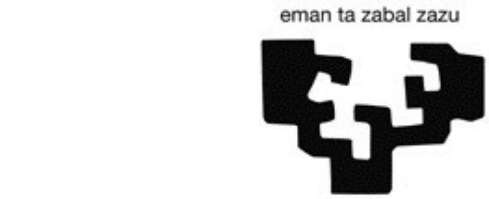


Facultad de Informática Informatika Fakultatea

TITULAZIOA: Sistemetako Informatikan Ingeniaritza Teknikoa

Bilaketan oinarritutako mashup aplikazioa ingelesez idazteko

Ikaslea: Eneko Lizardi Ituarte Jn.

Zuzendaria: Iñaki Alegria Loinaz Jn.

Karrera Bukaerako Proiektua, 2012ko ekaina

Aurkibidea

1. SARRERA

Ohikoa da ingelesezko testu bat idazterakoan hitz edo esaldi bat nola idazten den zalantzak izatea. Berehala *Google* erabiliz bilaketa azkar bat eginez, zalantzak argitzea da irtenbiderik erosoena.

Hemen aurkezten den aplikazio honek, ohitura hau jarraiki, ingelesez idazteko momentuan bilaketek eskaintzen duten laguntza ematen dio erabiltzaileari. Bilatzailean esaldi zatiak behin eta berriz idazten ibili eta emaitzak aztertzen egon beharrean, zuzentzaile honek oker egon daitezkeen zatiak identifikatu eta iradokizunak ematen ditu. Denbora hobeto aprobetxatzen laguntzen du, testuaren ideiak hobeto garatzen laguntzen baitu, hitzak nola idatzi bigarren mailako ekintza bihurtuz.

Edozein mailatako erabiltzaileei zuzenduta dago aplikazio hau. Institutuan dabilen ikasleek, adibidez, ingeleseko ikasgairako idazlanak egiteko erabilgarri dakioke okerrak non dauzkan berehala ikus baitezakete, ikasketa prozesuan ederki lagunduz. Ongi etor dakioke bulegari bati ere bere eguneroko lanean ingelesezko eskutitz bat egin nahi izan ezkero, zuzendu behar diren akats txikiak zuzendu ahal izateko.

Gauzak horrela, aplikazioa erabilgarri zein interesgarria dela esan daiteke. Edozein mailatako erabiltzaileei zuzenduta egonik eta *Web*-ean eskuragarri izateko prestatuta izanik etorkizun argia duela esan daiteke.

Hori dela eta, animatu nintzen proiektu hau burutzera. *API*-en bidez eskaintzen diren web zerbitzu bat edo gehiago hartu eta ideia bat garatu asmoz, nahi bezala moldatzea oso modan dago. Mota honetako aplikazioei *mashup aplikazio* deritze eta azken urteetan Interneteko fenomeno ezagun bilakatu dira. Gainera, gaur egun puri-purian dauden *REST* eta *JSON* moduko teknologien erabilera, *Web* zerbitzuen erabilera aplikazio berri bat sortzeko eta lengoaiaren tratamendua izan dira arrazoirik handienak aplikazio hau garatzera bultzatu ninduenak.

Hauxe da nire proiektua.

2. PROIEKTUAREN HELBURU DOKUMENTUA

2.1 Sarrera

Aplikazio hau ingelesarekin arazoak dituzten pertsonentzat pentsaturik dago. Idazteko orduan hitz edo esaldi bat eratzerakoan dudak izaten dira maiz, batez ere maila ertain eta baxuko erabiltzaileen artean. Arazo honi irtenbidea emateko askotan Interneteko bilatzaile gogokoenera joan eta zalantzan daukagun hori testu kutxan idatzi eta bilaketa batzuen ondoren, emaitzak aztertuta, nola idazten den behar bezala jabetzen gara. Idazlan bat egiteko orduan behin baino gehiagotan gerta daiteke zalantzak izatea eta beti bilatzailean sartu eta emaitzak aztertzea, denbora galtzeaz aparte, ez da izaten denon gustukoa. Hizkuntza hau ikasten dabiltzanentzat bereziki ongi datorkie ekintza hau automatizatzea, bilatzailean denbora alferrik galdu gabe lanean (eta ondorioz, ingelesaren ikasketan) erroreak non egiten dituzten konturatu eta beraien kabuz zuzendu ahal izango baitute eta.

2.2 Aurrekariak

Xuxen zuzentzaile ortografikoa da adibiderik garbiena. Oso ezaguna eta erabilia da euskaraz idazten duten erabiltzaileen artean. *Microsoft Office* eta *OpenOffice* ofimatika suitetarako plugin honek, hiztegi baten laguntzaz, idatzitako hitzak aztertu eta gaizki egon ezkero idatzi nahi izan den hitzaren antzekoak erakusten ditu. Bere Interneteko orrialdean web aplikazio moduan antzeko tresna edozeinen eskura dago, *xuxenweb* deritzona (xuxen.com).

Mashup aplikazioak oso erabiliak dira web osoan zehar zerbitzu bat edo batzuk konbinatuz beste aplikazio osatuago bat sortu ahal izateko. Hainbat modutan sortu daitezke, bai *mashup* aplikazioak sortzeko aplikazio bereziekin (grafikoki, kode lerro bat bera ere idatzi gabe *Yahoo Pipes* bezala), bai mahaigaineko beste edozein aplikazio bezala garapen tresna tradizionalagoekin ere.

Opentrad sistema ere aipatu beharreko adibidea da. Itzulpen automatikoa eskaintzen duen kode irekiko sistema bat da. Euskara, galiziera, katalana (valentziera barne) eta gaztelania hizkuntzen arteko itzulpenak egiten ditu. Bere lana burutzeko bi teknologietan oinarritzen da: *Apertium*, antzeko hizkuntzen arteko itzulpenetarako, eta *Matxin*, egitura ezberdinekoetarako. *Apertium* antzeko hizkuntzetarako diseinatu zen arren desberdinak direnen artean lan egiteko hedatua izan da. Horren adibide *erdaratu.eu* web orria da[1], euskaratik gaztelaniara itzulpenak egiteko sortu zena. *Matxin*ek hasieran gaztelaniatik euskarara itzultzen bazuen ere hurrengo bertsioan euskaratik gaztelaniara lan egiteko gai egin zuten. *Opentrad*-ek gaztelania eta galegoa moduko hizkuntzekin arazo gutxi baditu ere oraindik euskararekin lan egiteko buruhauste dezente ditu.

Google-k oso ezaguna eta erabilia den *Translator* aplikazioan euskara bere hizkuntza "itzulgaien" zerrendan sartu zuen orain dela gutxi. Alfa bertsioan dago oraindik eta itzulpenean maila kaxkarra eskaintzen du momentuz. Mota honetako aplikazio bat egiteko orduan estandar moduan hartzen da, hau da, *Googlek* ematen duena baino kalitate hobeagoko itzulpenak eman beharra dago esperantza pixkat izan nahi bada proiektua aurrera atera ahal izateko.

1 http://erdaratu.eu/index.eu.html

2.3 Helburuak

Mashup aplikazio bat garatzea da helburu nagusiena, egunean eguneko hizkuntza arazoei aurre egin ahal izateko edozein mailako erabiltzailek. Hasiera batean ingelesa izango bada ere proiektuaren ardatz, euskara, gaztelania edo frantsesa moduko beste hizkuntza batzuetarako garatzea ez litzateke gaizki egongo.

Lan honetan *mashup* aplikazioen garapena landuko da. Interneteko bilatzaileak emandako emaitzen bilaketa eta iradokizunak kontuan hartuz erabiltzaileari zuzen edo oker dagoen adierazi beharko zaio. Hau egiteko *Ajax, Javascript* eta *PHP* ikasi eta praktikan jarri beharko da.

2.4 Norainokoa

4.1 Azpiatazen zerrenda

- Prozesu taktikoak:

K – Kudeaketa
K1 – Bilerak
K11 – Aurrerapen bilerak
K2 – Artxiboaren kudeaketa
P – Planifikazioa
P1 – Helburu dokumentua egin

- Prozesu operatiboak:

G – Garapena
G1 – Eskakizunen bilketa
G11 – Erabilpen kasuen eredua egin
G12 – Domeinuaren eredua egin
G13 – Interfazearen prototipoa egin
G2 – Analisia
G21 – Sistemako sekuentzia diagrama egin
G22 – Kontratuak egin
G3 – Diseinua
G31 – Negozioaren logika diseinatu
G311 – Sekuentzia diagramak egin
G312 – Klase diagramak egin
G32 – Aurkezpen maila diseinatu
G33 – Probak diseinatu
G4 – Inplementazioa
G41 – Arkitektura definitu
G42 – Sistema inplementatu
G5 – Probak egin
D – Dokumentazioa
D1 – Teknologien azterketa

D1.1 – Ezagutzea
D1.2 – Ikastea
D1 – Eskuliburua egin
D2 – Memoria egin

4.2 Emangarrien zerrenda

1.- Helburuen dokumentua: Proiektua nola joango den laburtzen duen dokumentua. Bertan segituko den planifikazioa, zein emangarri sortuko diren eta artxiboa nola kudeatuko den azaltzen da, besteak beste.

2.- Eskakizunen bilketari dagokion dokumentua: Aplikazioaren erabilpen kasuak eta domeinuaren eredua.

3.- Sekuentzia diagramak.

4.- Klase diagramak.

3.- Iturburu kodea: Aplikazioaren kode osoa.

4.- Egindako proba kasuen dokumentua: Aplikazioan dauden erroreak aurkitzeko diseinatu diren proba guztien nondik norakoak laburtzen dituen dokumentua.

5.- Memoria: Proiektu osoan zehar egindako lanaz osaturiko dokumentua.

6.- Lan gaztiguak: Ataza bakoitzean pasatutako denbora islatzen duen taula.

2.5 Planifikazioa

5.1 Denboraren planifikazioa

Gutxi gorabehera, guztira 150 ordu sartzea espero da proiektua burutzeko. Lehengo azpiatazetan oinarriturik lana honela banatuko da:

-Helburuen dokumentua egitea. 4 ordu.
-Eskakizunen bilketa. 7 ordu.
-Teknologiak erabiltzen ikastea. 25 ordu.
-Analisia. 12 ordu.
-Diseinua. 16 ordu.
-Inplementazioa. 50 ordu.
-Proba kasuak diseinatu eta martxan jartzea. 6 ordu.
-Memoria osatzea. 30 ordu.

2.6 Arriskuak

Proiektua aurrera joan ahala egunean eguneko martxa galarazten duten hainbat arazo gerta daitezke. Honek proiektua behar den denboran aurrera atera ezin ahal izatea edo lan guztia galtzea ekarri dezakete eta horregatik ezinbestekoa da arrisku garrantzitsuenak zein diren identifikatu eta honen aurrean nola jokatu jakitea.

Datuak galtzea

- **Azalpena:** Ordenagailuari zerbait gertatu ezkero lan guztia gal daiteke. Oso kontuan hartu beharreko arazo hau hainbat arrazoirengatik izan daiteke: Eguneraketa bat gaizki instalatzea, ordenagailua galtzea, disko gogorra hondatzea...
- **Irtenbidea:** Arazoaren izaera kontuan hartu beharko litzatekeen arren oinarrizkoena babes kopiak egitea litzateke. *Dropbox* moduko aplikazioei esker kopiak Interneten gorde daitezke eta ordenagailuari edozer gertatu ezkero ere edonondik fitxategiak lor daitezke eta lanean segi.

Arazo pertsonalak izatea

- **Azalpena:** Gaixotzea, istripuren bat gertatzea edo beste antzeko arazoak ezin kontrola daitezke. Proiektua aurrera atera behar duenaren kasua da larriena, lana ezin jarraitu izatea dakarrelako. Zuzendariari ere gerta dakioke beste hainbeste, ikasleari berari laguntza gabe utziaz.
- **Irtenbidea:** Proiektua abian den bitartean emangarriak behar den momentua baino zertxobait lehenago eginak izatea. Horrela edozer gertatu ezkero ere aldez aurretik egin beharrekoa behar den momenturako burutua egongo da.

*API*a itxitzea eta zerbitzurik gabe geratzea

- **Azalpena:** Erabili behar den bilatzailearen *API*a edonoiz aldatu edo itxi daiteke. Proiektua martxan dagoen bitartean gertatuz gero ezin izango da bukatu proiektua.
- **Irtenbidea:** Beste bilatzaileen *API*-ak kontsultatu eta erabilgarri dauden ikusi, batek bere zerbitzuak kendu ezkero bestera pasatu eta lanean segitzeko.

2.7 Lan metodologia

7.1 Kudeaketa eta antolaketaren azalpena

Proiektua aurrera ateratzeko metodologia ondokoa izango da:

- 15 egunetik behin zuzendariarekin batu eta aurrerapen bilera bat egingo da. 2 aste horietan zehar egindakoaren analisia egingo da eta ondoren egin behar direnen planteamendu eta azterketa.
- Aplikazioaren barne lanak egiteko Bing bilatzaileak eskaintzen duen *API*a erabiliko da. *Google*-k duena aztertu zen bere momentuan eta zerbitzu bat itxi eta beste bat behar adinakoa baino sinpleagoa zenez alde batera utzi zen.

- Artxiboaren kudeaketa burutzeko babes kopia programatuak egingo dira 2 astean behin *Dropbox*-en laguntzaz.
- Garapen tresna bezala *Oxygen* erabiliko da eta *XAMPP PHP* eta web zerbitzuak emateko.

3. ANTZEKO APLIKAZIOAK

3.1 Sarrera

Proiektu honek hainbat aplikazioren ideia abiapuntutzat hartzen du. Hauek plazaratutako ideia hobetu nahian garatu da aplikazioa. Hori dela eta ezinbestekotzat jo da antzeko aplikazioak zeintzuk diren aurkitu eta aztertzea, nola dauden osaturik ikusi ahal izateko. Ondorengo hauek Interneten martxan zeuden edo dauden aplikazioak dira. Proiektuaren helburu eta isla dira, zein norabidetan mugitu esaten dute. Lortu nahi den ideiaren oso antzekoak direnez adibide egokiak dira aztertu eta dokumentatu ahal izateko.

3.2 Microsoft Research. ESL Assistant.

Microsoft-en lengoaia naturalen prozesamendurako taldeak sortutako aplikazio hau [1] 2008an publiko egin zen. Ingelesa bere ama hizkuntza ez eta eskolan ikasi behar izan duten pertsonentzat zuzenduta dago, sortu behar duten idazlanetan laguntza izan dezaten, hitz eta esaldiak behar bezala nola idatzi behar diren iradokizunak emanaz. Dena den, 2011ko apirilean arazo teknikoak medio eta, zerbitzua bertan behera uztea erabaki zen.

Web aplikazio honek azterketa estatistikoen bidez iradokizunak ematea zuen helburu, idazleak idatzi nahi zuena aukeratu ahal izateko. Funtzionamendu hau *Bing* bilatzailean oinarritzen zen. Honek emandako emaitzak aztertu eta emandako emaitza kopuruaren arabera ordenatuta ematen zituen iradokizun desberdinak.

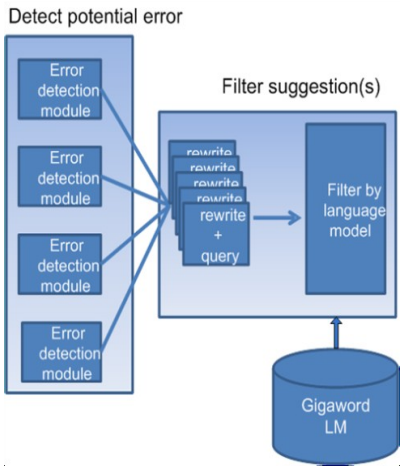
1. irudia: ESL Assistant martxan

Silverlight teknologia erabiltzen duen web interfaze baten bidez lan egin daiteke aplikazio honekin. Erabiltzaileak testu kutxan aplikazioak zuzentzea nahi duen testua sartzen du. Bukatzen duenean *Check* botoiari eman eta aplikazioa bere lana egiten hasten da. Erroreak nabarmendu eta esaldia egoki nola izango litzatekeen esaten du. Era berean, *Bing*-ek emandako emaitzen sintesi moduan tarta grafiko bat ematen du albo batean, bilaketak emandako emaitzen zenbatekoak ehunekotan azalduz.

Sistema honen arkitektura hiru zatitan bana daiteke:
Lehenengoak erroreak egon daitezkeen atalak identifikatzen ditu. Hainbat modulutan zatituta dago eta bakoitzak errore mota jakin baterako balio du. Heuristika zein ikasketa automatikoaren bidez lan egiten dute.

Bigarren zatiak, hizkuntza eredu zabal bat oinarri hartuta, iruzkinak aztertzen ditu.

Azkenik, bilaketa emaitzak sortzen dituen modulua dago, *Bing* bidez lan egiten duena.



2. irudia: ESL Assistant-en arkitektura

Outlook 2007 posta elektroniko bezeroarentzat *add-on* bertsio bat egin zen, idatzitakoa *Web* bidez *ESL Assistant*-en orrira bidali eta honek zuzenketak egin zezan.

On-line bidezko thesaurus zerbitzua ematen zuen, testuingurua kontuan hartuz aproposen datorren beste hitz bat eskaintzeko.

Denbora joan ahala hobekuntza txikiak egiten joan ziren eta bertsio berri bat ere argitaratu zuten, errore detekzio hobeagoarekin. Hor, oharreztatutako errorez osatuta zegoen corpus handi batean oinarrituta zekoen eredu estatistiko berri bat jarri zuten, lehendik zegoena alde batera utziz, erroreen berreskuratze tasa altuagoa baitzuen zehaztasun maila bera eskainiz. *Silverlightek* onura asko ematen zion arren aplikazioari oinarrizko funtzionalitate batzuk falta zitzaion. Dena den, teknologia hori ere pixkanaka-pixkanaka hobetzen joan ahala gabeziak gutxitzen joan ziren. Kopiatu eta itsasteko funtzionalitatea, adibidez, hasieran ez zegoen baina *Silverlighten* laugarren bertsioak ESL *Assistant-i* praktikotasun handia eman zion.

Denbora aurrera joan ahala gero eta erabiltzaile gehiagok eskatzen zuen *API* publiko bat. Hori ikusita proiektukideek *API*az gain zerbitzu guztia hodeira igotzea pentsatu zuten. Hala ere, handik hilabete batzuetara arazoak izan zituzten zerbitzarietara sartu ahal izateko segurtasun eguneraketa baten erruz eta egun batzuen buruan *ESL Assistant-en* zerbitzua bukatutzat jo zuten.

3.3 Nadaclair Language Technologies. SpellCheckPlus.

3. irudia: SpellCheckPlus -en logotipoa

Ingelesa ama hizkuntza ez duten pertsonei zuzendurik dago bereziki, normalean egiten duten okerrak ongi identifikatzen ditu eta.

ESL Assistant bezala [1], *SpellCheckPlus*[2] web bidez erabili daitekeen

zerbitzua da[2]. Orain dela gutxi mugikorretarako aplikazioa kaleratua izan da *iPhone* eta *Android* plataformetan hain zuzen ere, eta etorkizunean mahaigaineko aplikazio bat egiteko asmoa dago.

Nadaclair Language Technologies enpresak sortu zuen 2007. urtean. Terry Nadasdi eta Stéfan Sinclair unibertsitate irakasleek sortu zuten urteetan zehar lantzen egon ziren alorrak bultzaturik. Bata, linguista izanik, elebitasunean eta analisi gramatikoan espezializatua da eta bestea literatur analisian eta humanitateei zuzendutako informatikan.

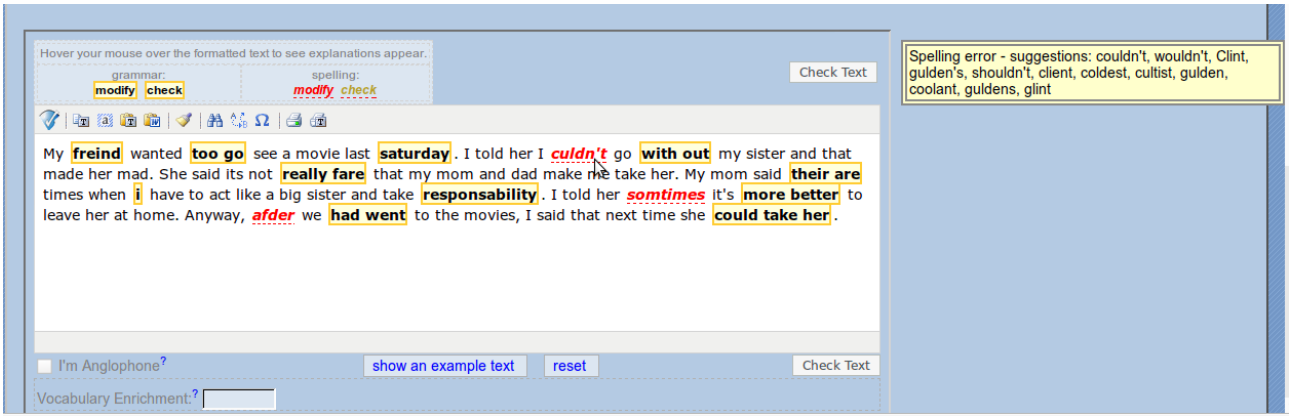
SpellCheckPlus programak ingeleserako balio du baina beste bi antzeko aplikazio ere garatu dituzte: *BonPatron* frantseserako eta *SpanishChecker* gaztelaniarako. Denak ere hizkuntza ikasteko tresna eskuragarri izateko sortuak dira. Hizkuntza ikasterako orduan ikasleak berak aurrerapenak lortu ahal izateko balio du aplikazio honek, akatsa nolakoa izan den esaten baitu eta adibideak emanez ongi nola izan beharko litzatekeen. Dena den, maila baxua ez ezik maila altuagoko erabiltzaileek ere erabil dezakete.

Funtzionamendu sinplea du: Web orri nagusian dagoen testu kutxan idatzi eta *Check text* botoiari jotzea besterik ez da. Orduan, Javascript bidez azterketa egiten du gramatika eta ortografia akatsak non dauden esanaz. 3 multzotan sailkatzen dira:

- Letren ordena hitz barruan (“*freind*” “*friend*” beharrean). Kasu honetan gaizki dagoen hitzaren gainetik arratoia igaroz gero hitza ongi nola idatzi beharko litzatekeen iradokizun zerrenda bat erakusten du. Hitza argi ikusten badu zein den zuzenean ongi nola idatzi beharko litzatekeen esango digu, hau da, “*Freind*”-en kasuan *friend* idazteko eskatzen du.
- Egiaztatu beharreko egiturak (“*he is home less*” adibidez)
- Zuzenketa behar duten egiturak (“*they're house*”).

Akatsa egon ezkero hitza bera nabarmendu eta zuzendu beharra den edo egiaztatu behar den esaten du. Arratoia gainetik pasatuz gero iradokizunak ematen ditu. Ingelesa bigarren hizkuntza duten erabiltzaileei zuzenduta dagoenez ez da gai edozein errore gramatiko identifikatzeko, egiten duten ohiko akats aurkitzeko baizik. Berdin ahoskatzen diren hitzak desberdindu (*there/their* eta *its/it's* adibidez), ongi dauden hitzak gaizki erabiltzea (*“never mined” “never mind”* idatzi behar denean), puntuazio/ hitzen banatzea (“*with out*”vs “*without*”), kapitalizatzea (“*saturday*” “*Saturday*” idatzi beharra) eta beste hainbat gramatika errore identifikatzeko gai da. Testuak ongi idazteko oso baliagarria den arren kontuan hartzekoa da ikasteko tresna bat dela eta horrekin ez dela nahi lortu ingelesa behar bezala ez dakien edozeinek akatsik gabeko testu aberats bat sortzerik.

2 http://spellcheckplus.com/



4. irudia: SpellCheckPlus martxan

Lexikoa aberasteko tresna sinple bat ere eskaintzen du, askotan erabiltzen diren hitzak bere sinonimoz ordezkatu ahal izateko.

Akatsen identifikazioa testuen bidalketetan oinarritzen da. Zenbat eta testu gehiago bidali eta errore bat behin baino gehiagotan agertu aplikazioak "ikasketa" prozesu bati esker hurrengo batean errore hori bera identifikatzeko gai izango da. *SpellCheckPlus*-ek ez du azterketa semantikorik egiten eta ondorioz, zentzurik ez duen esaldiren bat jarriz gero ez da ohartuko eta azterketa ortografiko eta gramatikala soilik burutuko du.

Doako bertsioa eta ordaintzekoa daude (*Pro* bertsioa, 10€ urte osorako lizentzia). Doakoan testu kutxa soila (500 hitz gehienez bidali daitezke zerbitzarira) eta iradokizunetan azalpen motzak azaltzen dira iragarkiekin batera. *Pro* bertsioak ordea testuan egindako errore guztien laburpen bat eskaintzen du, erabiltzaileak zein motatako erroreak egiten dituen ikusi eta ikasketako orduan zein arlo indartu behar duen jakin dezan. Horrekin batera testu kutxa handiagoa (500 hitz baino izan ditzake testuak), iragarkirik eza, lexikoa aberasteko tresna, ariketa pedagogikoak, testua zerbitzarian gorde eta hurrengo batean zuzentzen segi ahal izatea... ematen ditu.

3.4 Xuxen

1994an plazaratutako aplikazioa dugu[3][3]. EHUko informatika fakultatea, UZEI eta Baionako Hizkiaren arteko elkarlanaren fruitu, euskara batuan idatzitako testuen zuzenketa ortografiko eta tipografikoa egiteko balio du.

Hasieran ordenagailuan bertan instalatu behar zen aplikazioa bazen ere orain *XuxenWeb* izeneko web aplikazio moduan erabili daiteke edozein motatako konputagailu bat erabiliz ezer instalatu gabe eta dohainik erabil daitekeelarik.

Aplikazioa praktikoa izateaz aparte oso erabilerraza da. Bi zuzenketa modu ditu:

- Zuzendu. Testu osoa aztertu eta ezagutzen ez dituen hitzak azpimarratu eta bakoitzerako

3 http://www.xuxen.com/index.php

iradokizun desberdinak ematen ditu.

- Zuzenketa aurreratua. Azterketa sekuentziala egiten du bulego aplikazioetan egiten denaren antzera. Ezagutzen ez den lehenengo hitzean geratu eta iradokizunak ematen ditu. Hau zuzendu ondoren hurrengora igaroko da testu osoa zuzendu arte.

Hiztegi pertsonalizatu bat kudeatzeko aukera ere badago. Mahaigaineko aplikazioan ohiko den moduan kudeatzen bada ere web bertsioan erabiltzailea erregistratu beharra dago. Hiztegi propioa erabiltzeak abantaila ederra ematen du, Xuxenek berak ezagutzen ez dituen hitzak zuzen erabiltzean oker bezala ez nabarmentzeko balio du eta.

Oker bat antzeman ahal izateko dituenean analisi morfologikoa egiten da. Hau euskararen konplexutasun morfologikoaren dela eta aplikatu behar izan da. Beste hizkuntzetan hitz-zerrenda batean kontsultatzea nahikoa izaten da okerra identifikatzeko. Analisi morfologikoa egiteko Helsinkiko Unibertsitateko Koskenniemi irakasleak 1.983. urtean proposatutako formalismoa hartzen du kontuan euskara moduko hizkuntzetarako oso erabilgarria da eta. Eredu honen puntu garrantzitsuenak hauek dira : testuko hitzak (azaleko maila) eta lexikoa (maila lexikoa) argi bereiztea, analisi eta sintesirako baliagarri izatea eta programa eta ezaguera linguistikoa bereiztea.
Iradokizunak emateko gai ere bada Xuxen. Errore tipografiko baten erruz okerra gertatu ezkero antzeko hitzak aurkitzen saiatzen da, hizki bat galdu edo elkarren artean trukatuta egonik kontrako bidea hartuz. Euskararen erabilera okerraren ondorioz gertatu ezkero errorea azpilexiko eta erregela berezien bidez saiatzen da detektatu eta zuzentzen.

Hori guztia martxan jartzeko Euskaltzaindiaren proposamenak hartu dira kontuan, baina baita Ibon Sarasolaren Hauta-Lanerako Euskal Hiztegia, UZEIren Euskalterm eta EEBS datu-base lexikografikoa, Hiztegia 2000... laguntza moduan ere.

4. ERABILITAKO TEKNOLOGIA

4.1 Sarrera

Aztertu diren antzeko aplikazioek helburua zein den esaten dute eta erabilitako teknologiak oinarriak zehaztu. Nola osatuta dagoen eta zein abantaila eskaintzen duten beste aplikazioekin alderatuz. Teknologiaren hautaketa oso garrantzitsua da, aplikazioak nola funtzionatuko duen esaten baitu: Nola komunikatuko den *API*arekin, zer nolako datu motarekin lan egingo duen, nola tratatuko duen...

Ondoren proiektuaren garapenean erabili diren eta erabiltzeko asmoa zegoen (*Google API*) teknologiak aurkitzen dira. Bilatzaileetatik hasi, komunikazio arkitektura eta mezuen formatutik igaro eta programazio liburutegiekin bukatuz.

4.2 Bing

Microsoft etxeak garatutako web bilatzailea da[4]. Irudiak, web orriak, lekuak, berriak, bideoak... bilatzeko gauza da.

Zerbait bilatu nahi denean testu kutxan idaztea nahikoa da. Erabiltzailearen produktibitatea hobetze aldera idatzi ahala iradokizunak ematen ditu, horrela hitza edo esaldia osorik idatzi ordez zati bat idatzi eta osatutakoan klik eginez berehala bilaketa egin ahal izateko.

Egindako bilaketa bakoitzarekin, emaitzaz aparte, zerikusia duten bilaketen zerrenda bat erakusten du. Gainera, egindako azken bilaketen zerrenda bat ere albo batean erakusten du.
Adibide bat jartzearren, "*New York*" jarri ezkero testu kutxan, ohiko bilaketa emaitzaz gainera, zerikusia duten bilaketen zerrendan *New York Giants News, New York New York Weather, New York Attractions*... erakusten du. Erabiltzaileen hitz horiekin egindako bilaketen zerrenda dela suposatzen da, kontsulta gehien egin direnen lehen zortziak kontuan hartuz.

Bilaketak egiterako orduan leku geografikoa ere kontuan hartzen du. Kontsulta nondik egin den kontuan hartuta emaitzak moldatu egiten ditu. Estatu Batuetatik egindako kontsulta bat Japoniatik egindako beste baten desberdina izango da bertako erabiltzaileei "hurbilagoko" web orriak erakutsiko zaie lehenengo eta gero besteak. Honekin bilatu nahi dena errazago aurkitu eta kalitate handiagoko emaitzak eskaini nahi dira. Praktika hau gaur egungo bilatzaile eta hainbat web orrietan oso zabaldua da, nahiz eta oso polemikoa izan. Erabiltzailearen datu pertsonalak batzen dira, hala nola nongoa den eta bere gustuak, bisitatu dituen web orriak kontuan hartuz. *Bing*-en kasuan, ordea, hori ez ezik web orrien *geo* meta-etiketak ere kontuan hartzen ditu erabiltzailearekiko hurbilagoko web orriak emaitzen artean posizio altuagoan erakutsiz. Dena dela teknika honek badu bere alde txarrak ere. Web diseinatzaileek edozeinen txantiloietan oinarritu daitezke beraien web orri propioak sortzerako orduan eta orduan, txantiloia kopiatzean, *geo* meta-etiketak ere kopiatu eta ez dituzte beraien informazioarekin betetzen. Gauzak horrela, bilatzailearen emankizunen kalitatea okertu egiten da eta gainera errendimendu galera eragiten dute, behar ez den lekuan agertu eta bilaketa eta emaitza-prozesatze denbora handituz.

Erantzun azkarrak emateko gai da, ingelesezko bertsioan bakarrik bada ere. Unitate aldaketak (10

4 http://www.bing.com/

dollar zenbat euro diren, "*10 dollars in euros*"), hitzen definizioak (*"define", "definition"* edo *"what is..."* hitz- gakoak definitu nahi den hitzaren aurretik erabiliz), hegazkin bidaia eskaintzen bilaketa (Boston-era bidaiak *"flights to Boston"*), eragiketa matematikoak (*"sqtr 9"*, *"2y²+5y+10=40"*), eguraldia (*"weather"* edo *"forecast"* hitz-gakoak erabiliz hiri edo herriaren izenaren aurretik)...

Azken puntu aipagarri bezala *Facebook* sare sozialarekin konektatu ahal dela esan daiteke. Bilaketak *Facebook*eko lagunen ekintzekin integratzen ditu. Hori dela eta, *Bing* erabiltzen den bakoitzean informazio baliagarria erakusten du. Hona hemen adibide batzuk. Oporretarako bidaia bat antolatzean eta *Bing* erabiltzean, leku horretara zein lagun joan den eta bisitatutako bazterrak zein izan diren erakutsiko ditu ohiko beste emaitzekin batera. Era berean erosketak egiterako orduan produktuen saskia lagunei erakutsi eta iruzkinak egiteko aukera ematen du.

Beste hainbat aukera ematen dituen arren zehaztasunez aurkitu ahal izateko, punturik garrantzitsuenak aztertu diren honako hauek dira. Gauzak horrela *Bing* bilatzaile sakon zein boteretsua dela esan daiteke. Egunerokotasunerako oso praktikoa izateaz aparte bilaketa zehatzak egiteko ere gai da hainbat aukera eskainiz. Gainera *Facebooken* konexioa aprobetxatuz lortzen diren emaitzak ikusgarriak bezain interesgarriak dira. Gaur egungo beste bilatzaileek oraindik ez dute ezaugarri hau eta hemendik urte batzuetara nolako joera hartzen duen ikusi beharra dagoen arren, apustu ausarta egin dute.

4.2.1 Bing API

Bilaketa *API*ak[5] *SourceType* motako iturriak eskaintzen dio erabiltzaileari. Hainbat motatakoa izan daiteke. Parentesi artean idatzita daudenak *API*ak erabiltzen dituen hitz-gakoak dira.

- Web orriak (***Web***), irudiak (***Image***), berriak (***News***), galdera-erantzunak (***InstantAnswer***) eta bideoak (***Video***) aurkitzen ditu Internet osoan zehar.

- Zerikusia duten bilaketak egiteko (***RelatedSearch***) eta ortografia zuzenketak (***Spell***) iradokitzeko gai da.

- Negozio jakin bat emanda hartuta informazioa itzultzen du (***PhoneBook***), web orriaren helbidea, telefono zenbakia, helbidea eta erabiltzaileek emandako puntuazioa zehazki.

- Hitz eta esaldien itzulpenak eskaintzen ditu (***Translation***).

*API*aren bigarren bertsioa da orain eskuragarri dagoena. Aurreko bertsiotik hainbat hobekuntza egin dituzte, zerbitzua osatuz eta garatzaileei laguntza gehiago emanaz.
SourceType guztietatik *RelatedSearch, InstantAnswer, Video* eta *Translation* gehitu berri dira zebitzua osatu asmoz.

Aurreko bertsioan ez bezala, *API*arekin komunikatzeko hiru protokolo desberdinen artean aukeratu behar da: *JSON, XML* eta *SOAP*. Honek malgutasun handia ematen dio bilaketa zerbitzuari

5 http://msdn.microsoft.com/en-us/library/dd900818.aspx

garatzaileak oso kontuan hartzen baititu eta ondorioz aplikazio ahaltsuak errazago egiteko ahalmena eskaintzen baitie.

Bilaketa egin eta emaitzak itzultzen dituenean objektu gisa eskaintzen ditu. Aplikazioan zehar sortzen diren beste edozein objektu bezala erabil daitezke, hau da, beharra dagoenean sortuz eta emaitza lortzean bere atributuak hartu eta aztertuz. Bertsio zaharragoetan *SearchRequest* galdera-objektu eta *SearchResponse* erantzun-objektu bakarrak zeuden edozein zelarik galdera mota ere. *SearchRequest* objektuan galderaren parametroak sartzen ziren eta *SearchResponse*ren erantzunaren emaitzak. Galdera egiteko behar ziren parametroak eskaeran sartu ondoren bidali egiten zen eta erantzunean, galderaren arabera, *SearchResponse*-ren emaitza atributuak atzitu behar ziren. Zein motatako galdera zen kontuan hartuta (*web, news, image...*) atributu guztiak ez ziren beterik egongo beti. Orain zein motatako galdera den kontuan hartuta halako eskaera eta erantzun-objektua erabiliko da.

*API*a erabiltzen duten garatzaileentzat bereziki zuzendurik dagoen *Bing Developer Center* berria estreinatzen da. Hemen aplikazioaren identifikazio kodea lortu eta egindako kontsulta kopurua ikusi daiteke, besteak beste.

Bigarren bertsio honetan sartu den punturik interesgarriena kontsulta kopuru mugagabea da. Hau lortu ahal izateko, ordea, berebiziko garrantzia du erabilera-baldintza berrituei kasu egitea.

Zer egin behar den kontuan hartuta proiektu honetan erabiliko diren *SourceType* motak *Web, Spell* eta *RelatedSearch* izango dira.
Ondoren bakoitza nola dagoen osatuta azalduko da.

4.2.1.1 Web.

Web-ean bilaketa egiteko balio duen *SourceType*a da.

- Kontsulta egiterako orduan.

Beti bete beharreko parametroak.

- ***SearchRequest.AppId:*** Aplikazioaren identifikazio zenbakia zehazteko.
- ***SearchRequest.Query:*** Webean zer bilatu behar den esan behar da hemen.
- ***SearchRequest.Sources:*** Bilaketa zein *SourceType* motan oinarritu behar den adierazteko.

- Aukerako parametroak.

- ***SearchRequest.Market:*** Zein hizkuntza eta zein estatutarako emaitzak moldatzea nahi den. Bilatzailearen deskribapena egitean esan bezala, kontsulta berdina egin ezkero munduko bi leku desberdinetatik emaitza bera ez da ikusiko. Estatu, hizkuntza, kultura... desberdinak direnez emaitzak moldatu egiten dira erabiltzaileak aurkitu nahi duena bizkorrago lortu ahal izateko. Honi merkatura moldatzea esaten zaio. Hizkuntza eta estatu jakin bateko emaitzak lortu ahal izateko kode internazionala idatzi beharra dago hemen. Adibidez *es-MX* Mexikoko gaztelaniarako, *en-CA* Kanadako ingeleserako, *en-IN* Indiako ingeleserako edo *de-AT* Austriako alemanierako. Ezer idatzi ezean

zerbitzariak berak erabiltzailearen IP aztertuz, *cookieak* ikusiz... nongoa den asmatzen saiatuko da emaitza moldatu ahal izateko.

- ***SearchRequest.Version:*** *Bing API*aren zein bertsio ibiliko den zehazteko balio du. Ezer idatzi ezean 2.1 bertsioa erabiliko da.
- ***SearchRequest.Adult:*** *Web* iragazkia konfiguratu ahal izateko. Helduentzat zuzenduriko edukia emaitzen artean sartu behar duen ala ez esan ahal izateko hauta ematen du. Hiru dira aukerak: *Off* (iragazkiak ez du ezer egingo), *Moderate* (irudi eta bideo sexualik agertuko ez den arren emaitzan sexuari erreferentzia egiten diotenak ager daitezke) eta *Strict* (sexuari buruzko bideo, irudi eta testurik ez da agertuko).
- ***SearchRequest.Latitude:*** Bilaketa latitude jakin batean egiteko.
- ***SearchRequest.Longitude:*** Bilaketa longitude jakin batean egiteko.
- ***SearchRequest.Options:*** *DisableLocationDetection* (erabiltzailea non dagoen automatikoki antzemateko) edo/eta *EnableHighlighting* (itzulitako emaitzan bilatu nahi izan den hitz edo esaldia nabarmentzeko) aktibatzeko.
- ***WebRequest.Count:*** Bilaketa bakoitzeko gehienez zenbat emaitza itzuli nahi den zehaztu daiteke honekin.
- ***WebRequest.Offset:*** Emaitza multzoan elementu bat atzitu ahal izateko posizioa adierazi behar da hemen.
- ***WebRequest.Options:*** *DisableHostCollapsing* edo/eta *DisableQueryAlterations* aktibatu ahal izateko.
- ***WebRequest.FileType:*** *WebSource*-k itzulitako emaitza zein formatutan izan behar den esan behar da hemen. Aukera zabala eskaintzen du:
  - *Microsoft*-en dokumentu motak: *Word (DOC), PowerPoint (PPT), Excel (XLS)* eta *Microsoft*-en testu aberastuko *(RTF)* dokumentuak.
  - Testu fitxategi soilak *(TXT, TEXT)*
  - Beste motak: *HTML, PDF, RSS, DWF (Autodesk Drawing File)*.

- Erantzunaren zatiak.

SourceType guztietan berdinak diren eremuak.

- ***SearchResponse.Version:*** Erantzuna bidaltzen duen objektuaren bertsio zenbakia.
- ***Query.AlterationOverrideQuery:*** Egin beharreko kontsulta bera baina hitz-gako bakoitzari karaktere berezi bat aurretik gehituta, prozesatua izan dela adierazteko.
- **Query.SearchTerms:** Egindako kontsulta zein den esaten du.

Web SourceType-an bakarrik aurki daitezkeenak.

- ***WebResult.Url:*** Emaitzaren *URL* ematen du. Izenburua (*Title*, estekaren testua erakusteko) eta *URL*az (helbidea) osatuta dago.
- ***WebResult.CacheUrl*:** *Bing*-ek cacheratutako orriaren *URL* ematen du.
- ***WebResult.DateTime:*** Bilatzaileak azken eguneraketa (*crawl*) egin dueneko urtea, hilabetea, eguna, ordua, minutua eta segundua duen *DateTime* objektu bat itzultzen du
- ***WebResult.DeepLinks:*** *DeepLink* objetuen array bat itzultzen du. *DeepLink* bat web-orri baten barruko esteka da. Esteka bakoitzeko objektu bat dago eta *DeepLink.Title* (izenburua) zein *DeepLink.Url* (helbidea) propietateak ditu.

- ***WebResult.Description:*** Emaitzaren HTML gorputzeko zati batekin osaturiko *string* bat da. Bilatu nahi izan diren hitzak dituen zatia hartzen da horretarako.
- ***WebResult.DisplayURL:*** *URL* helbidea soilik ematen du.
- ***WebResult.SearchTags:*** Bilaketa-etiketen arraya bilaketa honetarako.
- ***WebResult.Title:*** Web orriaren *"title"* etiketan zehaztutako *string*-a bueltatzen du.

4.2.1.2 Spell.

Ortografia akatsak antzeman dezake *SourceType* honek.

- Kontsulta egiterako orduan.

Beti bete beharreko parametroak.

- ***SearchRequest.AppId:*** *API*ak zein aplikazio den identifikatu ahal izateko.
- ***SearchRequest.Query:*** Bilatu behar dena.
- ***SearchRequest.Sources:*** Bilaketa zein *SourceType* motan oinarritu behar den adierazteko.

- Aukerako parametroak.

- ***SearchRequest.Market:*** Zein hizkuntza eta zein estatutarako emaitzak moldatzea nahi den. Bilatzailearen deskribapena egitean esan bezala, kontsulta berdina egin ezkero munduko bi leku desberdinetatik emaitza bera ez da ikusiko. Estatu, hizkuntza, kultura... desberdinak direnez emaitzak moldatu egiten dira erabiltzaileak aurkitu nahi duena bizkorrago lortu ahal izateko. Honi merkatura moldatzea esaten zaio. Hizkuntza eta estatu jakin bateko emaitzak lortu ahal izateko kode internazionala idatzi beharra dago hemen. Adibidez *es-MX* Mexikoko gaztelaniarako, *en-CA* Kanadako ingeleserako, *en-IN* Indiako ingeleserako edo *de-AT* Austriako alemanierako. Ezer idatzi ezean zerbitzariak berak erabiltzailearen IP aztertuz, *cookieak* ikusiz... nongoa den asmatzen saiatuko da emaitza moldatu ahal izateko.
- ***SearchRequest.Version:*** *Bing API*aren zein bertsio ibiliko den zehazteko balio du. Ezer idatzi ezean 2.1 bertsioa erabiliko da.
- ***SearchRequest.Options:*** *DisableLocationDetection* (erabiltzailea non dagoen automatikoki antzemateko) edo/eta *EnableHighlighting* (itzulitako emaitzan bilatu nahi izan den hitz edo esaldia nabarmentzeko) aktibatzeko.

- Erantzunaren zatiak.

- ***SearchResponse.Version:*** Erantzuna bidaltzen duen objektuaren bertsio zenbakia.
- ***Query.SearchTerms:*** Egindako kontsulta zein den esaten du.
- ***SpellResponse.Total:*** Erantzun bakoitzeko zenbat emaitza lortu izan denaren estimazio bat ematen du.
- ***SpellResponse.Results:*** Bilaketaren emaitzen *array* bat.
- ***SpellResult.Value:*** Gaizki idatzita egon ezkero iradokizuna emango du.

4.2.1.3 RelatedSearch.

Guk emandako kontsultarekin erlazionatutako bilaketak zeintzuk diren eskaintzen du *SourceType* mota honek.

- Kontsulta egiterako orduan.

Beti bete beharreko parametroak.

- ***SearchRequest.AppId:*** Aplikazioaren identifikazio zenbakia zehazteko.
- ***SearchRequest.Query:*** Zer bilatu behar den esan behar da hemen.
- ***SearchRequest.Sources:*** Bilaketa zein *SourceType* motan oinarritu behar den adierazteko.

- Aukerako parametroak.

- ***SearchRequest.Market:*** Zein hizkuntza eta zein estatutarako emaitzak moldatzea nahi den. Bilatzailearen deskribapena egitean esan bezala, kontsulta berdina egin ezkero munduko bi leku desberdinetatik emaitza bera ez da ikusiko. Estatu, hizkuntza, kultura... desberdinak direnez emaitzak moldatu egiten dira erabiltzaileak aurkitu nahi duena bizkorrago lortu ahal izateko. Honi merkatura moldatzea esaten zaio. Hizkuntza eta estatu jakin bateko emaitzak lortu ahal izateko kode internazionala idatzi beharra dago hemen. Adibidez *es-MX* Mexikoko gaztelaniarako, *en-CA* Kanadako ingeleserako, *en-IN* Indiako ingeleserako edo *de-AT* Austriako alemanierako. Ezer idatzi ezean zerbitzariak berak erabiltzailearen IP aztertuz, *cookieak* ikusiz... nongoa den asmatzen saiatuko da emaitza moldatu ahal izateko.
- ***SearchRequest.Version:*** *Bing API*aren zein bertsio ibiliko den zehazteko balio du. Ezer idatzi ezean 2.1 bertsioa erabiliko da.
- ***SearchRequest.Options:*** *DisableLocationDetection* (erabiltzailea non dagoen automatikoki antzemateko) edo/eta *EnableHighlighting* (itzulitako emaitzan bilatu nahi izan den hitz edo esaldia nabarmentzeko) aktibatzeko.

- Erantzunaren zatiak.

SourceType guztietan berdinak diren eremuak.

- ***SearchResponse.Version:*** Erantzuna bidaltzen duen objektuaren bertsio zenbakia.
- ***Query.SearchTerms:*** Egindako kontsulta zein den esaten du.

RelatedSearch SourceTypean bakarrik aurki daitezkeenak.

- ***RelatedSearchResult.Title:*** Zerikusia daukan bilaketa iradokizuna *string* moduan.
- ***RelatedSearchResult.Url:*** Iradokitako bilaketa egin ahal izateko zuzeneko helbidea.

Hemen agertu diren *SearchRequest, WebRequest, SearchResponse, Query, WebResult, SpellResponse, SpellResult* eta *RelatedSearch API*an definitutako klaseak dira. Klase horietan

inplementatutako propietateak eta eragiketak dira hauek. Hainbat zerrenda (*enumerator* ingelesez) daude bilaketa erabili nahi den erara moldatu ahal izateko. Besteak beste *AdultOption* (iragazkiaren maila), *SearchOption* (bilaketa-aukerak aldatzeko) eta *SourceType* (*SourceType* mota bat aukeratu ahal izateko) daude. Erroreak ere gerta daitezke zerbitzua erabiltzean. Hori dela eta *Error* klasea definitu da. Hona hemen zer eskaintzen duen:

- ***Error.Code:*** Errore mezu bati dagokio kodea.
- ***Error.Message:*** Errore mezua, deskribapena.
- ***Error.HelpUrl:*** Kode eta mezuekin laguntza eskaintzen duen web orriko esteka.
- ***Error.Parameter:*** Parametro baten errua izan ezkero parametroa bera itzultzen da hemen.
- ***Error.SourceType:*** *SourceType* jakin batekin bereziki lotutako errorea bada *SourceType*aren izena agertuko da.
- ***Error.SourceTypeErrorCode:*** *SourceType* jakin batekin bereziki lotutakoa bada errore kode berezi bat egongo da.
- ***Error.Value:*** Parametro batekin lotutako errorea izan ezkero kontuan izan ez den balioa itzuliko da hemen.

4.3 Google

Ia mundu guztian (Txinan ez, esate baterako) ezagunena eta gehien erabiltzen den bilatzailea dugu. Sinpletasuna, erabilerraztasuna eta emaitzen kalitate ona direla eta da ezaguna. Bere punturik garrantzitsuenak ondorengo hauek dira.

Hainbat produktu eskaini eta *API* aukera zabala[6] edozeinen eskura jartzen duen arren soilik bilaketetan zentratuko gara, hori baita proiektuarekin lotuta dagoena.

Bilaketak egiterakoan ez da behar esaldi luzeak idazterik, hitz sinpleak erabiliz emaitza onak lortzeko ahalmena du eta. Praktikotasuna bultzatu asmoz bilaketak ahotsaren bidez egin daitezke, *Chrome* arakatzailerako soilik diseinaturik baldin badago ere momentuz. Emaitzen kalitatea hobetu asmoz hainbat teknika erabiltzen ditu, hala nola erabiltzailearen kokapena zein den asmatuz IP helbidea ikusiz eta cachea aztertuz, hitzen zuzenketa ortografikoa eginez, sinonimoak kontuan hartuz eta honen emaitzak ere erakutsiz... Bilaketa zehatza egin ahal izateko karaktere bereziak erabil daitezke[7]. *“”* erabili ezkero, adibidez, hitz edo esaldi batekin idatzi bezala eta ordena berean bilaketa egiteko esaten du. * karakterea, ordea, esaldi baten hutsune batean erabiltzeko balio du eta tarte horretan edozein zati sartzeko esaten dio, hau da, “*Google* *” jarri ezkero *Google*-ren produktu guztiak erakutsiko ditu. Beste karaktere erabilgarri bat “*site:*” da. Web orri edo domeinu baten barnean bilatzeko balio du.

Hau dena nahikoa ez eta erlazionatutako bilaketak ere egin daitezke. Egindako bilaketa zein den kontuan hartuz erlazionatuta dauden hitz-gakoekin emaitzak erakusteko aukera dago. Hainbat iragazki eskaintzen ditu. Bilaketa zehatzagoa egiteko web-ean zehar irudiak, bideoak, blogak, liburuak... aurkitzeko aukera aukera dago. Horretarako *Google*-k berak garatutako zerbitzu desberdinetan oinarritzen da, irudietarako *Images*, mapetarako *Maps*, liburuentzat *Books*...

6 https://developers.google.com/products/
7 http://support.google.com/websearch/bin/answer.py?hl=en&answer=136861

Iragazkiak ez dira emaitza zehatzak lortzeko erabiltzen bakarrik bilaketa segurua egiteko ere balio dute. Helduentzat zuzendutako web orriak iragazteko gai da eta hiru mailetan desberdintzen da bere zehaztasuna: *Strict* (sexuari buruzko web orri guztiak, esteka soilak izanik ere, emaitzetatik kanpo uzten ditu), *moderate* (sexuari errreferentzia egiten dioten web orriak emaitzetatik kanpo uzten ditu baina ez hauetara daramaten estekak dauzkaten orriak) eta *no filtering* (ez du iragazketarik egiten).

4.3.1 Google API

Google eskaintzen duen *API*a oso zabala da. Merkaturatuta dituen hainbat produkturen zerbitzuak eskaintzen ditu bertan, baita horien funtzionalitatea beste gauza batzuetarako optimizatutako aplikazioak eskuragarri jarri ere. *Maps* (mapa estatikoak, *Javascript* erabiliz, negoziorako zuzendutako bertsioa...), *Earth, Latitude, Youtube, Google+, Analytics* (ohikoa eta mugikorretarako...), *Chrome/Chromium*...

Bilaketen aldetik ere aukera handia dago: *Desktop* (indexatzeko ahalmena duten mahaigaineko *gadget*-ak egin ahal izateko), *Custom Search* (web orri jakin batean bilaketa kutxa bat jarriz nahi diren domeinuetan bilaketa pertsonalizatua egin ahal izateko), *Books (Googleren Books* aplikazioan liburuen bilaketa egiteko), *Search Appliance* (enpresentzat zuzenduta, produktibitatea lortu asmoz beharrezko informazioa lortu ahal izateko. Hardware berezia erabiltzen da.) eta *Web Search*.

Azken hau zaharkitutzat jo zuten 2.010eko azaroaren 1ean. Dena dela web aplikazio honetarako ederki etorriko litzateke, web osoan bilaketa egiteko gai baitzen hainbat aukera eskainiz gainera. Zerbitzu hori, ordea, zaharkitutzat geratu zen *Google*-ren aldetik beste ikuspuntu batetik birdefinitu nahi baitzuten bilaketa *API*a eta.

Honen eboluzio modura bilaketa pertsonalizatua (*Custom Search*) kaleratu zuten, aurreko Web bilaketaren antzerakoa baina *API*a berrituz eta komunikazioa erraztuz, lehengoan *SOAP* eta *WDSL* erabili beharra baitzegoen eta oraingoan *REST* arkitektura malguagoa onartuz. Honen diferentziarik handiena bere funtzionalitatean datza. Erabiltzaileari bilaketa motor bat egiteko eskatzen zaio, non bilatu behar duen esanaz. Orri horietan soilik edo web osoan, orri horien gaia eta edukia kontuan hartuta, bilaketa egiteko gai da. Behin oinarria zehaztuta aplikazioa martxan jar daiteke. Zerbitzua den bezala erabili nahi dutenentzat web orrian erraz txertatu daitekeen testu kutxa bat eskaintzen du *Googlek*. Bestalde, zerbitzuak ematen dituen emaitzak aprobetxatuz aplikazioak landu nahi dituzten erabiltzaileek *API*ak eskaintzen dituen funtzioak aprobetxa dezakete. Dena dela, zerbitzua mugatuta dago eta 100 eskaera eguneko soilik egin daitezke doan erabili nahi izan ezkero.

Bi bilatzaile hauek aztertuta Bing erabiltzea erabaki zen. Biak zerbitzu oso onak diren arren *Microsoft*enak zerbitzu mugagabea eskaintzen du. *Googlek* zerbitzua eguneko 100 kontsultara mugatzeak bere gainean sortutako web aplikazioen erabilera asko mugatzen du. Hori dela eta ia ezinbestekoa bihurtzen da ordaindu beharreko zerbitzua kontratatzea. *Microsoft*-ek, ordea nahi besteko kuota eskaintzen die garatzaileei.

Bestalde, bilaketa motorraren “eraikuntzak” dakarren mugak daude. *Googleren* bilaketa *API*a erabiltzen hasi baino lehen bilaketa motorra sortu behar da, hau da, *API*ari zein webgunetan oinarritu behar den bilaketak egiteko esan behar zaio. “Ohiko” bilaketa egin daitekeen arren webean zehar, bilaketa motorra oinarritzen den webguneek emaitzen kalitatean eragina dute. *Bing*

API-n ez dago horrelakorik. Betiko moduan bilatzaile bat erabiltzen den bezala erabil daiteke *API* bitartez eskainitako bilaketa zerbitzua. Ez dago inongo mugarik eta URL bitartez bilatu nahi denaren eskaria egin eta emaitzak aztertu behar dira soilik, honek eskaintzen duen kalitatea *Bing*-ek berak bere webgunetik eskaintzen duenaren berdinak izanik.

Bi bilatzaileak boteretsuak eta erabilgarriak direla kontuan hartuta, xehetasun horiek erabakigarriak izan dira *Microsoft*-en *Bing API*aren aldeko apustua egiteko.

4.4 REST

Re*presentational* ***S****tate* ***T****ransfer* [4] [6] 2000. urtean sortu zen komunikazio arkitektura bat da. Roy Fielding-ek, HTTP-ren 1.0 eta 1.1 bertsioen sortzaileetako bat, bere doktorego tesian definitu zuen lehen aldiz.

Web-a moduko sistema banatuetan erabili daiteke. Bere sinpletasuna dela eta beste komunikazio arkitekturak baino ezagunagoa eta erabiliagoa izaten ari da gaur egun.

REST estiloa HTTP 1.1-ekin batera paraleloan garatu zen. Web-aren egitura *REST*-en adibide garbia da eta espresuki Roy Fielding-ek bere tesian hitz hau berau erabili zuen Web-aren sare printzipioak deskribatzeko. Web-ean *REST* arkitekturaren puntu garrantzitsuenak betetzen dira: HTTP bidez informazioa bidali eta jasotzen da eta URL-en bidez baliabideak atzitu. Arkitektura hau jarraitzen duten aplikazioak *RESTful* [5] direla esaten zaie. Arkitektura bat izanik, ez du axolarik zein motatako informazioa trukatzen den, abstrakzio maila altua duelako. Hori dela eta edozein baliabide onartzen ditu, hala nola *XML, XHTML, RSS*...

Informazio iturriak helbideratu eta definitu daitezke. Orokorrean, HTTP bidez informazioa era sinple batean transmititu ahal izateko balio du *REST*-ek, inolako mezu-truke mailarik edo saioa hasi beharrik gabe.

REST-en oinarritutako web zerbitzu batek honako puntuak jarraitzen ditu:

- Bezero/zerbitzari motako arkitektura da.

  Eramangarritasuna eta mugikortasuna eskaintzen du plataformen artean. Gainera eskalabilitatea errazten du bai zerbitzari zein erabiltzailearen artean.

- Mailakatutako sistema.

  Erabiltzaile eta zerbitzariaren artean hainbat maila egon daitezke eta ez dio inondik inora erabiltzaile zein baliabidea duen zerbitzariari trabarik egiten, ezta komunikazioa zailtzen ere. Gainera elkarren arteko komunikazioari segurtasuna eta eraginkortasuna eman diezaiokete.

- *Stateless* da, hau da, ez da saioen kontrolik egiten.

  Zerbitzariak erabiltzailearen aldetik behar duen informazio guztia HTTP goiburukoak bidaltzen zaio. Horregatik saio hasieraren beharrik ez da izaten eta eraginkortasuna

handitu eta diseinu zein inplementazioa errazten da zerbitzariaren aldean.

- Interfazeen orokortasuna.

  Bezero eta zerbitzariaren arteko interfazea nahikoa orokorra izan behar da bakoitza independenteki garatu eta hobetu ahal izateko besteari ezer eragin gabe.
  - *URL*-en bidez lortzen dira baliabideak. *URL*ak erabiltzeak erabiltzaileari eta programatzaileari zein baliabide atzitzen ari den erraz antzematen laguntzen dio. Direktorio motako *URL*-ek baliabidea nolakoa den eta zer atzitzen ari den argi erakuts diezaioke eta gainera batetik bestera erraz igaro daiteke.
  - Baliabide bat lortu ezkero, eta zerbitzariaren baimena izanik, hau aldatzeko aukera dago. Mezu bakoitzak nahikoa informazio ematen du, metadatuen bidez, nola prozesatu behar den jakiteko.
  - Hipermedia aplikazioaren egoera makina bezala. Zerbitzariak ezagutzen dituen ekintzen bidez soilik egiten dira egoera transizioak. Hau da, hipertestuetako esteken bidez soilik igaro daiteke egoera batetik bestera. Adibide bat jartzearren, arakatzaile bat web orri baten barnean esteken bidez igaroko da bakarrik, egoera batetik bestera igaroz. Nahiz eta erabiltzailea askotan jabetu ez, zerbitzariak ezagutzen dituen *GET, POST* eta beste komandoak erabiltzen ditu horretarako HTTP erabiltzen baita Web-ean eta. Gehienbat *GET*, baliabide bat lortzeko, *POST*, baliabide bat sortzeko, *PUT*, baliabide baten egoera eguneratzeko, eta *DELETE*, baliabide bat ezabatzeko. *SOAP* eta *WSDL* erabili ezkero tarteko maila batean kokatzen diren komandoak definitu behar izaten dira web aplikazioan egin behar diren funtzioak kontutan hartuz, lortuErabiltzaileak() adibidez. *RESTful* izanik zuzenean Web-aren arkitektura eta izaera errespetatzen da, sinpleago eginez web aplikazioen garapena.

- Erantzuna cachean gordetzeko aukera.

  Eraginkortasuna handitu asmoz. Informazioa *XML, JSON* edo beste edozein motakoa izan daiteke.

- Erabiltzaile aldeko funtzionalitatea handitu daiteke eskuragarri dagoen kodea jaitsiz eta exekutatuz. Ez da derrigorrezkoa betetzea puntu hau.

Aurreko puntuetan ikusitakoaren ildotik hauek dira REST arkitekturaren helburu zehatzak:

- Eskalabilitatea ahalbidetzea.
- Interfazeak orokorrak izatea, bezero eta zerbitzariaren arteko komunikazioa beti egon dadin eta bakoitza bere kasa garatzeko aukera emanez.
- Aplikazioaren osagaien (erabiltzaile aldea eta zerbitzari aldea, adibidez) arteko independentzia.
- Tarteko osagaien aukerako erabilera, eraginkortasuna handitu eta segurtasuna lortzeko.

Laburbilduz, *REST* komunikazio arkitektura sinple bezain ahaltsua da. HTTP (normalean) aprobetxatuz, beste inongo komandorik sortu gabe, bezero eta zerbitzariaren arteko komunikazioa ahalbidetzen du *URL*-en bidez eskaerak eginez egoera batetik bestera igaroz. Orain arte sortutako

teknologia aprobetxatzea da *REST*en oinarria. *SOAP*, HTTP protokoloaz baliatzen bada ere, komando edo eragiketak definitzea eskatzen du, *GET, POST, DELETE* eta antzeko komandoak bigarren maila batean utziz eta garapena zailduz. Gainera, eragiketa horiek HTTPren *POST* komandoan oinarritzen dira, protokoloak eskaintzen dituen beste aukerak desaprobetxatuz. *REST*-ek HTTP errespetatzen duenez eraginkortasun eta sinpletasun handiagoa lortzen du. Hori dela eta azken urteetan gero eta zerbitzu gehiagok *SOAP* alde batera utzi eta *RESTful API*ak eskaintzen dituzte.

4.5 JSON

***J**ava**S**cript **O**bject **N**otation* [7] datuen elkartrukerako formatu arina da. *Javascript*-en oinarrituta, gizakiak erraz irakurri dezakeen zerrenda adierazten ditu, karaktere-kate eta balioen bidez. Ezaguna da *AJAX* aplikazioetan *XML*-ren ordez erabiltzeagatik.

Douglas Crockford izan zen espezifikazioa egin eta ezagutzera eman zuen lehena. *State Software* konpainian erabiltzen zen 2001 inguruan eta handik aurrera pixkanaka-pixkanaka zabaltzen joan zen, 2002an webgune ofiziala sortuz[8]. 2005. urtean *Yahoo*-k bere web zerbitzu batuk formatu honetan ematen hasi zen eta *Google*-k 2006an bere *GData* web protokolo propioari buruzko iturriak *JSON* formatuan eskaintzen zituen.

Javascripten oinarrituta egoteak ez du esan nahi beste programazio lengoaiekin bateragarritasunik ez duenik, independentea baita lengoaiarekiko. *JSONen* web orri ofizialean eskuragarri dauden *parser* guztietarako erreferentziak daude, hala nola *C, PHP, Phython, Java* eta *ASP*.

*JSON*ekin bi egitura sor daitezke: Zerrenda ordenatuak eta gako/balio motako array asoziatiboak (objektu deiturikoak). Zerrenda ordenatuak [karakterearekin hasi eta] karakterearekin bukatzen dira. Tartean balioak joan daitezke, komekin bereizita, eta *string*-ak, zenbakiak, balio boolearrak objektuak edo arrayak izan daitezke. Array asoziatiboak edo objektuak, { karakterearekin hasten dira eta } karakterearekin bukatu. Tartean gako/balio bikoteak doaz, bikote bakoitza komekin bereizita eta gakoa balioarengandik : karakterea erabiliz. Gakoa *string* baten bidez adierazten da eta balioa, lehen esan bezala, *string*, zenbaki, boolear, objektu... moduan. Hona hemen bakoitzaren adibide bana:

- Zerrenda ordenatua:

```
[ “Sagarra”, “Laranja”, “Gerezia”, “Melokotoia”]
```

8 www.json.org

- Array asoziatiboa (objektua):

```
{
	"Izena": "Joseba",
	"Abizenak": ["Goikoetxea", "Aranbarri"],
	"Helbidea":
	{
		"Herria": "Mutriku",
		"Posta-kodea": 20830,
		"Kalea": "Nasa",
		"Zenbakia":12
	} ,

	"Ikastetxeak":
	[
		{
			"Izena": "San Jose ikastola" ,
			"Helbidea":{
					"Herria": "Ordizia",
					"Posta-kodea":  21453,
					"Kalea": "Goienkale",
					"Zenbakia":5
				}
		},
		{
			"Izena": "Mutriku BHI",
			"Helbidea":{
					"Herria": "Mutriku",
					"Posta-kodea": 20830,
					"Kalea": "Madalena Auzoa, Eskolagunea",
					"Zenbakia":null
				}
		}

}
```

Ondorengo hauek dira oinarrizko datu motak:

- Zenbakiak.
- *String*-ak.
- Boolearrak.
- *Arrayak* (zerrenda ordenatuak).
- Objektuak (array asoziatiboak).
- *Null*.

Objektuak baliozkotzeko *XML*-k ez bezala ez du inongo eskemarik. Garrantzitsua izaten den arren objektu bat ongi osatuta dagoen ala ez jakitea, askotan ez dira erabiltzen eskemarik. Hori dela eta, zenbait baliozkotze tresna ez dira oso ezagunak eta gainera ez daude behar bezain beste garaturik. Adibiderik garbiena *JSON Schema* da. Garatzaileen artean ez dago oso zabaldurik, ezta eskuragarri programazio lengoaia askotarako ere, oso erabilgarria izan arren.

JSON formatuan dagoen bateko datu bat atzitu ahal izateko bi era daude. *Javascript*-en *eval()* funtzioa erabiliz edo *JSON parser* edo itzultzaile baten bidez. Lehenegoak *Javascript* konpilagailuari deia egiten dio lortutako testua objektu bihurtuz. Hala ere segurtasun arazoak egon daitezke, edozein *Javascript* funtzio konpilatu eta exekutatu dezake eta. Konfiantzazko leku batetik

lortu ez bada, *script* maltzur bat exekutatu daiteke nahi gabe. Hau gerta ez dadin *JSON* itzultzaile bat erabiltzea komeni da. *JSON* testua bakarrik onartzen du, *scriptak* alde batera utziz. Gainera, arakatzaile modernoetan *eval()* funtzioa bera baino azkarrago lan egiten du.

Javascript datu egiturak *JSON* testu formatura itzuli nahi izan ezkero *stringify()* funtzioa erabili behar da. Argi ibili behar da datu egitura ziklikoak ematearekin, ez ditu onartzen eta.

*XML*rekin alderatuz nahiko antzekoa den arren diferentzia nabarmen batzuk daude elkarren artean. JSON formatuan dagoen testua *XML* bidez hainbat modutara adieraz daiteke. *JSON* objektuak, adibidez, ume-elementuekin edo atributuak erabiliz lortu daitezke *XML*-n. Goiko objektuaren adibidea *XML*-z honela izango litzateke:

```
<ikaslea>
	<izena>Joseba</izena>
	<abizenak>
		<abizena>Goikoetxea</abizena>
		<abizena>Aranbarri</abizena>
	</abizenak>
	<helbidea>
		<herria>Mutriku</herria>
		<posta-kodea>20830</posta-kodea>
		<kalea>Nasa</kalea>
		<zenbakia>12</zenbakia>
	</helbidea>
	<ikastetxeak>
		<ikastetxea izena=”San Jose ikastola”>
			<helbidea>
				<herria>Ordizia</herria>
				<posta-kodea>21453</posta-kodea>
				<kalea>Goienkale</kalea>
				<zenbakia>5</zenbakia>
			</helbidea>
		</ikastetxea>
		<ikastetxea izena=”Mutriku BHI”>
			<helbidea>
				<herria>Mutriku</herria>
				<posta-kodea>20830</posta-kodea>
				<kalea>Makalena auzoa, Eskolagunea</kalea>
				<zenbakia></zenbakia>
			</helbidea>
		</ikastetxea>
	</ikastetxeak>
</ikaslea>
```

Helbidea adierazteko beste modu bat hauxe dugu:

```
	<helbidea herria=”Ordizia” posta-kodea=21453 kalea=”Goienkale” zenbakia=5/>
```

Datu bera hainbat eratan adierazi ahal izateak zaildu egin dezake honen tratamendua, eskema zorrotz baten mende ez badago behintzat. *XML*-n eskemen erabilpen zabala, *JSON*en erabilpen urria den bezala (nahiz eta egon, badaude), da beste desberdintasun nagusi bat. XML dokumentu baten zehaztasun eta baliozkotze maila altua behar bada, berez beti izan beharko litzatekeena, eskemak beharrezkotzat jotzen dira. Oraintxe ikusi dugu bezala, datuak hainbat eratan erakutsi daitezke eta eskemarik gabe tratamendua asko zailduko litzateke. *JSON*en, ordea, datuak ia beti modu berean idazten dira eta ez dago arazo handiegirik gaizki osatutako objektu bat dela eta. Dena

den, ongi letorke *XML*-k duen baliozkotze maila *JSON*ek izango balu.

Mime mota ofiziala *application/json* da eta fitxategiaren luzapena *.json* izaten da. *Javascript* lan ingurune ezagunenek JSON era natiboan tratatzen dute. Horietako batzuk *jQuery, MooTools* eta *Dojo Toolkit* dira.

4.6 jQuery

Javascript scripting lengoaian oinarritutako liburutegi ezaguna da. *Javascript*ek berak eskaintzen dituen funtzioez gain beste hainbat funtzionalitate gehitzen ditu programatzea sinpletu asmoz. AJAX eskaeren tratamendua, DOM zuhaitzen manipulazioa eta animazioen sorkuntza, esate baterako, askoz errazagoa da jQuery erabili ezkero lerro gutxi batzuk idatziz hainbeste ekintza egin baitaitezke.

2006ko urtarrilean John Resig-ek New Yorken izan zen *BarCamp* ekitaldian aurkeztu eta abuztuan kaleratu zuenetik oso ezaguna bilakatu zen munduko web orri ezagunenek erabiltzen hasi baitziren. Sintaxi erraza eta agindu sorta zabal eta boteretsua eskaintzen dio programatzaileari, animazioak sortu, *DOM* elementuak nahi bezala manipulatu, gertaerak maneiatu eta *AJAX* teknologia aplikazioan oso erraz erabili ahal izateko aukera emanik. Hau guztia dela eta gaur egungo 10.000 webgune ezagunen %55ek erabiltzen dute. Eta ez da gutxiagorako. Web orri eta aplikazio dinamiko zein ikusgarriak erraz egitea ahalbidetzen du liburutegi honek.

Dena dela, funtzionalitate guzti horiek nahikoa ez eta beste batzuk gehitu nahi izan ezkero ez dago arazorik, pluginak egin daitezke eta. Lan zehatzak burutzeko oso aproposak dira plugin hauek. *JQuery*ren web orrian hainbeste aurki daitezke, Horien artean *XML* eta *XSLT* tresnak, *cookie* maneiatzaileak, *AJAX*erako laguntzaileak eta baita Commodore 64 konputagailuaren emuladoreren bat ere.

JQuery liburutegia HTML orri bati gehitzea oso erraza da. Web orri ofizialetik iturburu kodea jaitsi eta gure web orrian beste edozein *Javascript* fitxategi gehitzen den bezala egitea da *head* atalean *script* etiketa gehituz.

```
<script type="text/javascript" src="jquery.js"></script>
```

Fitxategia konprimituta edo konprimitu gabe lortu daiteke. Nola funtzionatzen duen ikusteko araztailearekin oso egokia da konprimitu gabeko bertsioa lortzea, kodea garbi eta irakurterrazagoa baita. Iturburu kodea jaitsi gabe ere erabili ahal da *jQuery*, *Google* eta *Microsoft-ek* edozeinentzat eskuragarri jartzen baitute beraien zerbitzarietan.

4.6.1 Sintaxia eta aginduak

Aginduak idazteko orduan sintaxi berezia erabili behar da [9].

4.6.1.1 Aukeratze komandoak

Lehenbizi aukeratze komandoa erabili behar da eta ondoren aplikatu behar zaion eragiketa.

```
$("p").addClass("klaseBat");
```

Hemen HTML dokumentuko "p" elementu guztiei "klaseBat" balioko klase atributua gehitu zaie. Hainbat aukeratze komando daude:

- `$(".azpititulu")` : "azpititulu" balioko klasea duten elementuak hautatzen ditu
- `$(".azpititulu.ezkerMenu")` : "azpitutulu" balioko klase guztien artean "ezkerMenu" balioa duten klaseak soilik aukeratzeko.
- `$("#elementuID")` : "elementuID" identifikatzailea duten elementuak aukeratzeko.
- `$("p#IDbakarra .klaseBat")` : "klaseBat" klasedun elementu guztietatik "Idbakarra" identifikatzailea duten *div*-ak hautatzen ditu.

Aukeratze komando guztien zerrenda ikusteko joan web orri honetara[9].

4.6.1.2 Agindu bereziak

Lehen esan bezala jQueryk programatzea asko sinpletzen du agindu berezi batzuei esker Javascript hutsean hainbat lerro behar izaten dena honekin lerro gutxi batzuen bitartez nahikoa izaten baita. Hainbat agindu eskaintzen ditu jQuery-k lan egiteko baina hona hemen aipagarrienak direnak.

4.6.1.2.1 Gertaeren kudeaketa

JQuery-k eskaintzen duen funtzionalitateetako bat gertaeren kudeaketa da. Hona hemen adibide batzuk nola erabiltzen den ikusteko.

```
$(document).ready(function(){
      $("p").text("Dokumentua kargatu berri da.  Elementuak manipulatu daitezke
inongo arazorik gabe.")
});
```

ready DOM zuhaitza kargatu eta irudiak jaitsi direnean automatikoki abiarazten den gertaera da. Oso erabilia den arren, `<body onload = ... >` metodoarekin batera ez da erabili behar. Honek, dokumentua kargatu bezain laister *p* elementu bat sortuko du jarritako mezuarekin.

```
$("#elementuID").keyup(function(event) {
      kont++;
      mezua = kont + ". karakterea sartu da";
      $.print(mezua, 'html');
});
```

"elementuID" baliodun identifikatzailea duen elementuak fokua duenean eta teklatuko tekla bat askatzen denean mezu bat erakutsiko du, zenbat aldiz askatu den esanaz.

Animazio efektuak jartzeko, *CSS* propietateak eta HTML dokumentua aldatzeko eta beste hainbat funtzio burutzeko era berean egin daitezke.

4.6.1.2.2 AJAX eskaeren kudeaketa

AJAX erabili ahal izateko eskaintzen duen erraztasunak oso interesgarriak dira. Betidanik hainbat

9 http://api.jquery.com/category/selectors/

lerro idatzi behar izan dira *AJAX* eskaerak egin eta erantzuna kontrolatu ahal izateko. *JQuery*-k ordea metodo nagusi gutxi batzuen bidez edozein egoera kudeatzeko gai da.

```
$.ajax({
  url: "proba.html",
}).done(function(html) {
  $("#emaitzak").append(html);
});
```

proba.html orriaren eskaera egin eta tranferentzia burutzean gure dokumentuan “emaitzak” id duen elementuaren bukaeran txertatzen du.

$.ajax funtzioa boteretsu zein malgua da. Behe mailako interfazeko funtzio bat izanik aukera sorta handia du nahi bezala erabili ahal izateko. Aukera guztiak ikusi ahal izateko jo orri oinean dagoen helbidera[10]. Hainbeste aukera izanik, programatzaileari lana erraztu asmoz, “goi mailako” funtzioak ere badaude $.ajax funtzioan oinarritua direnak, eta lan jakin batzuetarako prestatuak izanik kode gutxiago idatzita emaitza berberak lortzeko ahalmena daukate.

```
$.get("proba.php", { izena: "Aitor", abizena: "Arrizabalaga" },
       function(erantz){
       alert("Erantzuna hau izan da: " + erantz);
});
```

$.get funtzio honek *get* komando baten bidez *proba.php* orrira `“izena: Aitor”` eta `“abizena: Arrizabalaga”` balioak bidaliko ditu eta erantzuna lortzean erakutsi egingo du. Honen “behe mailako” baliokidea hau izango litzateke:

```
$.ajax({
  url: proba.php,
  data: { izena: "Aitor", abizena: "Arrizabalaga" },
  success: function(erantz){
       alert("Erantzuna hau izan da: " + erantz);
       },
  dataType: HTML
});
```

Argi ikusten den bezala bigarren adibidean lerro gehiago idatzi beharra dago lan bera egiteko. Horregatik lan zehatzak burutzeko ona da funtzio jakin horiek erabiltzea eta malgutasun handiagoa lortu nahi izan ezkero, erantzunaren formatu mota dela eta, adibidez, behe mailakoak erabiliz nahi dena lortu daiteke.

4.6.1.2.3 Callback funtzioak

Kontuz jokatu behar da erantzuna jaso eta abiatu behar den funtzioa nola jartzen den. Animazioetan, adibidez, bukatu ondoren exekutatu behar den funtzioa gaizki jarri ezkero animazioa bera bukatu gabe dagoenean exekutatzen has daiteke, erroreak emanik. Hau ekiditeko funtzio bat bestearen barruan jarri beharra dago.

```
$("p").hide(1000);
alert("p guztiak ezkutatuta daude orain");
```

Callback funtzioak deritze mota honetakoei. 1000 milisegundutan *DOM* zuhaitzeko p etiketa guztiak ezkutatu egingo dira. Hau burutzean oharra erakustea nahi dugun arren ez da horrela

10 http://api.jquery.com/jQuery.ajax/

izango. Animazioa martxan dagoen bitartean oharra agertuko da. Hori saihesteko funtzio nagusiaren barruan joan beharko litzateke.

```
$("p").hide(1000,function(){
                alert("p guztiak ezkutatuta daude orain");
});
```

Era berean honako arazoa aurki dezakegu.

```
$.get('webOrriIkaragarria.html', nireCallbackFuntzioa('bat', 'besteBat'));
```

Hemengo arazo nagusiena zera da: *nireCallbackFuntzioa* deia *get* egin ondoren exekutatzea nahi dugun arren lehenik funtzio hori bera exekutatuko da eta ondoren *get* exekutatzen saiatuko da. Ez du zentzu handiegirik honek. *Javascript*-ek kasu hauetan funtzio erakusle bat espero izaten du eta hori dela eta modu honetan jartzea nahikoa izango litzateke.

```
$.get('webOrriIkaragarria.html', function(){
  nireCallbackFuntzioa('bat', 'besteBat');
});
```

Orain, *get* egin ondoren exekutatuko da guk nahi dugun funtzioa inongo ezusteko arazorik gabe.

4.6.2 Laburpena eta informazio osagarria

Hauxe da *jQuery*-k eskaintzen duena, laburbildurik. Ikusten den bezala erraz ikasi daiteke eta ez dauka inongo arazorik nola funtzionatzen duen ikusteko, *Javascript*-en oinarrituta baitago eta ondorioz, bata erabiltzen jakin ezkero bestea ulertzeko inongo arazorik ez dago. Adibide praktiko geiago ikusteko jo *W3Schools*en webgunera [10]. Biak konbinatuz web orri dinamikoak egiteko badaitezke ere, orokorrean DOM zuhaitza eta HTML dokumentua aldatzeko gauza dela har daiteke "soilik" (beste hainbat funtzionalitate izanda ere). Itxura aldetik gehiago landutako web orriak egin nahi badira, *jQuery UI* dago[11]. *JQuery*-k eskaintzen dituen animazio efektuetan oinarrituta, "goi mailako" abstrakzio maila ematen du *widget*ak, gai pertsonalizagarriak eta bezero-web orri interakzio aberatsa inplementatu ahal izateko, hala nola arrastatu eta utzi (*drag-and-drop*) egin ahal izatea eta aurrerapen barrak erakustea. Mugikor eta tablet-etarako optimizatuta dagoen bertsio bat ere argitaratu zen, 2010. urtean. *JQuery Mobile*[12]-kin mugikorrentzat zuzendutako web orri dinamikoak egin daitezke ukipenezko interakzio eta mugimenduak aprobetxatuz HTML5-ean, *jQuery* eta *jQuery UI*-n oinarrituz.

JQuery doako kode irekiko proiektua da eta bi lizentziarekin banatzen da: *MIT* lizentzia eta *GNU* lizentzia publikoko bigarren bertsiopean. Informazio gehiago *Wikipediako* artikuluan aurki daiteke[11].

11 http://jqueryui.com/
12 http://jquerymobile.com/

5. DISEINUA

5.1 Sarrera

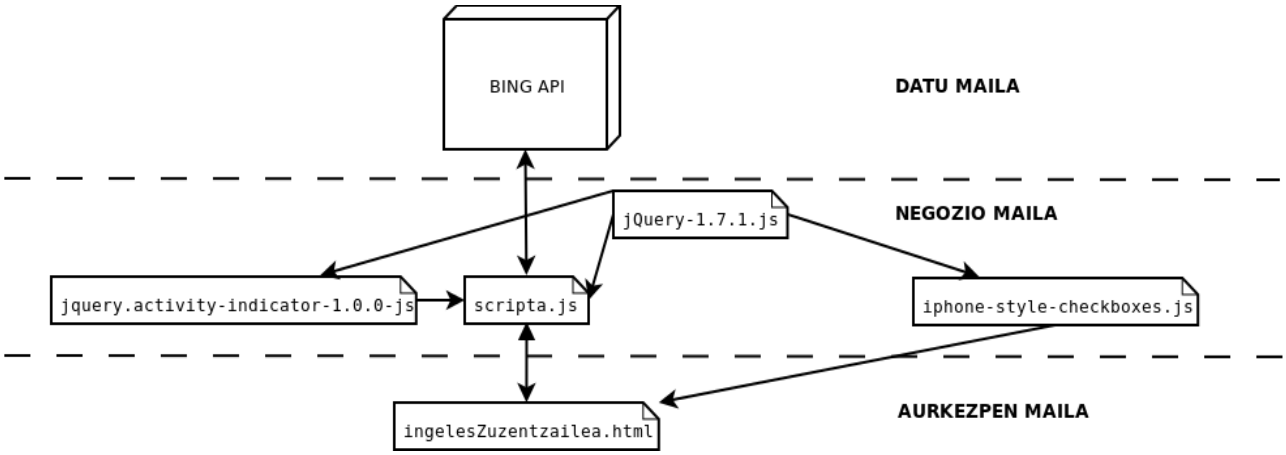
Hona hemen aplikazioaren funtzionalitateak deskribatzen dizuten erabilpen kasu eta sekuentzia diagramak, aplikazioaren arkitekturaren azalpena eta interfazearen hasierako zirriborro eta bukaerako irudiarekin batera.

Proiektuaren helburua *mashup* aplikazio bat egitea da, *API* baten erabileran oinarrituz eta web teknologiak erabiliz. Honela osatu da ingeles zuzentzailea.

5.2 Aplikazioaren arkitektura

Hasieratik ingeles zuzentzailea web aplikazio bat izatea nahi izan da, edonondik eta edozein motako makinatik atzigarri izan dadin. 3 mailatako arkitektura jarraitu du.

- HTML eta CSS bidez aurkezpen maila landu da.
- Negozio maila *Javascript*-en bidez egingo da. Aurkezpen mailatik lortutako testua datu mailak eskaintzen duen *API*aren interfazea erabiliz bilaketa egin eta emaitzak jasoko dira. Ondoren hauek aztertu egin beharko dira testuan okerrik ote dagoen jakiteko.
- Datu maila *Bing API*a izango da. Honekin ez da arduratu beharrik izango, *Microsoftek* eskaintzen duen web zerbitzu bat baita. Negozio mailatik *REST* arkitekturari esker honekin komunikatu beharko da bilaketen emaitzak lortzeko.



5. irudia: Aplikazioa nola antolatzen den adierazten duen diagrama

Maila bakoitza desberdintzeko fitxategi desberdinetan banatzea erabaki da. Aurkezpen maila alde batetik eta negozio maila beste aldetik. Datu maila Web-ean dagoenez ez dago ukitu beharrik.

Aztertutako teknologien atalean esan bezala *Bing API*-an oinarritzen da aplikazio hau. Mugarik gabeko doako kuota eta bilaketa aukera zabala ematen duelako ederki datorkio aplikazioari, oinarrian dagoen ideia garatzeko ezinbestekoa delarik. *API*ak *REST* arkitektura jarraituz datuen elkartrukerako XML zein JSON formatua eskaintzen du. Web aplikazio bat garatu nahi dela kontuan hartuta, *Javascript* lengoaia ezaguna erabiltzeko erabakia hartu da. Ondorioz, datuen

elkartrukerako *JSON* formatu sinple eta tratamendu errazekoa aukeratzea pentsatu da, *Javascriptekin* daukan erlazio estuaz aintzakotzat hartuta. Erabilerraztasuna eta eraginkortasuna bultzatu asmoz *jQuery*ren ahalmenaz baliatzea egoki datorrela iritzi da, honekin batera integratu daitezkeen pluginak erabili daitezkeelarik.

5.3 Garapen prozesuaren nondik norakoak

Helburuaren dokumentuan, lan metodologia atalean, *XAMPP* (eta *PHP*) erabili behar zirela erabaki zen. Dokumentazio falta zela eta idatzi zen hori. Denbora hartu ostean informazioa bilatu eta irakurtzen PHP eta web zerbitzaririk ez zirela behar erabaki zen, guztia *Javascript*ekin egin ahal zela ohartu nintzen eta.

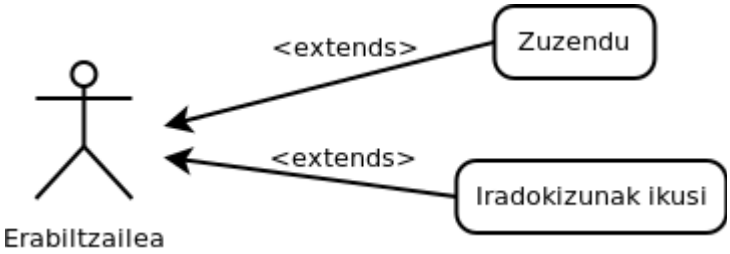
Aplikazioa garatzen joan ahala *Javascript* hutsa erabili ordez tresna ahaltsuagoen beharra izan da. Hori dela eta *jQuery* erabiltzea erabaki da. Aurkezpen mailan animazio dinamikoak sartzeko, DOM zuhaitza erraz manipulatzeko eta *AJAX* eskaeran modu sinple batean egiteko balio izan du. *Javascript*-eko funtzioak erabiliz j*Query*-k eskaintzen dituenak aprobetxatuz lana askoz errazago eta erosoago egin da. Gainera, *jQuery*-ren funtzionalitatea pluginen bidez zabaldu daiteke. Aplikazio honetan, testua aztertzen ari den bitartean adierazle bat sartzea erabaki da [12]. Gainera, idatzi ahalako iradokizunak aktibatu eta ezkutatzeko *checkbox* berezi bat erabili da [13].

Hasierako asmoa testu kutxa handi bat izatea zen. Bertan testua idatzi ahala okerrak erakustea hitzak kolore desberdinez azpimarratuz. Hitz oker bakoitzaren gainetik arratoia pasatu ezkero adibideak eman beharko lituzke. Hau HTML hutsean ezin zela inplementatu ikustean interfazea birdiseinatu behar izan zen. Puntu honetatik aurrera errore detekzioaz eta *API*arekin komunikazioaz arduratuko zen negozio maila inplementatzen hasi zen.

Pixkanaka-pixkanaka garapena aurrera joan ahala aplikazioak eraginkortasun baxua zuela iritzi zen, okerrak antzemateko denbora gehiegi igarotzen baitzuen. Arazo hori dela eta aplikazioa sakon-sakonetik birdiseinatzea erabaki zen, iterazioak motzago eginez (testu osoa hartu beharrean zatiak hartuz soilik) eta hobeto aprobetxatuz, barnean eragiketa gehiago eginez, beste iterazio batzuen beharra kenduz. Gainera, *Javascript* eta *jQuery*-rekin gero eta hobeto moldatzeak eraginkortasuna handitzea ahalbidetu zuten, kode zati batzuk eragiketa gutxi batzuekin ordezkatu izan baitziren.

Proiektuaren zuzendariak, aplikazioa hobetu asmoz idatzi ahala hurrengo hitza iradoki ahal izan zedin eskatu zuen. Hori dela eta, funtzionalitate berria diseinatu eta inplementatzeari ekin zen. Lehendik zeuden funtzioak aprobetxatuz berri batzuk gehitu ziren.

5.2 Erabilpen-kasuen eredua



Argi ikusten den bezala aplikazioak bi funtzio soilik eskaintzen ditu: Okerren detekzioa (testuaren zuzenketa deiturikoa) eta hitzen artean idatzi daitezkeen beste hitz batzuen iradokizunak.

5.2.1 EK1 erabilpen-kasua: Zuzendu

Aktorea: Erabiltzailea

Deskribapena: Erabiltzaileak zuzendu botoia jotzean testua aztertuko du eta oker egon daitezkeen zatiak erakusten ditu

Gertaera-fluxu normala

1.- Erabiltzailea: "Zuzendu" botoia joten du.

2.- Sistema: Testu kutxan idatzitako testua aztertzen du. Zuzenketa kutxan susmagarriak erakusten ditu.

Gertaera-fluxu alternatiboak

1.- Erabiltzailea: Testu kutxan ezer idatzi gabe zuzendu botoiari ematen dio.

2.- Sistema: Hutsik badago ezer ez du egingo.

5.2.2 EK2 erabilpen-kasua: Iradokizunak ikusi

Aktorea: Erabiltzailea

Deskribapena: Erabiltzailea testua idazten dabilen bitartean idazten ari den testu zatian jarri daitezkeen hitzak iradokitzen dira, bai hitz tartean zein bukaeran.

Gertaera-fluxu normala

1.- Erabiltzailea: Kurtsorea bi hitzen artean kokatzen du, hau da, hitz baten bukaeran. Hitz baten bukaeran edo hurrengoaren hasieran izan daiteke baina bukaeran izatea erabaki da, inongo arrazoi jakinik gabe.

2.- Sistema: Kurtsorearen posizioaren aurreko eta ondorengo hitzak hartu eta bien artean zein hitz jarri daitezkeen iradokiko du.

Gertaera-fluxu alternatiboak

1.- Erabiltzailea: Testuko azken hitzaren bukaeran kurtsorea jartzen du.

2.- Sistema: Azken bi hitzak hartuta, azkena nola bukatu (puntuazio ikurrik ez baitu, alajaina, eta ondorioz idazten bukatu gabe dagoela har daiteke) eta hurrengo hitza zein izan daitekeen iradokitzen du.

1.- Erabiltzailea: Hitz baten hasieran eda hitz barnean jarri du kurtsorea.

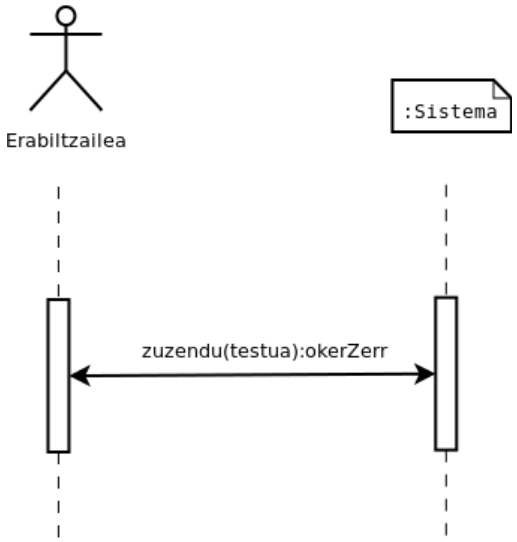
2.- Sistema: Ez du ezer egingo.

1.- Erabiltzaileak: Azken hitzeko azken posizioan kurtsorea jarri du baina puntuazio ikus bat dago.

2.- Sistema: Ez du ezer egingo, hitza eta esaldia bukatutzat jotzen baita.

5.3 Sekuentzia-diagramak

5.3.1 EK1: Zuzendu



Izena: zuzendu(testua):okerZerr

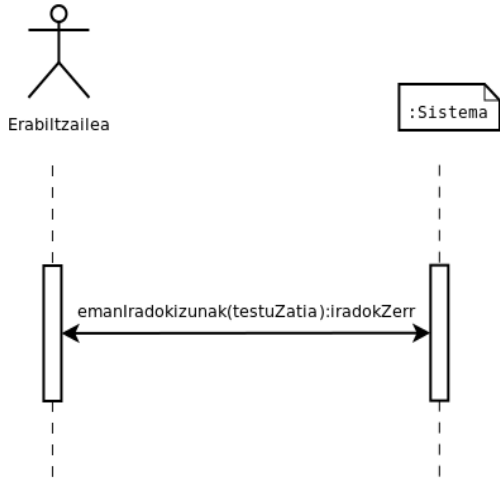
Erantzunkizunak: Testu osoa (edo zati bat) emanda okerrak zein diren antzeman eta itzultzen ditu

Aurrebaldintza: --

Postbaldintza: --

Irteera: okerZerr = zerrenda (<okerra>). Okerra = objektua(okerMota, azalpena, iradokZerr). IradokZerr = zerrenda (<iradokizuna>). Iradokizuna = *string*.

5.3.2 EK2: EmanIradokizunak



Izena: emanIradokizunak(testuZatia):iradokZerr

Erantzunkizunak: Testu zati bat emanda nola bukatu eta tartean zein hitz sartu iradokitzen du.

Aurrebaldintza: Testu zatia ez da izan behar hutsa. Kurtsorea hitz baten bukaeran egon behar da. Gainera hitz bukaeran ez da egon behar puntuazio ikurrik, hitza idazten bukatu dela kontsidera daiteke eta.

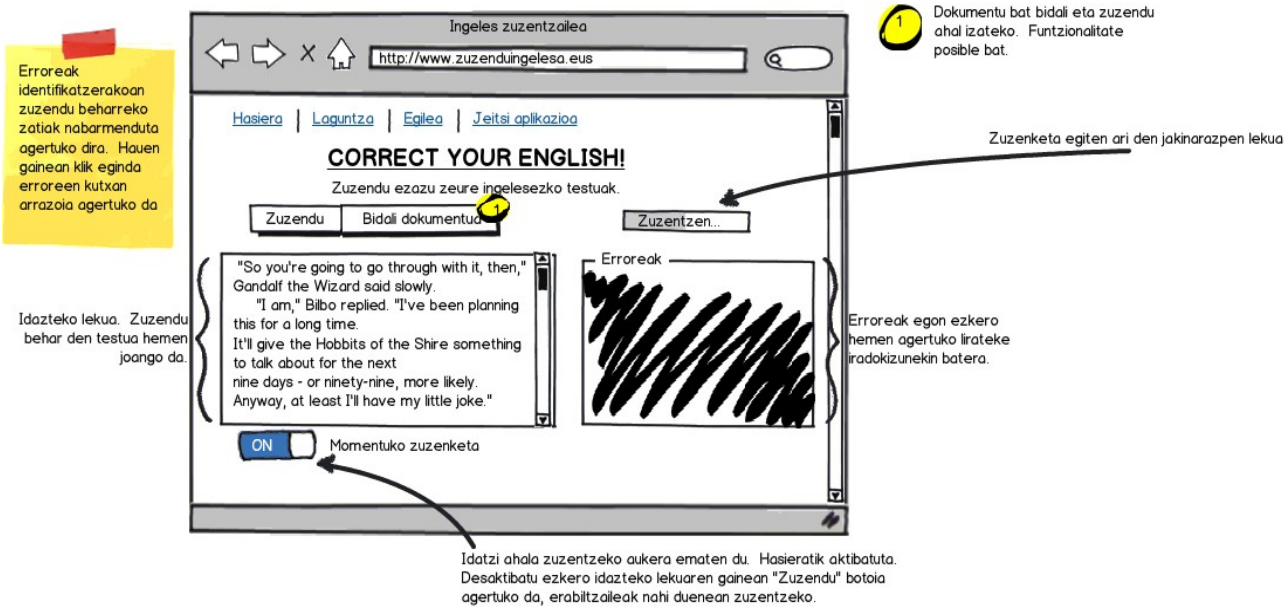
Postbaldintza: --

Irteera: IradokZerr = zerrenda (<iradokizuna>). Iradokizuna = *string*.

5.4 Interfazearen diseinua

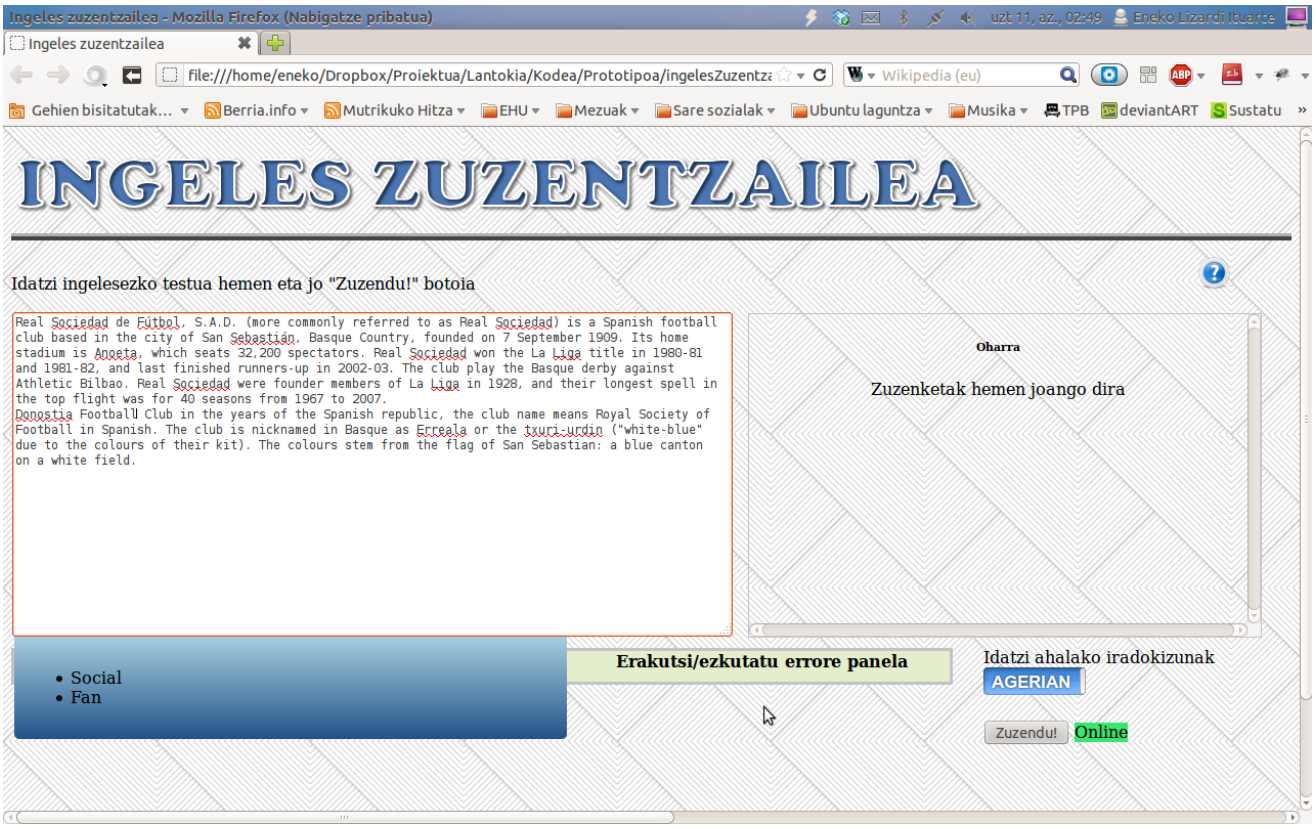
Hona hemen interfazea nola izan beharko litzatekeen hasieran egindako zirriborro bat. Ikusten den bezala ideia nagusia bukaeraraino zaindu den arren hobekuntza batzuk sartzea erabaki da.

Zirriborro hau egiteko *Balsamiq Mockup* softwarea erabili da[13].



6. irudia: HTML orriaren diseinua

13 http://www.balsamiq.com/products/mockups



7. irudia: Honela geratu da bukaeran aplikazioa

Elementuen tamaina eta kokapena erabiltzaileari modurik garbienean ahalik eta informazio gehien ematea kontuan hartuta egin da. Erraz ikus daitezke erroreak non dauden eta nahi izan ezkero testuan idazten zentratzeko iradokizun kutxa eta zuzenketa panela berehala eta erraz ezkutatu daitezke.

Kolore arinak erabili dira orokorrean ingurugiro atsegina izan dezan erabiltzaileak. Okerrak kolore desberdinetan jartzea ez da itxura kontua izan soilik. Arrazoi nagusiena okerraren izaerarekin bat datorren kolore bat hautatzea izan da. Ortografia akatsak larrienak (eta berehala zuzendu beharrekoak) direla esan nahi da. Hori dela eta gorri bizia du ohartarazpen kutxak. Gramatika akatsak berde argiarekin ageri dira. Susmagarriak diren heinean ezin da esan ziur okerra denik. Horregatik kolore argi hori.

Garatzerako orduan web arakatzaile desberdinak kontua hartu dira. *Firefox* 13.0.1, *Opera* 12.00, *Chromium* 18.0.1025.168 eta *Internet Explorer* 9 arakatzaileetan probatu da eta itxura aldetik arazorik egon ez daitezen egokitzapen script txiki batzuk sartu dira denetan berdin ikusi dadin. 1280x800 pixeleko bereizmena duen pantailan garatu da aplikazioa. Beste erresoluzio batekin ikusi ezkero (erresoluzio aldaketa handia egon ezkero 1280x800ekoarekin alderatuz) elementuak beraien berezko lekutik mugituta ikus daitezke.

6. INPLEMENTAZIOA

6.1 Kodearen azalpena gainetik

Kodea nola dabilen ulertzeko bi erabilpen kasuak ongi berezitu behar dira, nahiz eta elkarren artean erlazionatuta egon.

6.1.1 Zuzenketa

Alde batetik testuaren zuzenketa dugu. Erabiltzaileak klik egitean zuzenketa guztia kontrolatzen duen funtzio nagusira salto egiten dugu. Bertan testua osatzen duten hitz eta esaldiak datu egitura batean sartzen dira, zenbakiak eta puntuazio ikurrak alde batera uzteko eta azterketa errazagoa izateko. Behin datu egituran guztia sartuta hitzak banan banan aztertzen hasten da, ortografia akatsak antzeman ahal izateko. Hau *Bing API*ak eskaintzen duen *Spell* zerbitzuari esker egiten da.

Behin errore ortografikoak detektatuta gramatika akatsak antzeman behar dira. Esaldiz esaldi hitzak hirunaka batzen dira, trigramak osatuz. Teknika hau erabiliz zentzu minimo bat duten bilaketak egitea lortzen da, emaitzen artetik gure testura egokituko diren emaitzak lortuz.

Trigramen erabilera ongi dator aplikazioaren helburuak bete daitezen. Hitzak binaka hartuz zentzu nahikoa ez duten emaitzak lortuko lirateke eta ez da komeni. Tetragramak osatzeak, gure testuan esan nahi denarekin bat egiten duten emaitza hobeagoak lortzen badira ere, trigramekin alderatuz, bilaketetan emaitza oso gutxi lortzearen arriskua dago. Trigramekin zentzuzko emaitza kopuru nahikoa lortzen direlako hitzak hirunaka hartzearen apustu egitea iritzi da.

Behin trigrama sortuta, url-a sortu eta *AJAX* eskaera egiten da. Emaitza sorta lortzean zuzenketa modulura igarotzen da. Funtzio honetan oker eta zuzen zer dagoen erabakitzen da eta baita momentuko iradokizunetarako zein emaitza kontuan hartu eta zein ez.

Zuzenketa funtzioan trigramaren zati desberdinak kontuan hartzen dira: Trigrama berez den bezala, lehen bi hitzak, azken bi hitzak eta lehen eta azken hitzak. Ikusi adibide modura ondorengo trigrama:

made of wood

Trigramaren hiru zatiak hauexek izango lirateke:

1.- *made of wood* (trigrama bera).

2.- *made of* (lehen eta bigarrena).

3.- *of wood* (azken biak).

4.- *made wood* (lehen eta azkena).

Zati hauek gogoan, lortutako emaitza multzoan bilaketa egiten da espresio erregularren bidez ea zein agertzen den gehiago. 1, 2, eta 3 diren bezala bilatuko dira, agerpen bakoitzarekin kontagailuak gora eginez, baina 4.aukera *made* eta *wood*-en tartean hitzen bat bilatzen duenean egingo du gora bere kontagailuak. Gauzak horrela, emaitza guztiak aztertu ondoren gehien agertzen dena huraxe izango da zuzen dagoena. Lehena ez bada gehien agertu dena aztertzen gauden trigrama okertzat joko da. Hala ere kasualitateak egon daitezke emaitza multzo horretan eta hori

zuzendu asmoz azken konparaketa bat egiten da. Gehien agertu dena 2, 3 edo 4. zatia izan bada lehenaren agerpen kopuruari zenbateko aldea atera dion kalkulatzen da. Bost edo handiagoa bada kasualitatea dela alde batera uzten da. Iradokizunak lortu eta zuzenketa panelean jarri behar da.

Iradokizunak lortu ahal izateko, lehen aipatutako zatien agerpen bakoitzean emaitza multzoan zein posiziotan aurkitu den gordetzen da datu egitura batean. Iradokizunak errazago bilatu ahal izatea dakar honek. Zuzenean multzoko posizio jakin horietara joan eta trigrama zati horren agerpenaren testuingurua lortu behar da. Aipatu beharrekoa da emaitza multzo guztian agerpenak ikusi aurretik tratamendu bat egin behar dela. Askotan emaitzetan interesatzen ez diren puntuazio karaktereak hitzei lotuta agertzen dira. Hori dela eta aldez aurretik kendu ezean agerpenak identifikatzerakoan hainbeste oker egin daitezke, trigrama normalean okertzat hartuz, berez ez denean. Iragazki hori beti egin behar izaten da, bai zuzentzerako orduan zein iradokizunak aurkitzerakoan. *Javascripteko replace* funtzioa erabiltzen da horretarako. Espresio erregularrak erabiliz . (puntua), , (koma), *:* (bi puntu), *;* (puntu koma), _ (azpimarra), *?*(galdera ikurra), *!* (harridura ikurra), *(* (parentesia irekitzeko karakterea) eta *)* (parentesia ixteko karakterea) moduko karaktereak kentzen dira.

Iradokizunak lortu ostean zuzenketa panelean agertu behar da oharra. Horretaz arduratzen da *sortuErroreMezua* funtzioa. Errore motaren arabera kutxari kolore desberdina ematen dio (gorria ortografia akatsa denean eta berdea gramatikaren kasuan) eta gainera iradokizunak jarri eta azalpen bat ipintzen du.

6.1.2 idatzi ahala iradokizunak eman

Idatzi ahalako iradokizunek antzeko funtzionamendua dute. Erabiltzaileak idatzi ahala kurtsorea non dagoen ikusiko da. Hitz bukaeran egotean iradokizunak eman behar dira. Orduan iradokizunetaz arduratzen den funtzioak kudeatuko du guztia. Aurreko eta ondorengo hitzak lortu eta bigrama osatzen du. Kasu honetan ezin dugu trigrama osatu, okerra dagoen ala ez esan ordez esaldia osatuko duen beste hitz baten bila baikabiltza. Url-a modu berezi batean osatu ondoren, bigramaren lehen eta azken hitzaren artean edo azken hitzaren ondoren * karakterea jarriz (egoeraren arabera), bilaketa egiten da.

Emaitzak lortutakoan bigramaren bilaketaz *API*ak emandako multzoan zehar berriz ere *zuzendu* funtzioa arduratuko da. Honek, lehen bezala, emaitza bakoitzean zehar aztertzen joango da, oraingo honetan bigramaren tartean edo bukaeran zein hitz sartu daitezkeen bila. Hala ere, lehen ez bezala, hitza aurkitu ezkero beste funtzio bati deitu beharrean berorrek gehitzen du iradokizunak gordetzen dituen datu egiturara.

Behin guztia aztertuta iradokizunak erabiltzaileari erakusteko *sortuIradokizunKutxa* funtzioari deitzen dio. Honek iradokizunak erakusten dituen kutxa agertarazi eta bertan iradokizunak jartzen ditu.

6.2 APIarekin komunikatzen

Eskaera bat egiteko *REST* arkitekturan oinarriturik dagoenez, lehenik eta behin URL bat sortu beharra dago, behar diren parametroen artean guk nahi dugun kontsulta sartuz. Horretarako dago *sortuUrl* funtzioa.

URL-a sortu bezain laster *bidali* funtzioari dei egiten zaio. Hau *API*ari eskaera egiteaz arduratzen da *jQuery-ren ajax* funtzioa erabiliz. HTTP-ren *GET* eskaera bat egiten da aurretik sortu dugun URL-ra eta erantzuna jasotzean *zuzendu()* funtzioa martxan jartzen da okerra antzeman ahal izateko. Eskaeran errore bat egon ezkero *erroreTratamendua* izeneko funtzioa abiarazten da. Bidali funtzioak URL-arekin lan egin badezakeen arren beste hainbat parametro jasotzen ditu. URL-a ezik beste guztiak *zuzendu*ri parametro bezala pasatzeko erabiltzen dira.

Behin *API*aren erantzuna lortuta *JSON* formatuan, elementu bakoitza atzitu beharra dago. *Zuzendu* eta *lortuTestuingurua* funtzioetan egiten da hori. Demagun erantzuna parametro bezala eman digutela, *erantzuna* izenarekin. Zenbat emaitza aurkitu dituen jakiteko *erantzuna.SearchResponse.Web.Total* parametroa atzitu beharra dago. Bestalde, emaitza multzoa lortu ahal izateko *erantzuna.SearchResponse.Web.Results* egitea nahikoa da. Dena den, emaitza guztiek goiburuko berdinak ematen badituzte ere (*version* eta *query,* hau da, erabiltzen ari den APIaren bertsio zenbakia eta kontsulta zein izan den) *aztertutako teknologia* ataleko *Bing API* zatian agertu bezala SourceType bakoitzak emaitzean eremu desberdinak ematen ditu.

Hauxe da ortografia zuzentzaileak emandako *JSON* erantzunaren adibide bat:

```
if(typeof jQuery17109099192312561698_1341957688366 == 'function')
jQuery17109099192312561698_1341957688366(
{"SearchResponse":{ "Version":"2.0",
                    "Query":{"SearchTerms":"pinrate"},
                    "Spell":{"Total":1,
                             "Results":[{"Value":"pirate"}]
                    }
}
} /* pageview_candidate */);
```

erantzuna.SearchResponse.Query-k guk egindako kontsulta zein den esaten du: pinrate

erantzuna.SearchResponse.Spell.Total-ek zenbat emaitza ematen dituen. *Spell*-ek beti erantzun bakarra ematen du.

Erantzuna.SearchResponse.Spell.Results[0].Value-k emaitza sortatik 0.posizioan dagoena atzi dezakegu, hau da, *pirate*. Argi ikusten den bezala emaitzak (*Results*) array bat da eta horregatik erakusle batekin atzitu beharra dago bertako elementu guztiak, *Javascripteko* beste edozein *array*tan bezala.

6.3 Okerrak detektatzen

```
Nagusia(){

aztertuTestua();
for(esaldiGuztietarako){
	for(hitzGuztietarako){
		ortografiaAztertu//Url-a sortu eta bidali funtzioak egikaritu behar dira
				//horretarako
	}
}

for(esaldiGuztietarako){
	for(hitzGuztietarako){
		sortuTrigrama;
		bidaliBingAPIra//Url-a sortu eta bidali funtzioak erabili behar dira
	}
}

}
```

Bidalketaren ondorioz emaitzak jasotzerakoan *zuzendu* funtzioa egikarituko da. Idatzi ahalako iradokizunak emateko idatzita dagoen sasi kodean dago *zuzendu* funtzioaren kode-azalpena ere.

6.4 Idatzi ahalako iradokizunak ematen

```
Iradoki(){

	kurtsorePosizioa();
	if(bukaeranDago){
		sortuUrl(lortuAurrekoBiHitzak);
		bidali();
	}else{
		sortuUrl(lortuAurrekoHitza+lortuHurrengoHitza);
		bidali();

	}
}

zuzendu(){

if(webBilaketaEginDa){
	if(trigramaDa){
		trigrama;
		trigramaZati1;
		trigramaZati2;
		trigramaZati3;
		for(emaitzaMultzokoEmaitzaBakoitzean){
			if(trigramaAurkituDa){
				triKop++;
			}else{
				if(trigramaZati1AurkituDa){
					tri1Kop++;
				}else{
					if(trigramaZati2AurkituDa){
						tri2Kop++;
					}else{
						if(trigramaZati3AurkituDa){
							tri3Kop++;
						}
```

```
                                        }
                                }
                        }
                }
                zeinHandiena?(trigrama, trigramaZati1, trigramaZati2, trigramaZati3);
                if(berezkoTrigramariAldeHandia?){
                        lortuTestuingurua();
                        sortuErroreMezua();
                }
        }
        if(bigramaDa){
                for(emaitzaMultzokoEmaitzaBakoitzean){
                        if(bigramaOsatzenDuenHitzik?){
                                iradokizunaGehituZerrendara
                        }
                }
                sortuIradokizunKutxa(iradokizunZerrenda);
        }
}
if(spellErabiliDa){
        if(emaitzakZuzenketaEman){
                sortuErroreMezua();
        }
}
}
```

Hauek dira aplikazioa osatzen duten funtzio nagusiak. Beste hainbeste badaude ere, funtzionamendua erraz ulertu ahal izateko funtzio garrantzitsuenak sasikodez azaltzea erabaki da, xehetasun handietan sartu gabe.

6.5 Probak

Hasieran testu txiki eta sinpleekin lan egiten zen eta arazo handiegirik ez ziren antzematen. Testu tamaina handiagoekin hasterakoan ordea eraginkortasun galera itzela zegoela konturatu nintzen. Arazoa konponduta, hortik aurrera testu luzeagoekin lan egitea erabaki zen. *Wikipedia* eta *The Times*-eko artikuluak erabili dira orokorrean erroreak antzemateko. Proba bakoitza hasieran kutxa beltzeko proba moduan funtzionatzen duen arren, behin eta berriz testu berdinaren gainean probak egiten dira, kodea aldatuz eta testuan aldaketak sartuz. Hainbat proba egin dira baina beti prozedura bera jarraituz egin da.

- Testu ertain-luze bat hartu eta zuzenketa egin.
- Ikusi erroreak non dauden eta kodean aldaketak egin.
- Aldaketak ez badute funtzionatzen errorea konpondu arte testua ahalik eta gutxien aldatu.
- Erroreak zuzendu badira testuan aldaketak egin, kodeak non huts egin dezakeen kontuan hartuta.
- Testua aldatu akats berriak aurkitzeko.

Proba batzuk ere egin dira momentuan asmatutako esaldi txikiekin ere.

Proba hauek martxan jarri ondoren oraindik arazotxoak daudela esan beharra dago. Findu beharreko alderdiak daude, hala nola *API*ak emandako emaitzen tratamendu hobea egin beharra, bilatu nahi izaten den hitz edo esaldia ezin izaten baita aurkitu tartean puntuazio ikurren bat hitzen bati itsatsia badago. Gainera kontrolatu ezineko egoerak ere eman daitezke. *API*aren emaitzak ezin izaten dira kontrolatu eta Webean oker zein zuzen idatzitako testuak aurki daitezke. Askotan bilaketen emaitzen ordena garrantzitsua izaten da tratamendua egiterako orduan. Ezin dira *API*ak emandako emaitza guztiak aztertu, aplikazioa asko motelduko litzateke eta. Horregatik lehen 50 emaitzak soilik aztertzen dira. Hori dela eta oso garrantzitsua da emaitzak orden egoki batean etortzea. Bilatzaileetan webgune bat lehen postuan agertzea askotan interes ekonomikoetan oinarritzen da. Emandako emaitzetan %100-ean oinarritzea ez da ona izaten eta aplikazio honi horixe bera pasatzen zaio. Batzuetan interesatzen ez diren emaitzak lehen postuetan agertzen direnez okerren antzematea ez da behar bezala egiten.

Gainera zuzentzaile ortografikoak ez du behar besteko zehaztasunarekin lan egiten eta arazo honek batzuetan testu zati batzuk okertzat hartzen ditu.

7. ONDORIOAK

Proiektu guztia *Javascript*-en oso oinarrizko nozioak edukiz eta j*Query*-rekin inolako esperientziarik gabe egin da. Garapena aurrera joan ahala, ordea, trebezia handiagoz erabiltzen ikasi dut. Gainera titulazio osoan zehar pare bat proiektu handi soilik egin behar izateak, eta ondorioz esperientzia gutxi edukitzeak, hainbat programazio erronkei bakarrik aurre egin behar izatea eragin du. Hau dela eta proiektua aurrera ateratzeko, hainbat arazori aurre egin behar izan zaio lengoaia honen (*jQuery Javascript*-en oinarritzen denez lengoaia bat bezala har daiteke, nahiz eta erabiltzeko eta lan egiteko modua desberdina izan) ezjakintasuna dela eta. Hala ere arazoei bakarrik aurre egiteak, irtenbide batzuk Interneten bilatu eta besteak behin eta berriz aldaketak egiten aurkitu baitira, esperientzia ederra eman dute, sakonago ulertzen lagundu du programazio lengoaia eta bere *framework*aren nondik norakoak eta elkarren arteko desberdintasunak argi ikusiaz. Irtenbideak ahal diren lekutik atera behar izan dira eta Interneten ni bezala zeuden hainbat pertsonen arazoei esker irtenbideak aurkitu ahal izan ditut.

Aplikazioaren bukaera ordea laster etorriko da. Proiektuaren garapenaren azken zatian *Microsoft*-ek bere negozio eredua aldatu behar zuela ohartarazi zuen *Bing API*a zela eta. Zerbitzua hobetu, erabilera kuota zorrotza jarri eta segurtasuna hobetzea zen helburu nagusiena. Hau dela eta, aplikazioa garatzen jarrai daitekeen arren ez du pena merezi ordaindu behar den zerbitzu bat eskuratzea. Gainera barne aldaketak egin beharko lirateke kodean, segurtasun altuagoa lortu asmoz autentifikazio tokenen erabilera derrigorrezkoa da, eta orain ez bezala, komunikazio bide desberdin bat edukiko du *API*ak.

Aplikazio hau garatzeak ikasketa esperientzia ezin hobea ekarri dit. Dena den ez da zentzuzkoa hurrengo batean proiektu honen jarraipena egitea kode hau aprobetxatuz. Web bilaketak eginez testu bateko okerrak bilatzea puntu bateraino onargarria bada ere, zehaztasuna falta izaten da askotan. Web zerbitzuaren oso menpeko bihurtzen da aplikazioa. Zerbitzuaren kalitatea erdipurdikoa izan ezkero aplikazioak askorik ezin du egin okerren antzemate eta iradokizunak erakusteko momentuan. Hori dela eta ikasketa eta proiektu baten garapenean trebatzeko egokia bada ere, zehaztasun maila on bat lortzeko ez da nahikoa Web bilaketa eta *API*aren ortografia zuzentzailean oinarritzea proiektu osoa.

Bukatzeko, hasieran lortu nahi izan diren helburuak bete direla esan daiteke. *API* bidez eskainitako web zerbitzu baten oinarrituz *mashup* aplikazio bat egitea zen, web teknologiak erabiliz. *Bing API*an oinarrituta, *REST* arkitekturaren bidez *JSON* formatuan eta *AJAX* bidez eskuratutako emaitzen manipulazioa lortu da, web bilaketetan oinarrituta ingelesezko testuetan okerren detekzioa eta iradokizunak emanez. Proiektuak bere helburua lortzeko aldez aurretik dokumentu honetan azaldu diren teknologietan dokumentatu eta ikasi behar izan da, bibliografian aurki daitezkeen webguneetako informazioa barneratuz, adibideak probatuz eta foro espezializatuetan informazioa bilatuz.

7.1 Proiektuaren mugak

Hasieran ez bazen aintzakotzat hartzen ere, garapena aurrera joan ahala mugak eta ezintasunak aurkitu izan dira. Aplikazioak ezin ditu edozein motako akatsak aurkitu eta edozein hitzen iradokizunak eman ere. *API*ak eskaintzen duen ortografia zuzentzailea ez da nahi besteko zehatza eta askotan alde batera uzten ditu pare bat hitz ordenaz aldatuta dituen hitz bat, nahiz eta ikusmen

soilaz oker dagoela jakin. Hau dela eta gramatika akatsa etortzen dira askotan. Hainbeste akats dituzten hitzak zuzentzerako orduan ortografia zerbitzuak ez du esaten oker dagoenik eta ezin izaten da jakin zuzen dagoen edo oker dagoen, Web bilaketa batean emaitzak ikusi eta zorte pixka bat ez bada izaten.

Trigramak erabiliz okerrak detektatzea ideia ona den arren, aurretik antzeman ez diren erroreak tartean egon ezkero hainbat trigrama oker moduan agertzen dira askotan. Emaitzetan agertzen diren kointzidentzia kopurua kontuan hartuta trigrama "okerrak" berez zuzen daudela batzuetan antzematen diren aurren, beste askotan ezin izaten da ezer egin eta oker modura agertzen dira.

Beste arazoetako bat esaldiari zentzurik ezin hartzea da. Esaldiarekin zer esan nahi den ezin denez antzeman iradokizunak ez dute zentzu handiegirik izaten testuarekin alderatzen badugu.

Arrazoi hauek direla medio, proiektuak oso kontuan hartzeko mugak dituela esan daiteke. *API*aren emaitzen kalitatean oinarritu behar izateak askotan aplikazioa "salduta" uzten du, ezer egin ezinik. Gainera, Web-ean egindako bilaketek ez dute bermatzen ongi idatzitako edukia aurkituko dutela. Hainbat eta hainbat errore aurki daitezke eta ontzat hartzen direnez okerreko datuekin jokatzea ekartzen du. Oker idatzitako esaldi batek Web bilaketa batean hainbat kointzidentzia eman ditzake eta ondorio aplikazioak zuzen dagoela esan, nahiz eta modan dagoen musika disko baten izenburu soil bat izan bilatzen ari dena, adibidez.

7.2 Ikasi denari gainbegirada

Aplikazio handi baten garapena kudeatzeak esperientzia ederra ematen du. Dokumentatu, diseinatu, arazoei irtenbideak eman eta probak egin... ez dira askotan egiten horrelakorik titulazio osoan.

HTML eta CSS behar bezala erabiltzen ikasi dut interfaze atsegin eta erabilerrazak egiteko.

Javascript eta *jQuery* ongi barneratu ezkero zenbait trikimailu erabiliz *AJAX* aplikazio ahaltsuak erraz egin daitezkeela konturatu naiz. Animazio ikusgarriak ere egin daitezke dinamikoki sortutako elementuekin eta klik moduko gertaeren kudeaketa azkarren bidez, HTML elementuak nahi bezala kontrolaraziz. Gainera, lengoaia bera ongi erabiltzen jakitea ezinbestekoa da eraginkortasuna eta kodearen sinpletasuna landu nahi bada.

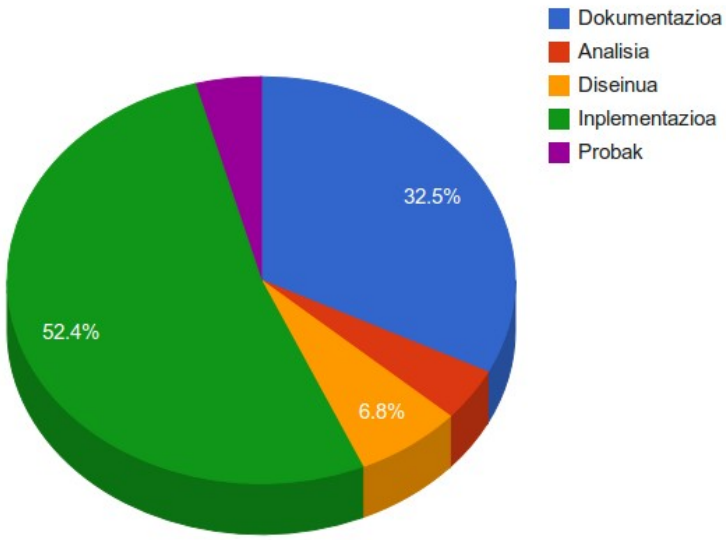
Bilaketak zehatzak egiteko espresio erregularrak erabiltzea ezinbestekoa dela jakin dut. Askotan klasean ikusi arren inoiz ezin da konturatu aplikazio praktiko bat egin arte zenbateraino erraztu dezaketen lana. Behar bezala ikastea ezinbestekoa izan da proiektu honetan.

Gainera, programazio arazoekin ikasiaz aparte, *JSON* eta *REST* moduko teknologiak erabiltzeak *API*en funtzionamendua eta datuen komunikazioaz jakintza berriak eman dizkidate. Proiektua hasi aurretik zer ziren jakin ez arren orain ongi uler dezaket zergatik hain ezagunak eta erabiliak diren gaur egun zerbitzuak martxan jartzeko momentuan.

Aplikazioa eraginkorra izateko teknika desberdinak jakitea beharrezkoa izan da, izan ere praktikan jarri behar izan dut aplikazioak hobetzeko behar handia baitzeukan.

7.3 Denboraren kudeaketa

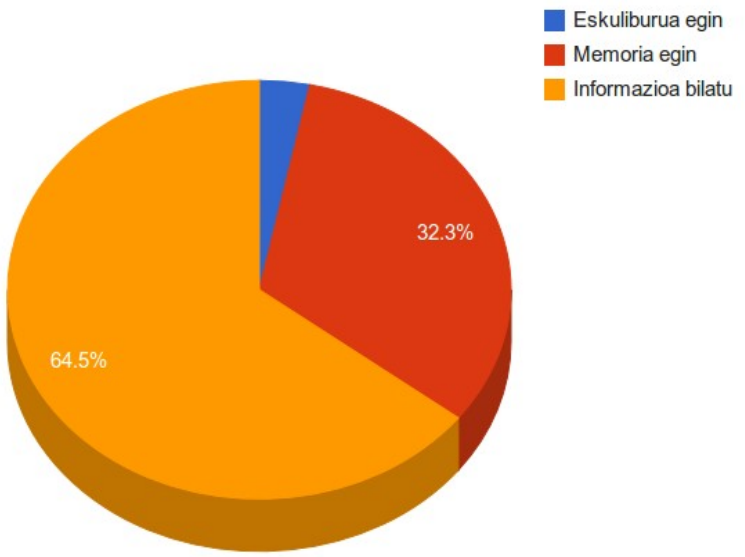
Hona hemen garapenean igarotako denboraren xehetasunak.



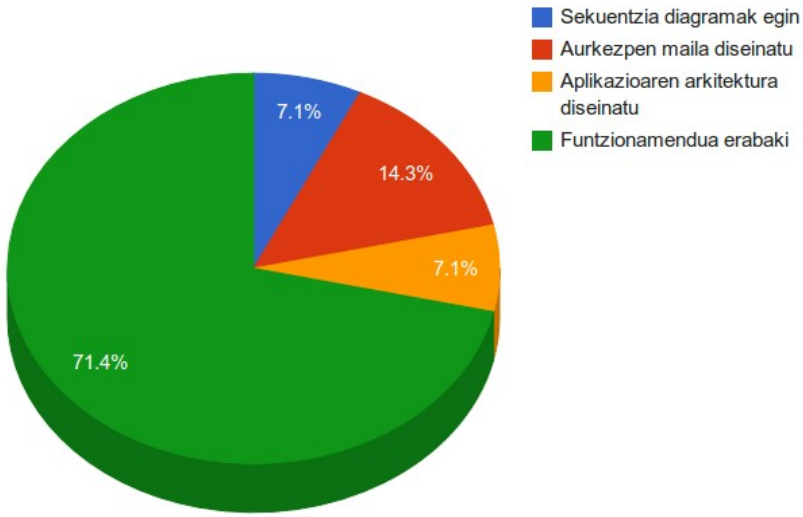
8. irudia: Sartutako orduen banaketa

- Dokumentazioa: 62 ordu.
  - Eskuliburua egin: 2 ordu.
  - Memoria osatu: 20 ordu.
  - Informazioa bilatu: 40 ordu.
- Analisia: 8 ordu.
- Diseinua: 13 ordu.
  - Sekuentzia diagramak egin: Ordu 1.
  - Aurkezpen maila diseinatu: 2 ordu.
  - Aplikazioaren arkitektura diseinatu: Ordu 1.
  - Funtzionamendua erabaki: 10 ordu.
- Inplementazioa: 100 ordu.
- Probak: 8 ordu.

GUZTIRA IGAROTAKO ORDUAK: 191 ordu.



9. irudia: Dokumentazioan igarotako orduen banaketa



10. irudia: Diseinatzen igarotako denboraren banaketa

8. BIBLIOGRAFIA

[1] http://blogs.msdn.com/b/eslassistant/
[2] http://spellcheckplus.com/Faq/
[3] http://zientzia.net/artikuluak/xuxen-euskararako-zuzentzaile-ortografikoa/
[4] http://en.wikipedia.org/wiki/REST
[5] https://www.ibm.com/developerworks/webservices/library/ws-restful/
[6] http://www.computerworld.com/s/article/297424/Representational_State_Transfer_REST_
[7] http://en.wikipedia.org/wiki/JSON
[8] http://www.json.org
[9] http://docs.jquery.com/Main_Page
[10] http://www.w3schools.com/jquery/default.asp
[11] http://en.wikipedia.org/wiki/Jquery

- Programazio arazoak izan direnean irtenbidea aurkitzeko ondorengo web orriak erabili dira:

http://www.w3schools.com/
http://api.jquery.com/
http://www.forosdelweb.com/
http://stackoverflow.com/

- Interfazea osatzerakoan koloreak eta irudiak hartzeko hona jo da:

http://openiconlibrary.sourceforge.net
Laguntza oharra erakusten duen irudia. Ikono libreak eskaintzen ditu.

http://www.flamingtext.com/
"Ingeles zuzentzailea" izenburua egiteko erabili den *online* tresna.
http://subtlepatterns.com/
Atzeko planoko irudiak eskaintzen dituen web orria.

[12] http://neteye.github.com/activity-indicator.html
Zuzenketa egiterakoan lanean ari dela erakusten duen ikonoa. *jQuery* plugin bat da.

[13] https://github.com/tdreyno/iphone-style-checkboxes
Idatzi ahalako iradokizunak kendu eta ipintzeko *checkbox*ari itxura berri bat emateko. *jQuery* plugin bat da.

9. ERANSKINA

9.1 Eskuliburua

11. irudia: Aplikazioa martxan

Aplikazioa martxan jartzeko oso sinplea da. Testu kutxan idatzi behar dena jartzen da. Nahi denean "Zuzendu" botoia jo eta okerrak antzematen hasten da. Testu kutxaren azpian errore panela ezkutatu eta erakusteko barra dago. Hasieran ezkutatuta badago ere barra honetan klik egin ezkero albo batean agertzen da hau.

Okerren bat detektatu ezkero errore panelean agertzen da. Bi errore mota bereizten dira: Ortografia akatsak, kutxa gorrian egoten dira, eta gramatika akatsak, kutxa berdean. Ortografia akats bat izan ezkero bertan adierazten da zein hitz idatzi behar izango litzatekeen. Kutxan klik egin ezkero testu kutxan hitza bera aukeratzen da, erabiltzaileak testuingurua irakurri eta egin beharreko aldaketak erraz egiteko.

Gramatika akatsak ere berdin funtzionatzen dute. Kutxan klik egin ezkero testuan esaldi zatia aukeratzen da. Hasieran iradokizunak agertzen ez diren arren hauek ikusi nahi izan ezkero "oker idatzita egon daiteke" testuan klik eginda kutxa handitu eta azpialdean iradokizun posible batzuk ematen ditu.

Zuzenketak egiterako orduan garrantzitsua da kontuan hartzea zenbait kontu.

- Testua puntuazio ikur batez bukatua egon behar da. Hala ez bada azken hitza ez da aztertuko, ez baitakigu idazten bukatuta dagoen ala ez.
- Ortografia akatsak badaude komenigarria da hauek zuzentzea lehenbizi. Gauzak horrela zenbait gramatika oker ere momentuan zuzendu daitezke, oker idatzitako hitzaren albokoa da eta.

Idatzi ahala iradokizunak ere ematen ditu automatikoki aplikazioak. Momentuan idazten dagoena kontuan hartuz ondoren zer idatzi dezakeen iradokitzen du. Hau gutxi balitz, kurtsorea hitz baten bukaeran jarri ezkero hitz horren eta hurrengoaren artean zein hitz sartu daitekeen esaten du. Dena den, iradokizun hauek edozein momentutan ezkutatu daitezke eskuinaldean dagoen botoiaren bidez. Hasieran idatzi ahalako iradokizunak “agerian” balioa badu ere bertan klik eginez gero “ezkutuan” jartzen da, eta alderantziz.

Goi eskuinaldean dagoen galdera ikurraren irudiaren gainetik arratoia igaro ezkero laguntza ohar bat agertzen da. Ortografia akatsak lehenbizi zuzentzea gomendatzen duen oharra erakusten du.

Azkenik, konexiorik gabe lanean aritu ezkero ohartarazpen leiho bat erakusten da, aplikazioak ezin duela konexiorik gabe lan egin esanez. Momentu oro Interneterako konexioa dagoen ala ez erakusten duen ohartarazpen lekua ikus daiteke zuzenketa botoiaren alboan.

9.2 Aplikazioa

Aplikazioa lortzeko jo helbide honetara:

https://www.dropbox.com/s/6cu8i6rqrqjnmsy/Ingeles%20zuzentzailea-Kodea-.tar.gz

Fitxategi guztiak erauzi eta direktorio berean gorde. *Behin betikoa* izeneko direktorioaren barnean daudenez dagoeneko hau erauztea nahikoa da.

Martxan jartzeko *ingelesZuzentzailea.html* fitxategia arakatzaile batekin ireki. Gogoratu 1280x800 pixeleko bereizmena duen pantailan egin dela garapena, *Firefox*13.0.1, *Opera* 12.00, *Chromium* 18.0.1025.168 eta *Internet Explorer* 9 arakatzaileenpean.