitan banatuko da. Lehenik, bilaketa-sistema aberastuko da, Relax NG eskema batetik abiatuz bilaketa-eremuak interfazean gehitu beharko dira. Ondoren bilaketak erabiltzen dituen XPath espresioak era errazean lortzeko sistema burutu beharko da, eskuz egiteak lan asko eskatzen duelako. Eta azkenik, eskematik abiatuz erabiltzaileek analisi linguistiko berriak sortu ahal izateko aplikazioa moldatu beharko dugu.

Proiektuaren hiru atalak ez dira independenteak, hirurak estuki lotuta daude, izan ere, hiruretan analisi linguistikoak definitzeko erabiltzen diren Relax NG eskemen erabilera aurreikusten da. Gainera, bilaketa-interfazearen sorkuntzan egindako lan handia analisien sorkuntzan berrerabiltzea espero da.

3.1.1 Betekizunak

Esan bezala, proiektua hiru atal nagusitan banatzen da:

- Web-aplikazioak egin ditzakeen bilaketak aberastu; horretarako, bilaketa-eremu gehiago gehitu beharko dira, eremu horiek Relax NG eskema batetik lortuz.
- Anotazio-amaraunean bilaketak egiteko erabiltzen diren XPath-ak automatikoki sortzea, Relax NG eskema erabiliz.
- Analisi linguistikoen sorkuntza, eskematik abiatuz.

Atal bakoitzaren betekizunak aztertuko ditugu orain.

Eskematik abiatuz Armiarmaren bilaketa-sistema aberastu

Helburua oraingo sistema hobetzea denez, bilaketa-sistemaren uneko egoera azter dezagun. Webgunearen bilaketa-interfazea eskuz kodetuta dago, hau da, bilaketa-eremu guztiak definituta daude, eta analisi linguistikoen egitura eta egitura hori definitzen duen Relax NG eskema aldatzen bada, interfazea aldatu beharko litzateke.

Atal honen helburua analisi linguistikoen Relax NG eskematik abiatuz bilaketa-formularioa sortzea da, eskeman agertzen diren ezaugarriak bilaketara gehituz. Ondoren, web-gunearen eskakizunak jasotzen dituen aplikazioa moldatu beharko da, honek ere bilaketa-eremu berriak onartzeko.

Proiektuaren zati hau aurrera eramateko betekizunak honako hauek dira:

- Analisiak definitzeko eta balidatzeko erabiltzen den Relax NG eskematik abiatuz, bilaketarako interfazea sortu behar da; horretarako:
  - Relax NG eskemak irakurtzeko liburutegi bat garatuko da, eta nabigatzailetik erabilgarria izateko Javascript lengoaia erabiliz kodetuko da.
  - Garatutako liburutegia erabiliz, bilaketa-interfazea dinamikoki sortuko da web-aplikazioan.
- Bilaketa-sistema moldatu behar da bilaketa-eremu berriak onartzeko.
  - Interfazetik bilaketa-parametroa jasotzen duen *HybridServer* modulua moldatu eremu berriak onartzeko.
  - *LibiXaML* liburutegia moldatu eremu berriak onartzeko.

Eskematik abiatuz bilaketa XPath-ak automatikoki sortu

Atal honetan zerotik hasiz egin beharko dugu lan. Bilaketa-sistemak XML datu-basean bilaketak egiteko XPath sententziak erabiltzen ditu; bilaketa-eremu berria gehitu nahi bada XPath berria ere gehitu behar da bilaketa-sisteman. XPath guztiak kodean idatzita daude, eta edozein aldaketak berriro konpilatzea eta probak egitea eskatzen du.

Helburua XPath horiek automatikoki sortzea da, eta horretarako Relax NG eskema erabiliko da. Bilaketa-sistemak bilaketa egiteko mementoan XPath-ak irakurri eta bilaketa egingo du. Hauek dira betekizunak:

- Programa bat garatuko da, zeinek Relax NG eskema bat jasota, eskema horretan dauden ezaugarriek analisi-linguistikoetan dituzten XPath espresioak itzuliko dituen.

Eskematik abiatuz analisi linguistikoen sorkuntza automatizatu

Atal honetan analisi linguistiko berriak sortzeko funtzionalitatea gehituko zaio web-aplikazioari; horretarako, Relax NG eskema erabiliko da, eta baliozkoa den eta ondo eratua dagoen XML analisi linguistiko berria sortzeko aukera emango zaio erabiltzaileari.

Gai honetan prototipo oso sinple bat jasoko da eta eskema erabiliz analisiak sortzeko moldatu beharko da. Hauek dira betekizunak:

- Relax NG eskema erabiliz analisi linguistikoak sortzeko funtzionalitatea gehitu web-aplikazioan; horretarako:
  - Bezeroek analisiak sortzeko interfazea sortu behar da.
  - RNG eskema erabiliz, dagokion analisi linguistikoa sortu.

3.1.2 Murriztapenak

Atal honetan proiektuaren murriztapenak aztertuko dira. Lehenik proiektu osoari dagozkion murriztapenak aztertuko ditugu eta ondoren atal bakoitzarenak.

Proiektu osoan zehar kontuan izan beharko dugu Relax NG eskemak aldatuko direla, oraindik hauen gainean lanean ari baitira, eta posible izango da edozein mementotan eskemaren bertsio berria jasotzea. Beraz, diseinua eta garapena egiterakoan kontuan hartu beharreko gauza izango da. Edozein eskemaren bertsiorekin lan egin beharko dute proiektuaren hiru ataletan garatutako modulu eta programak. Hala ere, aldaketa txikiak izango direla aurreikusten da.

Atal bakoitzari dagokionez hauek dira bete beharreko murriztapenak:

Armiarma web-aplikazioa:

- Armiarma web-aplikazioan sortuko den interfazea oso azkar sortu behar du, exekuzio-garaian sortuko baita.
- Garrantzitsua izango da interfazea ahalik eta intuitiboen mantentzea, informatikariak ez diren erabiltzaileek erabiliko baitute.

Analisien sorkuntza automatikoa:

- Analisiak ondo eratua egon eta baliozkoa izan behar du.
- Kode-erredundantziak ekiditeko, lehen atalean garatutako liburutegia berrerabiliko da

3.1.3 Irismena finkatu

Proiektua aurrera eramateko betekizunak eta murriztapenak bete behar dira. Irismena definitzeko bitan banatuko dugu, batetik proiektuaren zati guztiek izango duten irismena, eta bestetik bakoitzak izango duena.

Irismen komuna

- Aplikazioa hizkuntzalarientzat dago egina; beraz, erabiltzaile arruntek erabiltzeko aplikazioa izaten jarraitu behar du. Horretarako, laguntza-testuak eta interfaze ahalik eta intuitiboena garatzen saiatu behar da.
- Relax NG eskemekin lan egitean kontuan hartu behar da ez direla eskema definitiboak. Proiektuan zehar hauek aldatzea aurreikusten da; beraz, ahalik eta generikoen tratatu beharko dira, eskemak aldatzen badira, denak funtzionatzen jarraitzeko.
- Lana ez errepikatzeko, lehen zatia liburutegi moduan garatuko da, hirugarrenean bertako funtzioak berrerabiltzeko. Relax NG eskemak tratatzeko funtzioak liburutegi horretan barneratuko dira.

Lehen zatiaren irismena

- Relax NG eskemak erabiltzeko liburutegi bat garatuko da.
- Liburutegia eta Relax NG eskemak erabiliz bilaketa-interfazea sortuko da.
- Bilaketa egiten duen *HybridServer* modulua moldatu bilaketa-eremu berriak onartzeko.
- *LibiXaML* liburutegia moldatu bilaketarako behar diren XPath berriak gehituz.

Bigarren zatiaren irismena

- Programa bat garatu zeinek RNG eskema bat jaso, eta *feature* edo ezaugarri ezberdinen XPath-a itzultzen duen.

- Nahiz eta eskematik abiatu, sortzen diren XPath-ak analisiarenak izan behar dute, hau da, analisian bilaketak egiteko XPath-a lortu nahi da, eta ez eskeman bertan bilaketak egiteko. Hurrengo adibidean, ezkerreko eskematik abiatuz, espresio hau lortu beharko genuke: `f[@name="ADOIN"]/str`.

3.1 Kode-zatia: ADOIN ezaugarriaren RNG eskema.

```
<element name="f">
    <attribute name="name">
        <value>ADOIN</value>
    </attribute>
    <element name="str">
        <data type="string"/>
    </element>
</element>
```

3.2 Kode-zatia: Eskemak definitzen duen ezaugarria.

```
<f name="ADOIN">
    <str>balioa</str>
</f>
```

Hirugarren zatiaren irismena

- Erabiltzaileari analisi linguistiko berriak sortzeko aukera emango zaio.
- RNG eskema eta lehen ataleko liburutegia erabiliz sorkuntza interfazea eta analisi linguistikoak sortu behar dira.
- Horretarako eginda dagoen prototipotik abiatzen saiatuko gara.

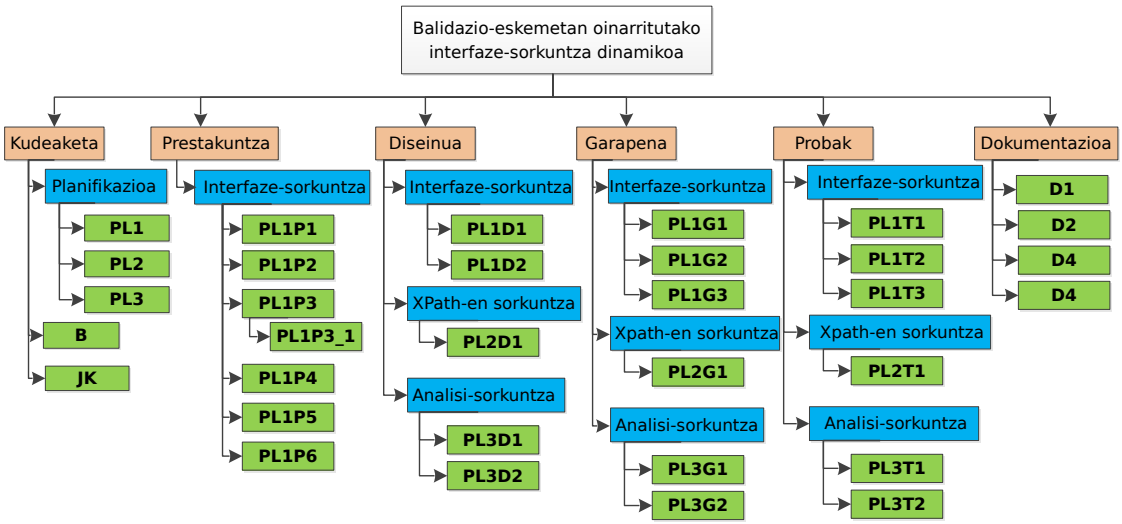
3.1.4 Lanaren deskonposaketa-egitura

Atal honetan lanaren deskonposaketa-egituraren diagrama azaldu eta lan-paketeen deskribapena egingo dugu. Proiektua zein atazatan banatuko den zehaztu eta hauek deskribatuko dira, eta honetan oinarrituko gara denboraren planifikazioa egiterako orduan.

LDE diagrama

Jarraian proiektuko lanaren deskonposaketa-egitura diagrama osoa azaltzen da.

3.1 Irudia: Proiektuaren LDE diagrama.



Lan-paketeen deskribapena

Atal honetan aurreko irudian azaltzen den lan-pakete bakoitza deskribatuko dugu.

Kudeaketa

- Planifikazioa
  - **PL1** - Interfazearen sorkuntzaren planifikazioa egin
  - **PL2** - Xpath-en sorkuntza automatikoaren planifikazioa egin
  - **PL3** - Analisien sorkuntzaren planifikazioa egin
- **B** - Bilerak egin
- **JK** - Jarraipen eta kontrola egin

Prestakuntza

- Interfaze-sorkuntza
  - **PL1P1** - Relax NG eskema-lengoaia ikasi
  - **PL1P2** - Armiarma web-aplikazioaren prototipoa ulertu

  - **PL1P3** - *LibiXaML* liburutegia erabiltzen ikasi
    * **PL1P3_1** - C++ ikasi
  - **PL1P4** - Anotazio-amarauna ulertu
  - **PL1P5** - Javascript menperatu
  - **PL1P6** - DOM eredua menperatu

Diseinua

- Interfaze-sorkuntza
  - **PL1D1** - Javascript liburutegia diseinatu
  - **PL1D2** - Interfazea diseinatu
- XPath-en sorkuntza
  - **PL2D1** - XPath-ak lortzeko estrategia diseinatu
- Analisien sorkuntza
  - **PL3D1** - Interfazea diseinatu
  - **PL3D2** - Analisi-sorkuntzaren estrategia

Garapena

- Interfaze-sorkuntza
  - **PL1G1** - Javascript liburutegia garatu
  - **PL1G2** - Interfazea garatu
  - **PL1G3** - *HybridServer* eta *LibiXaML* moldatu
- XPath-en sorkuntza
  - **PL2G1** - XPath-ak lortzeko *script*-a garatu
- Analisien sorkuntza: Honetan dagoen prototipotik lana berrerabiltzea espero da.
  - **PL3G1** - Analisien sorkuntzako interfazea garatu
  - **PL3G2** - Analisiak sortzeko programa garatu

Probak

- Interfaze-sorkuntza
  - **PL1T1** - Javascript liburutegia probatu
  - **PL1T2** - Interfazea sortzeko programa probatu
  - **PL1T3** - Denak ondo funtzionatzen duela probatu
- XPath-en sorkuntza
  - **PL2T1** - XPath-ak lortzeko *script*-a probatu
- Analisien sorkuntza
  - **PL3T1** - Analisien sorkuntzako interfazea probatu
  - **PL3T2** - Analisiak sortzeko programa probatu

Dokumentazioa

- **D1** - Eskuliburuak
- **D2** - Memoria
- **D3** - Aurkezpena
- **D4** - Jarraipena

3.2 Denboraren kudeaketa

Atal honetan denboraren kudeaketa azalduko dugu, horretarako atazak definitu eta denboraren estimazioa egingo dugu. Lehenik, proiektu osoaren planifikazioa egingo dugu, horretarako atal bakoitza aurrera eramateko beharko diren orduak estimatuko dira eta ondoren atal bakoitzaren denboraren planifikazioa azalduko dugu.

- Bilaketa sistema moldatzea lan gehien emango duen atala izango da. Prestakuntza gehiena hasieran egin beharko dugu eta lan konplexuena honetan egin beharko da; 180 ordu inguru beharko dira atal hau amaitzeko (maiatza - abendua bitartean).

- XPath-en sorkuntza automatikoa atala burutzean, ikasketa asko egina izango dugu, eta ondorioz, lana errazago eta azkarrago egitea espero da; 50 ordu inguru beharko dira atal hau amaitzeko (otsaila - martxoa bitartean).

- Analisien sorkuntzak lehen atalarekin zerikusi handia du, gainera, lehen ataleko liburutegia erabiliz gero lan gehiena eginda egotea espero da; 50 ordu inguru beharko dira atal hau amaitzeko (apirila - maiatza bitartean).

Hiru atalek guztira 280 ordu lan beharko dituzte; ordu hauei, jarraipena, dokumentazioa, memoria eta bestelako lanak egiteko beharreko orduak gehitu beharko zaizkie.

3.2.1 Atazak definitu

Jarraian atal bakoitzaren planifikazioan definitutako atazak azaltzen dira.

- **B** - Bilerak: Lana ikasle batek egingo duen arren, jende asko dago lanean proiektu honen inguruan, horregatik erabaki asko taldean hartu beharko dira. Horretarako astero bilera bat egingo da, betiere beharretara moldatuz.

- **JK** - Jarraipen eta kontrola: Proiektu-kudeaketak ez du ezertarako balio proiektuaren jarraipenik ez bada egiten; horregatik, proiektu osoan zehar honen jarraipena eta kontrola egin beharko da.

- **D1** - Eskuliburuak: Proiektu honetan hainbat tresna garatuko dira, tresna hauek etorkizunean erabilgarri izan daitezen bakoitzak eskuliburua izan beharko du.

- **D2** - Memoria: Proiektuaren memoria egin beharko da hau defendatu ahal izateko.

- **D3** - Aurkezpena: Proiektua epaimahai baten aurrean defendatu beharko da, eta horretarako aurkezpen bat prestatuko da.

- **D4** - Jarraipen dokumentua: Jarraipen eta kontrolaren ondorioz jarraipen-dokumentua lortuko dugu.

Lehen zatia: Web-aplikazioaren interfazearen sorkuntza

- **Prestakuntza**

  - **PL1P1** - Relax NG: Proiektu osoan zehar RNG eskemek garrantzi handia izango dute, horregatik ezinbestekoa izango da Relax NGk erabiltzen duen lengoaia menperatzea.
  - **PL1P2** - Armiarma prototipoa: Proiektu hasieran web-aplikazioaren prototipo bat jasoko da; honetan hainbat gauza inplementatu beharko dira, eta horregatik hau ondo ulertzea eta erabiltzen jakitea ezinbestekoa da.
  - **PL1P3_1** - C++ ikasi: Bilaketa-sistema hobetzeko C++ lengoaiaz idatzitako hainbat programa aldatu beharko dira, eta horretarako lengoaia hau ondo erabiltzen jakin beharko da.
  - **PL1P3** - *LibiXaML*: Dokumentuak kudeatzeko liburutegi hau menperatu beharko da, XML datu-basean bilaketak egiteko ezinbestekoa delako.
  - **PL1P4** - Anotazio-amarauna: IXA taldean analisi linguistikoak adierazteko erabiltzen den datu-eredu hau menperatu beharko da bilaketa-sistema eta analisien sorkuntza hobetu ahal izateko.
  - **PL1P5** - JS menperatu: Web-aplikazioan egingo den lan gehiena lengoaia honetan egingo da. Javascript erabiltzen ikasi beharko da lana ondo egin ahal izateko.
  - **PL1P6** - DOM eredua: XML datuen gainean lan egiteko eredu hau erabiliko da; ikaslearentzat dagoeneko ezaguna denez, DOM eredua menperatzeak ez du denbora asko beharko.

- **Diseinua**
  - **PL1D1** - Liburutegia diseinatu: Relax NG eskemak erabiltzeko liburutegi bat diseinatzea.
  - **PL1D2** - Interfazea sortuko duen programa diseinatu: Liburutegi hori erabiliz interfazea sortuko duen programa diseinatu.

- **Garapena**
  - **PL1G1** - Liburutegia garatu.
  - **PL1G2** - Interfazea sortuko duen programa garatu: horretarako lehen garatutako liburutegia erabiliko da.
  - **PL1G3** - *HybridServer* eta *LibiXaML* moldatu bilaketa-eremu berriak onartzeko. Behin interfazea aldatuta, interfazearen datuak jasotzen dituen modulua moldatu beharko da.

- **Probak**
  - **PL1T1** - Liburutegia probatu: egindako liburutegiak berrerabilgarria izan beharko du, edozein akatsek etorkizunean eragin handia izan lezake eta.
  - **PL1T2** - Interfazea probatu: lortuko den interfazea ondo egoteaz eta funtzionatzeaz gain, erabilgarria izan behar du.
  - **PL1T3** - Bilaketa-sistema probatu: egindako lana ondo dagoela probatzeko, atal honetan bilaketa-sistemak guztiz funtzionala egon beharko du.

Bigarren zatia: XPath-en sorkuntza automatikoa

- **Diseinua**
  - **PL2D1** - Xpath-ak sortzeko estrategia diseinatu: algoritmoa diseinatzeaz gain, Relax NG eskema-lengoaia aztertu beharko da, lengoaia honetan idatzitako eskema batetik XPath-ak lortu beharko ditugulako.
- **Garapena**
  - **PL2G1** - Scripta garatu: diseinatutakoa inplementatu eta XPath-ak *LibiXaML* liburutegitik erabiltzeko prestatu.
- **Probak**
  - **PL2T1** - Scripta probatu: sortutako XPath-ak egokiak direla ziurtatu.

Hirugarren zatia: Analisien sorkuntza automatikoa

- **Diseinua**
  - **PL3D1** - Interfazea diseinatu: analisiak sortzeko interfazea diseinatu; bilaketa-interfazearen oso antzekoa izango da.
  - **PL3D2** - Analisiak lortzeko estrategia diseinatu: XPath-en sorkuntzan ikasitakoa oso erabilgarria izango da atal honetan.
- **Garapena**
  - **PL3G1** - Interfazea garatu.
  - **PL3G2** - Analisiak sortzeko programa garatu.

- **Probak**
  - **PL3T1** - Interfazea probatu: interfaze egokia eta erabilgarria lortu nahi da. Horretarako erabilera- probak egin beharko dira.
  - **PL3T2** - Analisien sorkuntza probatu: sortzen diren analisiak ondo eratuta eta baliozkoak direla ziurtatu.

3.2.2 Denboraren estimazioa

Atal honetan, ataza bakoitza burutzeko beharko dugun denbora azalduko dugu.

- B: 30 ordu
- JK: 10 ordu
- D1: 20 ordu
- D2: 80 ordu
- D3: 10 ordu
- D4: 30 ordu

Lehen zatia: Web-aplikazioaren interfazearen sorkuntza

- PL1P1: 3 ordu
- PL1P2: 10 ordu
- PL1P3_1: 5 ordu
- PL1P3: 5 ordu
- PL1P4: 5 ordu
- PL1P5: 2 ordu
- PL1P6: 2 ordu
- PL1D1: 15 ordu
- PL1G1: 40 ordu

- PL1T1: 8 ordu
- PL1D2: 10 ordu
- PL1G2: 20 ordu
- PL1T2: 5 ordu
- PL1G3: 15 ordu
- PL1T3: 5 ordu

Zati honetan definitutako atazek **150 ordu** lan beharko dituzte; honi, bilerak, jarraipena, eta dokumentazio-lanak gehituta, hasieran aurreikusitako 180 orduak beteko genituzke.

Bigarren zatia: XPath-en sorkuntza automatikoa

- PL2D1: 10 ordu
- PL2G1: 20 ordu
- PL2T1: 5 ordu

Zati honetan definitutako atazek **35 ordu** lan beharko dituzte. Hasieran aurreikusitako 50 orduekin konparatuz motz geratzen garela dirudi; hala ere, dokumentazio eta jarraipen-lanekin, 50 orduetara hurbilduko gara.

Hirugarren zatia: Analisien sorkuntza automatikoa

- PL3D1: 5 ordu
- PL3G1: 10 ordu
- PL3T1: 3 ordu
- PL3D2: 10 ordu
- PL3G2: 20 ordu
- PL3T2: 3 ordu

Zati honetan definitutako atazek **51 ordu** lan beharko dituzte. Hasieran 50 ordu aurreikusi ziren eta argi dago motz geratu ginela, dokumentazio, jarraipen eta bilerekin 70 ordu inguruan egongo gara eta.

Jarraian azaltzen diren tauletan, proiektuan sartu beharreko ordu kopurua laburtu dugu.

3.1 Taula: Proiektuan zehar sartu beharreko orduen laburpena.

Ataza	**Ordu kopurua**
B	30
JK	10
D1	20
D2	80
D3	10
D4	30
Proiektuan zehar	**180**

3.2 Taula: Lehen zatian sartu beharreko orduen laburpena.

Ataza	**Ordu kopurua**
PL1P1	3
PL1P2	10
PL1P3_1	5
PL1P3	5
PL1P4	5
PL1P5	2
PL1P6	2
PL1D1	15
PL1G1	40
PL1T1	8
PL1D2	10
PL1G2	20
PL1T2	5
PL1G3	15
PL1T3	5
Lehen atala	**150**

3.3 Taula: Bigarren zatian sartu beharreko orduen laburpena.

Ataza	**Ordu kopurua**
PL2D1	10
PL2G1	20
PL2T1	5
Bigarren atala	**35**

3.4 Taula: Hirugarren zatian sartu beharreko orduen laburpena.

Ataza	**Ordu kopurua**
PL3D1	5
PL3G1	10
PL3T1	3
PL3D2	10
PL3G2	20
PL3T2	3
Hirugarren atala	**51**

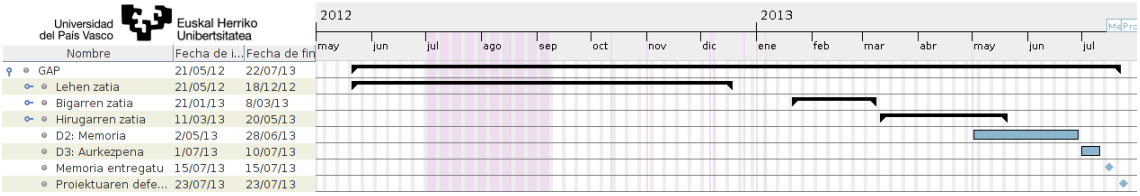
3.5 Taula: Proiektuan sartu beharreko orduen laburpena.

Proiektuan zehar	**180**
Lehen atala	**150**
Bigarren atala	**35**
Hirugarren atala	**51**
Guztira proiektuan	**416 ordu**

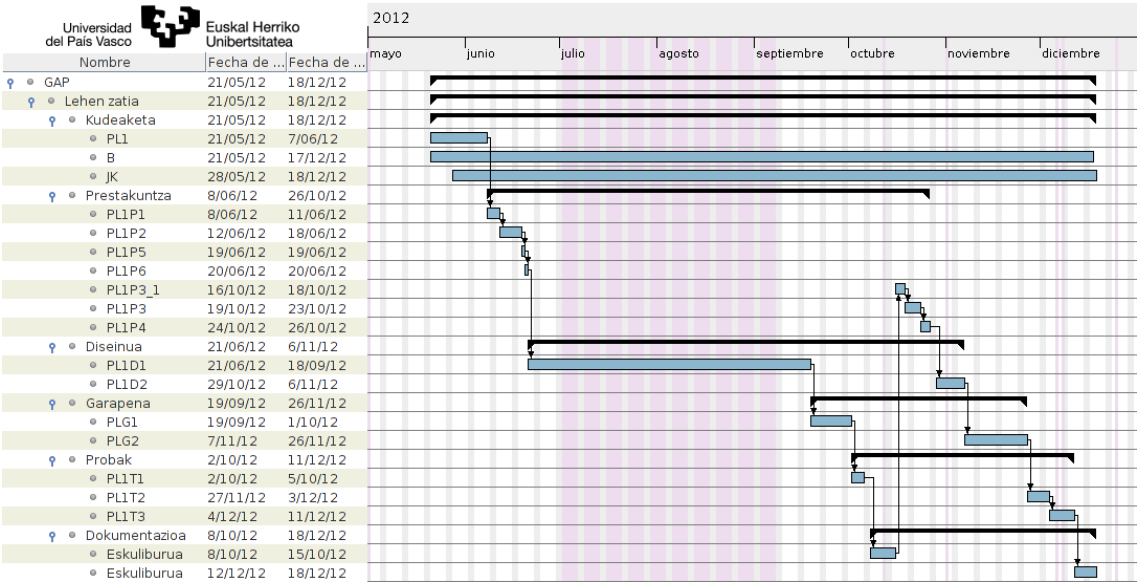
3.2.3 Gantt diagrama

Proiektuaren izaera dela-eta, ezinezkoa izango zen Gantt diagrama osoa irudi bakar batean gehitzea, horregatik lau diagramatan banatu dugu.

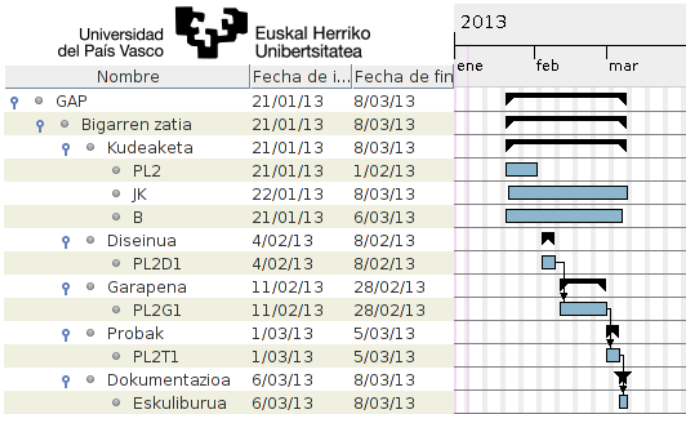
3.2 Irudia: Proiektu osoaren Gantt diagrama.



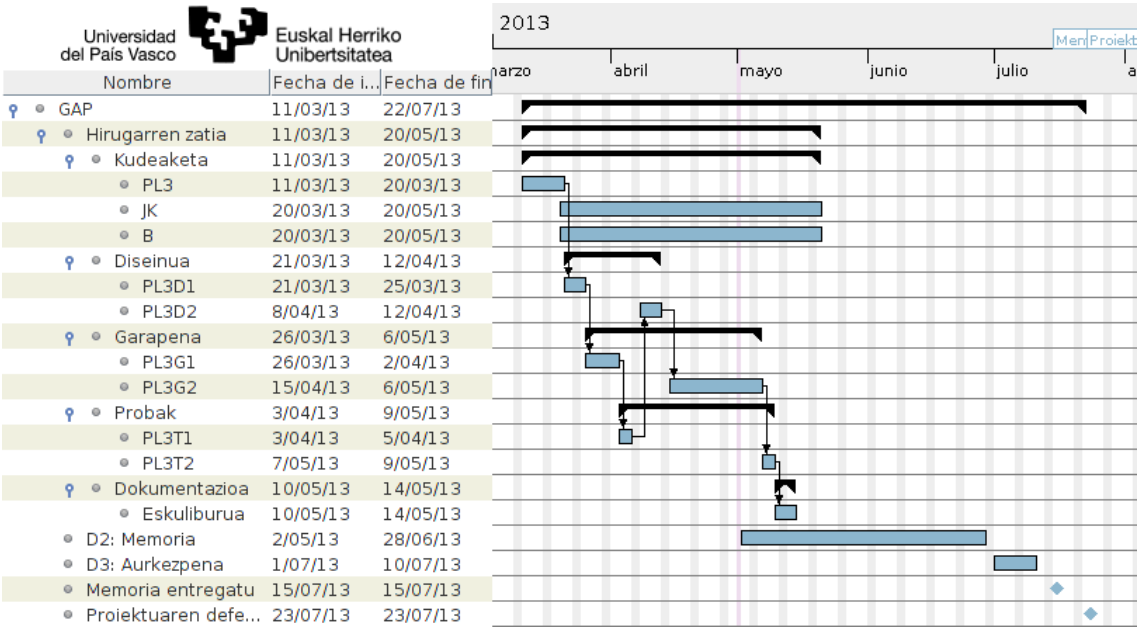
3.3 Irudia: Lehen zatiaren Gantt diagrama.



3.4 Irudia: Bigarren zatiaren Gantt diagrama.



3.5 Irudia: Hirugarren zatiaren Gantt diagrama.



3.3 Kalitate-plana

Kalitatea gero eta garrantzi handiagoa ari da hartzen informatikaren munduan. Gaur egun ez da nahikoa betekizun eta murriztapenak betetzearekin, bezeroak lortzeko kalitatea behar da. Proiektuan moldatuko den Armiarma web-aplikazioa Euskaltzaindiaren Lexikoaren Behatokia egitasmoan[1] sartzen da. Egitasmo horrek euskal hizkuntzalaritzan duen garrantzia dela-eta, kalitate maila altua mantentzea ezinbestekoa izango da. Horretarako kalitate-plana beharko dugu. Atal honetan kalitatearen planifikazioa eta kontrola nola egingo dugun azalduko dugu.

3.3.1 Kalitatearen planifikazioa

Kalitatezko proiektu batek ezinbestekoa du betekizun eta murriztapenak guztiz betetzea. Hala ere, hauek betetzea minimoa baino ez da, proiektuaren kalitatea ezin baita betekizunak eta murriztapenak betetzearekin bermatu. Kalitatea bermatzeko hainbat irizpide erabili behar dira, baina irizpide hauek zehaztea lan konplexua da, izan ere, pertsonek kalitatea era ezberdinean definitzen dute, kalitate-irizpide ezberdinak dituztelako. Horregatik ezinbestekoa da kalitate-irizpideak zein izango diren definitzea.

[1] http://lexikoarenbehatokia.euskaltzaindia.net/aurkezpena.htm

Kalitate-irizpideak

Proiektu hau kalitatezkoa dela esan ahal izateko, bete beharko diren hainbat irizpide definituko ditugu:

- **Egindako lanaren kalitatea:** Proiektuan ordu asko egin beharko dugu lan, ondorioz, ordu horiek aprobetxatzea ezinbestekoa izango da.
- **Emaitzen kalitatea:** Garatutako web-aplikazioaren kalitatea neurtzea oso zaila izango da; hala ere, erabilgarritasuna aztertuko da.
- **Kodearen kalitatea:** Proiektua amaitzean web-aplikazioa ez da guztiz amaituta egongo, eta beraz, proiektu honetan egindako lana beste norbaitek erabiliko du etorkizunean. Horregatik, proiektuaren etorkizuna bermatzeko kodearen kalitatea mantendu beharko da.
- **Dokumentazioaren kalitatea.**

3.3.2 Kalitatearen kontrola

Behin kalitate-irizpideak definituta, kalitatearen kontrola nola egingo dugun zehaztuko dugu.

- **Egindako lanaren kalitatea**
  - Lan-orduak aprobetxatzen direla ziurtatzeko ezinbestekoa da lan-giroa dagoen toki batean lan egitea. Hau bermatzeko, IXA taldeak ordenagailu bat utziko dio ikasleari, Informatika Fakultatean duen bulego batean lan egiteko.
- **Emaitzen kalitatea**
  - Aplikazioaren konplexutasuna dezente handituko da proiektu honen amaieran; horregatik, ahalik eta intuitiboen mantendu nahi da, informatikariak ez direnek ere erabiliko baitute. Horretarako, ikaslea ez den pertsonek emandako aplikazioaren erabilerari buruzko informazioa kontuan hartuko da interfazearen diseinuan.
- **Kodearen kalitatea**

  - Kalitatezko kodea izateko, ulerterraza izan behar du. Horretarako, behar denean kodea ondo komentatuko da.
  - Proiektuan zehar egingo diren eskuliburu ezberdinek honetan lagunduko dute.

- **Dokumentazioaren kalitatea**

  - Pertsona bakar batek egingo du. Beraz, formato eta estilo-ezberdintasunik ez da agertuko; hala ere, LaTeX txantiloi bat erabiliko da dokumentu ezberdinek itxura bera izateko.

3.4 Giza baliabideak

Proiektu honetan hainbat pertsonak hartuko dute parte. Proiektua aurrera eramango duena Jagoba Gascón Sánchez ikaslea izango da. Aitor Soroa Etxabe eta Xabier Artola Zubillaga irakasleak arituko dira zuzendari lanetan, haien lana ikaslea aholkatzea izango da. Proiektu hau aurrera eraman ahal izateko ezinbestekoa izan da aurrez Joseba Alberdi, Mikel Astiz eta Zuhaitz Beloki ikasleek egindako lana, eta bai IXA taldeak egindakoa ere.

3.5 Komunikazioak

Atal honetan proiektuan zehar jarraitu beharreko komunikazio-plana egingo dugu; horretarako interesatuak definitu eta komunikazioaren planifikazioa beharko ditugu.

3.5.1 Interesatuak definitu

3.6 Taula: Proiektuan interesatuak

Pertsona	**Rola**	**Interesa**	**Ezagutza maila**	**Ezagutza arloa**
Jagoba Gascón Sánchez	Ikaslea	Oso handia	Oso altua	Proiektua
Aitor Soroa Etxabe	Zuzendaria	Handia	Altua	*LibiXaML*
Xabier Artola Zubillaga	Zuzendaria	Handia	Altua	Relax NG
Zuhaitz Beloki Leitza	Ikasle ohia?	Handia	Altua	Relax NG eta Armiarma web-aplikazioa

3.5.2 Komunikazioen planifikazioa

Interes aldetik interesatu guztiek profil oso antzekoa dutenez, haiekin komunikatzeko estrategia bera erabiliko dugu. Hala ere, kontuan hartu beharko dugu bakoitzaren ezagutza arloa kontaktuan jarri baino lehen.

Posta elektronikoa

Komunikazio-sistema nagusia posta elektronikoa izango dela erabaki da. Interesatu kopurua txikia denez, mezu kopurua txikia izango da.

Bilerak

Esan bezala, astero bilera bat egitea planifikatu da. Bilera hauetan interesatu guztiak egongo dira, eta erabakiak hartzeko eta interesatuen artean informazioa trukatzeko erabiliko dira.

3.5.3 Informazioaren banaketa

Informazioa zentralizatuta mantentzeko proiektuaren garapenari buruzko wiki bat erabiliko da. Bertan, proiektuaren uneko egoera eta egin beharrekoak egongo dira, eta honela, edonork jakin ahal izango du proiektuaren egoera.

3.6 Kontingentzia-plana

Arriskuen kudeaketa egokia egiteko kontingentzia-plana erabiliko dugu. Proiektu honetan kudeaketa guztia ikasleak egingo duenez, bere lana izango da arriskuen aurrean erabakiak hartzea, mehatxuak murriztea, aukerak aprobetxatzea. Erabakiak hartzeko unean, kontuan hartu beharko da proiektuaren epeak atzeraezinak direla.

3.6.1 Arriskuak identifikatu

Atal honetan proiektuan zehar gerta daitezkeen arriskuak identifikatuko dira:

- **A01** - Zerbitzaria bertan behera gelditzea, edo honetara konektatu ahal ez izatea.
- **A02** - Kodea galtzea.
- **A03** - Plangintza ez betetzea.
- **A04** - Eskeman egindako aldaketa baten ondorioz, egindakoak funtzionatzeari uztea.

3.6.2 Arriskuak ekiditeko planifikazioa

A01

- Arrazoi posibleak:
  - Zerbitzariak berrasieratu behar izatea: zerbitzarien eguneraketa dela-eta, edo argia joan delako gerta liteke.
    - Gertatzeko arriskua: baxua.
  - Internet-konexiorik ez izatea: etxetik lan egitean Internet-konexioa gal genezake.
    - Gertatzeko arriskua: baxua.
  - VPN bidez konektatu ahal ez izatea: etxetik zerbitzaria atzitu ahal izateko VPN bidez konektatu behar da.
    - Gertatzeko arriskua: ertaina.
- Ondorioak: lanik egin ahal ez izatea. [A03] gertatzeko arriskua.
- Eragina proiektuan: txikia.
- Gertatzeko arriskua: ertaina.
- Arriskua ekiditeko hartu beharreko erabakiak:
  - Ezin da arrisku hau ekidin.

A02

- Arrazoi posibleak:
  - Zerbitzariko datuak borratzea.
    - Gertatzeko arriskua: oso baxua.
  - Ikasleak datuak nahigabe borratzea.
    - Gertatzeko arriskua: baxua.
- Ondorioak: galdutako lana berriro egin behar izatea; [A03] gertatzeko arriskua.
- Eragina proiektuan: oso handia.
- Gertatzeko arriskua: baxua.
- Arriskua ekiditeko hartu beharreko erabakiak:
  - Ikasleak *Git* bertsioak kontrolatzeko sistema erabiliko du egindako lanaren kopiak gordetzen joateko.

A03

- Arrazoi posibleak:
  - Ikaslea gaixotzea.
    - Gertatzeko arriskua: baxua.
  - Planifikazioan aurreikusi ez den arazo baten ondorioz denbora asko galtzea.
    - Gertatzeko arriskua: baxua.
- Ondorioak: proiektua epe barruan ez amaitzea.
- Eragina proiektuan: oso handia.
- Gertatzeko arriskua: baxua.
- Arriskua ekiditeko hartu beharreko erabakiak:
  - Jarraipen eta kontrol egokia eginez gero ezustezko baten eragina asko txikitzen da.

A04

- Arrazoi posibleak:
  - Eskeman aurreikusi ez den aldaketaren bat gehitzea.
    - Gertatzeko arriskua: ertaina.
- Ondorioak: lana berregin behar izatea arazoa konpontzeko; [A03] gertatzeko arriskua.
- Eragina proiektuan: eragin txikia, ertaina, edo oso handia izan dezake.
- Gertatzeko arriskua: ertaina.
- Arriskua ekiditeko hartu beharreko erabakiak:
  - Eskemak izan ditzakeen aldaketak aurreikusiz eta hauek, ahalik eta modurik orokorrenean tratatuz, arrisku honen eragina eta gertatzeko aukerak murriztuko ditugu.

3.6.3 Arriskuen kudeaketa, laburpena

3.7 Taula: Arriskuen kudeaketaren laburpen-taula.

Arriskua	**Eragina**	**Gertatzeko arriskua**	**Hartutako erabakiak**
A01	Txikia	Ertaina	Ezin ekidin
A02	Oso handia	Baxua	Bertsioak kontrolatzeko sistema erabiltzea
A03	Oso handia	Baxua	Jarraipena eta kontrola
A04	Ezezaguna	Ertaina	Aldaketak aurreikusi eta eskemak modu orokorrean tratatu

3.7 Eskuraketak

Proiektuan zehar eskuraketa gutxi egin beharko dira. Aurreikusten den eskuraketa mota bakarra softwarea da; zorionez, software libreak aukera asko eskaintzen ditu, eta beraz, softwarea erosten ez dugu gastu handirik egingo.

Proiektu honetan zehar XML teknologiak asko landuko direnez, Oxygen XML Editor tresna erabiltzea erabaki da. Tresna honek lizentzia erostea eskatzen du; hala ere, ikasleak erabili ahal izango du, eskura izango baititu Informatika Fakultateak eta IXA taldeak dituzten lizentziak.

4. KAPITULUA

Tresnak eta teknologiak

Proiektua aurrera eramateko orduan tresna eta teknologia ezberdinekin egin da lan, eta hauetako batzuk ezinbestekoak izan dira garapenean zehar. Atal honetan proiektuan zehar erabilitako garrantzitsuenak azalduko dira. Atala lau zatitan banatu da; lehenik, XML teknologiekin zerikusia duten tresnak eta web-teknologiak azalduko ditugu, hauek izan baitira proiektu honen oinarria. Ondoren erabilitako garapen-inguruneak eta bertsioen kontrol-sistemak azalduko ditugu.

4.1 XML teknologiak

4.1.1 XPath

XPath (*XML Path Language*) XML dokumentu baten gainean kontsultak egiteko balio duen lengoaia da, eta horretarako XML dokumentuaren egitura kontuan hartzen du. XMLren gainean transformazioak egiteko den XSLT lengoaiarekin batera erabiltzeko asmatu zen.

LibiXaML liburutegiak analisi linguistikoak bilatzeko orduan XQuery[1] erabiltzen du. XQuery-k XPath espresioak erabiltzen ditu XML dokumentuaren zati bat atzitzeko orduan. Proiektu honen helburuetako bat XPath horien sorkuntza automatikoa lortzea zen.

[1] XML datu bilduma baten gainean kontsultak egiteko balio duen lengoaia.

4.1.2 Relax NG

Analisi linguistikoen XML kodea balidatzeko eskemak Relax NG eskemak dira. Proiektuan zehar eskema hauek landu ez ditugun arren, sakonean aztertu behar izan ditugu eta egindako lanaren zati oso garrantzitsua izan dira. Hala ere lehen azaldu denez[2] ez dugu honetan sakonduko.

4.1.3 Oxygen XML Editor

Oxygen XML editor XML teknologiekin zerikusia duten lengoaiekin lan egiteko tresna da. XML dokumentuak bistaratu eta editatzeaz gain, XPath bilaketak eta XSLT transformazioak ere egiten ditu XML dokumentuen gainean.

Proiektuan zehar oso erabilgarria izan da erabili den Relax NG eskema bistaratzeko, izan ere eskema sei fitxategitan dago banatua eta erreferentzia asko erabiltzen ditu. Oxygen erabiliz eskema dokumentu bakar batean bezala ikusteko aukera izan dugu. Gainera, aldaketaren bat egiteko mementoan eskema ondo eratua zegoela ziurtatzen zuen.

Software hau ez da librea, lizentzia bat erostea eskatzen du. Gure kasuan bai IXA taldeak baita Informatika Fakultateak lizentzia ordaindua dute, eta beraz, ez da arazorik egon erabiltzeko.

4.1.4 Berkeley DB XML

Proiektu honetan, analisiak bilatu eta sortzeko garaian erabili da. Analisi linguistikoak XML egitura zutenez, XML motako datu-base batean gordetzea erabaki zen. Hauen artean Berkeley DB XML aukeratu zen. Berkeley DB XML, XML dokumentuak gordetzeko datu-base kudeaketa-sistema da, Oracle-rena. XQuery erabiltzeko aukera ematen du eta kudeaketa guztiaz (erabiltzaileak, konkurrentzia, etab.) arduratzen da.

[2] ikus 2.1.2 atala, 7. orrian.

4.1.5 Perl

Perl 1987. urtean sortutako lengoaia interpretatua da. Testuaren prozesamenduan eskaintzen zituen abantailengatik eta beste *script* motako lengoaien ezintasunak ez edukitzeagatik bere erabilpena asko handitu zen.

Testua prozesatzeaz gain, XML dokumentuak prozesatzeko ere erraztasun handiak eskaintzen ditu, eta horregatik aukeratu zen XPath-en sorkuntza automatikoa aurrera eramateko.

4.2 Web-tresnak eta teknologiak

Web-aplikazioaren garapenean tresna ugari erabili dira proiektuan; hauen artean, Javascript eta PHP lengoaiak, AJAX teknologia, JSON eta Chrome nabigatzaileak eskaintzen dituen kontsola eta debugger-a dira aipagarrienak. Tresna hauek oso ezagunak diren arren, merezi du batzuk laburrean azaltzea:

4.2.1 AJAX

Ingelesezko ***A****synchronous* ***J****avaScript* ***A****nd* ***XML*** hitzetik hartzen du izena. Web-aplikazio batean datuak era asinkronoan lortzeko metodoa da, honen bidez orri baten zati bat eguneratu daiteke orri osoa kargatu behar ez izateko. Armiarma web-aplikazioan asko erabiltzen da.

4.2.2 JSON

J*ava****S****cript* ***O****bject* ***N****otation*en laburdura. Testu laua erabiltzen duen datuen trukerako formatu arina da. Gainera, Javascript lengoaiak formatu honetan idatzitako testuak Javascript objektu bihurtzeko aukera ematen du. JSON erabiliz Relax NG eskemen tratamendurako liburutegiak behar duen konfigurazio-fitxategia idatzi da.

4.2.3 Chrome nabigatzailea: kontsola eta debugger-a

Proiektuan Javascript lengoaian kode asko idatzi behar izan da. Lengoaia honetan idatzitako *script*ak arazteko tresna oso erabilgarriak izan dira Chrome nabigatzaileak eskaintzen dituen garatzaileentzako tresnak.

4.3 Testu-editoreak eta garapen-inguruneak

Erabilitako testu-editore eta garapen-inguruneen artean Eclipse eta Gedit aurkitzen dira, eta Oxygen ere atal honetan sar genezake. Gedit testu laua editatzeko tresna da, kodea koloreztatzeko aukera ematen du, idazketa eta irakurketa asko erraztuz; Javascript, PHP, Perl, HTML eta landutako bestelako lengoaietan garatzeko erabili da. Eclipse lengoaia ezberdinetan lan egin dezakeen garapen-ingurunea da; bigarren hau sakonago azalduko dugu.

4.3.1 Eclipse

Kode irekiko IDE (*Integrated Development Enviroment* edo garapen ingurune integratu) bat da. IBM enpresak garatua baina gaur egin Eclipse Fundazioak mantentzen du. Aukera asko ematen dituen garapen-ingurunea da, lengoaia (C, C++, PHP, Java etab.) eta teknologia ezberdinak (modelo bidezko garapena, DSLen garapena, testak egiteko moduluak, etab.) erabiltzeko aukera ematen duelako.

Tresna honi esker kodearen idazketa eta akatsen zuzenketa asko azkartu da. Egungo IDE gehienek dituzten ezaugarri berak ditu (kodearen osatze automatikoa, debugger-a, konpiladorea etab.) baina honek komunitate handi bat du atzetik. *Plugin*-ak gehitzeko aukera ere ematen du, bere funtzionalitateak handitzeko aukera emanez.

Iturburu-kode zati handi bat idazteko erabili da, C++ lengoaiaz idatzitako zatia alegia. Jasotako prototipoan C++ lengoaiaz idatzitako moduluek ehun klasetik gora dituzte. Hasieran ezezagunak ziren modulu horiekin lan egitea ezinezkoa izango zen Eclipse-ren laguntzarik gabe.

4.4 Bertsioen kontrol-sistemak

4.4.1 Git

Git Linus Torvalds-ek garatutako VCS (*Version Control System* edo bertsioak kontrolatzeko sistema) softwarea da. Honek, iturburu-kodearen bertsioak gordetzen joateaz gain, *branch*[3] edo adar ezberdinak edukitzea ahalbidetzen du. Honela, ondo funtzionatzen duen kode batean zerbait aldatu edo funtzionalitate bat gehitu nahi bada, adar nagusia alde batera utzi eta beste adar batean lan egiten ahal da. Adar hori ondo dagoenean nagusiarekin batzen da. Honi esker, adar nagusian dagoen iturburu-kodea beti ondo funtzionatuko du.

Proiektuan garatutako sistema ezberdinetan erabili dugun bertsioen kontrol-sistema Git izan da. Proiektua osatzen duen modulo bakoitzarentzako Git errepositorio bat sortu da, eta honela aldaketen kontrola egin eta ezustekoen aurrean gure burua babesten dugu.

Subversion bertsioak kontrolatzeko sistema ere erabili behar izan da, IXA taldeak Subversion errepositorio batean gordetzen baititu Relax NG eskemak.

[3] Git-en uneko iturburu-kodearen kopia bat egin eta honen gainean era independentean lan egitea ahalbidetzen du

5. KAPITULUA

Diseinua eta garapena

Proiektu honetan diseinuak hiru fase izan ditu: Erabiltzaile-interfaze grafikoaren moldakuntza, kontsulten sorkuntza automatikoa eta analisien sorkuntza. Atal honetan, bakoitzari dagokion diseinua azalduko dugu.

5.1 Relax NG eskemen antolakuntza

Proiektuaren izaera dela-eta, Relax NG eskemen antolakuntza aztertuz asiko gara. Eskemaren egitura konplexuak irakurketa zailtzen du: sei fitxategiz osatuta dago eta errefe-rentzia asko erabiltzen ditu.

Eskemak hainbat analisi definitzen ditu: lematizazio, morfosintaxi eta segmentazio-analisiak, alegia.

5.1 Kode-zatia: Lematizazio analisi baten adibidea.

```
<fs id="L-A-ADB-ARR-46" type="lematizazioa">
    <f name="forma">
        <str>Jatorrizko</str>
    </f>
    <f name="lema-osatua">
        <str>jatorriz</str>
    </f>
    <f name="ezaugarri-morfologikoak">
        <fs type="goimailako-ezaugarri-lista">
            <f name="KAT">
```

```
                <sym value="ADB"/>
            </f>
            <f name="AZP">
                <sym value="ARR"/>
            </f>
            <f name="KAS">
                <sym value="GEL"/>
            </f>
        </fs>
    </f>
</fs>
```

5.1 adibidean ikus daitekeen bezala, analisia aipatutako TEI-P4 gidalerroak definitutako `f` eta `fs` egituraz dago osatua. Analisi mota bakoitzak ezaugarri ezberdinak onartu arren, ezaugarri guztiek egitura bera jarraitzen dute:

5.2 Kode-zatia: Kasua ezaugarriaren Relax NG eskema.

```
<element name="f">
    <a:documentation>Kasua</a:documentation>
    <attribute name="name">
        <value>KAS</value>
    </attribute>
    <element name="sym">
        <attribute name="value">
            <choice>
                <value>ABL</value>
                <a:documentation>ablatiboa</a:documentation>
                <value>ABU</value>
                <a:documentation>adlatibo bukatuzkoa</a:documentation>
                ...
                <value>GEL</value>
                <a:documentation>genitibo leku-denborazkoa</a:documentation>
                ...
            </choice>
        </attribute>
        <empty/>
    </element>
</element>
```

Esan bezala, ezaugarri guztiak TEI-P4 gidalerroak jarraituz definitzen dira, eta gidalerro hauen arabera, ezaugarriak `f` elementuak dira. Ezaugarriaren lehen umea `a:documentation` motako nodoa da, bertan pertsonek irakurtzeko moduan ezaugarriari buruzko dokumentazioa (izenaren azalpen moduko bat, eskuarki) adierazten da; `<attribute name='name'>` nodoak ezaugarriaren izena adierazten du. Ezaugarriak hainbat motatakoak izan daitezke: testua, EDBL gakoa edo aurredefinitutako balioa dutenak; gainera, aldi berean balio

mota ezberdinak onartzen dituzten ezaugarriak ere badaude. Jarraian datu mota bakoitza aztertuko dugu.

Testu motako ezaugarriak edozein karaktere-kate onartzen dute balio moduan. Hona hemen eskeman duen egitura:

5.3 Kode-zatia: Relax NG eskemako testua datu-mota.

```
<element name="f">
    ...
    <ref name="mota.str"/>
</element>

<define name="mota.str">
    <element name="str">
        <data type="string"/>
    </element>
</define>
```

EDBL[1] gakoa, sarrera eta homografo-zenbakiaz osatuta dago. Sarreran ezaugarriaren balioa gordetzen da eta homografo-zenbakia unitate lexikalak unibokoki identifikatzeko erabiltzen da.

5.4 Kode-zatia: Relax NG eskemako EDBL gakoa datu-mota.

```
<element name="f">
    ...
    <ref name="mota.fs.gako"/>
</element>

<define name="mota.fs.gako">
    <element name="fs">
        <attribute name="type">
            <value>Gako</value>
        </attribute>
        <element name="f">
            <attribute name="name">
                <value>Sarrera</value>
            </attribute>
            <ref name="mota.str"/>
        </element>
        <element name="f">
            <attribute name="name">
                <value>Homografo-Id</value>
            </attribute>
```

[1] Euskararen Datu Base Lexikala.

```
            <ref name="mota.nbr"/>
        </element>
    </element>
</define>
```

Balio jakin bat duten balioak `sym` elementu batez daude ostatuta, eta elementuaren `value` atributuan balio posible ezberdinak gordetzen dira. Hurrengo eskema-zatian, adibidez, `ARR` eta `GAL` balioak onartzen dira. Balioen azpian, ezaugarrietan bezala, `a:documentation` nodoa dago, bertan, gaineko balioaren azalpena ematen da.

5.5 Kode-zatia: Relax NG eskemako balio multzo datu-mota.

```
<element name="f">
    ...
    <element name="sym">
        <attribute name="value">
            <choice>
                <value>ARR</value>
                <a:documentation>adb. arrunta</a:documentation>
                <value>GAL</value>
                <a:documentation>adb. galdetzailea</a:documentation>
            </choice>
        </attribute>
        <empty/>
    </element>
</element>
```

Ezaugarri guztien artean, badago bat berezia dena: kategoria. Izan ere, eskemako ezaugarri gehienak kategoriaren menpekoak dira; kategoria aditza denean, adibidez, biziduna edo neurgarria bezalako ezaugarriak ez dira analisian gehitzen. Eskeman, goi-mailako ezaugarri guztiak kategoriaren arabera sailkatzen dira, eta hau lortzeko eskemak honako egitura du:

5.6 Kode-zatia: Kategorien arteko banaketa Relax NG eskeman.

```
<define name="goimailako-kategoria-ezaugarriak">
    <choice>
        <ref name="ADB-kategoria-eta-goimailako-ezaugarriak"/>
        <ref name="ADI-kategoria-eta-goimailako-ezaugarriak"/>
        ...
    </choice>
</define>

<define name="ADB-kategoria-eta-goimailako-ezaugarriak">
    <!-- Kategoria -->
```

```
11      <ref name="f.kat.adb"/>
12      <!-- Azpikategoria -->
13      <optional>
14          <ref name="f.azpkat.adb"/>
15      </optional>
16      <!-- Ezaugarriak -->
17      <ref name="ADB-ezaugarriak"/>
18      <ref name="flexio-ezaugarriak"/>
19      <ref name="funtzio-sintaktikoak"/>
20      <ref name="hiztegi-sarrera-konplexuen-ezaugarriak"/>
21      <ref name="morfemetatik-datozen-ezaugarriak-ADB"/>
22  </define>
23
24  <define name="f.kat.adb">
25      <element name="f">
26          <attribute name="name">
27              <value>KAT</value>
28          </attribute>
29          <element name="sym">
30              <attribute name="value">
31                  <value>ADB</value>
32              </attribute>
33              <empty/>
34          </element>
35      </element>
36  </define>
37
38  <define name="f.kat.adi">
39      <element name="f">
40          <attribute name="name">
41              <value>KAT</value>
42          </attribute>
43          <element name="sym">
44              <attribute name="value">
45                  <value>ADI</value>
46              </attribute>
47              <empty/>
48          </element>
49      </element>
50  </define>
```

Ikus daitekeen bezala, ez dago kategoria ezaugarriaren definizio bakarra, kategoria mota ezberdin bakoitzeko bat baizik; eta honek asko zailtzen du ezaugarriak lortzea. Azpikategoriarekin ere antzera gertatzen da, kategoria bakoitzak bere azpikategoria propioak ditu eta.

Eskeman ere ikus daiteke kategoria bakoitzak ezaugarri mota ezberdinak dituela: batetik, aipatutako azpikategoria dago. Bestetik, kategoriaren ezaugarriak, flexio-ezaugarriak,

funtzio sintaktikoak, hiztegi-sarrera konplexuen ezaugarriak eta morfemetatik datozen ezaugarriak. Multzo bakoitzean goi-mailako ezaugarri ezberdinak biltzen dira.

5.2 Erabiltzaile-interfaze grafikoaren moldakuntza

Proiektuaren lehen helburua bilaketa-sistema aberastea zen. Hori lortu nahian Relax NG eskema batetik abiatuz bilaketa-interfazea automatikoki eta dinamikoki sortu da. Atal honetan eskemak erabiltzeko liburutegiaren eta bilaketa-sistema moldatuaren diseinua azalduko dugu.

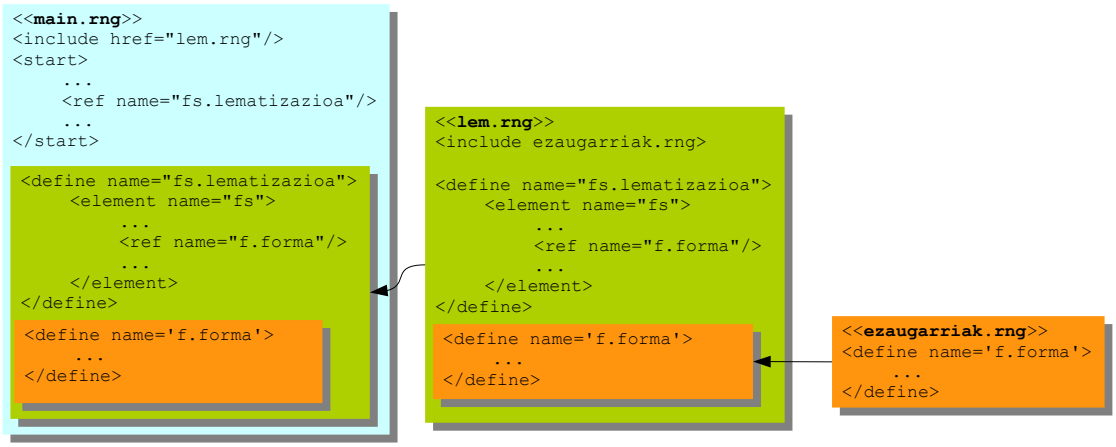
5.2.1 Relax NG eskemen analizatzailea

Kode errepikapen eta lan bikoizketa murriztearen helburuarekin, Relax NG eskematik informazioa lortzeko liburutegi bat garatu dugu. Liburutegi honek, Relax NG eskemekin erabili beharreko funtzio garrantzitsuenak eskaintzen ditu.

Analisien sorkuntzan bilaketan erabilitako liburutegia eta eskema berbera erabili behar direla kontuan hartuz, Relax NG eskema behin kargatu eta gordetzea erabaki da. Hori lortzeko, liburutegiak eskema kargatu eta gordetzen du, eta honekin lan egiteko funtzioak eskaintzen ditu.

Liburutegiaren funtzionalitateen artean, eskema kargatzea da garrantzitsuena. Esan bezala, eskema sei fitxategitan banatzen da, eta honekin lan egiteko, eskema kargatzean sei fitxategiak bakar batean biltzea erabaki zen. Horretarako, eskema kargatzeko funtzio bat gehitu da liburutegian. Funtzio honek eskemak osatzen duten fitxategiak ireki, fitxategiak batu eta XML dokumentu moduan itzultzen ditu.

5.1 Irudia: Fitxategiak batzeko prozesua.



Eskemaren karga liburutegiaren eskuetan uzten bada, erabiltzaileak ezin du jakin erabiltzeko prest noiz egongo den; izan ere, era asinkronoan kargatzen da. Hau konpontzeko, liburutegiak aukera ematen du eskema jasotakoan exekutatuko den funtzioa zehazteko.

5.7 Kode-zatia: Eskema kargatu ondoren exekutatu beharreko funtzioa nola zehaztu.

```
1  function loaded(){
2      alert("Eskema kargatu da!");
3      ...
4  }
5
6  var eskema = new rngSchema();
7  eskema.setOnReady(loaded);
8  eskema.open("main.rng");//Eskema kargatzean loaded funtzioa exekutatuko da.
```

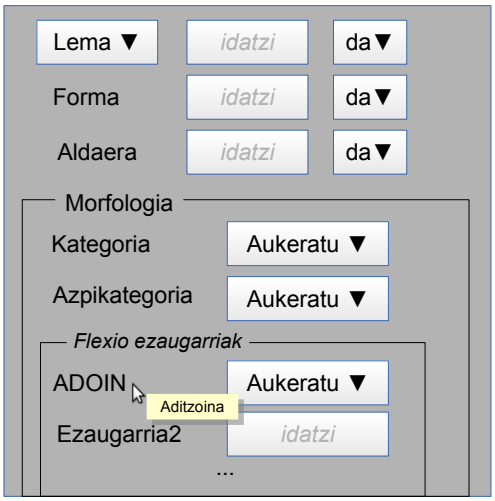
Goiko adibidean, eskema kargatu bezain pronto, `loaded` funtzioa exekutatuko da.

Behin eskema kargatzeko zailtasun guztiak konponduta, interfazearen diseinuarekin hasteko garaia da. Aipatu dugu kategoria bakoitzak ezaugarri mota ezberdinak dituela (flexio-ezaugarriak, morfemetatik datozenak etab.); erabiltzeko erraza izango den interfazea diseinatu nahian, interfazean ezaugarriek testuingurua behar zutela adostu zen, hau da, kategoria jakin baten ezaugarriak bistaratzeko orduan, hauek zein motatakoak diren azaldu behar du pantailan.

Erabiltzaileak hizkuntzalaritzan adituak izango direnez, pantailan ezaugarrien izena (KAT kategoriaren kasuan, AZP azpikategoriaren kasuan etab.) azaltzea erabaki zen, era-

biltzaileentzat ezagunak izango baitziren. Hala ere, laguntza gisa, `a:documentation` nodoetan zegoen testua *tooltip*[2] moduan eskaintzea erabaki zen, honela, identifikadore bat ezezaguna izanez gero, zer den jakitea erraza izango zen. Hona hemen interfazearen hasierako diseinua:

5.2 Irudia: Interfazearen diseinua.



Eskema kargatzeko arazoa konpondu eta interfazearen beharrak aztertu ondoren, datuak eskematik irakurtzeko garaia da. Ezaugarri guztiak kategoriaren menpekoak direla aipatu dugu, eta horregatik, kategoriak tratamendu berezia jaso behar du. Tratamendu berezi honek arriskuan jartzen du proiektua, izan ere, helburua ezaugarriak ahalik eta modurik orokorrenean lortzea da, eta hori ezinezkoa da ezaugarri batzuek tratamendu berezia behar dutenean.

Ezaugarrien arteko ezberdintasunak eta interfazearen beharrak zirela-eta, ezaugarri mota ezberdinei tratamendu ezberdina ematea erabaki zen. Honela, liburutegiak ezaugarriak lortzeko 7 funtzio eskaintzen ditu: kategoria zerrenda lortzeko, kategoria jakin baten azpikategoriak lortzeko, kategoria jakin baten ezaugarriak lortzeko, morfemetatik datozen ezaugarriak lortzeko, hiztegi-sarrera konplexuak lortzeko, funtzio sintaktikoak lortzeko eta flexio-ezaugarriak lortzeko funtzioak, alegia. Bakoitzak estrategia ezberdina jarraitzen du bere beharren arabera.

[2]Interfaze grafikoetan erabiltzen diren laguntza-testuak. Puxika moduan azaltzen dira elementu baten gainean kurtsorea jartzean.

Ezaugarrien bilaketan, Relax NG erreferentzia asko erabiltzen da eta horietako bat aurkitzen den bakoitzean, bere definizioa bilatu behar da. Lan hau motela eta errepikakorra da, eta interfazearen sorkuntzak azkarra izan behar duenez, ezaugarriak dauden azpizuhaitzaren kopia egin, eta bertako erreferentzia guztiak haien definizioekin ordezkatzea erabaki da. Honela, memoriaren kaltetan, azkartasunean irabazten dugu.

Azpizuhaitzeko erreferentziak ordezkatzeko orduan, erreferentziaren tokian definizio-nodo osoa gehitu beharra dago, izan ere, nahiz eta logikoki zentzuzkoa iruditu erreferentzia dagoen lekuan definizioaren nodoak gehitzea, honek eskema aldatzen du. Hurrengo adibideetan ikus daiteke:

5.8 Kode-zatia: Azpizuhaitzaren erreferentziak kentzeko adibidea: jatorrizko eskema.

```
<choice>
    <ref name='A'>
    <ref name='B'>
</choice>

<define name='A'>
    <element name='el1'>
</define>

<define name='B'>
    <element name='el2'>
    <element name='el3'>
</define>
```

5.9 Kode-zatia: Azpizuhaitzaren erreferentziak kentzeko adibidea: `define` nodo osoa gehitu.

```
<choice>
    <define name='A'>
        <element name='el1'>
    </define>
    <define name='B'>
        <element name='el2'>
        <element name='el3'>
    </define>
</choice>
```

Definizioa bera gehitu beharrean, definizioaren nodoak gehituko bagenitu, honela geldituko litzateke:

5.10 Kode-zatia: Azpizuhaitzaren erreferentziak kentzeko adibidea: definearen nodoak gehitu.

```
<choice>
    <element name='el1'>
    <element name='el2'>
    <element name='el3'>
</choice>
```

Baina honek eskema aldatzen du; jatorrizko eskeman eta 5.9 kode-zatian, `choice`-an bi elementuren artean aukeratu behar da; honetan, berriz, hiruren artean.

Liburutegiaren garapenean zehar eskematik lor daitekeen informazioa murriztua dela eta datu batzuk eskuz zehaztu behar direla ikusi da. Horretarako, konfigurazio-fitxategi bat erabiltzea erabaki da; bertan eskuz zehaztu beharreko datu guztiak zentralizatzeko helburuarekin.

Jarraian liburutegiak eskaintzen dituen funtzio ezberdinak azalduko ditugu: batetik, lehen aipatu ditugun, ezaugarriak lortzeko funtzioak daude; hauek, eskematik informazioa lortzeko pentsatuta daude. Bestetik, DOM ereduan dauden funtzioetan (`getElementsByTagName`, `childNodes` etab.) oinarrituz, eskema zuzenean irakurri ahal izateko funtzioak inplementatu dira.

Kategoriak eta azpikategoriak lortu

Aipatutako konfigurazio-fitxategiaren bidez kategoriak lortzea lan erraza da. Konfigurazio-fitxategian kategoriaren izena (KAT) gorde dugu; kategoria zerrenda lortzeko, azpizuhaitzeko KAT izen guztiak bilatu baino ez zen behar. Funtzio honek, kategorien identifikadore eta izenen zerrenda itzultzen du.

5.11 Kode-zatia: Adberbio kategoriaren eskema.

```
<element name="f">
    <a:documentation>Kategoria</a:documentation>
    <attribute name="name">
        <value>KAT</value> <!-- Balio hau aurkitu ondoren, kategoria-izena eta
    kategoria-azalpena lortzea oso erraza da. -->
    </attribute>
    <element name="sym">
        <a:documentation>adberbioa</a:documentation> <!-- Kategoria-azalpena -->
        <attribute name="value">
            <value>ADB</value> <!-- Kategoria-izena -->
```

```
        </attribute>
        <empty/>
    </element>
</element>
```

Azpikategoriaren kasuan ere, bere izena (AZP) dago konfigurazio-fitxategian. Kategoria baten azpikategoria lortzeko, kategoria horren azpizuhaitzaren barruan dagoen AZP atributua bilatu, eta kategorian bezala, izen eta azalpen-balioak itzultzen dira.

Beste ezaugarriak lortu

Kategoria eta azpikategoriaren kasuak tratatu ondoren, beste ezaugarriak nola lortu diren ikusteko garaia da. Ezaugarri hauek lortzeko, konfigurazio fitxategian ezaugarri mota ezberdinak aurkitzeko datuak gehitu behar izan dira. Hauek erabiliz, oso antzekoak diren hainbat funtzio garatu dira: kategoriaren ezaugarriak lortzeko, flexio-ezaugarriak lortzeko, funtzio sintaktikoak lortzeko, hiztegi-sarrera konplexuen ezaugarriak lortzeko eta morfemetatik datozen ezaugarriak lortzeko. Kasu hauetan guztietan honelako egiturekin aurkitzen gara:

5.12 Kode-zatia: Izenaren ezaugarrien eskema.

```
<define name="IZE-ezaugarriak">
    <optional>
        <ref name="f.plurala"/>
    </optional>
    <optional>
        <ref name="f.biziduna"/>
    </optional>
    <optional>
        <ref name="f.neurgarria"/>
    </optional>
    <optional>
        <ref name="f.zenbakarria"/>
    </optional>
</define>
```

Ezaugarri guztiak eskeman `<element name='f'>` nodoak direla aipatu dugu. Hala ere, badaude ezaugarriak ez diren eta egitura hori duten nodoak eskeman. Gako mota, adibidez, bi `<element name='f'>` nodoz osatuta dago, sarrera eta homografo-zenbakia. Ezaugarriak zirenak eta ez zirenak ezberdintzeko, ezaugarrien gurasoetan erre-

paratu behar izan genuen; ezaugarriak ez zirenen gurasoak `element` motakoak ziren; ezaugarrienak, ordea, `define` motakoak.

Ezaugarri hauen kasuan, balioak itzuli beharrean ezaugarrien erroaren nodoa eta ezaugarria aukerazkoa edo beharrezkoa zen itzultzea erabaki zen. Kategoria eta azpikategoriarekin ez bezala, hauetan, datu gehiago ezagutzea interesatzen zitzaigulako (zein motatakoak ziren adibidez).

Eskema manipulatzeko funtzioak

Eskematik datuak irakurtzeko funtzioak garatu ondoren, eskema linealki korritzeko beharra egongo zela ikusi zen, batez ere proiektuaren amaieran (analisien sorkuntza). HTML dokumentu bat irakurtzeko `getElementsByTagName` edo `childNodes` bezalako funtzioak erabiltzen dira. Gure kasuan, Relax NG eskema baten gainean aplikatu nahi ziren, eta horretarako hauen moldaketa egin da. Moldaketa hauek erreferentzien kudeaketarekin dute zerikusia gehienbat. `getElementsByTagName` funtzioak pasatutako izena duten nodoak itzultzen ditu, baina erreferentzia bat aurkitzean ez du erreferentzia hori jarraituko.

5.13 Kode-zatia: childNodes() funtzioaren adibide bat.

```
<optional>
    <ref name="a">
    <element name="b"/>
</optional>

<define name="a">
    <element name="a"/>
</define>
```

Goiko kasuan, `optional` nodoan `getElementsByTagName("element")` funtzioa aplikatzean, `<element name='a'/>` eta `<element name='b'/>` jasotzea da helburua. Garatutako funtzioak honakoak dira: `firstChild`, `hasChildNodes`, `childNodes` eta `getElementsByTagName`. Sobera ezagunak direnez, hauek zer egiten duten ez dugu azalduko. Hala ere, A eranskinean liburutegiaren funtzioak nola erabiltzen diren azaltzen da.

Beharrezkoak izango diren beste funtzio batzuk ere gehitu dira: erreferentzia bat emanda, erreferentzia horren definizioa aurkitzeko funtzioa, eta liburutegitik kargatutako eskemaren `start` nodoa itzultzen duen beste bat, eskema zuzenean atzitzeko. Aipatutako funtzioekin analisien sorkuntzan egin beharreko lan asko aurreratzea espero da.

5.2.2 Interfaze-sorkuntza

Relax NG eskemak irakurtzeko liburutegia garatzean interfazea sortzeko lan gehiena egin da. Interfazea sortzeko garaian, ezaugarri bakoitza pantailaratzea baino ez da behar.

Kategoria eta azpikategoriaren kasuan, liburutegitik balio zerrenda bat lortzen dugu, eta balio zerrenda hori pantailaratzeko lista hedagarriak erabili. Beste ezaugarrien kasuan, ordea, nodo zerrenda jasotzen dugu eta hauek pantailaratzeko nodo horiek tratatu behar ditugu; beraz, ezaugarri horiek HTML bihurtzea da hurrengo pausoa. Horretarako, ezaugarriak motaren arabera sailkatu ditugu: edozein testu onartzen dutenak, EDBL gako motakoak, bai/ez motakoak eta aurredefinitutako balio batzuen artean bat aukeratzea eskatzen dutenak. Mota hori, jasotako ezaugarriak aztertuz lortzen genuen:

5.14 Kode-zatia: Relax NG eskemak irakurtzeko liburutegiak itzultzen dituen ezaugarrietako bat.

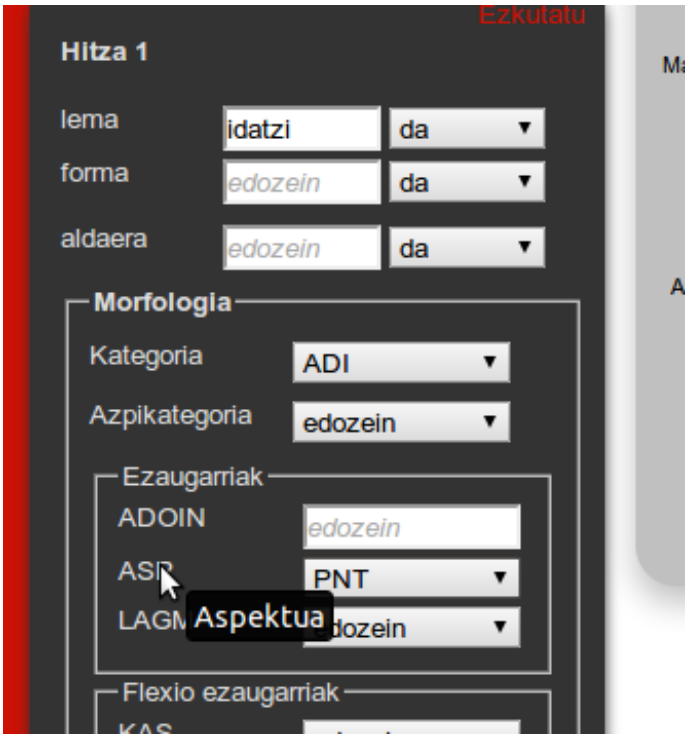
```
<element name="f">
    <attribute name="name">
        <value>ADZ</value>
    </attribute>
    <ref name="mota.fs.gako"/> <!-- EDBL gako motako ezaugarria -->
</element>
```

Mota bakoitzak, era ezberdinean pantailaratzen da: bai/ez motakoak eta aurredefinitutako balioak dituztenak pantailaratzeko lista hedagarriak erabiltzen dira; eta edozein testu onartzen dutenentzat testu-kutxak.

EDBL gako motakoa ere testu-kutxa moduan pantailaratzea erabaki da. Mota honetako ezaugarriak bi balio eduki arren, sarrera eta homografo-zenbakia, bigarrena alde batera uztea eta interfazean sarrera bakarrik eskatzea erabaki da; izan ere, homografo-zenbakia unitate lexikalak unibokoki identifikatzeko Euskararen Datu-Base Lexikalak erabiltzen duen kodea da, eta hau kontsultetan ez zela erabiliko aurreikusi zen.

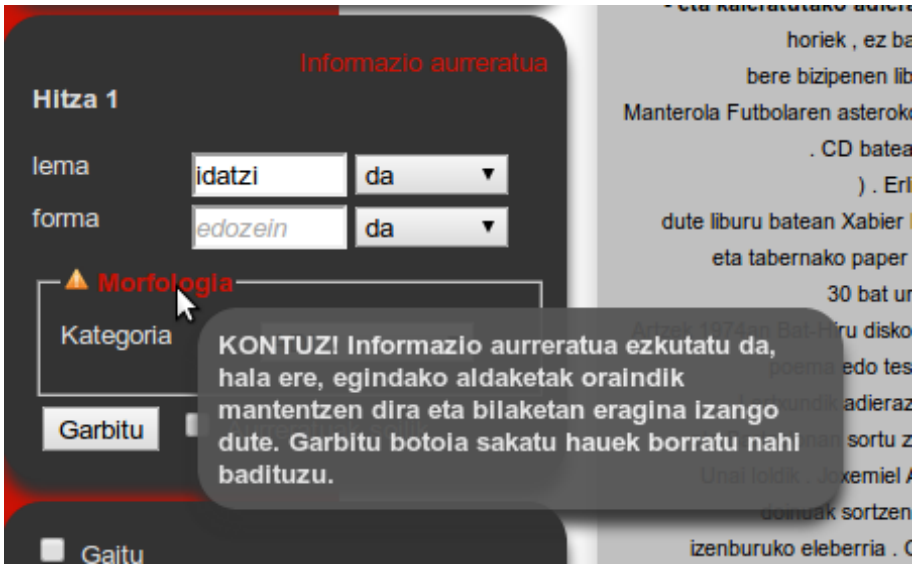
Ezaugarriak HTML bihurtzeko, ezaugarrien zerrenda bat jaso eta HTML kodea itzultzen duen programa garatu da. Ondoren HTML kode hori kontsulta interfazean gehitzen dugu, honelako interfazea lortuz:

5.3 Irudia: Garatutako interfazea.



Erabilgarritasunari dagokionez, laguntza-mezuak gehitu dira ezaugarri guztietan, horretarako eskemako `a:documentation` testuak erabiliz. Bestalde, jasotako prototipoaren interfazean, bilaketa aurreratua azaldu eta ezkutatzeko aukera ematen zen; gure diseinuan hori mantentzea erabaki da. Hala ere, bilaketa aurreratua ezkutatzean, posible da balioren bat duten ezaugarriak ere ezkutatzea; eta bilaketa egitean, ezkutatutako balio horiek kontuan hartuko direnez, erabiltzailea ohartarazteko mezu bat agertzea erabaki dugu.

5.4 Irudia: Ezkutatutako ezaugarriak badaudela ohartarazteko mezua.



5.3 Kontsulten sorkuntza automatikoa

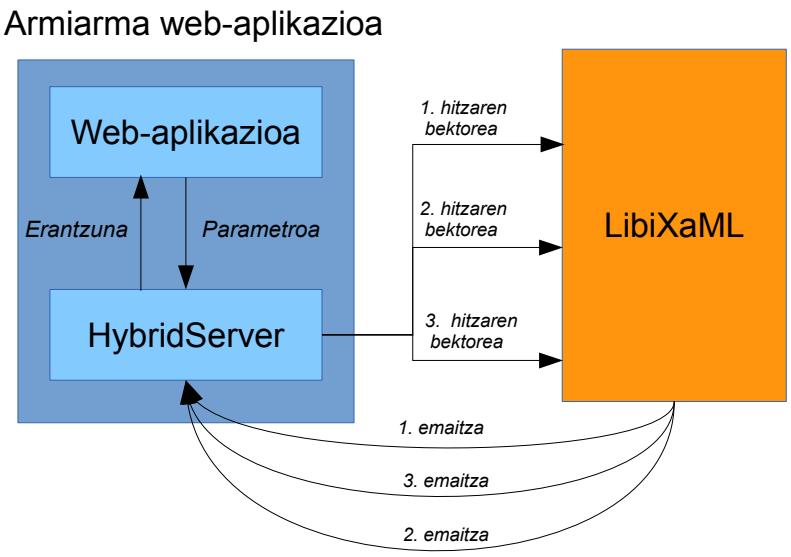
5.3.1 Bilaketa-sistemaren moldaketa

Interfazea sortu ondoren, bilaketak egiteko garaia da. Jasotako prototipoan kontsultak egiteko *HybridServer* moduluari parametro bat pasatzen zitzaion. Bilaketa-eremu berriak gehitu ahal izateko, parametro hori, *HybridServer* modulua eta *LibiXaML* liburutegia moldatu behar izan ditugu.

Parametroari dagokionez, prototipoaren egitura bera jarraitzea erabaki da, honela, *HybridServer* moduluan egin beharreko aldaketak gutxiago izango direlako (ikus D eranskina parametroa ikusteko).

Behin parametroa moldatuta, *HybridServer* modulua moldatzeko garaia iritsi zen. *HybridServer* moduluaren eginkizuna, parametro hori hartu eta hitz bakoitza (nagusia eta bere testuingurua osatzen duten beste biak) egitura batean gordetzea da. Ondoren, egitura horiek hash taula moduan *LibiXaML*ri banaka pasatzeko. *LibiXaML*k datu basean hitz bakoitzarekin bilaketa egin, eta erantzunak *HybridServer* moduluari pasatzen dizkio. Ondoren, *HybridServer* moduluak erantzun horiek jaso eta hitz nagusiaren erantzuna beste biekin filtratzen du. Azkenik, web-aplikazioari bilaketaren erantzuna bidaltzen dio.

5.5 Irudia: Web-aplikazioaren arkitektura.



HybridServer moduluan aldaketa gutxi egin behar izan dira. Garrantzitsuena, hitzak gordetzen diren egitura moldatzea izan da. Egitura horrek, parametroko hitz bakoitza osatzen duten ezaugarri guztiak biltzen ditu, bertan, kontsultan erabiltzen diren ezaugarri berriak gehitu dira.

LibiXaML liburutegia, ezaugarrien hash taula jaso eta horrekin XML datu-basean egingo den kontsulta sortzeaz arduratzen da. Kontsulta *XQuery* erabiliz egiten da: honek, XPath espresioak erabiltzen ditu XML dokumentuak atzitzeko. Bilaketa-interfazean egindako aldaketen ondorioz, kontsulta egiteko *XQuery* espresioa sortzen duen programa moldatu behar izan dugu.

XQuery espresioak erabiltzen dituen XPath espresio horietako bakoitza ezaugarri bati dagokio, hau da, galde daitekeen ezaugarri bakoitzak XPath espresio bat du.

5.15 Kode-zatia: Lematizazio analisi baten adibidea.

```
<fs type="lematizazioa">
    <f name="lema-osatua">
        <str>idatzi</str>
    </f>
    <f name="forma">
```

```
        <str>idatziak</str>
    </f>
</fs>
```

Aurreko analisian, adibidez, lema osatua atzitzeko honako XPath espresioa erabiliko genuke:

5.16 Kode-zatia: Lema osatua atziteko erabilitako XPath espresioa.

```
f[@name='lema-osatua']/str = 'idatzi'
```

XPath honek, ordea, lema osatuak idatzi balioa duenean soilik balio du, horregatik, honelako XPath espresioak erabiltzen dira:

5.17 Kode-zatia: Lema atziteko erabilitako XPath espresioa

```
f[@name='lema-osatua']/str = '$'
```

Bilaketa egiteko garaian, *LibiXaML*k behar duen XPath espresioa hartu, eta $ ikurraren tokian, *HybridServer* modulutik jasotako balioa jarriko du.

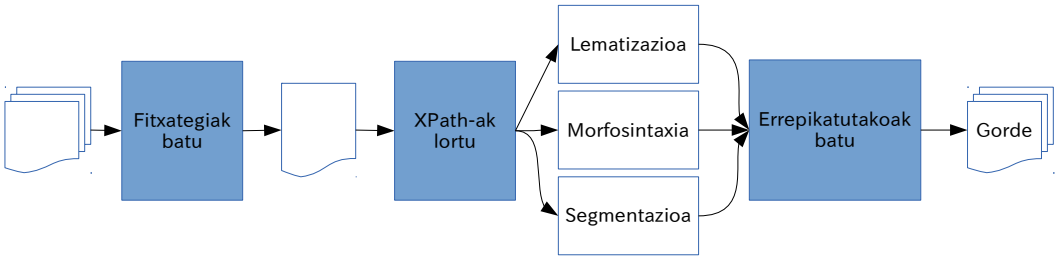
5.3.2 Kontsulten sorkuntza automatikoa

Bilaketa sistema moldatu ondoren berrogei ezaugarri genituen bilaketan, eta ezaugarri horietako bakoitzak bere XPath espresioa behar zuen *LibiXaML* liburutegian; gainera, eskema aldatuz gero, XPath horiek berriro lortu beharko ziren. Horregatik, XPath horiek automatikoki sortzeko beharra ikusi zen. Helburua, Relax NG eskematik abiatuz XPath espresioak sortu, fitxategi batean gorde, eta *LibiXaML* liburutegiak XPath bat behar zuenean bertatik hartzea zen.

Testuak prozesatzeko eskaintzen dituen abantailengatik, Perl lengoaiaz script bat idaztea erabaki zen. Egindako programak, Relax NG eskema irakurri eta hiru fitxategi sortzen ditu: analisi mota bakoitzeko bat (lematizazio, morfosintaxi eta segmentazio). Fitxategi horietan `ezaugarri_izen:XPath` pareak gordetzen ditu, honela *LibiXaML*k era errazean irakurtzen dituelako.

XPath-ak lortzeko hainbat pauso jarraitzen dira: lehenik, Relax NG eskema osatzen duten fitxategiak irakurri eta, Javascript liburutegiak egiten duen antzera, fitxategi bakar batean biltzen dira; honela, hauek tratatzea askoz errazagoa da. Ondoren, programa batek eskemako XPath espresio guztiak lortzen ditu; lortutakoen artean errepikatutako ezaugarriak egon daitezke (ezaugarri batek bi datu-mota onartzen dituenean), eta horregatik beste programa batek errepikatutako ezaugarrien XPath-ak batzen ditu. Azkenik, XPath-ak aipatutako hiru fitxategietan gordetzen dira.

5.6 Irudia: XPath-ak lortzeko jarraitzen diren pausoak.



Jarraian XPath-ak lortzeko eta errepikatutako ezaugarrien XPath-ak batzeko programak azalduko ditugu: XPath-ak lortzeko programak diseinu errekurtsiboa du, sarrera moduan, tratatu beharreko nodoa, nodo horren gurasoaren XPath-a eta zein analisi motari dagokion jasotzen du; eta jasotzen duen XPath-ean, jasotako nodoari dagokion XPath-a gehitzeaz eta nodoaren ume guztiekin dei errekurtsiboa egiteaz arduratzen da. Aurkitzen dituen XPath-ak taula batean gordetzen ditu, ondoren errepikatutakoak tratatu ahal izateko.

Beraz, programa errekurtsibo horren lan nagusia nodo bakar baten XPath-a lortzea eta aurreko XPath-ari gehitzea da. Horretarako, Relax NG eskeman aurki ditzakegun nodo ezberdinak aztertuko ditugu:

- **`element`** nodoak: XML elementuak definitzeko dira eta `name` atributuan elementuaren izena daramate. Mota honetako nodo bat aurkitzean, XPath espresioan elementuaren izena gehitu beharko dugu.
- **`attribute`** nodoak: elementu baten atributua definitzen dute, nodo hauek beti `element` nodo baten umeak izango dira.

- **`value`** nodoak: atributuen balioak zehazteko erabiltzen dira.
- **`ref`** nodoak: erreferentziak dira, eta hauek ez dute XPath-ean eragin zuzenik; hala ere, bereziki tratatu behar dira hauen definiziora jo behar baita erreferentzia bat aurkitzean.
- **`data`** nodoak: Relax NG datu-motak erabiltzeko dira. Nodo hauek umerik ez dute, eta horregatik XPath gehienak hauetako batean amaitzen dira.
- Besteek: `oneOrMore`, `zeroOrMore`, `optional`, `choice`, `group` eta `a:documentation`, ez dute XPath-en sorkuntzan eraginik.

XPath-ak sortzeko orduan elementuak `/` ikurraz banatzen dira; atributuak, ordea, kortxete artean jartzen dira elementuen atzetik; horregatik, hauen tratamendua elementuenarekin batera egitea erabaki zen, eta honela, elementu bat aurkitzean, bere atributu guztiak lortu eta elementuaren atzetik gehitzen dira honen umeak tratatu baino lehen.

XPath-ak sortzeko orduan, kontsultan behar diren $ ikurren kokapena aztertu behar izan da; izan ere, ikur horrek kontsulta balioak non jarri behar diren adierazten du. Relax NG eskemak aztertuta, ikur hori bi kasutan azaltzen dela konturatu gara: atributu baten balioa ezezaguna denean, hau da, `data` motako nodo bat aurkitzean; eta atributu batek balio asko har ditzakenean, hau da, atributuaren lehen umea `choice` motako nodoa denean.

5.18 Kode-zatia: XPath-en sorkuntzan $ ikurrak non jarri behar diren azaltzeko adibidea.

```
<element name="f">
    <attribute name="name">
        <data type="string"/>
    </attribute>
    <element name="sym">
        <attribute name="value">
            <choice>
                <value>lehen balioa</value>
                <value>bigarren balioa</value>
            </choice>
        </attribute>
    </element>
</element>
```

Goiko adibideko eskema jasota, honako XPath-a sortuko dugu:
`f[@name='$']/sym[@value='$'].`

Ezaugarrien balioak gehitzeko $ ikurren kokapena zehaztu ondoren, XPath-ak noiz gorde zehaztu behar dugu. Horretarako, ezaugarrien eskema nola amaitzen den aztertu dugu: ezaugarri guztietan definitzen den azkeneko gauza datu-mota da, beraz, datu-mota horiek XPath espresiora gehitu ondoren espresioa gordetzen badugu ezaugarri guztien XPath-a lortuko dugu. XPath-ak gorde behar diren kasuak hauek dira: `data` nodo baten ondoren, eta `sym`, `plus` edo `minus` atributu balioen ondoren. Izan ere, `data` nodoak testu motako balioak errepresentatzeko, `sym` elementuak balio jakin batzuk errepresentatzeko, eta `plus` eta `minus` elementuak bai/ez motako datuak errepresentatzeko erabiltzen dira.

Behin hau zehaztuta, XPath-en sorkuntzari ekin genion. XPath ezberdinak sortzen genituen analisi mota bakoitzeko:

5.19 Kode-zatia: Analisi mota ezberdinak banatzen diren eskema-zatia.

```
<element name="body">
    <choice>
        <!-- Segmentazioa -->
        <oneOrMore>
            <element name="p">
                <ref name="fs.segmentazioa"/>
            </element>
        </oneOrMore>
        <!-- Morfosintaxia -->
        <oneOrMore>
            <element name="p">
                <ref name="fs.morfosintaxia"/>
            </element>
        </oneOrMore>
        <!-- Lematizazioa -->
        <oneOrMore>
            <element name="p">
                <ref name="fs.lematizazioa"/>
            </element>
        </oneOrMore>
    </choice>
</element>
```

Analisi mota bakoitzaren errotik hasiz, elementuak tratatu eta XPath-ak eraikitzen dira. Hona hemen algoritmoa:

5.20 Kode-zatia: XPath-ak sortzeko algoritmoa.

```
sortuXPath (nodoa, XPath, analisiMota)
    if (nodoa == element)
        XPath + '/' + nodoaren_izena
```

```
        XPath + '['
        for (nodoa.atributuNodoak)
            baldin (atributuak balio asko)
                XPath + (atributuIzena = '$')
            ----------
            bestela
                XPath + (atributu_izena = 'atributu_balioa')
            ----------
        ----------
        XPath + ']'
        baldin (nodoaren_izena == "sym" edo "plus" edo "minus")
            GORDE(XPath)
        ----------
        for (nodoaren_umeak)
            sortuXPath(umea, XPath, analisiMota)
        ----------
    ----------
    else if (nodoa == ref)
        bilatu_definea
        for (definearen_umeak)
            sortuXPath(umea, XPath, analisiMota)
        ----------
    ----------
    else if (nodoa == data)
        XPath + (='$')
        GORDE(XPath)
    ----------
    else
        for (nodoaren_umeak)
            sortuXPath(umea, XPath, analisiMota)
        ----------
    ----------
----------
```

XPath-en sorkuntzan aurkitutako lehenengo arazoa, bilaketa-interfazean bezala, eskeman kategoria eta azpikategoriak duten egitura berezia izan zen. Izan ere, kategoria ezberdin bakoitzeko behin definitzen dira; eta XPath espresioak lortzeko orduan honakoa lortzen genuen:

5.21 Kode-zatia: XPath sorkuntza: kategoria eta azpikategoriarekin arazoak.

```
(...)/fs[@type='goimailako-ezaugarri-lista']/f[@name='KAT']/sym[@value='ADI']
(...)/fs[@type='goimailako-ezaugarri-lista']/f[@name='KAT']/sym[@value='ADB']
(...)/fs[@type='goimailako-ezaugarri-lista']/f[@name='KAT']/sym[@value='IZE']
                                        ...
(...)/fs[@type='goimailako-ezaugarri-lista']/f[@name='AZP']/sym[@value='ARR']
(...)/fs[@type='goimailako-ezaugarri-lista']/f[@name='AZP']/sym[@value='GAL']
(...)/fs[@type='goimailako-ezaugarri-lista']/f[@name='AZP']/sym[@value='FAK']
                                        ...
```

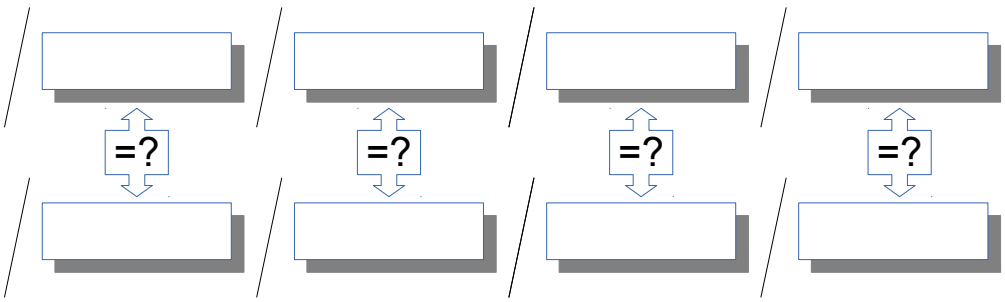
Emaitza hauekin, kategoria eta azpikategoriaren balio ezberdin bakoitzeko XPath ezberdina lortuko genuke; baina, bilaketa sistemak ezaugarri bakoitzeko XPath espresio bakarra erabiltzen duenez, kasu hauek tratatzeko metodoa bilatu behar izan dugu. Errepikatutako ezaugarriak batzeko programa, ezaugarriek izan ditzaketen balio mota ezberdinak, eta ondorioz XPath ezberdinak, batzea da. Kasu hau ordea ezberdina da, honetan XPath-ak batzeaz gain, kontsultan erabili beharreko $ ikurra ere gehitu behar baitugu. Helburua hau lortzea da:

5.22 Kode-zatia: XPath-en sorkuntza: kategoria eta azpikategoriko XPath-ak.

```
1 (...)/fs[@type='goimailako-ezaugarri-lista']/f[@name='KAT']/sym[@value='$']
2 (...)/fs[@type='goimailako-ezaugarri-lista']/f[@name='AZP']/sym[@value='$']
```

XPath-ak lortu eta errepikatutako ezaugarriak batu bitartean, lortutako XPath espresioak taula batean gordetzen ditugu; aurreko arazoa XPath-ak taulan gordetzean konpontzea erabaki genuen. Horretarako, XPath espresio bat taulan gordetzean, bertako espresio guztiekin konparatzen dugu. Konparaketa egiteko, XPath espresioak `'//'` ikurra erabiliz banatu, eta banatutakoan lortutako zatiak banaka konparatzen ditugu.

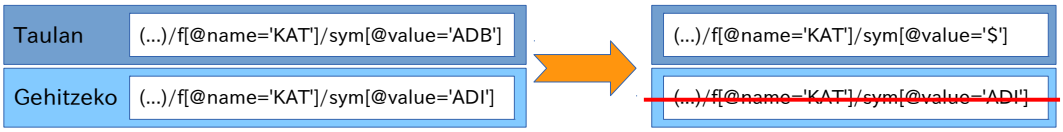
5.7 Irudia: XPath-ak konparatzeko metodoa.



Konparaketa horretan oso antzekoak diren XPath-ak bilatzen ditugu. Gure kasuan antzeko XPath-ak honela definitzen ditugu: luzera berbera eta gehienez espresioaren zati bat izango dute ezberdina; gainera, ezberdina den zati horrek honako forma izan beharko du: `elementu_izen[@value='balioa']`.

Baldintza hauek betetzen dituzten bi espresio aurkitzean, ezaugarri izen berdina (`value` atributuaren aurretik beti ezaugarriaren izena egongo da: `f[@name = 'izena']`) eta balio ezberdina duten bi XPath izango ditugu. Hau gertatzean, taulan gordeta dagoen XPath espresioko `value` atributuaren balioan $ ikurra gehitzen dugu, eta honela kategoria eta azpikategoriarekin genituen arazoak konpontzen ditugu.

5.8 Irudia: Antzeko XPath-en tratamendua.



Puntu honetan, hiru taula ditugu; bakoitzean analisi mota bati dagozkion XPath-ak gordetzen dira. Hurrengo helburua, errepikatutako ezaugarrien XPath-ak, baliokide izango den beste XPath batean bihurtzea da, izan ere, honelako XPath espresioak ditugu taulan:

5.23 Kode-zatia: XPath-en sorkuntza: errepikatutako espresio baten adibidea (AURL).

```
1 f[@name='ezaugarri-morfologikoak']/fs[@type='goimailako-ezaugarri-lista']/f[@name='
      AURL']/str='$'
2 f[@name='ezaugarri-morfologikoak']/fs[@type='goimailako-ezaugarri-lista']/f[@name='
      AURL']/fs[@type='Gako']/f[@name='Sarrera']/str='$'
3
4 f[@name='ezaugarri-morfologikoak']/fs[@type='goimailako-ezaugarri-lista']/f[@name='BIZ
      ']/plus
5 f[@name='ezaugarri-morfologikoak']/fs[@type='goimailako-ezaugarri-lista']/f[@name='BIZ
      ']/minus
```

Aurreko adibidean bi ezaugarriren XPath espresioak daude. Lehena, aurrizki lexikala (AURL) ezaugarriaren bi espresio dira; batek, ezaugarri honen karaktere-kate mota errepresentatzen du, eta besteak ezaugarriaren EDBL gako motako sarrera eremua (gogoratu homografo-identifikadorea ez dela kontsultan galdetzen). Bien espresioen baliokide den espresioa honako hau da:

5.24 Kode-zatia: XPath sorkuntza: AURL ezaugarriaren espresio baliokidea.

```
1 f[@name='ezaugarri-morfologikoak']/fs[@type='goimailako-ezaugarri-lista']/f[@name='
      AURL']//str='$'
```

Bigarrena, ordea, biziduna (BIZ) ezaugarriari dagokio, eta honetan ez dugu espresio baliokiderik sortu beharko. Izan ere, bai/ez motako ezaugarria da, `plus` eta `minus` balioak dituena, eta kontsultarako biak beharko ditugu.

Egin beharreko lehenengo gauza errepikatutako ezaugarriak aurkitzea da; horretarako, XPath-ak zein ezaugarriri dagokion jakin beharko dugu. XPath espresioen egitura aztertuz honako hau ondoriozta dezakegu:

- Ezaugarria zehazten duen testua honako hau da: `(...)/f[@name='ezaugarria']/(...)`.
- XPath espresio bakoitzean `f` elementu bat baino gehiago egon arren, oro har, ezaugarria adierazten duena XPath espresioko azken `f`-a izango da.
- Honen salbuespen izango dira EDBL gako motako ezaugarriak: hauen azken `f`-ak `Sarrera` izena du.

Hauek kontuan izanik, XPath sententzia baten ezaugarri-izena lortuko zuen espresio erregularra erabiltzea erabaki dugu.

5.25 Kode-zatia: Ezaugarriak lortzeko espresio erregularra.

```
1 .*f\[@name='(((?!Sarrera|').)*)'[^\]]*\]/((?!f\[@name='((?!Sarrera).)*').)*$
```

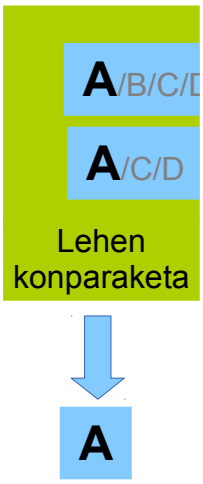
Aurreko espresio erregularrak `Sarrera` balioa ez duen azken `f` elementua bilatzen du. Perl lengoaiak testu baten gainean espresio erregular bat aplikatzeko eta ondoren espresio erregularraren parentesien arteko balioak lortzeko aukera ematen du; honela, ezaugarrien izena lortzea erraza da.

5.26 Kode-zatia: XPath batetik ezaugarri-izena lortzeko modua.

```
1 $expr=m/.*f\[@name='(((?!Sarrera|').)*)'[^\]]*\]/((?!f\[@name='((?!Sarrera).)*').)*$/;
2 $XPatha =~ $expr;
3 $ezaugarriIzena = $1;
4 # $1-ek lehen parentesi arteko balioa hartzen du; gure kasuan, ezaugarri izena.
```

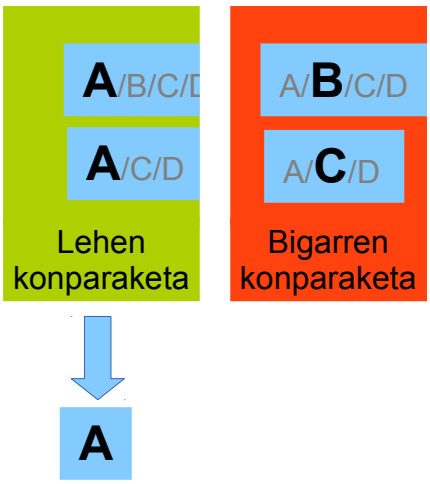
Izen bereko bi ezaugarri aurkitu ondoren, bien baliokide den XPath-a aurkitzeko metodoa honako hau da: XPath-ak `'//'` erabiliz zatitzen dira, eta zati bakoitza konparatzen goaz. Algoritmoa azaltzeko `A/B/C/D` eta `A/C/D` XPath-en gainean aplikatuko dugu. Lehenik, XPath bakoitzaren lehen bi zatiak konparatzen ditugu:

5.9 Irudia: A/B/C/D eta A/C/D XPath-ak batzeko metodoa.



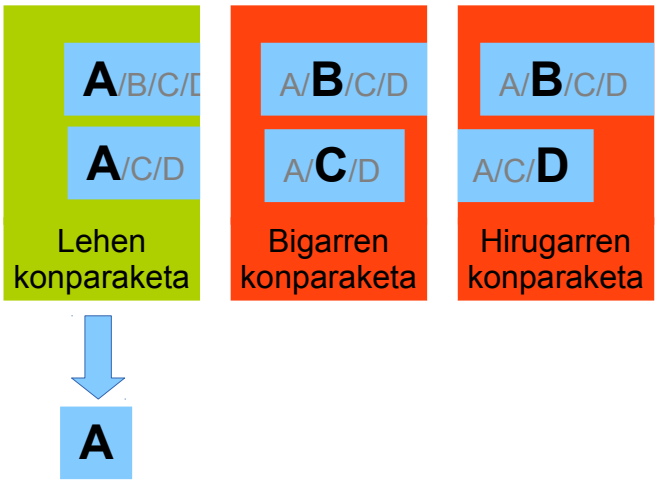
Berdinak direnez, `A` XPath baliokidera gehitzen dugu eta bi espresioetan aurrera egiten dugu:

5.10 Irudia: A/B/C/D eta A/C/D XPath-ak batzeko metodoa.



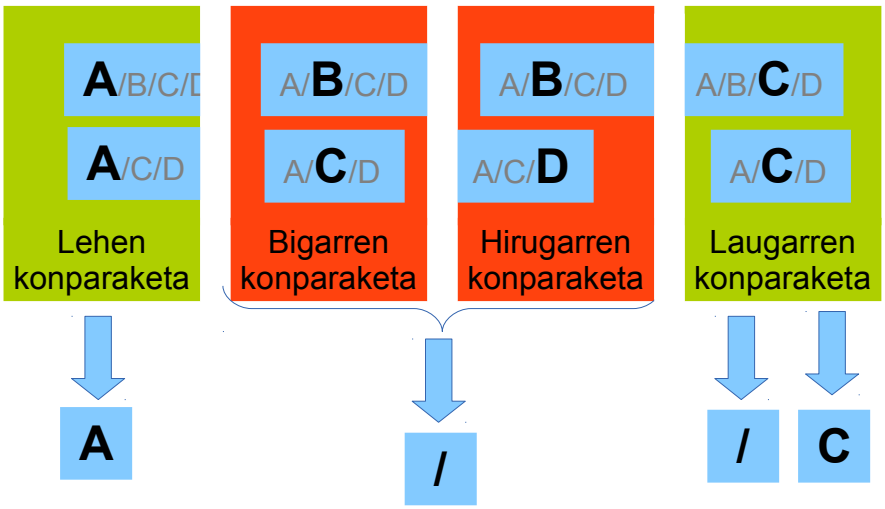
Kasu honetan `B` eta `C` ezberdinak dira, eta orduan bigarren espresioak egiten du aurrera, berdina den XPath zatia bilatu nahian:

5.11 Irudia: A/B/C/D eta A/C/D XPath-ak batzeko metodoa.



Berdina den zatirik aurkitu gabe bigarren XPath espresioaren amaierara iritsi gara. Orduan, lehenak aurrera egiten du, eta bigarrena ezberdina zen lehen zatira bueltatzen da, hau da, `C`-ra:

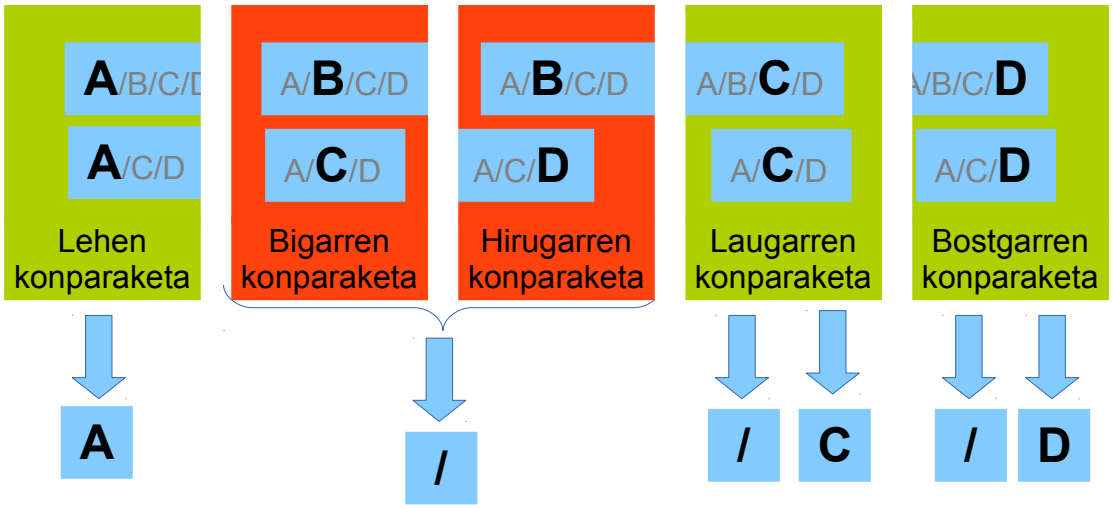
5.12 Irudia: A/B/C/D eta A/C/D XPath-ak batzeko metodoa.



Orain, bi zatiak berdinak dira, hasieran ez gaudenez `/C` gehitu beharko genuke, baina tartean ezberdinak ziren zatiak aurkitu direnez, `//C` gehituko dugu. Orain ere bi espre-

sioek aurrera egingo dute:

5.13 Irudia: A/B/C/D eta A/C/D XPath-ak batzeko metodoa.



Berriro ere berdinak dira. Kasu honetan, tartean ez da ezberdina den zatirik aurkitu, eta beraz, XPath baliokidera `/D` gehituko dugu. Prozesu honen bidez bi XPath espresioen baliokide den XPath-a aurkitu dugu: `A//C/D`.

Algoritmo hau errepikatutako ezaugarrien gainean aplikatu ondoren, gordetzeko prest egongo diren XPath espresioak izango ditugu. Gordetzeko `ezaugarri_izen:XPath` pareak erabiliko dira[3], eta horretarako errepikatutako XPath-ak aurkitzeko erabili dugun espresio erregularra berrerabili dugu. Pare horiek gordetzea baino ez da behar.

Hurrengo pausoa XPath-ak fitxategitik irakurtzeko *LibiXaML* moldatzea zen. Horretarako, *LibiXaML*k ezaugarri bat jasotzean, fitxategia ireki, eta ezaugarriaren izenarekin hasten den lerroa kargatzen du, bertatik XPath-a hartuz.

5.4 Analisi-sorkuntza

Proiektuaren azkeneko helburua, Relax NG eskematik abiatuz analisi-sorkuntza sistema garatzea zen. Honetan asko lagundu du hasieran garatutako liburutegiak, izan ere, honek egiten baititu lanik errepikakorrenak.

[3] Adibidez: `KAT:(...)/f[@name='KAT']/str='$'`

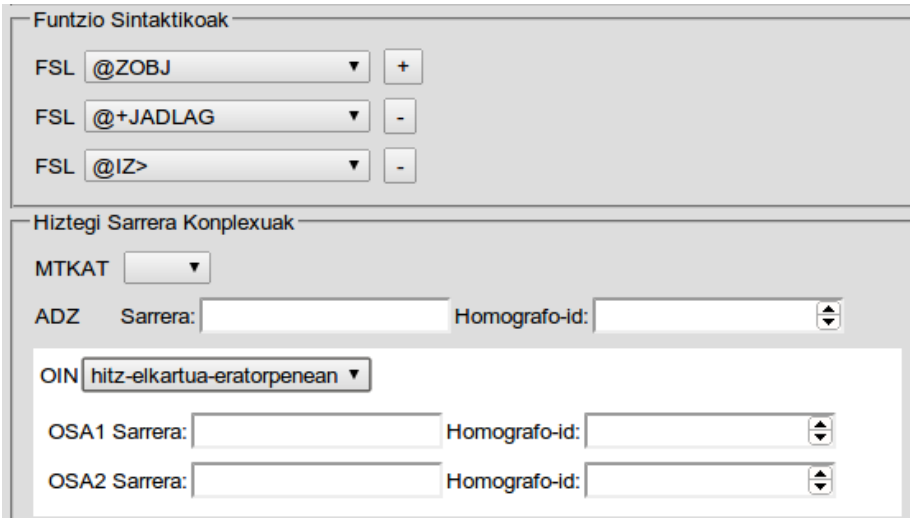
Analisi-sorkuntzan prototipo bat garatuta zegoen, baina honek arazo bat zuen. Izan ere, prototipoak eskema linealki irakurtzen zuen eta, datu bat behar zuenean, erabiltzaileari eskatzen zion; datua gaizki sartuz gero, analisi-sorkuntza bertan behera utzi eta berriro hasi behar zen. Hala ere, prototipo hori oso lagungarria izan da atal honen diseinu eta garapenean.

5.4.1 Interfaze-sorkuntza

Analisiak sortzeko, kontsultan erabilitako bilaketa-interfazea berrerabiltzea erabaki zen; helburua jasotako prototipoak erabiltzen zuen analisiak sortzeko sistema berrerabiltzea zen, baina, kasu honetan, lehenik interfazea erabiliz datuak sartuko ziren, eta, ondoren, datuak interfazetik lortuz baliozko analisia sortu.

Interfazea sortzeko, kontsulta-interfazean erabilitako metodo berbera jarraitu da: ezaugarriak lortu, eta bakoitzaren motaren arabera pantailaratzen dira. Kasu honetan, ezaugarriak pantailaratzeko sisteman aldaketa batzuk egin behar izan dira: batetik, garapenaren fase honetan Relax NG eskemak izandako eguneraketa batean datu-mota konposatuak gehitu ziren, eta interfazean ere gehitu dira. Datu-mota konposatuak beste ezaugarri batzuez osatutakoak dira: oinarri (OIN) ezaugarriari, adibidez, datu-mota konposatu horietako bat gehitu zaio: "eratorpen-oin egituratua", mugatzailea (OSA1) eta mugakizuna (OSA2) ezaugarriez osatua. Bestetik, kontsulta-interfazean kontuan eduki ez ziren balio zerrendak pantailaratu dira analisi sorkuntzan.

5.14 Irudia: Analisi-sorkuntzako interfazea: datu-mota berriak.



5.4.2 Analisi-sorkuntza

Interfazea sortu ondoren, analisia sortzeko garaia da, eta horretarako prototipoan garatutako sistema berrerabili da. Sistema hau, prototipoaren antzera, Relax NG eskema irakurri eta analisia sortzen joaten da, baina ezaugarri baten balioa behar duenean, interfazeko formulariotik lortzen du.

Interfazetik zein balio hartu behar duen jakiteko, identifikadore-sistema bat erabili behar izan dugu. Identifikadore-sistema honelakoa da:

- Identifikadoreak sortzeko ezaugarrien izenak biltzen goaz, hau da, `OIN` ezaugarriaren barruko OSA1 ezaugarriaren identifikadorea `OIN_OSA1` izango da. Honen bidez, `OSA1` ezaugarria eta `OIN` ezaugarriaren `OSA1` ezaugarria ezberdintzen ditugu.
- EDBL gako motako `Sarrera` eta `Homografo-Id` eremuak `f` elementuak direnez, hauek ere ezaugarri moduan hartzen dira. `OIN` ezaugarriaren barruko `OSA1` ezaugarria, EDBL gako motakoa da, eta beraz, bi balio izango ditu: `OIN_OSA1_Sarrera` eta `OIN_OSA1_Homografo-Id`.

Interfazea sortzean, interfazeko sarrera-elementu bakoitzak dagokion identifikadorea izango du, eta ezaugarri baten balioa aldatzean, identifikadorea erabiliz zerrenda batean gordetzen dugu; honela, analisiak behar izango dituen balio guztiak zerrenda bakar batean izango ditugu.

5.27 Kode-zatia: Ezaugarrien balioak gordetzeko zerrenda.

```
1 ezaugarriBalio["OIN"] = "";// OIN string mota
2 ezaugarriBalio["OIN_Sarrera"] = "";//OIN gako mota
3 ezaugarriBalio["OIN_Homografo-Id"] = "";//OIN gako mota
4 ezaugarriBalio["OIN_OSA1_Sarrera"] = "";//OIN_OSA1 gako mota
5 ezaugarriBalio["OIN_OSA1_Homografo-Id"] = "";//OIN_OSA1 gako mota
6 ezaugarriBalio["OIN_OSA2_Sarrera"] = "";//OIN_OSA2 gako mota
7 ezaugarriBalio["OIN_OSA2_Homografo-Id"] = "";//OIN_OSA2 gako mota
```

Analisiak sortzea da hurrengo pausoa. Analisiak sortzeko metodoa oso sinplea da, XPath-en sorkuntzaren antzera, nodo mota bakoitza tratatuz analisia sortzen joaten gara. Ezaugarri batera (`<element name='f'>` nodo bat) iristean, zerrendan zein balio

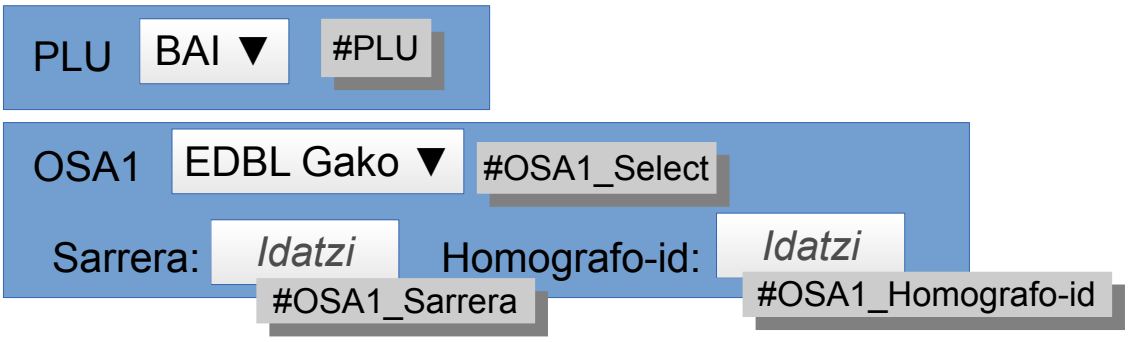
duen konprobatzen da: hutsa badago, erabiltzaileak ez du ezaugarri hori bete, eta beraz, analisian ez da gehitzen; balioa badu, eskemaren irakurketarekin jarraitzen dugu.

Analisiak sortzeko orduan bi arazo aurkitu genituen: lehena, `choice` nodo batera iristean zein bidetik joan, izan ere, ez dakigu erabiltzaileak zein bide jarraitu duen. `OIN` ezaugarriak adibidez, hiru motako balioak izan ditzake, eta soilik eskema erabiliz ezin dezakegu jakin erabiltzaileak zein aukeratu duen. Eta `oneOrMore` edo `zeroOrMore` nodo batera iristean zenbat ume gehitu analisira, hauek, zerrenda-balioen kasuan erabiltzen dira, eta kasu honetan ere, ezin dugu jakin zenbat balio sartu dituen erabiltzaileak.

Lehen arazoari dagokionez, eskeman `choice` nodo ugari daude; hala ere, hauek bitan sailka daitezke: ezaugarri baten parte direnak, datu-mota ezberdina aukeratzeko, edo bai/ez motako nodoetan balioa aukeratzeko, adibidez; eta ezaugarrien parte ez direnak.

Planteamendua sinplea da, eskema irakurri ahala, ezaugarri bat tratatzen ari garen edo ez jakingo dugu; honela, `choice` nodo batera iristean, ezaugarri baten `choice` nodoa bada, interfazetik hartuko dugu balioa. Honetan bi kasu aurkituko ditugu: datu-mota anitzeko ezaugarrien kasuan, ezaugarriaren zein datu-mota erabili den jakin beharko dugu, hau da, ezaugarriak datu-mota aukeratzeko izango duen lista hedagarritik hartuko dugu `choice` nodoan aukeratu beharreko bidea. Bai/ez motako ezaugarrien kasuan, `choice` nodoa erabiltzen da bai edo ez balioa duelako; kasu honetan, `choice` nodoaren bidea interfazeko ezaugarriaren balioa aztertuz aukeratuko dugu: bai bada, `plus` nodotik; ez bada, `minus`-etik.

5.15 Irudia: Interfazetik datuak hartzeko sistema: identifikadoreak.



Ezaugarri baten barruan ez dagoen `choice` nodoa aurkitzean, planteamendua ezberdina izango da. Ezaugarri barruan ez dagoen `choice` nodo bat aurkitzean zein bide jarraitu jakiteko, bide horietatik guri interesatzen zaizkigun ezaugarri gehien duena hartuko dugu. Interesatzen zaizkigun ezaugarriak honakoak dira:

- Ezaugarriak interfazean balioa izan behar du, hau da, ezaugarrien balioa gordetzeko taulan balioa izan behar du.
- Ezaugarriak eskema balio jakina badu, gurearen berdina izan behar du. Adibidez: 5.6 kode-zatian ikus daitekeen bezala, kategoria ezberdinak `choice` nodo baten barruan banatzen dira; kasu horretan, `choice` nodoaren bide bakoitzak kategoriaren definizio bat du, eta definizio horrek balio jakina izango du. Gure kategoria ezaugarriak eta aukeratuko dugun adarraren kategoria ezaugarriak balio bera izan beharko dute.

Honekin `choice` batean bidea aukeratzeko gai izango gara. Hurrengo arazoa `oneOrMore` nodoak ziren. Hauek, elementu bat errepikatzkeo erabiltzen dira, baina ezin dugu eskema bidez jakin zenbat aldiz errepikatu beharko dugun. Nodo hauek balio zerrendetan erabiltzen dira (`FSL`, `ATZL`), eta beraz, horrelako bat aurkitzean, ezaugarri hauetako baten barruan aurkituko gara. Gure lana, `oneOrMore` nodoaren umeak zenbat aldiz prozesatu jakitea izango da, eta horretarako, ezaugarrien balioak dauden zerrendaz baliatuko gara.

Ezaugarrien balio zerrendan, hauen balioak daude; zerrenda motako ezaugarrien kasuan, balioa beste zerrenda-egitura bat da. `oneOrMore` nodo bat aurkitzean, zein ezaugarritan gauden ikusi, eta ezaugarri horren elementu kopurua irakurtzen dugu. Ondoren, elementu kopuru adina aldiz prozesatzen dugu `oneOrMore` nodoa, aldi bakoitzean indize bat bidaliz. Indize hori prozesua ezaugarriaren balio-zerrendako zein elementuri dagokion jakiteko da, prozesuetako bakoitzak zerrendako balio ezberdin bat har dezan.

E eranskinean ikus daiteke sortutako analisi baten adibidea.

6. KAPITULUA

Jarraipena eta kontrola

Atal honetan proiektuaren jarraipen eta kontrola azalduko dugu. Horretarako egindakoa planifikatutakoarekin alderatuko dugu.

6.1 Proiektuaren jarraipena eta desbideraketak

Proiektuan zehar burututako atazen jarraipen eta kontrola egin da. Hona hemen ataza bakoitzak behar izan duen denbora eta izandako desbideraketak.

6.1 Taula: Proiektuan zehar burututako atazen desbideraketak.

Ataza	**Egindakoa**	**Planifikazioa**	**Desbideraketa**
B	30	30	0
JK	10	10	0
D1	10	20	-10
D2	110	80	+30
D3	10	10	0
D4	18	30	-12
Proiektuan zehar	**188**	**180**	**+8**

6.2 Taula: Lehen atalean burututako atazen desbideraketak.

Ataza	Egindakoa	Planifikazioa	Desbideraketa
PL1P1	2	3	-1
PL1P2	6	10	-4
PL1P3_1	10	5	+5
PL1P3	10	5	+5
PL1P4	8	5	+3
PL1P5	1	2	-1
PL1P6	1	2	-1
PL1D1	22	15	+7
PL1G1	35	40	-5
PL1T1	6	8	-2
PL1D2	5	10	-5
PL1G2	18	20	-2
PL1T2	10	5	+5
PL1G3	20	15	+5
PL1T3	8	5	+3
Lehen atala	**162**	**150**	**+12**

6.3 Taula: Bigarren atalean burututako atazen desbideraketak.

Ataza	Egindakoa	Planifikazioa	Desbideraketa
PLNA1	10	-	+10
PL2D1	5	10	-5
PL2G1	18	20	-2
PL2T1	8	5	+3
PLNA2	10	-	+10
Bigarren atala	**51**	**35**	**+16**

6.4 Taula: Hirugarren atalean burututako atazen desbideraketak.

Ataza	Egindakoa	Planifikazioa	Desbideraketa
PL3D1	2	5	-3
PL3G1	5	10	-5
PL3T1	2	3	-1
PL3D2	6	10	-4
PL3G2	15	20	-5
PL3T2	6	3	+3
Hirugarren atala	**36**	**51**	**-15**

6.5 Taula: Proiektuan burututako atazen desbideraketak.

Ataza	Egindakoa	Planifikazioa	Desbideraketa
Proiektuan zehar	**188**	**180**	**+8**
Lehen atala	**162**	**150**	**+12**
Bigarren atala	**51**	**35**	**+16**
Hirugarren atala	**36**	**51**	**-15**
Guztira proiektuan	**437 ordu**	**416 ordu**	**+21**

Bi tauletan, planifikatu gabeko bi ataza ikus daitezke:

- PLNA1: Perl lengoaia ikasi. Proiektuaren bigarren atalerako ezinbestekoa.
- PLNA2: Aplikazioan izandako arazo bat konpontzeko behar izan zen denbora da. Kontsultak funtzionamendu arraroa zuen, bilaketak emaitza arraroak itzultzen zituen. Hau konpontzea dezente kostatu zen.

Guztira 20 ordu inguruko (%4) desbideraketa ikus daiteke proiektuan. Hala ere, egindako kudeaketaren ondorioz, proiektua burutzeko arazorik ez da izan. Gainera, denbora gehiena aurreikusi ez ziren atazak burutzen eman dugu (20 ordu); beraz, denboraren kudeaketa egokia egin dela esan dezakegu.

6.2 Komunikazioak

Proiektuan zehar komunikazioak aurrera eramateko metodoa posta elektronikoa eta bilerak izan dira. Posta elektronikoaren bidez gauzatutako komunikazioa ezin hobea izan da; garrantzitsuak ziren gaietarako erabili da, eta formaltasuna eta egokitasuna mantendu da. Premiazkoak ez ziren gaiak bileretan komentatzen ziren, gutxi gorabehera astean behin egin dugu bilera, eta hauetan, garrantzizko erabakiak hartu eta lanaren jarraipena egin da.

Informazioa banatzeko sistema wiki bat izan da; bertan, proiektuaren garapen-egoera eguneratzen eta egin beharrekoak gehitzen joan da ikaslea.

6.3 Kalitatea

Atal honetan proiektuaren kalitatearen jarraipena azalduko da, bai proiektuarena eta bai produktuarena ere. Gure proiektuaren kalitatea neurtzeko irizpideak honako hauek izan dira:

- **Egindako lanaren kalitatea.**
- **Emaitzen kalitatea.**
- **Kodearen kalitatea.**
- **Dokumentazioaren kalitatea.**

Irizpide hauen betetze maila oso altua izan da. IXA taldeari esker ikasleak lantoki eroso bat izan du, eta bertan giro ezin hobean aritu da lanean urte osoa. Emaitzen kalitatea neurtzea zaila da; hala ere, erabilgarritasunaren aldetik emaitza onak lortu dira. Kodearen kalitateari dagokionez, erabilitako teknikak eta diseinuak errepikatutako kodea murriztu dute; gainera, ondo komentatutako kodea lortu da. Dokumentazioaren aldetik gutxi esan beharra dago: ikasleak LaTeX erabili du dokumentazio guztia garatzeko, eta honek dokumentazioren txukuntasunean lagundu du.

6.4 Arriskuak

Garapenean zehar, planifikazioan aipatutako arriskurik ez da gertatu, ez eta proiektua arriskuan jarri duen bestelakorik.

7. KAPITULUA

Ondorioak eta etorkizuneko lana

7.1 Ondorioak

Proiektuaren helburuak, orokorrean, bete dira. Proiektuaren helburua aplikazioa hobetzea zen, eta horretarako, etorkizunean jasango dituen aldaketen aurrean erantzuteko gaitasuna eman zaio.

Lehenik, interfazea egokitu dugu. Aplikazioak bilaketa-sistema eskasa zuen, kodean bertan idatzita zeuden bilaketa-eremu gutxi batzuk baino ez zituen onartzen, eta ondorioz, bilaketa-sisteman aldaketak egin nahi izanez gero kodea moldatu behar zen. Proiektu honetan dinamikoki sortutako interfazea garatu dugu, bilaketa-sistema aberatsa eta eguneratua mantenduko duen interfazea lortuz.

Ondoren, eskema batetik abiatuz kontsultan erabiltzen diren XPath-en sorkuntza automatikoa garatu dugu. XPath horiek denboran zehar aldaketak izateko arriskua zela-eta, automatikoki lortzeko bidea bilatu dugu.

Azkenik, erabiltzaileei analisi berrien sorkuntza-sistema eskaintzea. Aplikazioaren helburuetako bat analisi linguistiko berriak sortzea zen; hasieran, ordea, funtzionalitate hori ez zegoen inplementatua.

Bilaketa-sistemak izan duen hobekuntza handia izan da: kontsulta hiruzpalau termino zituen bilaketa-sistema izatetik, berrogei baino gehiago onartzen dituen batera pasa da. Gainera, kontsulta-interfazea, kontsulta XPath espresioak eta analisiak automatikoki sortzen direnez, aplikazioak etorkizunean jasan ditzakeen aldaketen aurrean erantzuteko gaitasuna ere handitu da.

Kontsulta XPath-ak automatikoki sortzeak abantaila handia eskaintzen du, eskeman ezaugarri berria gehituz gero, script baten bidez bere XPath espresioa lor baitaiteke. Analisien sorkuntzari dagokionez, prototipoan analisiak sortzeko aukerarik ez zegoen. Proiektu honen ondoren, web-aplikazioa erabilita analisi berriak sor daitezke; gainera, analisiak eskema erabiliz sortzen dira, eta honela sortutako analisiak beti egokiak izango dira.

Oro har, aplikazioaren moldagarritasuna handitu da. Relax NG eskema baten bidez, analisiak bilatu eta sortu daitezke, koderik ukitu gabe.

Ikaslearen aldetik, proiektuan zehar gauza asko ikasi du. Batetik, informazioa bilatzen trebetasuna hartu du, proiektua aurrera eramateko behar zen ezagutza bere kabuz bilatu behar izan baitu. Ezagutza hori, batzuetan taldeko kideengan aurkitzen zuen, eta beste batzuetan bibliografian bilatu behar izan du. Bestetik, bere denbora kudeatzen trebetasun handia lortu du, ia urtebetez aritu baita lanean eta denbora horretan era konstantean egin behar izan du lan. Lortutako trebetasun teknikoak ere ugariak dira, izan ere, tresna eta teknologia berri asko erabili behar izan ditu.

7.2 Etorkizuneko lanak

Proiektuan planifikatutakoa bete dugun arren, aplikazioa erabiltzaileen esku uzteko moduan egoteko lana gelditzen da.

Bilaketa-sistemari dagokionez, kasu konplexu batzuk ez dira kontuan hartu: hitz-elkarketaren osagaia eratorria denean (berri-emaile) eta eratorriaren oina hitz-elkartua denean (analisi-bildumatxo). Kasu konplexu hauek bilaketa-sistema moldatu ondoren gehitu ziren eskeman, eta denbora faltagatik, aurrera jarraitzea erabaki zen. Hala ere, hau inplementatzeko oinarri izango da analisi-sorkuntzan egindako lana.

Bestalde, bilaketa oso orokorrak egitean, aplikazioak erantzuna emateko denbora dezente behar du. Erabiltzaile askok erabiltzeko moduko aplikazioa lortzeko, bilaketak egiteko behar den denbora murriztu beharko da.

Analisi-sorkuntzari dagokionez, analisi batean *homografo-identifikadorea* gehitzeko Euskararen Datu-Base Lexikala kontsultatu behar da. Analisi-sorkuntza erraztu nahian, etorkizunean aplikaziotik Euskararen Datu-Base Lexikala kontsultatzeko aukera inplementatu nahi da.

Eranskinak

A. ERANSKINA

Relax NG eskemak kudeatzeko liburutegiaren erabiltzaile-eskuliburua

Gida honetan Relax NG eskemak irakurtzeko liburutegia nola erabili aztertuko da.

Kontuan hartu beharko dira, batetik, analisi linguistikoen Relax NG eskemak irakurtzeko dagoela egina eta ez edozein Relax NG eskema. Bestetik, Javascript lengoaiaz dagoela idatzita; beraz, hau erabiltzeko Javascript lengoaia erabili beharko da.

A.1 Erabiltzeko prestatu

Liburutegia hainbat fitxategik osatzen dute. Batetik, Javascript lengoaiaz idatzitako script-a dago; bestetik, JSON formatua duen konfigurazio-fitxategia. Liburutegia erabiltzeko hainbat pauso jarraitu beharko dira:

- Script-a erabili nahi den web-orritik erreferentziatu:

```
<script lang="javascript" src="RNGSchemaParser.js"></script>
```

- Konfigurazio-fitxategian bete beharreko eremuak bete:
  - `pathFromAPI` eremuan Relax NG eskemak dauden karpeta zehaztu beharko da.

- Liburutegia erabiltzeko `rngSchema` objektu bat sortu

```
var eskema = new rngSchema();
```

- Konfigurazio-fitxategia kargatu

```
eskema.loadPreferences("../config","options.json");
```

A.2 Liburutegia nola erabili

Liburutegia erabiltzeko bi aukera ditugu: batetik, liburutegia erabiliz eskema kargatzea; eta horretarako `open` funtzioa erabili beharko da; bestetik, eskema guk kargatzea eta eskaintzen diren funtzio lagungarriak erabiltzea.

A.2.1 Eskemaren karga liburutegiari utzi

Eskema bat kargatzeko orduan, hainbat gauza eduki beharko dira kontuan: batetik, karga era asinkronoan egingo dela, eta beraz, erabiltzaileak ez du jakingo karga hori noiz amaitu den. Arazo hori konpontzeko `rngSchema` objektuaren `onready` eremuan fitxategien karga amaitzen denean exekutatu beharreko funtzioa jarriko dugu. Bestalde, liburutegia analisi linguistikoen eskemekin lan egiteko dago prestatua; beraz, liburutegitik kargatzen diren eskemak IXA taldearen anotazioa eskemak izan beharko dute.

Jarraitu beharreko pausoak hauek dira:

- `onReady` eremuan nahi dugun funtzioa gorde.

```
function eskemaIrakurri() {
    (...)
}
eskema.setOnReady(eskemaIrakurri); // Funtzio hau  eskema kargatzean exekutatuko
     da.
```

- Eskemaren fitxategi nagusia pasatu liburutegiari.

```
eskema.open("main.rng");
```

Behin eskema kargatuta, funtzio hauek erabili ahal izango ditugu:

- Eskema-erroa lortzeko:
  - Sarrera: –
  - Irteera: Eskemaren erro-nodoa.
  - Adibidea:

```
var start = eskema.getStart();
print start;
>>  <start>
        <...>
    </start>
```

- Kategoriak lortzeko:
  - Sarrera: –
  - Irteera: Hitz baten kategoria posibleak biltzen dituen `array` egitura.
  - Adibidea:

```
var kategoriak = eskema.lortuKategoriak();
print kategoriak;
>> [["ADI", "Aditza"], ["ADB", "Adberbioa"], ...]
```

- Azpikategoriak lortzeko.
  - Sarrera: Kategoriaren identifikatzailea ("ADI", "ADB", ...).
  - Irteera: Sarrerako kategoriaren azpikategoria posibleak biltzen dituen `array` egitura.
  - Adibidea:

```
var azpikategoriak = eskema.lortuAzpiKategoriak("IZE");
print azpikategoriak;
>> [["ARR", "Arrunta"], ["LIB", "leku-izen berezia"],...]
```

- Goi-mailako ezaugarri guztiak lortzeko. Funtzio honek ezaugarriak prestatzen ditu bilaketa-formularioan pantailaratzeko.
  - Sarrera: Kategoriaren identifikatzailea eta hitzaren identifikadore-zenbakia.

  - Irteera: Ezaugarrien HTML kodea.
  - Adibidea:

```
var html = eskema.lortuEzaugarriak("IZE", 0);
print html;
>> <bilaketa interfazea>
```

- Morfemetatik datozen ezaugarriak lortzeko.
  - Sarrera: Kategoriaren identifikatzailea.
  - Irteera: Morfemetatik datozen ezaugarriak dituen zerrenda. Zerrendako elementu bakoitzak bi azpielementu ditu: ezaugarriaren nodoa (erabiltzaileak nahi duen moduan "parsea"dezan) eta aukerazkoa edo beharrezkoa den.
  - Adibidea:

```
var mfe = eskema.lortuMorfemetatikDatozenak("ADI");
print mfe;
>> [[nodo1, false], [nodo2, true], ...]
```

- Kategoriaren ezaugarriak lortzeko.
  - Sarrera: Kategoriaren identifikatzailea.
  - Irteera: Sarrerako kategoriaren ezaugarriak dituen zerrenda. Zerrendako elementu bakoitzak bi azpielementu ditu: ezaugarriaren nodoa (erabiltzaileak nahi duen moduan "parsea"dezan) eta aukerazkoa edo beharrezkoa den.
  - Adibidea:

```
var kate = eskema.lortuKategoriaEzaugarriak("ADI");
print kate;
>> [[nodo1, false], [nodo2, true], ...]
```

- Hiztegi-sarrera konplexuen ezaugarriak lortzeko.
  - Sarrera: Kategoriaren identifikatzailea.
  - Irteera: Hiztegi sarrera konplexuen ezaugarriak dituen zerrenda. Zerrendako elementu bakoitzak bi azpielementu ditu: ezaugarriaren nodoa (erabiltzaileak nahi duen moduan "parsea"dezan) eta aukerazkoa edo beharrezkoa den.
  - Adibidea:

```
var hske = eskema.lortuHiztegiSarreraKonplexuak("IZE");
print hske;
>> [[nodo1, false], [nodo2, true], ...]
```

- Funtzio sintaktikoen ezaugarriak lortzeko.
  - Sarrera: Kategoriaren identifikatzailea.
  - Irteera: Funtzio sintaktikoen ezaugarriak dituen zerrenda. Zerrendako elementu bakoitzak bi azpielementu ditu: ezaugarriaren nodoa (erabiltzaileak nahi duen moduan "parsea"dezan) eta aukerazkoa edo beharrezkoa den.
  - Adibidea:

```
var fse = eskema.lortuFuntzioSintaktikoak("ADI");
print fse;
>> [[nodo1, false], [nodo2, true], ...]
```

- Flexio-ezaugarriak lortzeko.
  - Sarrera: Kategoriaren identifikatzailea.
  - Irteera: Flexio-ezaugarriak dituen zerrenda. Zerrendako elementu bakoitzak bi azpielementu ditu: ezaugarriaren nodoa (erabiltzaileak nahi duen moduan "parsea"dezan) eta aukerazkoa edo beharrezkoa den.
  - Adibidea:

```
var fe = eskema.lortuFlexioEzaugarriak("ADB");
print fe;
>> [[nodo1, false], [nodo2, true], ...]
```

A.2.2 Eskemaren karga erabiltzaileak egin

Analisi linguistikoak ez diren eskemekin lan egiteko hainbat funtzio eskaintzen ditu liburutegiak. XML formatua duen RNG eskema bat kargatu ondoren, funtzio hauek erabili ahal izango ditugu:

- Nodo baten lehen umea lortzeko:
  - Sarrera: Nodo bat.

  - Irteera: Nodoaren lehen umea.
  - Adibidea:

```
var fchild = eskema.firstChild(node);
```

- Nodo batek umeak dituen edo ez jakiteko:
  - Sarrera: Nodo bat.
  - Irteera: `true` umeak baditu, `false` bestela.
  - Adibidea:

```
var hasChilds = eskema.hasChildNodes(node);
```

- Nodo baten umeak itzultzen ditu:
  - Sarrera: Nodo bat.
  - Irteera: Nodoaren umeak.
  - Adibidea:

```
<start>
    <element name="el1">
    <ref name="ref1"/>
</start>

<define name="ref1">
    <element name="el2"/>
</define>
```

```
node =  eskema.getStart(); //edo guk kargatutako RNG eskema baten nodoa.

var childs = eskema.childNodes(node);
print childs;
>>  [<element name="el1"/>, <element name="el2"/>]
```

- RNG eskemaren gaineko `getElementsByTagName` funtzioa:
  - Sarrera: Nodo-izena eta nodoa.
  - Irteera: Nodoaren azpitik dauden sarrerako nodo-izena duten nodoak.
  - Adibidea:

```
<start>
    <element name="el1">
    <ref name="ref1"/>
</start>

<define name="ref1">
    <element name="el2">
        <element name="el3" />
    </element>
</define>
```

```
node =  eskema.getStart();
var nodes = eskema.getElementsByTagName(node, "element");
print nodes;
>>  [<element name="el1"/>, <element name="el2">...</element>, <element
    name="el3" />]
```

- Erreferentzia motako nodo baten definizioa aurkitzea:
  - Sarrera: Erreferentzia-nodoa.
  - Irteera: Sarrerako erreferentzia-nodoari dagokion `define`-a.
  - Adibidea:

```
<start>
    <element name="el1">
    <ref name="ref1"/>
</start>

<define name="ref1">
    <element name="el2"/>
</define>
```

```
node =  <ref name="ref1"/>;
var def = eskema.getDefine(node);
print def;
>>  [<define name="ref1">...</define>]
```

A.2.3 Bi metodoak erabili

Analisi linguistikoen kasuan liburutegia erabiltzeko erarik errazena bi metodoak erabiltzea da: eskemaren karga liburutegiaren bidez egin eta `getStart()` funtzioaren bidez eskema lortu eta nahi den moduan erabili.

B. ERANSKINA

Relax NG eskemak kudeatzeko liburutegiaren programatzaile-eskuliburua

Dokumentu honetan RNG eskemak erabiltzeko liburutegiaren aspektu teknikoak azalduko ditugu.

B.1 Konfigurazio-fitxaregia

Konfigurazio fitxategia JSON lengoaiaz dago idatzita. Atal honetan konfigurazio-eremu bakoitza azalduko dugu.

B.1 Kode-zatia: RNG eskemen liburutegiaren konfigurazio-fitxategia.

```
{"config": {
    "kategorienErroa": "goimailako-kategoria-ezaugarriak",
    "kategoriAtributua": "KAT",
    "azpikategoriAtributua": "AZP",
    "ezaugarrienDefinea": "-ezaugarriak",
    "hske": "hiztegi-sarrera-konplexuen-ezaugarriak",
    "fse": "funtzio-sintaktikoak",
    "fe": "flexio-ezaugarriak",
    "mde": "morfemetatik-datozen-ezaugarriak-",
    "morfemenEzaugarriak": "morfemetatik-datozen-ezaugarriak",
    "pathFromAPI": "../rng"
   }
}
```

`kategorienErroa` Kategorien `define`-a zein den zehazten du.

`kategoriAtributua` Kategoriaren atributua zein den zehazten du.

`azpikategoriAtributua` Azpikategoriaren atributua zein den zehazten du.

`ezaugarrienDefinea` Kategoria baten ezaugarrien `define`-a zein den zehazten du (adib. `ADI-ezaugarriak`).

`hske` Hiztegi-sarrera konplexuen ezaugarriak dauden `define`-a zehazten du.

`fse` Funtzio sintaktikoak dauden `define`-a zehazten du.

`fe` Flexio-ezaugarriak dauden `define`-a zehazten du.

`mde` Morfemetatik datozen ezaugarriak dauden `define`-a zehazten du.

`pathFromAPI` RNG eskemak liburutegiarekiko non dauden zehazten du.

C. ERANSKINA

XPath-en sorkuntza automatikoaren script-aren eskuliburua

C.1 Konfigurazio-fitxategia

XPath-ak sortzeko script-a Perl lengoaiaz dago idatzita. XPath-ak sortzeko orduan, hauetan nahi ez diren atributu eta ezaugarriak daude; hauek kudeatzeko, konfigurazio-fitxategi bat erabiltzen da. Honetan bi balio mota daude: `ignoreAttributes`, baztertu nahi diren atributuek zerrendatzeko, eta `ignoreFeatures`, baztertu nahi diren ezaugarriak zerrendatzeko.

C.1 Kode-zatia: XPath-ak sortzeko script-aren konfigurazio-fitxategia

```
<parserconfig enable='enable'><!-- edo disable -->
    <ignoreAttributtes>
        <attribute value='id'/>
        <!-- <attribute value='besteBat'/> -->
    </ignoreAttributtes>
    <ignoreFeatures>
        <feature value='osagaiak'/><!-- Ezaugarri honen eta bere umeen XPath-a ez da
    nahi -->
    </ignoreFeatures>
</parserconfig>
```

C.2 Script-a nola exekutatu

Script-a exekutatzeko Perl interpretatzailea eduki behar da instalatuta. Script-a komando lerrotik exekutatzen da, eta bi parametro jasotzen ditu.

C.2 Kode-zatia: Script-a exekutatzeko komandoa.

```
1 $ perl xpath_sortzailea.pl rng_eskema.rng [0|1]inprimatu?
```

Lehen parametroa testu-kate bat da, eskemaren fitxategi-izena zehazteko erabiltzen da. Eta bigarren parametroa boolearra da, script-a zer egiten ari den pantailaratzeko erabiltzen da (0 ez bada ezer pantailaratu nahi, eta 1 bestela).

Komandoa exekutatu ondoren hiru fitxategi sortuko dira: *lem.txt*, *seg.txt* eta *morf.txt*. Bakoitzean analisi mota bakoitzari dagozkion XPath-ak gordetzen dira.

D. ERANSKINA

HybridServer moduluak jasotzen duen parametroa

Atal honetan kontsulta-parametroan eskatzen diren ezaugarriak zerrendatzen dira.

- Parametroa: **db_zuz_hitzKop**_*hitza_hitza_hitza*
- hitza: **lema,modua1,forma,modua2,aldaera,modua3**,*goimailako-ezaugarriak*,**nondik,nora**
- goimailako-ezaugarriak: galde daitezkeen ezaugarri guztiak komaz banandurik.

Jarraian galde daitezkeen ezaugarri guztiak zerrendatzen dira:

`kat` Hitzaren kategoria.

`azp` Hitzaren azpikategoria.

`adbm` Adberbio mota.

`adoin` Aditz-oina.

`asp` Aspektua.

`lagm` Laguntzaile mota.

`izaur` Izenaren aurretik etor liteke? (boolearra).

`err` Erroa.

`mdn` Modua/Denbora.

nor Nor.

nori Nori.

nork Nork.

hit Alokutiboa (hitanoa).

mtkat Metakategoria.

adz Adierazia (erreferentziazkoetan).

nmg Numeroa/Mugatasuna.

num Numeroa.

per Pertsona.

plu Plurala? (boolearra).

biz Biziduna? (boolearra).

neur Neurgarria? (boolearra).

zenb Zenbagarria? (boolearra).

erl Erlazioa.

klm Klausula-muga.

mod Modalitatea.

kas Kasua.

mug Mugatasuna.

fsl Funtzio sintaktikoen lista

oin Oinarria.

aurl Aurrizki lexikala.

atzl Atzizki lista.

osa1 Lehen osagaia (hitz-elkartea).

`osa2` Bigarren osagaia (hitz-elkartea).

`elk` Elkarte mota.

`adm` Aditz mota.

`mai` Gradu-maila.

`kas-plus` Kasu-metatzea.

E. ERANSKINA

Analisi-sorkuntzan sortutako analisi baten adibidea

Hurrengo analisia web-aplikazioaren bidez sortu da.

E.1 Kode-zatia: Sortutako analisi baten adibidea.

```
<fs type="lematizazioa">
    <f name="forma">
        <str>bilatzea</str>
    </f>
    <f name="lema-osatua">
        <str>bilatze</str>
    </f>
    <f name="ezaugarri-morfologikoak">
        <fs type="goimailako-ezaugarri-lista">
            <f name="KAT">
                <sym value="IZE"></sym>
            </f>
            <f name="AZP">
                <sym value="ARR"></sym>
            </f>
            <f name="KAS">
                <sym value="ABS"></sym>
            </f>
            <f name="NUM">
                <sym value="S"></sym>
            </f>
            <f name="MUG">
                <sym value="M"></sym>
            </f>
            <f name="OIN">
                <fs type="Gako">
                    <f name="Sarrera">
                        <str>bilatu</str>
                    </f>
```

```
                    <f name="Homografo-Id">
                        <nbr value="0"></nbr>
                    </f>
                </fs>
            </f>
            <f name="ATZL" org="list">
                <fs type="Gako">
                    <f name="Sarrera">
                        <str>tze</str>
                    </f>
                    <f name="Homografo-Id">
                        <nbr value="2"></nbr>
                    </f>
                </fs>
            </f>
            <f name="ADOIN">
                <str>bila</str>
            </f>
        </fs>
    </f>
</fs>
```

Bibliografia

[1] X. Artola, A. Díaz de Ilarraza, A. Soroa, eta A. Sologaistoa. (2009). Dealing With Complex Linguistic Annotations Within a Language Processing Framework. *IEEE Transactions on Audio, Speech and Language Processing*, 17(5):904-915.

[2] Z. Beloki. (2011). Armiarma: Anotazio linguistiko konplexuak kudeatzeko web-aplikazioa. Karrera Bukaerako Proiektua, UPV/EHU.

[3] I. Aldezabal, O. Ansa, X. Artola, A. Ezeiza, K. Gojenola, J.M. Insausti eta M. Lersundi. (1999). Euskararen Datu-Base Lexikala (EDBL): eskema berriaren proposamena.

[4] A. Koenig eta B. E. Moo. (2000). *"Accelerated C++: Practical Programming by Example"*. O'Reilly Media.

[5] A. Astigarraga, K. Gojenola, K. Sarasola eta A. Soroa. (2009). *"TAPE Testu-analisirako PERL erremintak"*. Udako Euskal Unibertsitatea (UEU).

[6] E. van der Vlist. (2003). *"RELAX NG"*. O'Reilly Media.

[7] D. Sklar. (2004). *"Learning PHP 5"*. O'Reilly Media.