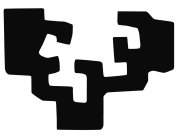
eman ta zabal zazu



Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Ingeniaritzako Gradua

Software Ingeniaritza

Gradu Amaierako Proiektua

Magnet loturak: bezero-zerbitzari API baten garapena

Egilea

Alain Mendizabal Agirre

2015

Laburpena

Dokumentu honek magnet lotura eta web API arloak lantzen dituen Gradu Amaierako Proiektua deskribatzen du. Teknologia horiek liburu elektronikoak partekatzeko erabili dira eta sortutako aplikazioari Liburutruk izena eman zaio.

Honakoak dira Liburutruk-en ezaugarri nagusienak:

- Zerbitzariak web API api publiko eta dokumentatua eskaintzen du.
- APIaren bidez liburuak eta horiei lotutako baliabideak kudeatu daitezke.
- Gehitutako liburuak BitTorrent bidez partekatzen ditu eta liburu horien magnet loturak sortzen ditu.
- Web-nabigatzaileetan exekutatzen den bezeroa du, orri bakarreko aplikazioa dena.
- Bezeroak zerbitzariaren APIa erabiltzen du.
- Aplikazioa eta kodea edonork eskura ditzake.

Hitz gakoak: Liburutruk, web API, bezero-zerbitzari, BitTorrent, magnet lotura, liburu elektroniko.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Sarrera

Gaur egun, Internet arlo guztietara zabaltzearekin batera, web bidezko APIen gorakada ikusten ari gara. API horiek programatu daitezkeen interfazeak dira, eta HTTP bidez zerbitzatu ohi dira. Lehen genituen web-orri edo aplikazio monolitikoen ordez, orain hainbat web-zerbitzuetatik elikatzen diren guneak ditugu. Horrela, API ezberdinak konbinatuz mashup[1] gisako guneak sor ditzakegu, besteak beste.

BitTorrent sarea edukiak partekatzeko erabiltzen da, eta Interneteko trafikoaren zati handi bat hartzen du. Horrelako sareetan, badu garrantzia partekatutako baliabideak nola identifikatzen diren. Hain zuzen ere, azken urteotan, igo egin da baliabideak URI baten bidez identifikatzen dituzten "magnet"izeneko loturen erabilera.

BitTorrent fitxategi handiak transmititzeko erabili ohi den arren, liburu elektronikoak partekatzeko ere egokia izan daiteke, magnet loturak erabiliz.

Proiektu honetan, bi arlo horiek bilduko dira: web APIak eta magnet loturak. Teknologia horiek liburu elektronikoak partekatzeko erabiliko dira.

[1] Hainbat iturritako edukia konbinatzen duen web-orri edo aplikazioa.

1.1 Web APIak

Web-zerbitzu deritzona sarearen bidez 2 makina komunikatzeko sistema bat da[2], eta WSDL[3], UDDI[4] eta SOAP[5] teknologiak erabiltzen ditu. Web 2.0rekin batera zerbitzuen bilakaera gertatu da, eta REST[6] moduko APIak erabiltzeari ekin zaio. API horietan, lehen aipatutako teknologien ordez, zuzenean HTTP metodoen bidez (GET, POST, PUT, DELETE...) baliabideak atzitzen dira. Transmisiorako formatua, berriz, XML edo JSON izaten da (azken horrek indar gehiago hartu du). Esan daiteke sakonki espezifikatutako zerbitzu astunetatik zerbitzu arin eta sinpleetarako pausoa eman dela, hau da, mutur batetik besterako jauzia egin dela.

Horrela, web APIak aipatzen ditugunean, REST motako zerbitzuei buruz ariko gara.

1.2 Magnet loturak

1999an Napster[7] zerbitzua abian jarri zenean hasi ziren zabaltzen P2P (*peer-to-peer*) gisa ezagutzen ditugun sareak. P2P sareetan, bezero-zerbitzari ereduan ez bezala, komunikazioa maila berean dauden kideen artean gauzatzen da. Horregatik, edukia lortu nahi duten erabiltzaileen kopurua igo ahala, igo egiten da sarearen ahalmena. Bestalde, batzuetan nolabaiteko zentralizazio-maila bat izan arren (Napster zentralizatua zen), sare deszentralizatuak izaten dira. Ezaugarri horiek garrantzitsuak dira sare horien gorakada ulertzeko, eduki bat partekatzeko behar den azpiegituraren kostua txikia baita.

Dauden P2P sareetatik, BitTorrent da erabiliena. Protokoloak indexatzeko aukerarik ematen ez duenez, edukiak aurkitzeko webguneak egon ohi dira, The Pirate Bay[8]

[2] Definizio bat baino gehiago aurkitu daiteke; W3C erakundearena erabili da: `http://www.w3.org/TR/ws-arch/#whatis`

[3] Web Services Description Language: web zerbitzuaren funtzionalitatea deskribatzen du.

[4] Universal Description Discovery and Integration: web zerbitzuak erregistratzeko eta aurkitzeko erabiltzen da.

[5] Simple Object Access Protocol: makinen artean informazioa trukatzeko protokoloa.

[6] Representational State Transfer: baliabideak HTTP metodoen bidez atzitzen dira.

[7] Musika fitxategiak, MP3 formatukoak, partekatzeko erabiltzen zen. Diskoetxeekin arazo legalak eduki ondoren itxi egin zuten.

[8] Urteetan edukien indize handiena izan da. 2014ko abenduaren 9an Suediako poliziak webgunearen zerbitzariak konfiskatu zituen. Testu hau idatzitako unean, webgunea berriz martxan jarrita zegoen, 2015eko otsailaren 1erako kontagailu batekin.

ezaguna kasu. BitTorrent-en azken urteotako joera bat izan da .torrent fitxategiak[9] erabiltzetik magnet loturetara pasatzea.

Magnet loturak URN[10] motako URIak[11] dira. Edukiaren edo metadatuen hash balioaren bidez baliabideak identifikatzen dituzte. Hau da, ez dute kokapena zehazten, edukia baizik. ISBNren[12] antzera, baliabide bakoitzeko identifikatzaile bakarra egoten da.

Magnet loturak askotariko sareetan erabil daitezke (BitTorrent, eDonkey, Gnutella...) eta horren araberako formatua izaten dute. Loturen parametro nagusienetako bat `xt` ("exact copy") izenekoa da eta hash identifikatzailea edukitzen du; BitTorrenten kasuan BTIH (BitTorrent Info Hash) izenekoa erabiltzen da. Balio hori lortzeko, .torrent fitxategia hartzen da, metadatuak dituena, eta bertako info zatia hartzen da. Zati horri SHA-1 hash funtzioa aplikatuta lortzen da BTIHren balioa.

Beste atal bat, beharrezkoa ez dena baina proiektu honetan erabili dena, `tr` ("address tracker") da. Atal horretan tracker baten helbidea adierazten da. BitTorrent sarean, tracker bat fitxategiak non dauden dakien zerbitzari bat da. Fitxategi bat lortu nahi bada, trackerrak fitxategi hori nork duen jakiten laguntzen du, eta, behin jabeak aurkituta, ez da haren beharrik izaten. BitTorrent bezeroak DHT[13]-rentzako euskarria badu, trackerrik gabe ere funtziona dezakete. Hala ere, normalean azkartu egiten dute prozesua, fitxategia nork duen azkarrago jakiten baita trackerrari galdetuz gero.

1.3 Liburu elektronikoak

Liburu elektronikoak -e-book edo e-liburuak ere deitzen zaie- formatu digitala duten liburuak dira. E-book irakurgailuen bidez irakurtzeko prestatuak daude, baina gailu gehiagotan ere erabil daitezke: ordenagailuak, tabletak, telefono adimendunak (smartphoneak)...

Hasiera batean gailuek hainbat formatutako liburuak onartzen zituztenez, formatu estandar bat garatzen hasi ziren. Hala, 2007an, EPUB estandar libre eta irekia argitaratu zen, Open eBook-en ondorengoa hain zuzen. Estandar hori darabilten

[9] Baliabideak BitTorrent bidez partekatzeko metadatuak gordetzen dituzte.
[10] Uniform Resource Name
[11] Uniform Resource Identifier
[12] International Standard Book Number
[13] Distributed Hash Table (Hash Taula Banatuak): sistema banatu deszentralizatua, baliabideak aurkitzeko hash taula moduko zerbitzua eskaintzen duena.

liburuek `.epub` luzapena dute, eta, oraindik gailu guztietan erabiltzea ezinezkoa den arren, liburu bat modu digitalean banatzeko formatu egokia da.

1.4 Motibazioa

Webgune asko daude torrent fitxategien indize gisa funtzionatzen dutenak. Baina normalean ez dute APIrik eskaintzen, eta orria zuzenean bisitatu behar izaten da. Modu berean, fitxategi batetik torrent fitxategia sortzeko onlineko zerbitzuak ere badaude, baita torrent-etatik magnet loturak sortzekoak ere. Halere, fitxategi bat emanda torrent-a (eta magnet lotura) sortu eta indize gisa jokatzen duen webgunerik ez da aurkitu. Proiektu honekin hutsune hori bete nahi izan da.

P2P sareen ezaugarrietako bat da fitxategiak toki zentralizatu batean ez edukitzea. Fitxategiak erabiltzaileen artean partekatzen direnez, eduki jakin baten eskari handia dagoenean, erraztu egiten da banaketa. Edukia zerbitzari batean zentralizatuta egonez gero, berriz, bezero asko konektatzeak zaildu egiten du banaketa.

P2P sareetan geroz eta gehiago erabiltzen diren magnet loturek oso informazio gutxirekin (testu kate bat) baliabideak identifikatzeko balio dute. Horri esker, oso arinak dira gordetzeko edo sarean zehar garraiatzeko. Bestalde, web-ingurunean erabili ohi diren HTTP URL-ek kokapena adierazten duten moduan, magnet loturek edukia hartzen dute kontuan. Beraz, edukia sarean dagoen bitartean, loturak funtzionalak dira, eta ez dira hain erraz iraungitzen. URL jakin batean adierazitako fitxategia zuen zerbitzaria desagertzen denean, berriz, URL horrek ez du fitxategia eskuratzeko balioko. Magnet loturak DHT-ekin erabilita, gainera, trackerrekiko menpekotasuna kentzen dute, eta sarea deszentralizatu egiten da.

Web APIak interesgarriak iruditzen zaizkit, informazioa modu errazean tratatzeko aukera ematen baitute. Webguneetako edukia nabigatzaileentzako prestatuta egon ohi da, eta zaila da hori beste testuinguru batzuetan berrerabiltzea; erabilera finkatuta geratzen da nolabait. API batekin, berriz, modu programatuagoan lan egin daiteke. Garatzaile baten ikuspegitik, APIak aukera gehiago eskaintzen ditu. Zerbitzu baten APIa eskura jartzeak, beraz, garatzaileak hurbiltzen ditu eta zerbitzu horren inguruko ekosistema hobetzen du. Azken urteotan, web-zerbitzuek gora egin dute, eta APIak eskaini ohi dituzte[14].

[14] Steve Yegge-k Amazonen bilakaerari buruz idatzitakoa interesgarria da: `https://plus.google.com/+RipRowan/posts/eVeouesvaVX`

Web APIek bi rol biltzen dituzte: garatzailearen eta erabiltzailearen rolak.

Proiektu honetan bi rolak jokatu nahi izan dira: batetik, magnet loturak erabiltzeko aukera ematen duen web API bat sortu, eta, bestetik, API hori erabiliz, liburuak trukatzeko aplikazio bat garatu nahi izan da.

Gaur egun, asko zabaldu da JavaScript-en erabilera eta, horrekin batera, geroz eta orri bakarreko aplikazio[15] gehiago daude. Joera horri jarraituz, APIa kontsumitzen duen orri bakarreko aplikazioa eraiki nahi da.

1.5 Helburuak

Proiektu honek "motibazioa"atalean aipatutako hutsunea bete nahi du. Horrekin batera, aurreko ikasturtean gauzatutako proiektu bat[16] sakondu nahi izan da.

Proiektuaren helburu nagusienetako bat API bat diseinatzen ikastea izan da. Behin APIa diseinatuta, API hori inplementatzen duen zerbitzaria garatzea eduki da helburutzat.

Bestalde, orri bakarreko aplikazioak nola eraikitzen diren ikasi nahi izan da. Horretarako, diseinatutako APIa erabiltzen duen bezeroa garatu da.

Azkenik, Gradu Amaierako Proiektu guztiekin bezala, proiektu informatiko baten kudeaketan eta garapenean esperientzia hartzea helburu garrantzitsua izan da. Horrekin batera, memoriaren idazketan ere trebatu nahi izan da.

1.6 Aspektu teknikoa

Proiektu honetan, magnet loturak erabili nahi ziren, haien sorrera ahalik eta gehiena automatizatuz, eta API bat eraiki nahi zen. Bi aspektu horiek garbi izanik abiatu da proiektua. Bestalde, orri bakarreko aplikazio bat ere eraiki da, APIa erabiltzeko modu egokia delako.

[15] Orri bakarrean sartzen diren web-aplikazioak, ingelesez Single-page application (SPA).
[16] Magnet loturen inguruko proiektu bat. "Hasierako egoera"atalean azalduko da.

1.7 Aspektu funtzionala

Proiektuaren motibazio nagusia ez da izan liburu elektronikoak partekatzea. Lehenik teknikoki zer egin nahi den zehaztu da eta geroago pentsatu da liburuak ardatz hartzea. Domeinu jakin baten inguruan lan egingo zela argi zegoen, noski, baina ez liburuen inguruan.

Hainbat domeinuren artean zalantza izan da: bideoak, argazkiak eta liburuak. Bideoek (luzeek behintzat) tamaina handia edukitzen dute, eta egokiak izan ohi dira BitTorrent bidez partekatzeko. Baina torrent fitxategiak automatikoki sortu nahi direnez, ikusi da oso astuna dela tamaina horietako fitxategiak zerbitzariari pasatzea eta tratatzea. Argazkiak ere, taldeka hartuz gero, partekatzeko egokiak izan daitezke. Argazkiak, ordea, webguneetan zuzenean ikusi ohi dira, eta, nahi izanez gero, ikusitako argazkia bera jaisten da edota argazki hori kalitate hobean duen lotura bat klikatzen da. Domeinu hori aukeratuz gero, beraz, albumak edukiko genituzke, eta haietako bakoitzaren aurrebista. Pentsatu da hori ere ez dela oso praktikoa.

Horrenbestez, liburuak hautatu dira. Partekatzeak zentzua du, eta, fitxategi handiak izan ohi ez direnez, tratatzeko egokiak dira.

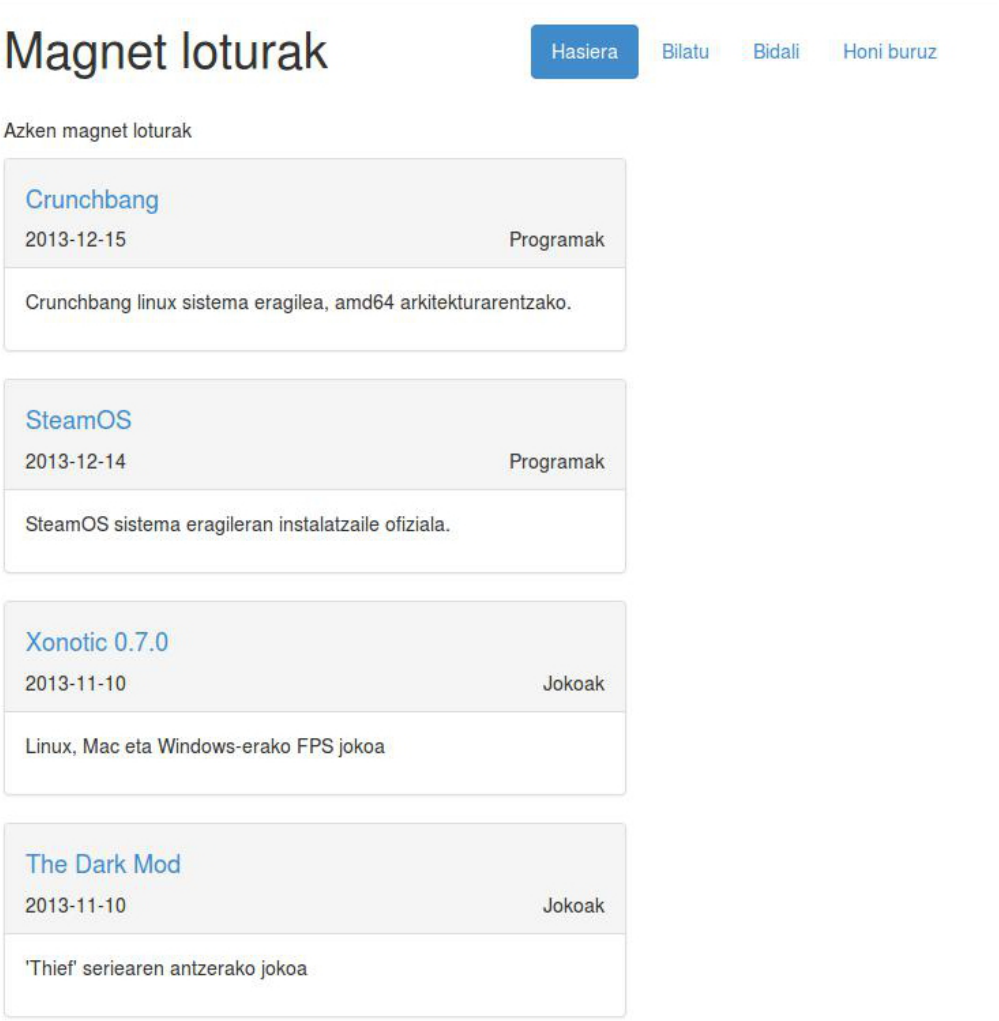
1.8 Hasierako egoera

2013/2014 ikasturtean, Web Sistemak ikasgaiko proiektu gisa, magnet loturak partekatzeko eta arakatzeko webgune bat eraiki zen. Orriaren atzean, web APIa eskaintzen zuen zerbitzari bat zegoen, eta JavaScript bidez zerbitzari horretako datuak erabiltzen ziren. 1.1 irudian ikus daiteke zer itxura zuen.

Funtzionalitate gutxi zuen aplikazioa zen, baina kontzeptu interesgarria zela iruditu zitzaion irakasleari[17]. Horrela, Gradu Amaierako Proiektuaren bidez hasierako ideia hura sakontzea eta garatzea erabaki zen.

Bestalde, lehen esan bezala, torrent fitxategien indizeak badaude. Fitxategi bat emanda torrent-a (eta magnet lotura) sortzeko zerbitzuak ere badaude. Web APIak ugariak dira, eta arlo ezberdinetakoak. Baina hiru ezaugarri horien konbinaketarik ez dago.

[17] Xabier Arregi, proiektuaren tutorea.



1.1 Irudia: Web Sistemak-en eraikitako webgunea.

2. KAPITULUA

Planifikazioa

2.1 Proiektuaren irismena

1. **Betekizunak**

   Lehen aipatu bezala, proiektu honetan web API bat garatu nahi magnet loturak kudeatzeko. Horrekin batera, web APIa funtzionalki erabili nahi da liburuak trukatzeko sistema bat garatzeko. Sistema hori bi ataletan banatuta dago: zerbitzaria eta bezeroa.

   Atal horiek honako betekizunak dituzte:

   Zerbitzaria:

   - Web API baten bidez ondorengo baliabideak eta beraien arteko erlazioak kudeatzeko aukera eskaintzea:
     - Erabiltzaileak.
     - Erabiltzaileen saioak.
     - Liburuak.
     - Iruzkinak.
     - Administratzaileak.
   - Liburuen fitxategietatik torrent fitxategiak eta magnet loturak sortzea.
   - Liburuak BitTorrent bidez partekatzea.

- Erabiltzaileen saioen kontrola burutzea.

Bezeroa:

- Web APIa erabiliz ondorengo aukerak eskaintzea:
  - Liburu berrietatik gogokoena ikusteko.
  - Azken iruzkinak ikusteko.
  - Liburuak ikusteko.
  - Liburu baten informazio guztia ikusteko.
  - Liburu bati lotutako iruzkinak ikusteko.
  - Erregistratzeko.
  - Erabiltzaile gisa saioa hasteko.
  - Liburuak gehitzeko.
  - Norberaren liburuak kudeatzeko.
  - Liburu bati lotutako iruzkinak idazteko.
  - Norberaren iruzkinak kudeatzeko.
  - Liburuak gogoko gisa markatzeko.
  - Norberaren gogokoak kudeatzeko.
- Aplikazioa orri bakarrean exekutatzea.

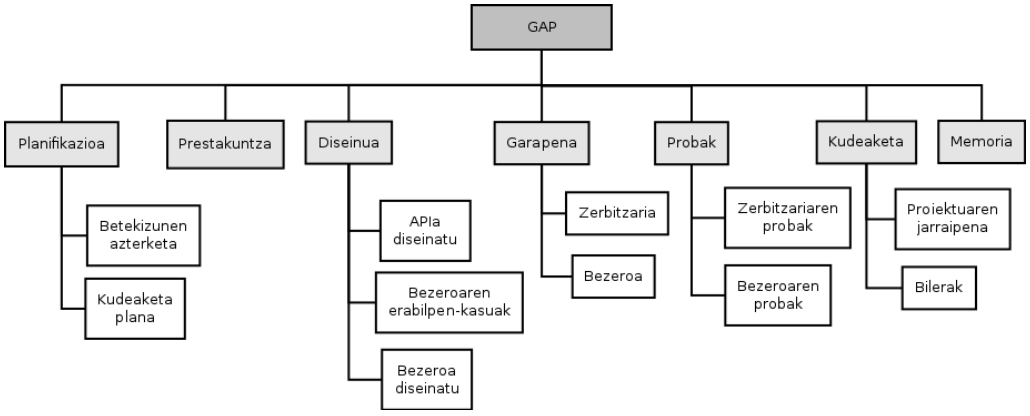
Erabilitako tresna guztiek software irekiaren filosofia jarraituko dute. Sortuko den softwarea ere irekitasun printzipioan oinarrituko da.

2. **LDE diagrama**

   2.1 irudian lanaren deskonposaketa modu grafikoan ikus daiteke. Honako elementuak ditu:

   - **GAP:** Gradu Amaierako Proiektua.
   - **Planifikazioa:** proiektua garatzeko plana zehazten da.
     - **Betekizunen azterketa:** proiektuak bete behar dituen ezaugarriak.
     - **Kudeaketa plana:** kudeaketa egokia izateko plana.
   - **Prestakuntza:** garapena ongi gauzatzeko gaitasunak lortzea.
   - **Diseinua:**
     - **APIa diseinatu:** web APIa diseinatu eta dokumentatu.

  - **Bezeroaren erabilpen-kasuak**
  - **Bezeroa diseinatu**
- **Garapena:**
  - **Zerbitzaria**
  - **Bezeroa**
- **Probak:**
  - **Zerbitzariaren probak:** zerbitzariak APIa ongi inplementatzen duela probatu.
  - **Bezeroaren probak:** bezeroaren funtzionamendua zuzena dela probatu.
- **Kudeaketa:** lanaren kudeaketa.
  - **Proiektuaren jarraipena**
  - **Bilerak**
- **Memoria**



2.1 Irudia: LDE diagrama

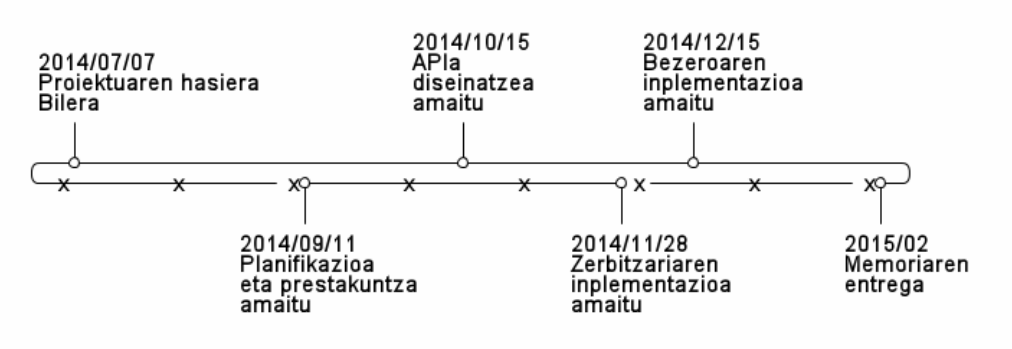
3. **Emangarriak**

   Proiektuan bi emangarri mota bereizi daitezke:

   - Produktua:
     - Zerbitzaria.
     - Bezeroa.
     - Instalazio eskuliburua (zerbitzariarentzat eta bezeroarentzat).
     - Erabilpen eskuliburua.

- Proiektuari lotutakoak:
  - Proiektua garatzeko plana.
  - Proiektuaren memoria.

4. **Mugarriak**



2.2 Irudia: Mugarrien diagrama.

2.2 irudian, diagrama baten bidez, proiektuaren mugarriak agertzen dira. Estimazioak izanik, datak ez dira zehazki bete.

2.2 Denboraren planifikazioa

1. **Atazak**

   Proiektuari 300 ordu eskaintzea espero da. Hori kontutan edukiz, ondorengo estimazioak egin dira:

   (a) **Planifikazioa: 15 ordu**

   - Betekizunen azterketa: 5 ordu
   - Kudeaketa plana garatu: 10 ordu

   (b) **Kudeaketa: 15 ordu**

   - Proiektuaren jarraipena: 5 ordu
   - Bilerak: 10 ordu

   (c) **Prestakuntza: 20 ordu**

   - Magnet loturak: 5 ordu
   - API garapena: 5 ordu

- Orri bakarreko aplikazioak (SPA): 10 ordu

(d) **Diseinua: 50 ordu**

- APIa diseinatu: 20 ordu
- Bezeroaren erabilpen-kasuak landu: 10
- Bezeroa diseinatu: 20 ordu

(e) **Garapena: 87 ordu**

- APIari eusten dion zerbitzaria inplementatu: 40 ordu
- Bezeroa inplementatu: 47 ordu

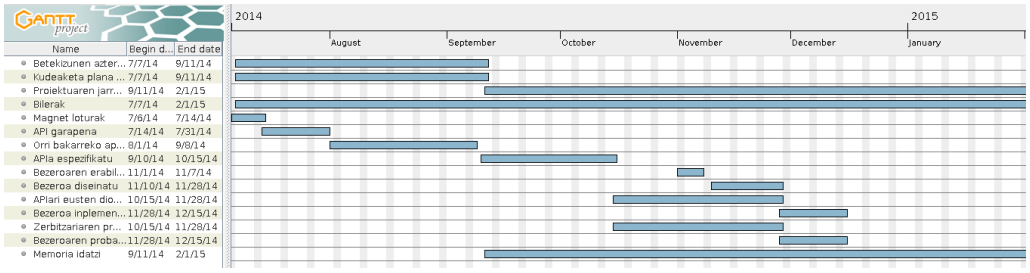
(f) **Probak: 13 ordu**

- Zerbitzariaren proba funtzionalak inplementatu: 7 ordu
- Bezeroaren probak inplementatu: 6 ordu

(g) **Memoria idatzi: 100 ordu**

2. **Atazen antolaketa**

2.3 irudiko Gantt diagraman aurreko ataleko ataza bakoitza noiz burutzen den adierazten da.



2.3 Irudia: Gantt diagrama.

2.3 Komunikazio-plana

Proiektu honetako interesatuak bi dira:

Ikaslea	Alain Mendizabal	amendizabal017@ikasle.ehu.es
Tutorea	Xabier Arregi	xabier.arregi@ehu.es

Komunikazioaren zati handi bat posta elektronikoz burutuko da. Bestalde, ikasleak noizean behin bilera bat burutzea eskatuko dio tutoreari eta, honen agendaren arabera, egun eta ordu bat finkatuko da. Bileretan ordura arte egindakoa aztertuko da, arazoak aurkitu eta konpontzen saiatu eta ondorengo helburuak ezarri. Ikaslea arduratuko da bileren laburpena idazteaz, eta uneoro egon dira laburpen horiek kontsultagai.

Kode guztia GitHub-eko biltegietan egongo da eta ikaslea kodea biltegira egunero igotzeaz arduratuko da. Horrela tutoreak kodea eskura edukiko du edozein momentutan.

2.4 Kalitatearen kudeaketa-plana

1. **Proiektuaren kudeaketa**

   Gutxienez bi astetik behin interesatuen arteko komunikazioa burutuko da, bilera batekin behar izanez gero. Horrela ikasleak proiektuaren egoeraren berri emango dio tutoreari eta proiektua bide onetik doan ikusiko da.

2. **Garatutako produktua**

   Produktuaren kalitateari dagokionez, zerbitzariari jarriko zaio arreta gehien. Hori horrela izateko bi arrazoi daude:

   - Zerbitzariaren kalitatea neurtzea errazagoa da.
   - Bezeroak zerbitzaria erabiltzen duenez, kalitate eskaseko zerbitzari batek bezeroaren funtzionamenduan eragina izango du.

   Zerbitzariaren funtzionamendua egokia dela baieztatzeko proba automatizatuak erabiliko dira. APIa definitzearekin batera zerbitzaria espezifikatzen denez, API hori erabiliko da gida gisa.

   Zerbitzariaren funtzionalitate bati dagokion kodea idazten hasi aurretik horri dagokion proba idatziko da. Probak huts egingo du, espero bezala, eta jarraian proba gainditzeko beharrezko kodea idatziko da. Behin probak ez duela huts egiten ikusitakoan, kodea berregituratu daiteke. Ziklo hori errepikatuko da funtzionalitate guztiekin, espezifikazio guztia inplementatu arte. Aipatutako prozesua TDD[1] gisakoa da. TDD normalean unitate probekin erabiltzen da, baina kasu honetan sistema osoa probatzen da.

[1] Test-driven development

Proba horiek automatizatuak direnez ez dute pertsona baten esku-hartzea eskatzen. Hori aprobetxatuz, Travis CI bidez integrazio jarraitua burutuko da. Tresna honek, GitHub-eko biltegian aldaketa bat dagoenean, kodea hartu eta probak exekutatzen ditu. Horrez gain, hainbat inguruneta exekutatzeko aukera dago (adb: JDK 7 edo JDK 8). Travis CI-n exekutatutako proben egoeraren berri ematen duen irudi bat ere gehitu da GitHub-eko orrian, uneoro informazio hori eskura edukitzeko.

Probak modu horretan antolatzeak hainbat abantaila ekartzen ditu:

- Funtzionalitate jakin batean zentratu daiteke, pixkanaka, dena batera hartu gabe. Lan txikiagoak egin behar dira, burutzeko errazagoak izanik.
- Zati berri bat modu azkarrean idatzi daiteke eta funtzionatzen duela ikustean txukundu.
- Kodea berrantolatzeko konfiantza ematen du, aldaketa batek zerbait hausten badu berehala ikus baitaiteke.
- Garatzeko makinan ez dauden inguruneetan probatu daiteke.

3. **Memoria**

   Tutoreari bere iritzia eskatuko zaio sortze prozesuan. Horrez gain, proiektuarekin loturarik ez duen baten iritzia ere jasoko da, batez ere ideia orokorrak ongi azaltzen diren jakiteko. Proiektuan murgilduta dagoen batek testuingurua ongi ezagutzen du eta informazio osoa jaso ez arren ulertzeko modua du. Hori ez da gertatzen lehen aldiz irakurtzen duen batekin.

   Memoriarentzat 80-100 orri inguruko tamaina hartu da erreferentzia gisa. Halere, edukiak orri gehiago edo gutxiago eskatzen baditu, ez da moztu edo luzatuko. Luzera behartzeak kalitatea jaitsi dezake, adibidez, edonolako edukia gehituz edo garrantzitsua den zerbait kenduz.

2.5 Arriskuen kudeaketa-plana

Jarraian, proiektua kolokan jarri dezaketen arriskuak aipatuko dira. Zer eragin eduki dezaketen eta nola ekidin daitezkeen ere azalduko da. Arrisku mailaren arabera ordenatu dira, eragin gehiena duena lehenengoa izanik.

1. **Magnet loturak automatikoki ezin sortzea**

    - **Azalpena:** Erabiliko den teknologiarekin agian ez da posible fitxategiak aukeratu eta automatikoki magnet loturak sortzea edo posible izan arren proiektu honetarako kostu handia du.
    - **Eragina:** Ertaina. Bezeroaren funtzionalitatea aldatzea eskatzen du baina ez du proiektua oztopatzen.
    - **Gertatzeko probabilitatea:** Handia.
    - **Ekiditeko estrategia:** Eginkizuna burutu daiteken aztertuko da eta ez bada bideragarria funtzionalitatea aldatuko da: modu automatikoan sortu ordez erabiltzaileak sortuko du magnet lotura.

2. **Metadatuak fitxategietatik modu automatikoan ezin lortzea**

    - **Azalpena:** Erabiliko den teknologiarekin (HTML5 eta JavaScript) agian ez da posible fitxategietatik metadatuak modu automatikoan lortzea edo posible izan arren kostu handia du.
    - **Eragina:** Ertaina. Bezeroaren funtzionalitatea aldatzea eskatzen du baina ez du proiektua oztopatzen.
    - **Gertatzeko probabilitatea:** Handia.
    - **Ekiditeko estrategia:** Eginkizuna burutu daiteken aztertuko da eta ez bada bideragarria funtzionalitatea aldatuko da: metadatuak modu automatikoan sortu ordez erabiltzaileak eskuz pasako ditu.

3. **Denbora-estimazio okerrak egitea**

    - **Azalpena:** Ataza bakoitza burutzeko denborak gaizki estimatzea.
    - **Eragina:** Ertaina.
    - **Gertatzeko probabilitatea:** Handia.
    - **Ekiditeko estrategia:** Ataza bakoitzean pasatako denbora kontrolatuko da eta estimazioetatik urruntzen badoaz zergatia aztertuko da. Behar izanez gero estimazioak egokituko dira.

3. KAPITULUA

Analisia

3.1 Analisi funtzionala: erabilpen eszenatokiak

Erabilpen nagusia liburuak trukatzea izango litzateke, noski. Erabiltzaile batek interesgarriak iruditzen zaizkion liburuak beste batzuekin partekatuko lituzke eta alderantziz, besteek partekatutakoak ere jasoko lituzke. Iruzkinak idazteko aukerarekin iritziak eta eztabaidak zabaltzen dira komunitate kutsua emanez. Erabiltzaileen arteko komunikazioa hobetuz gero (lagun gisa gehituz, lagunei mezuak idatziz, lagunen gogoko liburuak ikusteko aukera gehituz. . .) komunitate sendoagoa lortzea posible da. Beraz liburuekin lotutako euskal komunitate bat eraikitzeko balioko luke.

Web zerbitzaria hasieratik modu ezberdinetan erabiltzeko pentsatua dago, betiere APIak eskaintzen duenaren mugen barnean. Hori horrela izanik alde teknikotik erabilpen-eszenatoki ezberdinak eduki ditzakegu, adibidez: liburuak trukatzeko webgunea, HTML eta JavaScript bidezko widgetak eta aplikazio natiboak.

1. **Liburuak trukatzeko webgunea**

   Honako hau da proiektu honetan bezeroaren aldeari eman zaion erabilpena. Kasu honetan zerbitzariarekin JavaScript bidez komunikatzen den orri bakarreko aplikazioa eraiki da, baina beste lengoaia bat edo arkitektura bat ere erabil liteke, atzean APIa erabiltzen duen zerbitzari batekin adibidez.[1]

[1] Eranskineko "Instalazio-eskuliburua"atalean PHP bidez idatzitako adibide bat azaltzen da.

2. **HTML eta JavaScript bidezko widgetak**

   Widget horiek kode-zatiak izango lirateke, konfigurazio lan gutxirekin edozein orritan erraz txertatzekoak. Adibidez, demagun erabiltzaile baten liburuak erakusten dituen widget bat sortzen dugula eta erabiltzailearen izena pasata konfiguratzen dugula. Erabiltzaile batek igota dituen liburuak bere blogean erakustea edukiko luke honelako zerbait txertatuz:

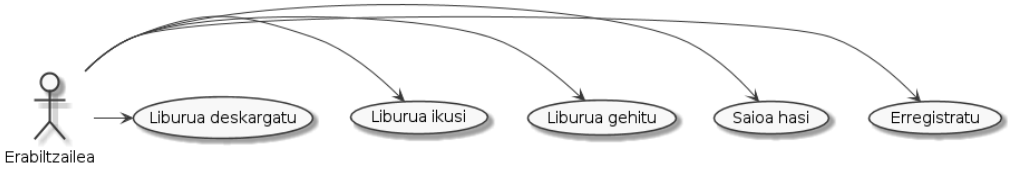
```
<script src="erabiltzaileLiburuWidget.js"></script>
<script>
  erabiltzaileLiburuWidget.abiarazi("nireErabiltzailea");
</script>
```

3. **Aplikazio natiboak**

   Sistema eragile edo plataforma jakin baterako garatutako aplikazioak. Web orriekin alderatuz hedapen mugatuagoa dute, baina plataformaren baliabideak hobeto aprobetxatzen dituzte. Adibide batzuk: Windows sitema eragilerako aplikazioa, smartphonetarako Android edo iOS aplikazioa, liburu berri bat gehitzen denean abisatzen duen scripta...

3.2 Erabilpen-kasuak

Aurreko atalean azaldu den bezala, zerbitzariaren APIa erabiltzeko modu asko dago. Horregatik, atal honetan, bezeroan inplementatzea pentsatu diren erabilpen-kasu garrantzitsuenak esplikatuko dira. Hiru aktore daude: erabiltzailea, administratzailea eta superadministratzailea. Administratzailea erabiltzailearen espezializazio bat da. Superadministratzailearean eta administratzailearen arteko erlazioa ere berdina da. Beraz, ondorengo erabilpen-kasuetan erabiltzailea aktorea azalduko da, baina administratzaileak eta superadministratzaileak burutu ditzaketen ekintzak dira, erabiltzaileak baitira.



3.1 Irudia: Erabilpen-kasu garrantzitsuenak.

1. **Erregistratu**

    - **Laburpena:** Erabiltzailea erregistratu egiten da.
    - **Aurrebaldintza:**
    - **Postbaldintza:** Erabiltzailea erregistratuta dago eta saioa hasita du.
    - **Gertaera fluxu normala:**
        (a) Erabiltzailea: Erregistratu orrira joaten da.
        (b) Erabiltzailea: Erabiltzaile-izena, pasahitza eta izena eremuak betetzen ditu (deskribapena aukeran).
        (c) Aplikazioa: Erabiltzailea erregistratzen du eta bere saioa hasten du profila orrira eramanez.
    - **Gertaera fluxu alternatiboak:**
        – 3. urratsa: Beharrezko eremuak ez dira bete.
            (a) Aplikazioa: Beharrezko eremuak ez direla bete adierazten du.
        – 3. urratsa: Aukeratutako erabiltzailea existitzen da.
            (a) Aplikazioa: Erabiltzailea existitzen dela adierazten du.

2. **Saioa hasi**

    - **Laburpena:** Erabiltzaileak saioa hasten du.
    - **Aurrebaldintza:**
    - **Postbaldintza:** Erabiltzaileak saioa hasita du.
    - **Gertaera fluxu normala:**
        (a) Erabiltzailea: Saioa hasi orrira joaten da.
        (b) Erabiltzailea: Erabiltzaile-izena eta pasahitza eremuak betetzen ditu.
        (c) Aplikazioa: Erabiltzailearen saioa du, menua aldatzen du eta hasierako orrira eramaten du.
    - **Gertaera fluxu alternatiboak:**
        – 3. urratsa: Beharrezko eremuak ez dira bete.
            (a) Aplikazioa: Beharrezko eremuak ez direla bete adierazten du.
        – 3. urratsa: Erabiltzaile-izena edo pasahitza okerra da.
            (a) Aplikazioa: Erabiltzaile-izen edo pasahitz okerra idatzi dela adierazten du.

3. **Liburua gehitu**

   - **Laburpena:** Erabiltzaileak liburu berri bat gehitzen du.
   - **Aurrebaldintza:** Erabiltzaileak saioa hasita du.
   - **Postbaldintza:** Erabiltzailearen liburua sisteman gehituta eta BitTorrent sarean partekatuta dago.
   - **Gertaera fluxu normala:**
     (a) Erabiltzailea: Liburua gehitu orrira joaten da.
     (b) Erabiltzailea: Ondorengo eremuak osatzen ditu: epub fitxategia, titulua, egileak, hizkuntza, sinopsia, urtea eta azala. Nahi izanez gero argitaletxea, generoa eta etiketak gehitzen ditu.
     (c) Aplikazioa: Erabiltzailearen liburua gehitzen du, liburuaren torrent fitxategia eta magnet lotura sortzen ditu eta BitTorrent sarean partekatzen du. Liburuaren orria erakusten du erabiltzaileak bidalitako datuekin eta magnet loturarekin.
   - **Gertaera fluxu alternatiboak:**
     – 3. urratsa: Beharrezko eremuak ez dira bete.
       (a) Aplikazioa: Beharrezko eremuak ez direla bete adierazten du.
     – 3. urratsa: Epub formatua ez duen fitxategia aukeratu da.
       (a) Aplikazioa: Fitxategiak epub formatua ez duela adierazten du.
     – 3. urratsa: JPG formatua ez duen azala aukeratu da.
       (a) Aplikazioa: Fitxategiak JPG formatua ez duela adierazten du.

4. **Liburua ikusi**

   - **Laburpena:** Erabiltzaileak liburu bati dagozkion datuak ikusten ditu.
   - **Aurrebaldintza:**
   - **Postbaldintza:**
   - **Gertaera fluxu normala:**
     (a) Erabiltzailea: Liburuaren orrira joaten da.
     (b) Aplikazioa: Liburuaren datuak erakusten ditu: titulua, magnet lotura, egileak, erabiltzailea (liburua gehitu duena), hizkuntza, igotze data, generoa, urtea, argitaletxea, etiketak, sinopsia eta azala. Horien azpian liburuaren iruzkinak zerrendatzen ditu.

- **Gertaera fluxu alternatiboak:**
  - 2. urratsa: Erabiltzaileak saioa hasita du.
    (a) Aplikazioa: Azalaren azpian "Liburu hau gogoko dut!"lotura erakutsiko du liburua gogokoetan sartzeko edo "Liburua gogoko dut"mezua liburua jada gogokoetan badu. Iruzkinen azpian iruzkin berri bat gehitzeko formularioa gehituko du.

5. **Liburua deskargatu**

- **Laburpena:** Erabiltzaileak liburu bat deskargatzen du.
- **Aurrebaldintza:** Erabiltzailea liburu baten datuak eskura ditu (ohikoena liburuaren orrian egotea da).
- **Postbaldintza:** Erabiltzaileak epub fitxategia eskuratu du.
- **Gertaera fluxu normala:**
  (a) Erabiltzailea: Magnet lotura hartu eta BitTorrent bezeroarekin irekitzen du.
  (b) Aplikazioa: Aplikazioaren *torrent-partekatze-sistema*-k BitTorrent bidez epub fitxategia bezeroarekin partekatzen du.
  (c) Erabiltzailea: Liburua deskargatu dela ikusita, liburua partekatzen jarraitzea edo uztea erabakitzen du.
- **Gertaera fluxu alternatiboak:**
  - 2. urratsa: Erabiltzaile bat edo gehiago liburua partekatzen ari dira.
    (a) Partekatzen ari diren erabiltzaileak: Liburua deskargatu nahi duenarekin fitxategia partekatzen dute.

4. KAPITULUA

Aplikazioaren diseinua

4.1 Arkitektura

Proiektu hau bitan banatuta dago: APIa eskaintzen duen zerbitzaria eta hori kontsumitzen duen bezeroa. Zerbitzaria web-zerbitzari bat da, HTTP bidez eskaerak jaso eta erantzunak bidaltzen dituena[1]. Bezeroa, berriz, HTML eta JavaScript-ez idatzitako web-orria exekutatzen duen web-nabigatzaile bat da[2]. Bi zati horiek HTTP bidez lotzen dira.

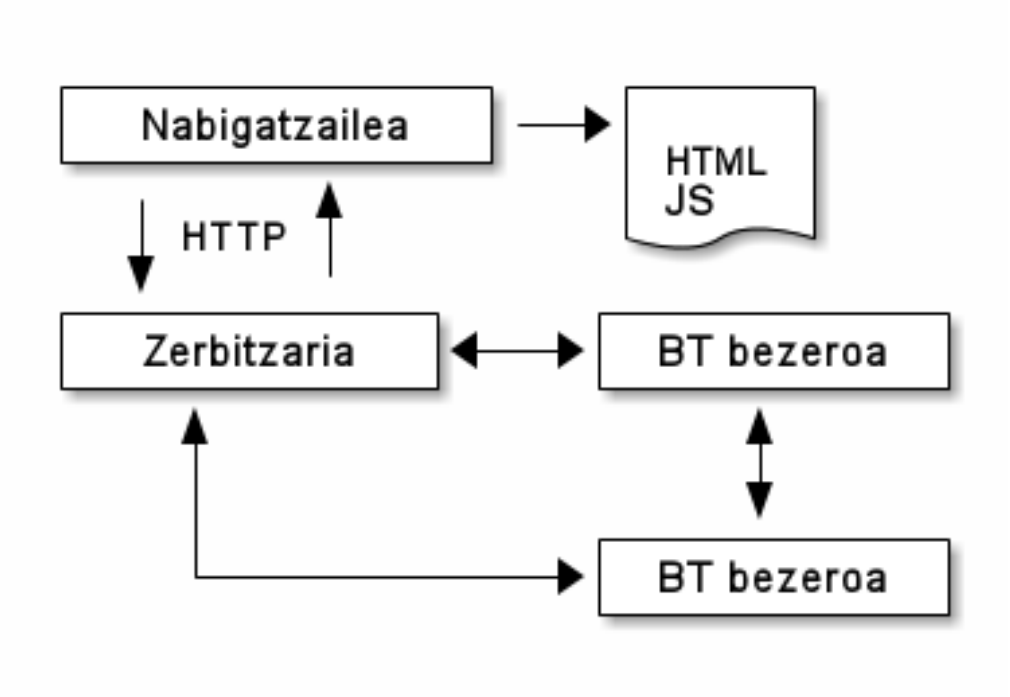
Banaketa horri esker, zerbitzaria edo bezeroa bien arteko interfazea errespetatzen duen beste osagai batekin ordezkatu daiteke. Web-nabigatzailearen ordez, adibidez, aplikazio natibo bat eduki genezake.

Erabiltzaileak, web-nabigatzaileaz gain, BitTorrent bezeroa ere erabil dezake. Tresna horren bidez bi eginkizun burutzeko aukera du: liburu bat deskargatu edo lehendik deskargatuta dituen liburuak partekatu. Bi kasuetan, gainontzeko BitTorrent sareko kideekin komunikatuko da. Kide horiek zerbitzaria eta liburua partekatzen duten erabiltzaileak izango dira.

1. **Zerbitzaria**

[1] HTTPS ere erabili daiteke, eta banaketan hori erabiltzea gomendatzen da, ikusi eranskineko "Instalazio-eskuliburua"atala.

[2] Aurrerago, HTML eta JavaScript kode-multzoari ere bezeroa izena emango zaio, "bezeroak exekutatzen duen kodea"moduko adierazpenak ez erabiltzearren. Kodea nabigatzaileak exekutatzen duela ulertu behar da.



4.1 Irudia: Sistemaren arkitektura orokorra.

Esan dugunez, zerbitzaria web-zerbitzari bat da, hau da, HTTP eskaerak jaso eta erantzuten ditu, eta REST[3] estiloko arkitektura du. Eskaera horietan baliabideekin burutu nahi diren eginkizunak HTTP metodoen bidez adierazten dira:

- GET: baliabide bat edo bilduma lortu.
- POST: baliabidea sortu.
- PUT: baliabidea aldatu.
- DELETE: baliabidea ezabatu.

Eskaera eta erantzun guztien edukia JSON formatuan bidaltzen da eta ez dago eskaeren jatorrien murriztapenik. Horretarako, edozein jatorri onartzen duen CORS[4] mekanismoa gaituta dago. Eskaera batetik bestera zerbitzariak ez du

[3]Representational State Transfer. SOAP/WSDL web-zerbitzuen ordezko arkitektura estiloa, geroz eta gehiago erabiltzen dena. "APIaren deskribapena"atalean sakonago azalduko da.

[4]Cross-Origin Resource Sharing. Web-nabigatzaileek "same-origin policy"segurtasun mekanismoa erabiltzen dute. Mekanismo horrek webguneko scriptak bakarrik uzten du exekutatzen; kanpoko API deiak ere ezin dira egin. Zerbitzariak CORS goiburuak erabiltzen baditu murriztapen hori saihestu daiteke. Gainera, *all cross-origin requests* baimentzea baino seguruagoa da.

testuingururik gordetzen; beraz, egoerarik gabeko protokoloa erabiltzen dela esan daiteke.

Zerbitzaria, lehen aipatu den bezala, bezeroarekin HTTP erabiliz komunikatzen da, baina instalazio prozesuan alderantzizko proxy bat gehitzen bazaio (ikus eranskineko "Instalazio-eskuliburua"atala) proxy horrek egingo du bitartekari lana. Bezeroaren eta proxyaren arteko komunikazioa HTTP edo HTTPS bidezkoa izan daiteke. Proxyaren eta zerbitzariaren artekoa, berriz, HTTP bidez gauzatuko da.

Datuen iraunkortasuna bermatzea zerbitzariaren eginkizuna da, eta, horretarako, datu-base erlazional bat erabiltzen du. Zerbitzariak erabiltzen dituen fitxategiei dagokienez, 2 taldetan banatzen dira: pribatuak eta publikoak. Lehenak EPUB fitxategiak eta horiei dagozkien .torrent fitxategiak dira, eta ezin dira kanpotik atzitu. Publikoak, berriz, JPG formatuko azalak dira eta bezeroak zerbitzariari eskatuz gero lor ditzake.

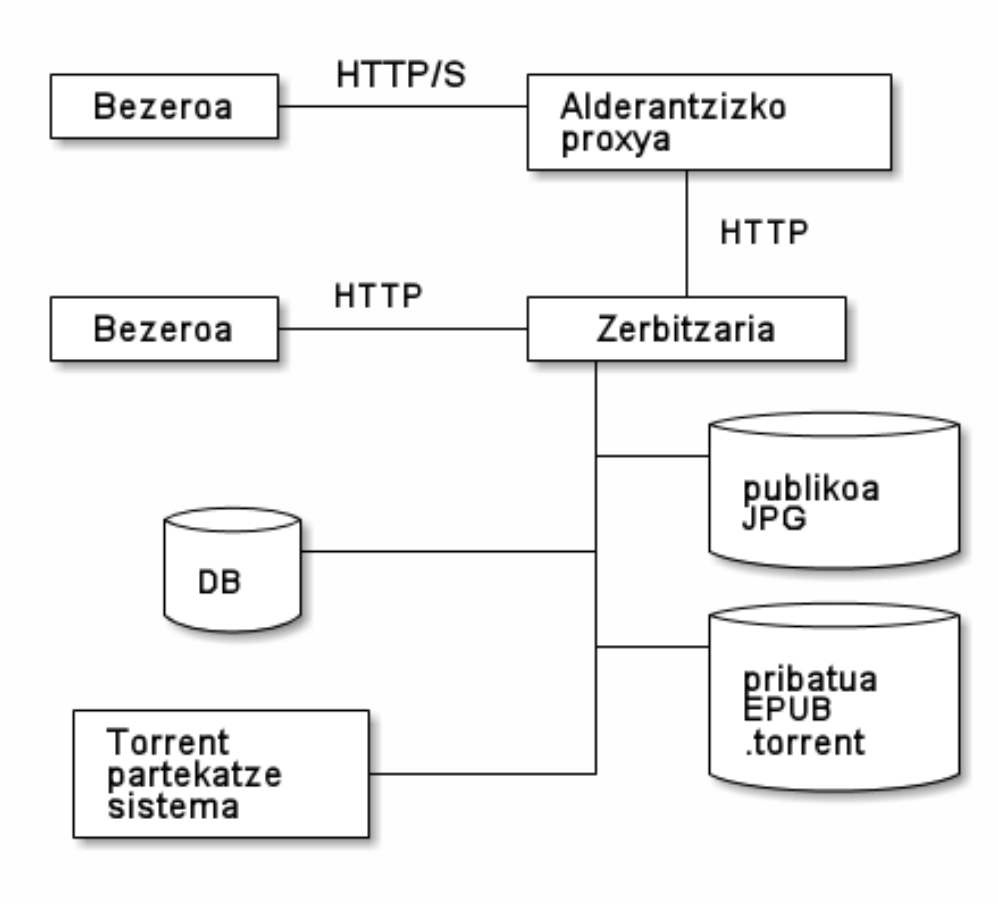
Azkenik, liburuak partekatzeko *torrent partekatze-sistema* deitu zaion osagaia erabiltzen da. Zerbitzaria, liburu bat partekatu nahi duenean, sistema horrekin komunikatzen da. Zerbitzariak, bezeroak pasatako liburuaren edukiarekin, EPUB fitxategia sortu, dagokion torrent fitxategia eraiki eta *torrent partekatze-sistemari* fitxategi hori partekatzeko eskatzen dio.

2. **Bezeroa**

   Web-nabigatzaileak, lehen esan bezala, HTML eta JavaScriptez idatzitako fitxategiak[5] kargatzen ditu eta, jarraian, JavaScript kodea exekutatzen du. Fitxategi horien jatorria disko lokalean edo web-zerbitzari batean egon daiteke. Orri bakarreko aplikazioa izanik, bezeroari dagozkion fitxategiak behin kargatzearekin aski da. Hori kontuan edukiz, bezeroaren banaketari HTML5eko *Cache Manifest* gehitu zaio. Manifestu horren bidez fitxategiak diskoan gorde daitezke, cache gisa. Jatorrizko fitxategiak webgune batean egonez gero, adibidez, lehen bisitaren ostean fitxategiak cachean gordeta geratuko dira. Beranduago, webgunea eskuragarri ez badago ere, cachean gordetako fitxategiak erabiliko dira. Behin dena martxan jarritakoan, AJAX[6] deien bidez, zerbitzariarekin komunikatzen da.

[5] CSS eta irudiak ere kargatzen dira, baina fitxategi horiek koderik ez dutenez ez dira aipatzen.

[6] Asynchronous JavaScript and XML: JavaScript dei asinkronoak, kasu honetan JSON erabiltzen da XML ordez.

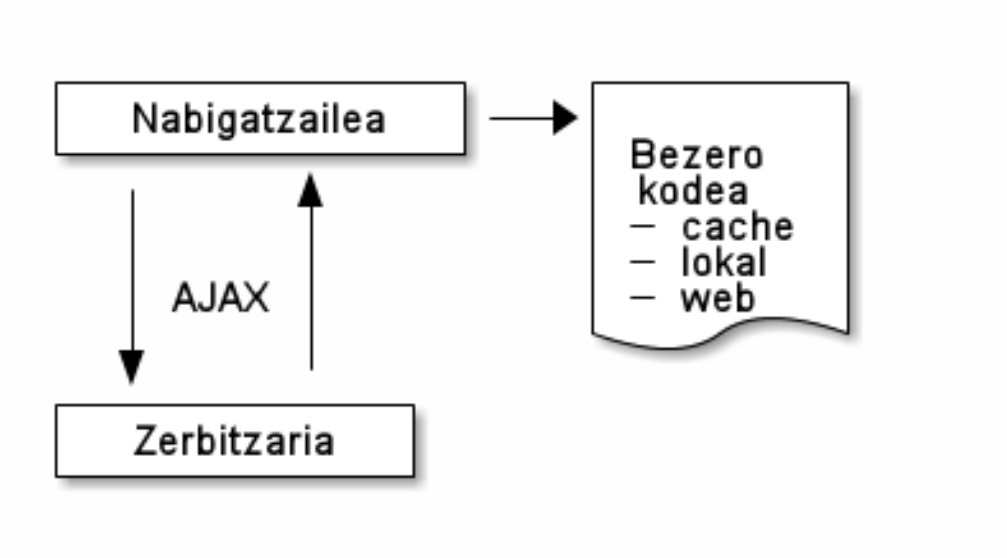


4.2 Irudia: Zerbitzariaren arkitektura.

4.2 Baliabideen deskribapena eta datuen modelizazioa

Proiektuko baliabide garrantzitsuena liburua da, dudarik gabe. Liburuak zehazteko orduan, hasiera batean, *ONIX for books* edo *Dublin Core* estandarra erabiltzea pentsatu da, baina estandar horietako bat jarraitzeak abantaila askorik ez zekarrela ikusi da. Nahiko konplikatuak dira, eta, proiektu honentzako, deskribapen sinpleagoa aski da. Hala ere, eremuak zehazterako orduan erreferentzia gisa erabili dira. Horrela, liburuak ondorengo eremuekin definitu dira:

- Magnet lotura: EPUB fitxategitik lortutako torrent fitxategiari dagokiona.
- Titulua.
- Egileak: Liburuaren egileen zerrenda (bat edo gehiago).



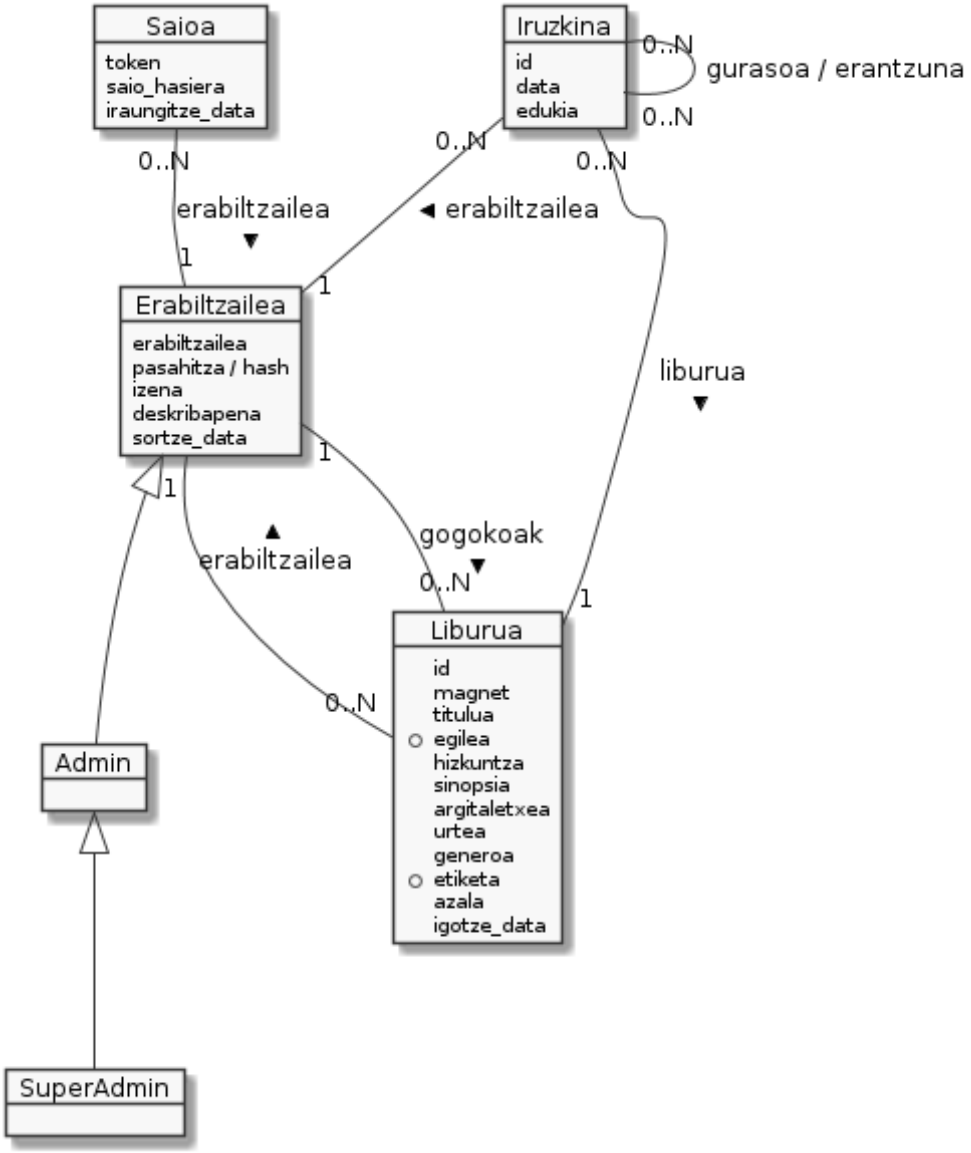
4.3 Irudia: Bezeroaren arkitektura.

- Hizkuntza: ze hizkuntzatan idatzita dagoen.
- Sinopsia.
- Argitaletxea: Argitaletxearen izena.
- Urtea: Liburua ze urtetan argitaratu den.
- Generoa: Literatura-generoa, liburua sailkatzeko. Hautazkoa izatea erabaki da, liburu batzuk zailak baitira genero batean sailkatzen.
- Etiketak: Etiketen zerrenda. Zero etiketa eduki ditzake. Etiketa bat liburua deskribatzen duen gako-hitza da.
- Azala: Liburuaren azala duen irudia, liburua nolakoa den ideia bat edukitzeko.
- Igotze-data: Erabiltzaileak liburua noiz igo duen.

Liburuez gain, erabiltzaileak, iruzkinak, saioak eta administratzaileak ere badaude. Erabiltzaile batek hainbat liburu gehitu ditzake; eta, liburu bakoitza erabiltzaile bati lotuta dago. Liburuak kudeatzeaz gain, horiekin beste bi eginkizun ere burutu ditzake: gogoko gisa gehitu eta iruzkinak idatzi. Liburu bati lotuta, hainbat iruzkin aurkitu daitezke, eta iruzkin horiek beraien artean erlazionatuta egon daitezke, guraso/erantzun erlazioaren bidez. Eginkizun horiek burutzeko, erabiltzaileak saioa hasi behar du. Azkenik, bi erabiltzaile berezi ere badaude: administratzailea (admin)

eta superadministratzailea (superadmin). Adminek erabiltzaile normal batek baino gaitasun gehiago ditu. superadminek, berriz, adminek baino gaitasun gehiago ditu.

4.4 irudian beraien atributuak eta erlazioak ikus daitezke. Egilea eta etiketa (zirkulu batez markatuta) eremuak balio aniztunak dira. Erabiltzaileak pasahitza du, baina hash balioa gordeko da (aurrerago aipatuko da zergatia). Iruzkin bakoitzak gurasoak eta erantzunak ditu, zeinak iruzkinak diren.



4.4 Irudia: Domeinuaren eredua.

Liburuari id atributu gehitu zaio gehitutako liburu bakoitza identifikatzeko. Liburuen datuetan identifikatzaile egoki bat aurkitzeko ahalegina egin da, baina ez da egokirik aurkitu. Tituluak ez du balio, hainbat liburuk titulu berdina eduki baitezakete. Titulu

eta egileak gisako konbinazioak ere ez, liburu batek hainbat edizio eduki baititzake. Edizioa edo ISBN eremua gehitzea pentsatu da, baina, liburu elektronikoak izanik, liburu guztiek ediziotan kaleratzeko eredua ez dutela jarraituko pentsatu da. Beste aukera bat edukiaren bidez identifikatzea litzake, hau da, magnet loturako *info hash* balioaren bidez. Aukera hori, ordea, baztertu egin da, posible baita fitxategi ezberdinek liburu berdina adieraztea (nahikoa da edukiko zatitxo bat aldatzea). Hori guztia ikusirik, liburuak gehitzeko murriztapenik ez da gehitu. Horren ordez, liburu bat gehitzerako orduan, titulua berdina edo antzerakoa duen liburu bat existitzen denean erabiltzaileari abisatzeko erabakia hartu da. Errepikatutako liburuekin arazorik balego, administratzailea arduratuko litzake horien kudeaketaz.

4.3 APIaren deskribapena

1. **Sarrera**

   Domeinuarekin lan egiteko, web API bat diseinatu da. Atal honetan API hori deskribatuko da. Ondorengo zatitan banatu da:

   - Sarrera
   - Saioak
   - Erabiltzaileak
   - Liburuak
   - Iruzkinak
   - Gogokoak
   - Administratzaileak

   APIak REST motako web-zerbitzuen arkitektura jarraitzen du. URLak baliabideak irudikatzeko erabiltzen dira eta izenez osatuta daude. Baliabide bakoitzeko normalean 2 oinarri-URL daude: bilduma irudikatzen duena eta bildumako elementu zehatz bati erreferentzia egiten diona. Adibidez, "liburu"baliabidearentzako:

   ```
   /liburuak
   /liburuak/42
   ```

Lehenak liburu bilduma bat adierazten du eta bigarrenak bilduma horretako 42 identifikatzailea duen liburua. APIa deskribatzerakoan, baliabide konkretu baten balioak (42 liburua bezala) ipini ordez identifikatzailearen izen generikoa erabiliko da (`/liburuak/:id`).

Adibideko URLari zerbait falta zaio, ordea: URL oinarria. Oinarri hori `http(s)://ostalaria:portua/api_bidea` gisako helbidea da. *api_bidea* zatiak errotik ahal den hurbilen egon beharko luke. Horretarako, eta webgune nagusiarekin ez nahasteko, `dev.nirewebgunea.com` gisako azpidomeinu bat erabil daiteke. APIa etorkizunean aldatu daitekeenez eta bertsioen artean nahasketarik ez egoteko, bertsio bakoitzeko oinarri ezberdina egotea komeni da. Horregatik APIaren bertsio zenbakiak ere azaldu behar du oinarrian. Oinarri hau, URL guztietan erabiliko denez, hemendik aurrera ez da adieraziko. Hori guztia jakinda, helbideak honela interpretatu behar dira (`http://localhost:3000/v1` oinarria erabiliz):

```
/liburuak    => http://localhost:3000/v1/liburuak
/liburuak/42 => http://localhost:3000/v1/liburuak/42
```

APIarekin lotutako ekintzak honako HTTP metodoen bidez gauzatzen dira: POST, GET, PUT eta DELETE. Lau metodo horiek CRUD[7] eragiketen parekideak dira, berdinak ez diren arren. Beraz, URLak (baliabideen izenak) eta metodoak (ekintzak) elkartuz baliabideak sortu, irakurri, eguneratu edo ezabatu ditzakegu.

Baliabide bat beste baten barnean egon daitekeenez, URLak habiaratu daitezke, baina habiaratze-maila altu bat nahasgarria izan daiteke. Horregatik, gehienez habiaratze-maila bat gomendatzen da. Adibidez, `/liburuak/42/iruzkinak` liburu baten iruzkinak adierazteko.

Eskaera konplexuagoak burutzeko (filtro edo bilaketak) *query string*-ean[8] parametroak gehitzen dira:

```
/liburuak?argitaletxea=etxea
/liburuak?argitaletxea=etxea&urtea=2009
/liburuak?muga=15
```

[7] Create, Read, Update and Delete. Iraunkortasun mailako funtzio basikoak: sortu, irakurri, eguneratu eta ezabatu.

[8] URL batean, egitura hierarkikotik kanpo geratzen den zatia, `?` karakterearen atzetik azaldu ohi da.

Bilduma bat jasotzean, handia izan daitekeenez, ez dira osagai guztiak itzultzen. Muga bat dago ezarrita eta muga hori aldatzeko *muga* parametroa erabil daiteke (0 balioarekin baliabide guztiak jasoko dira). Elementu jakin batetik aurrerakoak hartu nahi badira *desplazamendua* parametroa erabil daiteke. Bilduma bakoitzeko *guztira* eremua itzultzen da, hau da, bildumak dituen elementu kopurua. Eremu hori muga eta desplazamendua parametroen balioak kalkulatzeko erabil daiteke.

Adibidez, l1, l2, l3, l4, l5, l6 eta l7 baliabidez osatutako bilduma edukita:

```
GET /liburuak?desplazamendua=2&muga=3
```

```
{
    "desplazamendua": 2,
    "muga": 3,
    "guztira": 7,
    "liburuak": [l3, l4, l5]
}
```

Erroreei dagokienez, ekintzekin egiten den antzera, HTTP errore kodeak erabiltzen dira. 2xx familiakoek dena ongi joan dela adierazten dute, 4xx bezeroaren akatsa eta 5xx zerbitzariaren akatsa. Hona erabilitako zerrenda:

Zenbakia	Izena	Deskribapena
200	OK	Dena ongi joan da
201	Created	Baliabidea ongi sortu da
400	Bad Request	Eskaera ezin da prozesatu sintaxi okerra duelako
401	Unauthorized	Baimenik ez, kautotzea beharrezkoa
404	Not Found	Baliabidea ez da aurkitu
422	Unprocessable Entity	Eskaerak sintaxi egokia du baina errore semantikoak ditu
500	Internal Server Error	Zerbitzarian errore bat gertatu da

Errorearekin batera mezu laburra eta deskribapena itzultzen da:

```
{
    "mezua": "Ez duzu eragiketa burutzeko baimenik",
```

```
    "deskribapena": "Eragiketa hau burutzeko saioa hasita eduki behar da.
      Saioarekin batera token bat eskuratzen da eta beharrezkoa da
      token hori parametro gisa pasatzea"
}
```

Azkenik, hurrengo ataletan ekintzak adierazteko erabiltzen den egitura eta notazioa azalduko da. Ekintza bakoitza honela antolatuta dago:

(a) HTTP metodoa eta URLa. *Query string*-neko parametroak ez dira adierazten. Baliabide konkretuak adierazteko identifikatzaileak erabiltzen dira, `:` karakterearekin hasita (`:id`).

(b) Parametroak. *Query string*-ean gehitu daitezkeen parametroak.

(c) Sarrera. Eskaeraren gorputzean bidali beharreko datuak eta adibide bat (JSON formatuarekin).

(d) Erantzuna. Sarrerako adibideari erantzuten dion erantzunaren gorputza (JSON hau ere).

Ekintza guztiek formatu berdina ez dutenez, posible da zati horietako bat ez agertzea. Adibidez, sarrera gisa ezer ez jasotzea. *Parametroak* eta *sarrera* ataletan, beharrezkoak diren parametro eta datuak izartxo (*) batekin markatu dira.

2. **Saioak**

   APIarekin hainbat eginkizun burutzeko saioa hasita eduki behar da. Horretarako, lehenbizi, APIa erabiltzen ari denak erabiltzaile bat (hurrengo atalean) sortuta eduki behar du. Saio berri bat lortzeko erabiltzailea eta pasahitza bidaltzen dira. Datuak zuzenak badira, zerbitzariak token bat eta horren iraungitze data itzuliko du. Tokena ausaz sortutako testu-kate bat da.

   Saioa hasita eduki behar den eginkizunak burutzeko, URLari lortutako tokena parametro gisa erantsiko zaio. Adibidez, `DELETE /erabiltzaileak/era1` burutzeko `DELETE /erabiltzaileak/era1?token=niretokena` bidez egingo litzateke.

   Tokena iraungitzen bada baliogabetuta geratuko da eta saio berri bat hasi beharko da. Horrela, tokenak betirako ematea ekiditen da. Saioa amaituz gero ere tokena baliogabetzen da.

   (a) **Saioa hasi**

`POST /saioak`

Sarrera

Izena	Mota	Deskribapena
*erabiltzailea	testua	Erabiltzaile izena
*pasahitza	testua	Pasahitza

```
{
  "erabiltzailea": "erab1",
  "pasahitza": "1234"
}
```

Erantzuna

```
{
    "erabiltzailea": "erab1",
    "token": "fjepsfhesphfpesfs",
    "saio_hasiera": "2010-05-16T02:15:15Z",
    "iraungitze_data": "2010-05-21T02:15:15Z"
}
```

(b) **Saioa amaitu**

`DELETE /saioak/:token`

3. **Erabiltzaileak**

Erabiltzaileen pasahitzak besterik ezean gordetzeak hainbat arazo ekartzen ditu. Segurtasun arazo batengatik datu-baseko datuak eskuragarri geratzen badira pasahitz guztiak agerian geratzen dira. Horregatik normalean hash kriptografiko bat aplikatuta gordetzen dira. Jatorrizko testua ez da inoiz gordetzen, eta horregatik, ezin dugu pasahitzaren balioa lortu. Beraz, erabiltzaile batek pasahitza atributua duen arren ez da inoiz itzultzen.

Erabiltzaile bat aldatzeko eta ezabatzeko saioa hasita eduki behar da.

(a) **Erabiltzaileak lortu**

`GET /erabiltzaileak`

Parametroak

Izena	Mota	Deskribapena
desplazamendua	zenbakia	
muga	zenbakia	

Erantzuna

```
{
    "desplazamendua": 0,
    "muga": 25,
    "guztira": 23,
    "erabiltzaileak": [{
"erabiltzailea": "era1",
"izena": "Era",
"deskribapena": "Erabiltzaile bat naiz",
"sortze_data": "2010-04-14T02:15:15Z"
    }, {
"erabiltzailea": "era2",
"izena": "Era bi",
"deskribapena": "Beste erabiltzaile bat naiz",
"sortze_data": "2010-04-14T02:15:15Z"
    },
    ...
    ]
}
```

(b) **Erabiltzaile bat lortu**

```
GET /erabiltzaileak/:erabiltzailea
```

Erantzuna

```
{
    "erabiltzailea": {
"erabiltzailea": "era1",
"izena": "Era",
"deskribapena": "Erabiltzaile bat naiz",
"sortze_data": "2010-04-14T02:15:15Z"
    }
}
```

(c) **Erabiltzailea gehitu**

`POST /erabiltzaileak`

Sarrera

Izena	Mota	Deskribapena
*erabiltzailea	testua	Erabiltzaile izena, identifikatzailea izango dena
*pasahitza	testua	Pasahitza
*izena	testua	Izena
deskribapena	testua	Erabiltzailea deskribatuko duen testua

```
{
    "erabiltzailea": "era1",
    "pasahitza": "1234",
    "izena": "Era",
    "deskribapena": "Erabiltzaile bat naiz",
}
```

Erantzuna

```
{
    "erabiltzailea": {
"erabiltzailea": "era1",
"izena": "Era",
"deskribapena": "Erabiltzaile bat naiz",
"sortze_data": "2010-04-14T02:15:15Z"
    }
}
```

(d) **Erabiltzaile bat aldatu**

`PUT /erabiltzaileak/:erabiltzailea`

Parametroak

Izena	Mota	Deskribapena
*token	testua	Erabiltzailearen tokena

Sarrera

Izena	Mota	Deskribapena
*pasahitza	testua	Pasahitza
*izena	testua	Izena
deskribapena	testua	Erabiltzailea deskribatuko duen testua

```
{
    "pasahitza": "pasahitz berria",
    "izena": "Era berria",
    "deskribapena": "Erabiltzaile bat naiz",
}
```

Erantzuna

```
{
    "erabiltzailea": {
"erabiltzailea": "era1",
"izena": "Era berria",
"deskribapena": "Erabiltzaile bat naiz",
"sortze_data": "2010-04-14T02:15:15Z"
    }
}
```

(e) **Erabiltzailea ezabatu**

```
DELETE /erabiltzaileak/:erabiltzailea
```

Parametroak

Izena	Mota	Deskribapena
*token	testua	Erabiltzailearen tokena

4. **Liburuak**

Liburuak gehitzeko, aldatzeko edo ezabatzeko saioa hasita eduki behar da.

(a) **Liburuak lortu**

Liburu guztiak:

```
GET /liburuak
```

Erabiltzaile baten liburuak:

```
GET /erabiltzaileak/:erabiltzailea/liburuak
```

Egile jakin batzuen liburuak:

```
GET /liburuak?egileak=:egile1+:egile2
```

Argitaletxe jakin bateko liburuak:

```
GET /liburuak?argitaletxea=:argitaletxea
```

Genero jakin batzuetako liburuak:

```
GET /liburuak?generoak=:gen1+:gen2
```

Etiketa jakin batzuk dituzten liburuak:

```
GET /liburuak?etiketak=:etiketa1+:etiketa2
```

Urte jakin batean argitaratutako liburuak:

```
GET /liburuak?urtea=:urtea
```

Hizkuntza jakin batean argitaratutako liburuak:

```
GET /liburuak?hizkuntza=:hizkuntza
```

Hainbat parametro nahastuta:

```
GET /liburuak?egilea=:egilea&etiketak=:etiketa
```

Parametroak

Izena	Mota	Deskribapena
desplazamendua	zenbakia	
muga	zenbakia	
egileak	testu zerrenda	Egileak, '+' bidez bereizita
argitaletxea	testua	Argitaletxe baten izena
generoak	testu zerrenda	Generoak, '+' bidez bereizita
etiketak	testu zerrenda	Etiketak, '+' bidez bereizita
urteak	zenbaki zerrenda	Urteak, '+' bidez bereizita
hizkuntza	testua	Liburuaren hizkuntza

Erantzuna

```
{
    "desplazamendua": 0,
    "muga": 25,
    "guztira": 2,
    "liburuak":
    [{
"id": 1
"magnet": "magnet:?xt=urn:btih:372e39f275f3886b37e857b394e52cf73c1f9bb6"
```

```
"titulua": "Kaixo mundua",
"egileak": ["Joxe", "Patxi"],
"hizkuntza": "euskara",
"sinopsia": "Duela urte asko...",
"argitaletxea": "Etxea",
"urtea": "2009",
"generoa": "Eleberria",
"etiketak": ["kaixo", "joxe", "zaharra"],
"azala": "/azalak/kaixo.jpg",
"erabiltzailea": "erab1"
"igotze_data": "2010-04-14T02:15:15Z",
"iruzkin_kopurua": 3,
"gogoko_kopurua": 2,
    }, {
"id": 2,
"magnet": "magnet:?xt=urn:btih:937e39f275f3886b37e857b394e52cf73c1f9bb6"
"titulua": "Kaixo unibertsoa",
"egileak": ["Joxe"],
"hizkuntza": "euskara",
"sinopsia": "Galaxia batean...",
"argitaletxea": "BesteEtxeBat",
"urtea": "2010",
"generoa": "Fikzioa",
"etiketak": ["joxe", "unibertsoa", "galaxia"],
"azala": "/azalak/kaixo_unibertsoa.jpg",
"erabiltzailea": "erab2"
"igotze_data": "2010-05-14T02:15:15Z",
"iruzkin_kopurua": 5,
"gogoko_kopurua": 4,
    }]
}
```


(b) **Liburua lortu**

```
GET /liburuak/:id
```

Erantzuna:

```
{
    "liburua": {
"id": 1
"magnet": "magnet:?xt=urn:btih:372e39f275f3886b37e857b394e52cf73c1f9bb6"
"titulua": "Kaixo mundua",
"egileak": ["Joxe", "Patxi"],
"hizkuntza": "euskara",
"sinopsia": "Duela urte asko...",
"argitaletxea": "Etxea",
"urtea": "2009",
"generoa": "Eleberria",
"etiketak": ["kaixo", "joxe", "zaharra"],
"azala": "/azalak/kaixo.jpg",
"erabiltzailea": "erab1"
"igotze_data": "2010-04-14T02:15:15Z",
"iruzkin_kopurua": 3,
"gogoko_kopurua": 2,
    }
}
```

(c) **Liburua gehitu**

```
POST /liburuak
```

Parametroak

Izena	Mota	Deskribapena
*token	testua	Erabiltzailearen tokena

Sarrera

Izena	Mota	Deskribapena
*epub	base64	Epub fitxategiaren edukia, base64 bidez kodetua
*titulua	testua	Liburuaren titutulua
*egileak	testuen arraya	Liburuaren egileen zerrenda (bat edo gehiago)
*hizkuntza	testua	Liburua idatzita dagoen hizkuntza
*sinopsia	testua	Liburuaren sinopsia
argitaletxea	testua	Argitaletxearen izena
*urtea	zenbakia	Liburua argitaratu den urtea
generoa	testua	Literatura-generoa, liburua sailkatzeko
*etiketak	testuen arraya	Etiketen zerrenda
*azala	base64	Azalaren irudiaren edukia, base64 bidez kodetua

Erabiltzailea tokenetik lortzena da, inplizitua da.

```
{
    "epub": "TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx [...]",
    "titulua": "Kaixo mundua",
    "egileak": ["Joxe", "Patxi"],
    "hizkuntza": "euskara",
    "sinopsia": "Duela urte asko...",
    "argitaletxea": "Etxea",
    "urtea": "2009",
    "generoa": "Eleberria",
    "etiketak": ["kaixo", "joxe", "zaharra"],
    "azala": "dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcm [...]"
}
```

Erantzuna

```
{
    "liburua": {
"id": 1
"magnet": "magnet:?xt=urn:btih:372e39f275f3886b37e857b394e52cf73c1f9bb6"
"titulua": "Kaixo mundua",
"egileak": ["Joxe", "Patxi"],
"hizkuntza": "euskara",
"sinopsia": "Duela urte asko...",
"argitaletxea": "Etxea",
```

```
"urtea": "2009",
"generoa": "Eleberria",
"etiketak": ["kaixo", "joxe", "zaharra"],
"azala": "/azalak/kaixo.jpg",
"erabiltzailea": "erab1"
"igotze_data": "2010-04-14T02:15:15Z",
"iruzkin_kopurua": 0,
    }
}
```

(d) **Liburua aldatu**

Aldatzeko orduan, epub edo magnet eremurik ez da erabiltzen. Fitxategia liburuaren oinarria da eta hori aldatuko balitz beste liburu bat igoko litzake.

```
PUT /liburuak/:id
```

Parametroak

Izena	Mota	Deskribapena
*token	testua	Erabiltzailearen tokena

Sarrera

Izena	Mota	Deskribapena
*titulua	testua	Liburuaren titutulua
*egileak	testuen arraya	Liburuaren egileen zerrenda (bat edo gehiago)
*hizkuntza	testua	Liburua idatzita dagoen hizkuntza
*sinopsia	testua	Liburuaren sinopsia
argitaletxea	testua	Argitaletxearen izena
*urtea	zenbakia	Liburua argitaratu den urtea
generoa	testua	Literatura-generoa, liburua sailkatzeko
*etiketak	testuen arraya	Etiketen zerrenda
*azala	base64	Azalaren irudiaren edukia, base64 bidez kodetua

Erabiltzailea tokenetik lortzen da, inplizitua da.

```
{
    "titulua": "Kaixo mundua",
    "egileak": ["Joxe", "Patxi"],
    "hizkuntza": "euskara",
```

```
    "sinopsia": "Duela urte asko...",
    "argitaletxea": "Etxea",
    "urtea": "2009",
    "generoa": "Eleberria",
    "etiketak": ["kaixo", "joxe", "zaharra"],
    "azala": "/azalak/kaixo.jpg",
}
```

(e) **Liburua ezabatu**

Liburu bat ezabatzean liburu horri lotutako iruzkinak eta gogokoak ere ezabatzen dira.

```
DELETE /liburuak/:id
```

Parametroak

Izena	Mota	Deskribapena
*token	testua	Erabiltzailearen tokena

(f) **Liburuekin lotutako hainbat datu**

Titulu guztiak:

```
GET /tituluak
```

Egile guztiak:

```
GET /egileak
```

Argitaletxe guztiak:

```
GET /argitaletxeak
```

Genero guztiak:

```
GET /generoak
```

Etiketa guztiak:

```
GET /etiketak
```

Urte guztiak:

```
GET /urteak
```

Hizkuntza guztiak:

`GET /hizkuntzak`

Erantzuna

```
{
    "desplazamendua": 0,
    "muga": 25,
    "guztira": 2,
    "egileak" : ["Patxi", "Joxe"]
}
```

Erantzun guztiek formatu berdina dute: bildumaren metadatuak eta eskatutako datuen zerrenda. Egileen ordez, generoak edukiz gero, adibidez, `[...] "generoak": ["genero1", "genero2"] [...]` edukiko genuke.

5. **Iruzkinak**

   Iruzkin batek hainbat erantzun eduki ditzake. Erantzunen id zenbakiak iruzkinarekin batera jasotzen dira. Bestalde, erantzun bat ez doa iruzkin bakarrarekin lotuta, hainbat iruzkini erantzutea posible da. Horiei gurasoak deituko zaie eta horien id zenbakiak ere lortzen dira.

   Iruzkinak gehitzeko, aldatzeko eta ezabatzeko saioa hasita eduki behar da.

   (a) **Iruzkinak lortu**

   Iruzkin guztiak lortu:

   `GET /iruzkinak`

   Liburu baten iruzkinak lortu:

   `GET /liburuak/:id/iruzkinak`

   Erabiltzaile batek idatzitako iruzkinak lortu:

   `GET /erabiltzaileak/:erabiltzailea/iruzkinak`

   Parametroak

Izena	Mota	Deskribapena
desplazamendua	zenbakia	
muga	zenbakia	

   Erantzuna

```
{
    "desplazamendua": 0,
    "muga": 25,
    "guztira": 5,
    "iruzkinak": [{
"id": 3,
"liburua": 23,
"erabiltzailea": "era1",
"gurasoak": [],
"erantzunak": [4],
"data": "2010-05-21T02:15:15Z",
"edukia": "Iruzkin bat"
    },{
"id": 4,
"liburua": 23,
"erabiltzailea": "era2",
"gurasoak": [1, 3],
"erantzunak": [],
"data": "2010-05-21T02:17:35Z",
"edukia": "Beste iruzkin bat"
    }
    ...
    ]
}
```


(b) **Iruzkina lortu**

```
GET /iruzkinak/:id
```

Erantzuna


```
{
    "iruzkina": {
"id": 3,
"liburua": 23,
"erabiltzailea": "era1",
"gurasoak": [],
"erantzunak": [4],
```

```
"data": "2010-05-21T02:15:15Z",
"edukia": "Iruzkin bat"
    }
}
```

(c) **Iruzkina gehitu**

`POST /liburuak/:id/iruzkinak`

Parametroak

Izena	Mota	Deskribapena
*token	testua	Erabiltzailearen tokena

Sarrera

Izena	Mota	Deskribapena
*gurasoak	zenbakien arraya	Iruzkinaren gurasoen zerrenda
*edukia	testua	Iruzkinaren edukia

```
{
    "gurasoak": [3],
    "edukia": "Hau iruzkin bat da"
}
```

Erantzuna

```
{
    "iruzkina": {
"id": 5,
"liburua": 23,
"erabiltzailea": "era1",
"gurasoak": [3],
"erantzunak": [],
"data": "2010-05-21T02:15:15Z",
"edukia": "Hau iruzkin bat da"
    }
}
```

(d) **Iruzkina aldatu**

`PUT /iruzkinak/:id`

Parametroak

Izena	Mota	Deskribapena
*token	testua	Erabiltzailearen tokena

Sarrera

Izena	Mota	Deskribapena
*gurasoak	zenbakien arraya	Iruzkinaren gurasoen zerrenda
*edukia	testua	Iruzkinaren edukia

```
{
    "gurasoak": [1, 3, 4],
    "edukia": "Eduki berria"
}
```

Erantzuna

```
{
    "iruzkina": {
"id": 5,
"liburua": 23,
"erabiltzailea": "era1",
"gurasoak": [1, 3, 4],
"erantzunak": [],
"data": "2010-05-21T02:15:15Z",
"edukia": "Eduki berria"
    }
}
```

(e) **Iruzkina ezabatu**

Iruzkina ezabatzean guraso/erantzun erreferentziak ere ezabatzen dira.

```
DELETE /iruzkinak/:id
```

Parametroak

Izena	Mota	Deskribapena
*token	testua	Erabiltzailearen tokena

6. **Gogokoak**

Erabiltzaile bati liburu bat gustatuz gero, gogokoen bilduman sartu dezake. Horrela, erabiltzaile batek gogoko dituen liburuak batetik eta liburu bat gogoko duten erabiltzaileak bestetik edukiko ditugu.

Gogokoak gehitzeko edo ezabatzeko saioa hasita eduki behar da.

(a) **Gogokoak lortu**

Erabiltzaile baten gogoko liburuak:

`GET /erabiltzaileak/:erabiltzailea/gogoko_liburuak`

Parametroak

Izena	Mota	Deskribapena
desplazamendua	zenbakia	
muga	zenbakia	

Erantzuna

```
{
    "desplazamendua": 0,
    "muga": 25,
    "guztira": 2,
    "gogoko_liburuak":
    [{
"id": 1
"magnet": "magnet:?xt=urn:btih:372e39f275f3886b37e857b394e52cf73c1f9bb6"
"titulua": "Kaixo mundua",
"egileak": ["Joxe", "Patxi"],
"hizkuntza": "euskara",
"sinopsia": "Duela urte asko...",
"argitaletxea": "Etxea",
"urtea": "2009",
"generoa": "Eleberria",
"etiketak": ["kaixo", "joxe", "zaharra"],
"azala": "/azalak/kaixo.jpg",
"erabiltzailea": "erab1"
"igotze_data": "2010-04-14T02:15:15Z",
"iruzkin_kopurua": 3,
"gogoko_kopurua": 2,
    }, {
"id": 2,
"magnet": "magnet:?xt=urn:btih:937e39f275f3886b37e857b394e52cf73c1f9bb6"
"titulua": "Kaixo unibertsoa",
```

```
"egileak": ["Joxe"],
"hizkuntza": "euskara",
"sinopsia": "Galaxia batean...",
"argitaletxea": "BesteEtxeBat",
"urtea": "2010",
"generoa": "Fikzioa",
"etiketak": ["joxe", "unibertsoa", "galaxia"],
"azala": "/azalak/kaixo_unibertsoa.jpg",
"erabiltzailea": "erab2"
"igotze_data": "2010-05-14T02:15:15Z",
"iruzkin_kopurua": 5,
"gogoko_kopurua": 4,
    }]
}
```


Liburu bat lorturik, liburu hori gogoko duten erabiltzaileen zerrenda ere jaso daiteke:

```
GET /liburuak/:id/gogoko_erabiltzaileak
```

Parametroak

Izena	Mota	Deskribapena
desplazamendua	zenbakia	
muga	zenbakia	

Erantzuna


```
{
    "desplazamendua": 0,
    "muga": 25,
    "guztira": 23,
    "gogoko_erabiltzaileak": [{
"erabiltzailea": "era1",
"izena": "Era",
"deskribapena": "Erabiltzaile bat naiz",
"sortze_data": "2010-04-14T02:15:15Z"
    }, {
"erabiltzailea": "era2",
"izena": "Era bi",
```

```
"deskribapena": "Beste erabiltzaile bat naiz",
"sortze_data": "2010-04-14T02:15:15Z"
    },
    ...
    ]
}
```

(b) **Gogokoak gehitu**

Liburu bat gogokoen bilduman sartzeko.

```
POST /erabiltzaileak/:erabiltzailea/gogoko_liburuak
```

Parametroak

Izena	Mota	Deskribapena
*token	testua	Erabiltzailearen tokena

Sarrera

Izena	Mota	Deskribapena
*id	zenbakia	Liburuaren id zenbakia

```
{
    "id": "1"
}
```

Erantzuna

```
{
    "gogoko_liburua": {
"id": 1
"magnet": "magnet:?xt=urn:btih:372e39f275f3886b37e857b394e52cf73c1f9bb6"
"titulua": "Kaixo mundua",
"egileak": ["Joxe", "Patxi"],
"hizkuntza": "euskara",
"sinopsia": "Duela urte asko...",
"argitaletxea": "Etxea",
"urtea": "2009",
"generoa": "Eleberria",
"etiketak": ["kaixo", "joxe", "zaharra"],
"azala": "/azalak/kaixo.jpg",
```

```
"erabiltzailea": "erab1"
"igotze_data": "2010-04-14T02:15:15Z",
"iruzkin_kopurua": 3,
"gogoko_kopurua": 2,
    }
}
```

(c) **Gogokoak ezabatu**

Liburu bat gogokoen bildumatik kendu (id liburuarena da):

```
DELETE /erabiltzaileak/:erabiltzailea/gogoko_liburuak/:id
```

Parametroak

Izena	Mota	Deskribapena
*token	testua	Erabiltzailearen tokena

7. **Administratzaileak**

2 administratzaile mota bereizten dira: administratzaileak eta superadministratzaileak. Laburtzeko admin eta superadmin deituko zaie, hurrenez hurren. Lehenak, aurrez aipatu diren eginkizun guztiak burutu ditzake. Hau da, GET, POST, PUT eta DELETE eragiketa guztientzako baimenak ditu. Superadminek, hori guztiaz gain, administratzaileekin lotutako eginkizunetarako baimenak ditu, jarraian aipatuko direnak.

Sistema hasieratzean superadmin bat egongo da. Hau APItik kanpo sortuko da.

Adminak gehitzeko, aldatzeko edo ezabatzeko superadmin gisa saioa hasita eduki behar da.

(a) **Adminak lortu**

```
GET /adminak
```

Parametroak

Izena	Mota	Deskribapena
desplazamendua	zenbakia	
muga	zenbakia	

Erantzuna

```
{
    "desplazamendua": 0,
    "muga": 25,
    "guztira": 23,
    "adminak": [{
"erabiltzailea": "admin1",
"izena": "Admin bat",
"deskribapena": "Ezer",
"sortze_data": "2010-04-14T02:15:15Z",
"superadmin": true
    }, {
"erabiltzailea": "admin2",
"izena": "Admin bi",
"deskribapena": "Ezer",
"sortze_data": "2010-04-14T02:15:15Z",
"superadmin": false
    },
    ...
    ]
}
```

(b) **Admin lortu**

```
GET /adminak/:erabiltzailea
```

Erantzuna

```
{
    "admina": {
"erabiltzailea": "admin1",
"izena": "Admin bat",
"deskribapena": "Ezer",
"sortze_data": "2010-04-14T02:15:15Z",
"superadmin": true
    }
}
```

(c) **Admin gehitu**

```
POST /adminak
```

Parametroak

Izena	Mota	Deskribapena
*token	testua	Erabiltzailearen tokena

Sarrera

Izena	Mota	Deskribapena
*erabiltzailea	testua	Erabiltzaile izena, identifikatzailea izango dena
*pasahitza	testua	Pasahitza
*izena	testua	Izena
deskribapena	testua	Erabiltzailea deskribatuko duen testua
*superadmin	boolearra	Superadmin izango den ala ez

```
{
    "erabiltzailea": "admin2",
    "pasahitza": "1234",
    "izena": "Admin bi",
    "deskribapena": "Admin berri bat",
    "superadmin": false
}
```

Erantzuna

```
{
    "erabiltzailea": {
"erabiltzailea": "admin2",
"izena": "Admin",
"deskribapena": "Admin berri bat",
"sortze_data": "2010-04-14T02:15:15Z",
"superadmin": false
    }
}
```

(d) **Admin aldatu**

```
PUT /adminak/:erabiltzailea
```

Parametroak

Izena	Mota	Deskribapena
*token	testua	Erabiltzailearen tokena

Sarrera

Izena	Mota	Deskribapena
*pasahitza	testua	Pasahitza
*izena	testua	Izena
deskribapena	testua	Erabiltzailea deskribatuko duen testua
*superadmin	boolearra	Superadmin den ala ez

```
{
    "pasahitza": "pasahitz berria",
    "izena": "izen berria",
    "deskribapena": "deskribapen berria",
    "superadmin": false
}
```

Erantzuna

```
{
    "erabiltzailea": {
"erabiltzailea": "admin2",
"izena": "izen berria",
"deskribapena": "deskribapen berria",
"sortze_data": "2010-04-14T02:15:15Z",
"superadmin": false
    }
}
```

(e) **Admin ezabatu**

`DELETE /adminak/:erabiltzailea`

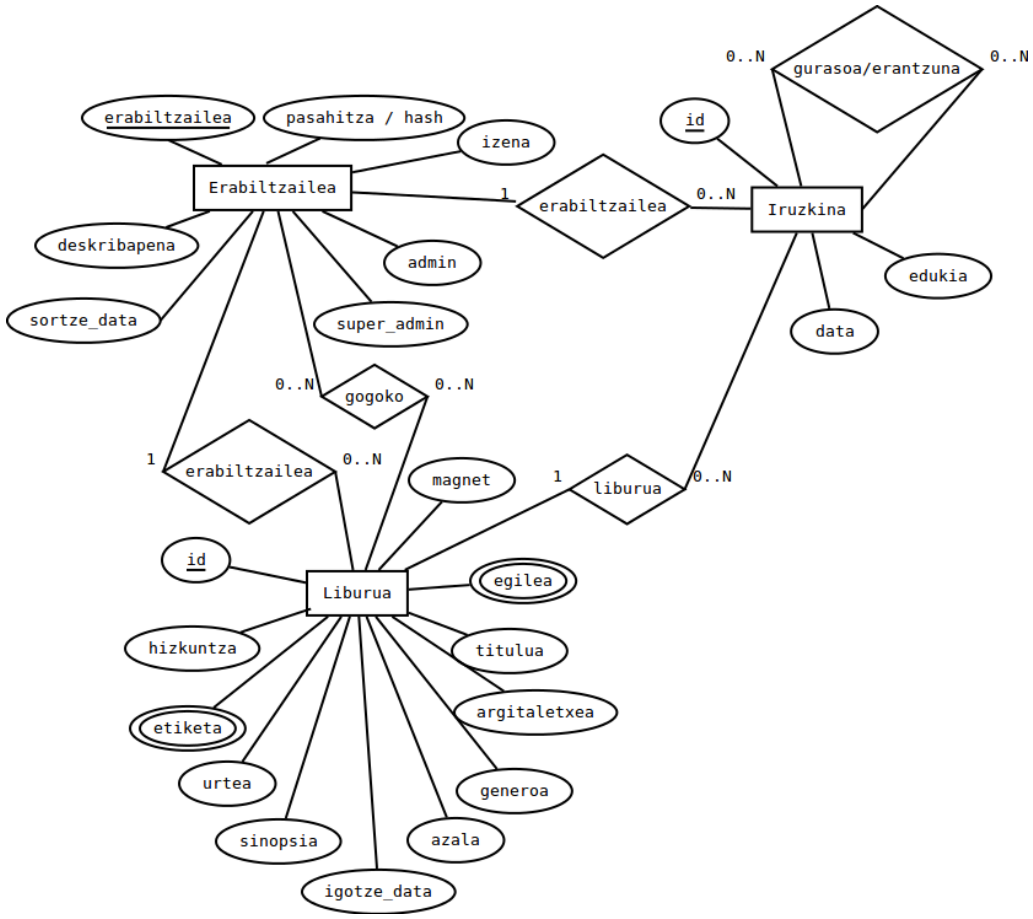
Parametroak

Izena	Mota	Deskribapena
*token	testua	Erabiltzailearen tokena

4.4 Datuen iraunkortasuna

Datuen iraunkortasuna bermatzeko datu-base erlazional bat erabili da, 4.5 irudiko entitate-erlazio diagrama jarraitzen duena.

Domeinuarekin alderatuz, ezberdintasun nagusi bat du: saioak ez dira datu-basean gordetzen, zerbitzariaren memorian baizik. Arrazoia saioen kudeaketa erraztea eta azkartzea da. Datu-basean gordez gero, tokenarekin egindako eskaera bakoitzeko (okerrak ere) datu-basea atzitu beharko litzake. Gainera iraungitutako saioak periodikoki ezabatu behar dira eta zerbitzaria arduratzen da hortaz. Erabaki horrek dakarren arrisku bat zerbitzaria eroriz gero uneko saioak galtzea da, baina eragin handia ez duela pentsatu da. Bestalde, datu guztiak iraunkortasun mailan modu berean ez gordetzeak APIaren bereizle funtzioa nabarmentzeko balio du. APIaren kontsumitzaileak ez daki, eta ez zaio axola, iraunkortasuna nola inplementatuta dagoen.



4.5 Irudia: Entitate-erlazio diagrama.

5. KAPITULUA

Inplementazioa

5.1 Tresnak eta teknologiak

Atal honetan proiektuaren garapenean erabilitako tresnak eta teknologiak azalduko dira.

1. **Hardwarea**

   Proiektuaren izaera dela eta, baliabide fisiko gutxi erabili dira. Guztira 3 ordenagailu erabili dira: ordenagailu pertsonala (finkoa), ordenagailu pertsonal eramangarria eta Raspberry Pi ordenagailua. Garapen ia guztia ordenagailu batean (finkoa edo eramangarria, unearen arabera) egin da. Fitxategien partekatzea probatzeko soilik erabili dira hiru ordenagailuak aldi berean: 2 ordenagailu pertsonalak BitTorrent bezeroekin eta Raspberry Pia tracker lana eginez. Raspberry Piaren ordez beste ordenagailu bat erabili zitekeen, eskura zegoelako erabili da.

   Azpiegiturari dagokionez, sare lokala eta Internet erabili dira. Sare lokala aurrez aipatutako partekatze probak burutzeko erabili da. Internet, proba horiek Interneteko tracker batekin burutzeko (bi ordenagailu pertsonalak erabiliz) eta tresnen kodea eta dokumentazioa lortzeko erabili da.

2. **Softwarea**

Proiektu honetako eginkizun nagusi bat software egokia aukeratzea izan da. Erabilitako software bilduma handia izanik, erabaki asko hartu behar izan dira. Aukeratzerako orduan, jarraibide orokor hauek jarraitu dira:

(a) Kode irekikoa izatea.

(b) Plataforma ezberdinetan erabili ahal izatea. Sistema eragile nagusienak hartu dira kontuan: Linux, OS X eta Windows.

Aukeratutako tresna batzuk berriak dira, eta horregatik, eragozpen batzuk dituzte. Lehenik, bertsio asko argitaratzen dira eta ez dira oso egonkorrak izaten (APIak asko aldatzen dira). Horren aurrean, bertsio jakin batzuk hautatu dira eta ez da eguneraketarik burutu, eguneratzeak kostu txikia izan duenean izan ezik. Bestetik, berria izatean, ez dira asko erabiliak izan eta horiei buruz gutxi idatzi da. Horregatik, batzuetan aurrera nola egin jakitea ez da erraza izan.

(a) **Sistema Eragilea**

Linux sistema eragilea erabili da, Debian[1] banaketa zehazki. Garapen guztia sistema horrekin egin den arren, aplikazioa Windows sistema eragilean ere probatu da. OS X-ik eskura eduki ez denez ez da bertan probarik egin.

(b) **Garapen ingurunea: GNU Emacs eta IntelliJ IDEA**

GNU Emacs[2] hedatzeko erraza den testu-editore bat da. Horregatik, hainbat eginkizunetarako erabili da:

- Kodea editatu.
- Kodea exekutatu, probak burutu.
- REPL[3] (CIDER[4] erabiliz).
- Bertsio kontrola (Git erabiliz, aurrerago azalduko da).
- Fitxategiak maneiatu.
- Atazen jarraipena burutu (org-mode[5] erabiliz).
- Memoria idatzi (org-mode eta LaTeX[6] erabiliz).

[1] `https://www.debian.org/`
[2] `https://www.gnu.org/software/emacs/`
[3] Read–Eval–Print Loop: kodea irakurri, ebaluatu eta emaitza inprimatzen duen begizta.
[4] `https://github.com/clojure-emacs/cider`
[5] `http://orgmode.org/`
[6] `http://www.latex-project.org/`

Java kodea editatzeko, berriz, IntelliJ IDEA[7] erabili da, eginkizun horretarako Emacs baino egokiagoa baita.

Esan daiteke, beraz, ia garapen guztia Emacs barnean gauzatu dela.

(c) **Web-nabigatzaileak: Mozilla Firefox eta Chromium**

Web-nabigatzailea ezinbestekoa izan da. Hasteko, informazioa lortzeko (dokumentazio guztia Interneten dago) eta, bestetik, zerbitzaria eta bezeroa probatzeko. Horretarako Mozilla Firefox[8] web-nabigatzailea erabili da. Chromium[9] web-nabigatzailea ere erabili da, bezeroaren zatia beste nabigatzaile batean probatzeko. Zerbitzariaren erantzunak hobeto ikusteko Firefoxentzako JSONView[10] gehigarria instalatu da. Bestalde, bezeroarekin lan egiterako orduan, Firefoxek dakarren Firefox Developer Tools erabili da (batez ere API deiak aztertzeko).

(d) **Bertsio kontrol sistema: Git**

Git[11] bertsio kontrol banaturako sistema da, hau da, ez du biltegi zentralizatu bat eskatzen. Antzerako sistema gehiago badaude (Mercurial, Darcs, Bazaar...), Mercurial izanik, agian, Git ondoren ezagunena. Git aukeratu da, ondorengo arrazoiengatik:

- Ezagunena da eta indar gehien duena. Zabalduta egotean jende gehiagok ezagutzen du.
- Adar lokalak sortzea merkea da eta eginkizun ez konbentzionalak burutzeko erraztasunak ematen ditu (rebase historia aldatzeko, adibidez).
- Web-tresna asko Git bidez erabiltzeko pentsatuak daude (GitHub, Travis CI).

(e) **Zerbitzariaren garapena: JDK, Clojure, H2, Transmission**

Zerbitzaria idazteko Clojure[12] lengoaia aukeratu da hainbat arrazoigatik:

- Java Makina Birtuala (JVM), Common Language Runtime (CLR) eta JavaScript motorraren gainean exekutatzen da. Zerbitzariaren kasuan JVM erabili da. Horrekin, banaketa prestatu ondoren, JVM duten plataforma ezberdinetan arazorik gabe exekutatzea lortzen da.

[7] http://www.jetbrains.com/idea/
[8] https://www.mozilla.org/en-US/firefox/new/
[9] http://www.chromium.org/
[10] https://addons.mozilla.org/en-US/firefox/addon/jsonview/
[11] http://www.git-scm.com/
[12] http://clojure.org

- Lisp motako lengoaia da eta oso espresiboa.
- Ongi diseinatuta dago.
- Balio aldaezinak (*immutable values*) eskaintzen ditu datu-egitura iraunkorren (*persistent data structures*) bidez[13].
- Programazio funtzionalera bideratuta dago. Zerbitzariak jasotzen duen eskaera bakoitza sarrera bat emanik erantzun bat itzultzen duen funtzio gisa ulertu daiteke; beraz, zerbitzariarentzako paradigma egokia da.

Zerbitzaria exekutatzeko JRE (*Java Runtime Environment*) aski den arren, garatzerako orduan JDK (*Java Development Kit*) erabili behar izan da. Horretarako OpenJDK 7[14] erabili da, sisteman horixe instalatuta baitzegoen.

Datu-base sistema gisa H2[15] aukeratu da, zeina Javaz idatzita dagoen eta aplikazioan txertatu daitekeen. Horri esker, zerbitzariak ez du datu-base sistema baten instalazioa eskatzen.

Zerbitzariak fitxategiak kanpoko aplikazio baten laguntzarik gabe partekatzea lortu nahi izan da hasiera batean. Horretarako *ttorrent* liburutegia[16] aztertu da. Liburutegi horrek, ordea, torrent fitxategi bakoitzarentzako portu bat irekitzen du eta libre dagoen porturik ez aurkitzean errorea ematen du. *ttorrent*-en liburutegia baztertuta, Transmission[17] BitTorrent bezeroa erabiltzea erabaki da, ezagutzen delako eta torrentak komando bidez gehitzeko aukera ematen duelako.

(f) **Bezeroaren garapena: ClojureScript, React, Reagent, Foundation**

Gaur egun, orri bakarreko web-aplikazioak HTML eta JavaScript bidez idatzi ohi dira (CSS itxurarentzako erabiliz). Horregatik, web-ingurunerako JavaScript frameworkak aztertu dira. Horien artean AngularJS[18] erabiltzea pentsatu da hasiera batean, Web Sistemak ikasgaiko proiektua garatzeko erabili baitzen. Baina, egia esanda, mantsoa eta konplikatuegia da burutzen duenerako. Gainera, Angular 2.0 bertsioa prototipo egoeran dago eta 1 bertsiotik asko aldentzen da. Oraindik ez dago garbi 1 bertsioa noiz

[13] Iraunkortasunak kasu honetan ez du datuak diskoan gordetzearekin loturarik. Datuaren balioa aldatuz gero, balio zaharra kontserbatzean datza

[14] `http://openjdk.java.net/`

[15] `http://h2database.com/html/main.html`

[16] `https://github.com/mpetazzoni/ttorrent`

[17] `https://www.transmissionbt.com/`

[18] `https://angularjs.org/`

zaharkituko den, baina etorkizun ez oso urrunean mantendu gabe geratzeko arriskua du. Hori guztiagatik, AngularJS ez erabiltzea erabaki da.

Aukerak aztertzean, React[19] aurkitu da eta, berria izan arren, aukera interesgarria dela ikusi da (Facebook eta Instagramek garatutakoa da). Izenean antzeman daiteken bezala, "erreaktiboa"da, interfazea modu deklaratibo batean definitzen da eta datuak aldatu ahala eguneratzen da.

Om[20] liburutegiari esker, ClojureScript[21] bidez React erabili daiteke. Bezeroan ClojureScript erabiliz gero, zerbitzari eta bezeroa lengoaia berdinarekin garatu daitezke. Hori ikusirik, bide ClojureScript-en bidea jarraitzea erabaki da. Halere, tresna ahaltsua izan arren, Om nahiko konplexua da. Horren ordez, beste liburutegi bat erabili da: Reagent[22]. Reagent-ek, Om-ek bezala, React-en gaineko interfaze bat eskaintzen du, baina erabiltzeko sinpleagoa da.

Bezeroa zuzenean JavaScriptez idatzi zitekeen, baina ClojureScript bidez idaztea erabaki da, hiru arrazoirengatik:

i. Zerbitzarian jada Clojure erabiltzea pentsatuta zegoen. ClojureScriptekin bi zatietan lengoaia bera erabil daiteke.
ii. Clojureren datu aldaezinak Reacten arkitekturarentzako egokiak dira.
iii. Ez da JavaScript. JavaScript orain dela 20 urte, 10 egunetan, sortutako lengoaia da. Hobekuntzak egin diren arren, oraindik ez da lengoaia oso "txukuna".

Reagent-en bidezko bezeroaren garapenari Figwheel[23] tresna gehitu zaio. Tresna horrek kodea, aldatu ahala, nabigatzailean automatikoki kargatzen du, programa berrabiarazi izan behar gabe. Hau da, aldaketak nabigatzailean berehala ikus daitezke, REPL batean egongo bagina bezala. Lan egiteko modu hori oso lagungarria izan da eta interfaze grafikoa garatzeko prozesua asko erraztu du.

Garapena errazteko erabili den beste tresna bat Foundation[24] *front-end framework*-a izan da, *responsive* diseinua duelako eta lehendik ezagutzen zelako.

[19] `http://reactjs.com/`
[20] `https://github.com/swannodette/om`
[21] ClojureScript konpilatzaile bat da, Clojuretik JavaScriptera konpilatzen duena. Lehen esan bezala, Clojure JavaScripten gainean erabili daiteke.
[22] `https://reagent-project.github.io/`
[23] `https://github.com/bhauman/lein-figwheel`
[24] `http://foundation.zurb.com/`

(g) **Clojure garapena: Leiningen**

Zerbitzaria eta bezeroa Clojure proiektuak izanik, horiek kudeatzeko Leiningen[25] erabili da. Tresnak horrek garapeneko hainbat prozesu errazten ditu: dependentziak definitu eta instalatu, programa abiarazi, probak gauzatu, REPL abiarazi, banaketa prestatu...

(h) **Fitxategiak partekatzea: Opentracker eta FrootVPN**

BitTorrent sareetan, trackerrek fitxategiak partekatzen laguntzen dute. Sare lokalean tracker bat edukitzeko Opentracker[26] erabili da.

Internet bidez partekatzerako orduan, berriz, arazo bat aurkitu da: bi ordenagailuek kanpoko IP berdina edukitzea toki berdinetik konektatuta egoteagatik. Bi IP ezberdinen arteko konexioa edukitzeko FrootVPN[27] VPN[28] zerbitzua erabili da. Tresna horren laguntzarekin, ordenagailu batekin berezko konexioa erabili da eta bestearekin VPN. Lehenbizi EHUko VPN zerbitzua probatu da, baina sare horretan BitTorrentek ez zuen funtzionatzen.

(i) **Web-baliabideak**

Kode guztia GitHub-era[29] igo da eta zerbitzariko probak Travis CI[30] bidez exekutatu dira. Bi zerbitzu horiek doako planak dituzte.

(j) **Diagramak: PlantUML, Ditaa, Dia, GanttProject**

- PlantUML[31]: UML diagramak sortzeko.
- Ditaa[32]: diagramak sortzeko.
- Dia[33]: diagramak sortzeko.
- GanttProject[34]: Gantt diagrama sortzeko.

[25] `http://leiningen.org/`
[26] `https://erdgeist.org/arts/software/opentracker/`
[27] `https://www.frootvpn.com/`
[28] Virtual Private Network: sare pribatu birtuala.
[29] `https://github.com/`
[30] `https://travis-ci.org`
[31] `http://plantuml.com/`
[32] `http://ditaa.sourceforge.net/`
[33] `https://wiki.gnome.org/Apps/Dia`
[34] `http://www.ganttproject.biz/`

5.2 Zerbitzaria

1. **Probek bideratutako garapena**

   Kalitatearen kudeaketa-planean zerbitzaria TDD bidez garatzea erabaki da. TDD (Test-Driven Development) softwarea garatzeko prozesu bat da. Prozesu horretan garapen-ziklo motzak errepikatzen dira. Ziklo bakoitzak honako sekuentzia jarraitzen du:

   (a) Proba bat idatzi. Ezaugarri berri bat garatu nahi denean, lehenengo ezaugarri horri dagokion proba idazten da. Horretarako, ezaugarriaren eskakizunak eta espezifikazioak ongi zehaztuta egon behar dira.

   (b) Probak exekutatu eta ezaugarri berriari dagokion probak huts egiten duela ikusi.

   (c) Proba gaindituko duen kodea idatzi.

   (d) Probak exekutatu. Proba berria gainditzeaz gain, lehendik idatzitako probek ere ez dute huts egin behar. Era horretan, ezaugarri berriak sistemaren gainontzeko zatiei ez diela eragin ziurtatzen da.

   (e) Kodea berrantolatu. Kodea hobetu eta txukuntzen da. Berrantolatu ondoren, probak berriz exekutatzen dira funtzionalitatean eragin ez dela ikusteko.

   TDD unitate txikiak (klase bat edo erlazionatutako funtzio batzuk) probatzeko erabili ohi da. Proiektu honetan, ordea, zerbitzari osoa probatzeko erabili da. Horretarako, probetan, API deien bidez zerbitzariarekin komunikatu da. APIa aurrez diseinatu denez, APIaren dokumentazioa erabili da gida gisa.

   Probak Clojure lengoaian idatzita daude eta Midje[35] bidez exekutatzen dira. Probak isolatuta burutu behar direnez, proba bakoitzaren aurretik datu-basea garbitu eta zerbitzaria hasieratzen da. Modu horretan, proba bat exekutatu aurretik, zerbitzaria beti hasierako egoeran dagoela ziurtatzen da.

   Hasieraketa horren eta proba asko egotearen ondorioz, proba guztiak exekutatzea ez da berehalako prozesua. Garapen-zikloa arintzeko, probak moten arabera (saioak, liburuak, iruzkinak...) sailkatu dira. Horrez gain, uneko proba ere bereizi da. Ezaugarri bat garatzerako orduan, lehenenik ezaugarriari dagokion proba exekutatu da. Jarraian, ezaugarri motari dagozkien probak, eta azkenik,

[35] Clojurentzako proben *frameworka*.

proba guztiak. Horri esker, ezaugarriari dagokion probak huts egiten duenean, ez da beharrezkoa horren aurretik idatzita dauden proba guztiak exekutatu arte zain egotea.

2. **APIaren inplementazio maila**

   Proiektuari eskaintzeko denbora mugatua izanik, eta, APIa zabala dela kontuan edukiz, ezin izan da API guztia inplementatu. Ondorengo zatiak geratu dira inplementatu gabe:

   - Liburuak parametroen bidez lortzea. Parametroak honakoak dira: egileak, argitaletxeak, generoak, etiketak, urtea eta hizkuntza. Parametroak gehitzeak, eta horiek nahasteko aukera eskaintzeak, kodea gehiago adarkatuko luke. Inplementazioak erabaki gehiago hartu beharko lituzke, baina, horrez gain, ez da lan konplikatua.
   - Administratzaileak eta superadministratzaileak. Bezeroan administratzailerik ez gehitzea erabaki denez, inplementazioan zati hori ez da kontutan eduki. Lan gehiago eskatuko luke, baina inplementatzeko erraza da (erabiltzaileak moten arabera bereiztea eskatuko luke).

3. **Zerbitzariaren funtzionamendua**

   Zerbitzaria Clojure lengoaiaz idatzita dago eta Java Makina Birtualaren gainean exekutatzen da. Sistema eragileko erabiltzaile arrunt gisa abiarazten da eta, behin martxan jarrita, prozesua ez da amaitzen.

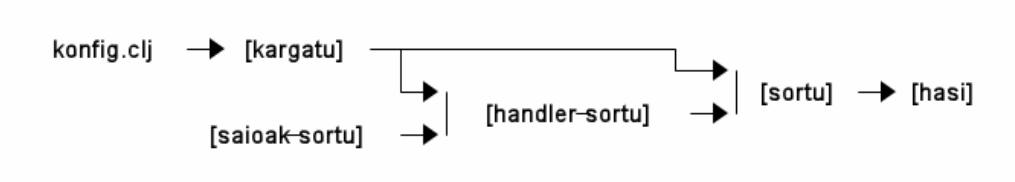
   Aplikazioak ez du ia egoerarik gordetzen (datu-basea kontuan hartu gabe) eta gordetzen dena toki jakin batean, erroan, dago. Egoera hori eta konfigurazioa argumentu gisa pasatzen da barneko funtzioetara, modu esplizituan. Horrela, funtzioek behar duten guztia argumentuen bidez jasotzen dute eta erantzun bat itzultzeaz soilik arduratzen dira.

   Aplikazioa abiarazteko orduan hainbat pauso burutzen dira (5.1 irudia). Lehenik, konfig.clj fitxategitik konfigurazioa lortzen da[36], eta, hori erabiliz, zerbitzaria sortzen da. Horretarako, lehenbizi handler-a[37] eta saioen osagaia (saioak

[36] Aurrerago aipatuko den gisa, konfiguratzeko aukera gehiago daude.
[37] Bideen kudeatzailea, eskaerekin zer egin erabakitzen duena.

gordetzeko) sortzen dira. HTTP zerbitzariarentzako[38] tokia ere prestatzen da. Gero zerbitzaria[39] abiarazten da, modu honetan:

- ";; Zerbitzaria [portu-zenbakia] portuan abiarazten"mezua erakusten du
- HTTP zerbitzaria abiarazten du eta horren erreferentzia gordetzen du. HTTP Zerbitzariak lehen sortutako handlerra erabiltzen du.
- Konfigurazioan partekatzea gaitu bada, torrent karpetako fitxategiak partekatzeko eskatuko dio torrent-gehitze-programari (konfigurazioan zehazten da).



5.1 Irudia: Zerbitzaria abiarazteko prozesua.

Zerbitzaria geratuz gero (modu interaktiboan, REPL bidez, bakarrik), berriz, sortu ondoren zuen egoerara pasatzen da:

- ";; Zerbitzaria geratzen"mezua erakusten da
- HTTP zerbitzaria geratzen da.

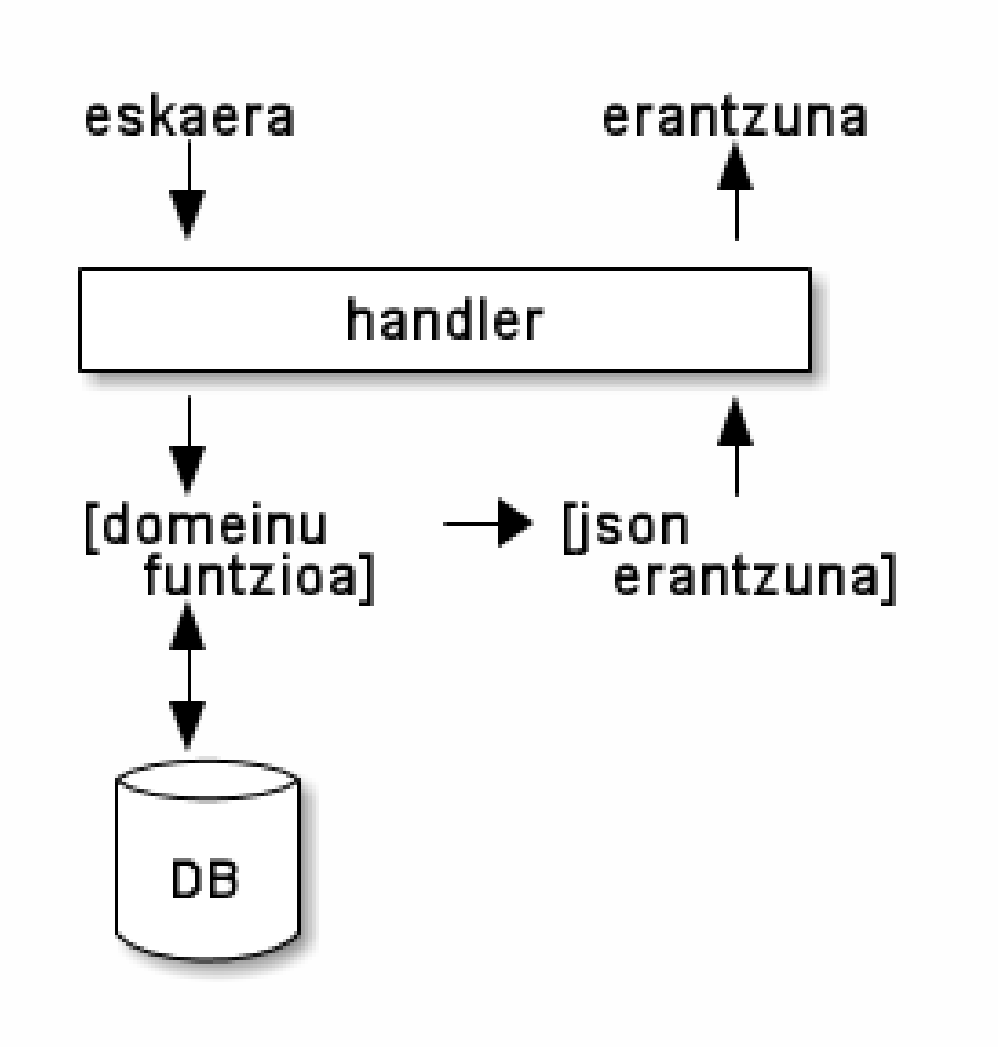
Beraz, aplikazioa martxan dagoen bitartean, HTTP zerbitzari bat martxan dago. Zerbitzari horrek konfigurazioan adierazitako portuko eskaerei erantzuten die. Lehen aipatutako handlerra da eskaerekin zer egin erabakitzeaz arduratzen dena. handler.clj fitxategiko *bideak-sortu* funtzioan daude bide guztiak; hau da, eskaera bat iristean, bide horietako batetik pasatzen da. Bide guztiek egitura berdina dute (*hainbat* motakoek sintaxi ezberdina dute, baina oinarrian berdinak dira) eta *api-erantzuna* makro laguntzailearen bidez definitu dira. *api-erantzuna*-k Clojure datu-egiturak JSON-era pasatzen ditu, eta, behar izanez gero, egoera zenbakia eta gorputza gehitzen ditu.

Bideek domeinuarekin lotutako funtzioei deitzen diete, ondorengo fitxategietan definituta daudenak: erabiltzaileak.clj, hainbat.clj, iruzkinak.clj, liburuak.clj eta saioak.clj. Funtzio horiek domeinuko eragiketak gauzatzen dituzte, gehienetan

[38] HTTP Kit erabili da, baina antzerako beste zerbitzari bat erabili daiteke, Jetty adibidez.

[39] "Zerbitzaria"aipatzean aplikazioari egiten zaio erreferentzia. "HTTP zerbitzaria", berriz, aplikazioak, bere barne, erabiltzen duen HTTP zerbitzaria izendatzeko erabiltzen da.

datu-basea atzituz. Domeinuko funtzioek itzultzen dituzten erantzunek bi osagai dituzte: egoera eta datuak (datuak hautazkoa da). Saioei dagokienez, beste atal batean aipatu den bezala, ez dira datu-basean gordetzen. Hasieran sortutako saioak osagaia-ren gainean egiten da lan (hash taula bat da). 5.2 irudiko diagraman azaltzen da funtzionamendu orokorra.



5.2 Irudia: Eskaerei erantzuteko prozesua.

Liburu bat gehitzerako orduan, datu-basean atzitzeaz gain, fitxategiekin ere lan egiten da. liburuak.clj fitxategiko *liburua-gehitu!* funtzioa arduratzen da eskaera horiei erantzuteaz. Bertan, prozesu hau jarraitzen da:

- Liburu berriaren datuak datu-basean gorde.
- Epub fitxategia sortu.

- Torrent fitxategia sortu eta dagokion magnet lotura eskuratu.
- Magnet lotura datu-basean gorde.
- Partekatzea gaituta badago torrent fitxategia erabiliz liburua partekatu.
- Azala fitxategia sortu.
- Azala datu-basean gorde.
- Liburuaren datuak itzuli.

Torrent fitxategia sortzeko eta partekatzeko torrent.clj moduluaren laguntza erabiltzen da. Horrek Javaz idatzi den zati bakarra erabiltzen du: *com.magnet/Torrent* klasea, Torrent.java fitxategian dagoena. 5.3 irudian liburua gehitzeko prozesua azaltzen da.

4. **Fitxategiak partekatzea**

   Fitxategiak partekatzeko, torrent fitxategiak sortu eta horietatik magnet loturak ateratzea lortu behar da. Horretarako, lehenik, torrent fitxategien egitura aztertu da. Kodetze kontuetan sartu gabe, hash taula gisako egitura dute, ondorengo eremuekin:

   - announce: trackerraren URLa.
   - info: info zatia duen egitura, ondorengo eremuekin:
     - name: fitxategiaren izena.
     - length: fitxategiaren tamaina.
     - piece length: zati bakoitzaren tamaina[40].
     - pieces: zati bakoitzaren SHA1 hash balioa, zerrenda batean.

   Eremu gehiago egon daitezke, BitTorrentek hedapen asko baititu[41], baina horiek aski dira fitxategi bat partekatzeko. Torrent fitxategiak probatzeko, fitxategi horiek Transmission BitTorrent bezeroarekin ireki dira eta formatu egokia zuten begiratu da.

   Behin torrent fitxategia sortzeko modua edukita, magnet loturak sortzea ez da horren zaila: info zatiaren SHA1 hash balioa kalkulatu behar da. Balio horri

[40] BitTorrent protokoloan fitxategiak piezaka partekatzen dira.

[41] Hobetzeko proposamenak BEP (BitTorrent Enhancement Proposals) izeneko dokumentuetan biltzen dira. Asko oraindik estandarizatu gabe daude: `http://www.bittorrent.org/beps/bep_0000.html`

BTIH deritzo (BitTorrent Info Hash) eta magnet loturaren oinarria da. Loturari fitxategiaren izena eta trackerrak (bat baino gehiago erabili daiteke) gehitzea dago, modu honetan:

```
"magnet:?xt=urn:btih:" + infoHash + "&dn=" + izena \
+ "&tr=" + tracker1 + "&tr=" + tracker2 + [...]
```

Konfigurazioan trackerren zerrenda zehazten da. Zerrendako lehen trackerra torrent fitxategiaren tracker gisa ezartzen da. Gainontzekoak magnet loturan gehitzen dira, tracker nagusiarekin batera. Torrent fitxategian tracker bakarra gehitzen denez, magnet loturako gainontzeko trackerrak ez dira beharrezkoak; baina, posible da norbaitek torrent horrekin tracker gehiago erabiltzea. Horregatik, badaezpada ere, aukera guztiak gehitu dira.

Fitxategiak magnet loturen bidez partekatzen asmatzea ez da erraza izan. Ordenagailu batean zerbitzaria martxan ipini da, fitxategia partekatzen, eta beste ordenagailu bat fitxategia eskuratzeko erabili da, zerbitzariak adierazitako magnet loturaren bidez. Hasieran, sare lokalean zeuden 2 ordenagailurekin probatu da, baina ez ziren beraien artean komunikatzen. Zerbitzaria Interneteko tracker bat erabiltzeko konfiguratuta zegoenez, tracker horrekin komunikatzen ziren. NAT erabiltzearen ondorioz kanpora begira biek IP berdina zuten eta ez ziren sare lokalean elkar komunikatzen.

Arrakastarik gabeko proba asko egin ondoren, 3. ordenagailu bat tracker gisa erabiltzea pentsatu da. Horrela, Raspberry Pi batean Opentracker instalatu da. Jarraian, zerbitzariaren konfigurazioa aldatu da, torrentetan tracker hori azaltzeko (helbide lokala erabiliz). Era horretan, sare lokalean fitxategiak partekatzen zirela ikusi da.

Oraindik fitxategiak Internet bidez partekatu zitezkeen probatzea falta zen. Horretarako, tokiz mugitzen ibili beharrean, ordenagailu batean FrootVPN zerbitzua erabili da. Modu horretan, bi IP helbide ezberdinetatik partekatzea probatu da eta ongi zebilela ikusi da. Amaitzeko, kasu errealago bat probatzeko, ordenagailu bat partekatzen piztuta utzi da eta, fisikoki beste sare batetik konektatuta, fitxategia eskuratzea posible dela berretsi da.

5. **Garapen ingurunea**

   Zerbitzaria garatzeko orduan Leiningen tresna erabili da. Leiningenek 2 JVM abiarazten ditu: bat bere kodearentzako eta bestea proiektuaren kodearentzako.

Gainera, Clojure hasieratu behar du. Horren ondorioz, komando lerrotik *lein* komandoak[42] abiaraztea motela da eta etenaldi ezerosoak egin behar dira, denbora galduz.

Garapena azkartzeko eta erosoagoa izateko, lan saio bakoitzeko, Emacs barnean, REPL bat ireki da. Bertan, aurrez aipatu den moduan, zerbitzaria sortu, hasi eta geratu daiteke, konfigurazioa aldatuz nahi bada[43]. Leiningen bidez komando lerrotik burutzen diren eragiketa guztiak ere REPL barnetik exekutatu dira: probak exekutatu, dokumentazioa sortu, banaketa prestatu... Gainera, aplikazioa martxan dagoela kodea aldatu eta ebaluatu daiteke, aldaketak momentuan aplikatuz. Hori guztia lan egiteko orduan lagungarria izan da, ataza horiek burutzeko prozesua azkartu baitu.

5.4 irudian ikus daiteke garapeneko momentu bat: goian kodea editatzen, behean zerbitzaria REPL bidez abiarazten eta eskuinean torrentak gehitu direla adierazten duten mezuak.

Aplikazioaren ezaugarri berri bat idatzitakoan, edo aldaketak egin ondoren, probak exekutatu dira (*lein midje* komandoaren bidez). Proba guztiak gaindituz gero, aldaketak GitHubeko biltegira bidali dira. Travis CI zerbitzuak biltegi horretako kodea aldizka begiratzen du, eta, aldaketa berri bat dagoenean (*commit*) probak exekutatzen ditu. Proben emaitzak errazago bistaratzeko, GitHubeko proiektuaren orrian Traviseko proben egoeraren berri ematen duen botoi bat gehitu da. Orri hori noizean behin bisitatu da, probak ongi exekutatu diren begiratzeko.

Travisek probak hainbat ingurunetan exekutatzea ahalbideratzen duenez, posible da proba lokalekin arazorik ez edukitzea baina Travisen probak ez gainditzea. Hain zuzen, horixe gertatu da. Hasiera batean ondorengo inguruneak zehaztu dira: OpenJDK 6, OpenJDK 7, OracleJDK 7 eta OracleJDK 8. Baina torrentak sortzearen zatia gehitu denean, Traviseko egoeraren botoia gorriz zegoela ikusi da. Arrazoia Java NIO erabiltzea izan da. Java NIO JDK 7 bertsioan gehitu zen. Garapeneko makinan OpenJDK 7 instalatuta zegoenez, probek ez zuten arazorik ematen. Traviseko OpenJDK 6, ordea, ez zen kodearekin bateragarria. Hori ikusirik, aplikazioa JDK 6-rekin ez dela bateragarria ondorioztatu da, ingurune hori instalatu behar izan gabe.

[42] Leiningen *lein* izeneko script baten bidez abiarazten da.

[43] Aplikazioa hasieratzean konfigurazioa fitxategitik irakurtzen da, baina REPL barnean konfigurazioa gordetzen duten datu-egitura alda daiteke zerbitzaria abiarazi aurretik.

6. **Kodearen antolaketa**

   Zerbitzaria osatzen duten fitxategiak modu honetan daude antolatuta:

   - files/: fitxategi pribatu eta publikoak dituen karpeta.
   - src/: zerbitzariaren kodea
     - cljs/magnet/: Clojurez idatzitako kodea
       * core.clj: sarrera, main funtzioa bertan dago.
       * erabiltzaileak.clj: erabiltzaileekin lotutako funtzioak.
       * hainbat.clj: hainbat datu eskuratzeko funtzioak.
       * handler.clj: aplikazioaren bideak zehazten ditu, eskaerak eta erantzunak prozesatuz, eta domeinuko funtzioak erabiltzen ditu.
       * iruzkinak.clj: iruzkinei lotutako funtzioak.
       * lagun.clj: funtzio laguntzaileak (datu-basea hasieratu, denborarekin lotutakoak, orriztatzea...).
       * liburuak.clj: liburuekin lotutako funtzioak.
       * saioak.clj: saioekin lotutako funtzioak.
       * torrent.clj: torrent fitxategiei lotutako funtzioak.
       * zer.clj: zerbitzaria sortu, hasi eta geratzeko funtzioak.
     - java/com/magnet/Torrent.java: torrent fitxategiak eta horien magnet loturak sortzeko kodea (Java).
   - test-files/: probetarako fitxategiak gordeko dituen karpeta.
   - test/magnet/t_api.clj: proben kodea.
   - .gitignore: bertsio-kontrolean sartu behar ez diren fitxategiak adierazten ditu.
   - .travis.yml: Travis CI bidez probak nola exekutatu behar diren zehazten duen fitxategia.
   - LICENSE: kodearen lizentzia.
   - README.md: dokumentazioa.
   - before_script.sh: Travis CI bidez probak baino lehen exekutatuko den scripta.
   - konfig.adb.clj: konfigurazio adibidea.
   - proba-konfig.adb.clj: proben konfigurazio adibidea.

- project.clj: proiektuaren konfigurazioa, Leiningen-ek erabiltzen duena.
- test-azala.jpg: probetarako azala.

5.3 Bezeroa

1. **Bezeroaren funtzionamendua**

   Bezeroa HTML, CSS, JavaScript eta ClojureScript-ez osatua dago (ClojureScript, erabili aurretik, JS[44]-era konpilatzen da). index.html web-nabigatzaile batekin irekiz gero oinarrizko HTML, CSS eta JS laguntzailea, eta, aplikazioaren JS kodea kargatzen da. Zerbitzariaren antzera, bezeroarekin ere konfigurazio fitxategi bat (konfig.js) erabiltzea erabaki da. Fitxategi hori index.html orrian kargatzen da, aplikazioaren kodea baino lehen, eta bertako balioak aplikazioari (magnet.core.run) pasatzen zaizkio argumentu gisa.

   Bezeroak zerbitzariak baino egoera gehiago gordetzen du eta eskaera->erantzuna baino egitura konplikatuagoa du. Aplikazioaren egoera magnet.core moduluaren hasieran definituta dago, hainbat aldagaitan banatuta. Jarraian, aldagai horiekin eta APIarekin lan egiten duten funtzioak daude. Gero kudeatzaileak datoz; bideen, saioen, liburuen eta iruzkinen gertaerekin zer egin erabakitzen dutenak. Azkenik *errendatu* eta *run* funtzioak datoz, gero esplikatuko direnak.

   Aplikazioren sarbidea *run* funtzioa da eta pauso hauek jarraitzen ditu (5.5 irudia):

   - Jasotako argumentuekin konfigurazioa ezarri.
   - Kudeatzaile bakoitzarentzako kanal[45] bat sortu.
   - Errendaturi deitu (aurrerago azalduko da).
   - Kanal bakoitzean entzuten jarri eta zerbait jasotakoan kudeatzaileari deitu.
   - Bidea ezarri (index).
   - Liburuak lortu.
   - Azken gogokoena lortu.
   - Azken iruzkinak lortu.

[44] JavaScript, JS gisa laburtu ohi da.

[45] Kanalak Clojure core.async kanalak dira: `http://clojure.com/blog/2013/06/28/clojure-core-async-channels.html`

- Figwheel erabiltzea erabaki bada aldaketak daudenean eguneratu (garapenarekin lotuta, hurrengo atalean azalduko da).

Beraz, hasieratu ondoren hainbat datu lortuta egongo dira: liburuak, azken gogokoena eta azken iruzkinak. Jarraian zer egin gertaeren arabera erabakitzen da. Horretarako 4 kanal daude: saioena, bideena, liburuena eta iruzkinena. Kanal horiek jasotzen dutenaren arabera, kudeatzaile batek edo besteak erabakitakoa egingo da (kanal bakoitza kudeatzaile bati lotuta dago). Adibidez, bideak definitzean, bide bakoitzeko identifikatzaile bat zehazten da eta, bide horrekin topo egindakoan, identifikatzailea bideen kanalean sartzen da. Balio hori bideen kudeatzaileak hartzen du eta, balioaren arabera, ekintzak gauzatzen ditu.

Orain arte azaldutakoarekin, ordea, ez dago interfaze grafikorik, datuen errepresentaziorik. Hortaz arduratzen da *errendatu*. Funtzio horrek Reagenti *bistak/main* osagaia errendatzeko eskatzen dio. Osagai horrek bi motatako argumentuak jasotzen ditu:

- Kanalak: aurrez aipatutako 4 kanalak. Osagaiek, gertaera bat bultzatu nahi dutenean, bertan idazten dute.
- Datuak: errendatu nahi diren datuak.

bistak/main funtzio bat da, argumentu berdinekin beti interfaze bera itzultzen duena. Bere egitura berdina duten hainbat osagai ditu eta horiek ere hainbat argumentu jasotzen dituzte, datu guztiak modu esplizituan pasaz. Erabaki horren ondorioz, osagai batzuk argumentu asko dituzte eta horiek behin eta berriz pasa behar dira osagaira iritsi arte. Hau da, zuhaitz baten moduan antolatuta dago, eta datuak errotik hostoetaraino pasatzen dira (5.7 irudia).

Hainbeste aldagai eduki ordez, aldagaiak taldekatuz gero txukunago geratuko litzake, baina erabaki horrek arrazoi bat badu: Reagentek datuak errendatzeko modua. Osagai batek errendatzeko jasotzen dituen balioak atom[46] motakoak dira. Atom bat aldatzean, atom horren balioa erabiltzen duen osagaia bakarrik errendatzen da. Osagai batek hainbat datu biltzen dituen atom bat jasoko balu, datu horietako bat aldatzearekin batera osagaia osorik errendatu beharko litzateke, osagaiak datu hori erabiltzen ez badu ere (beste osagai batentzako

[46] Erreferentzia motako balioak dira: `http://clojure.org/atoms`

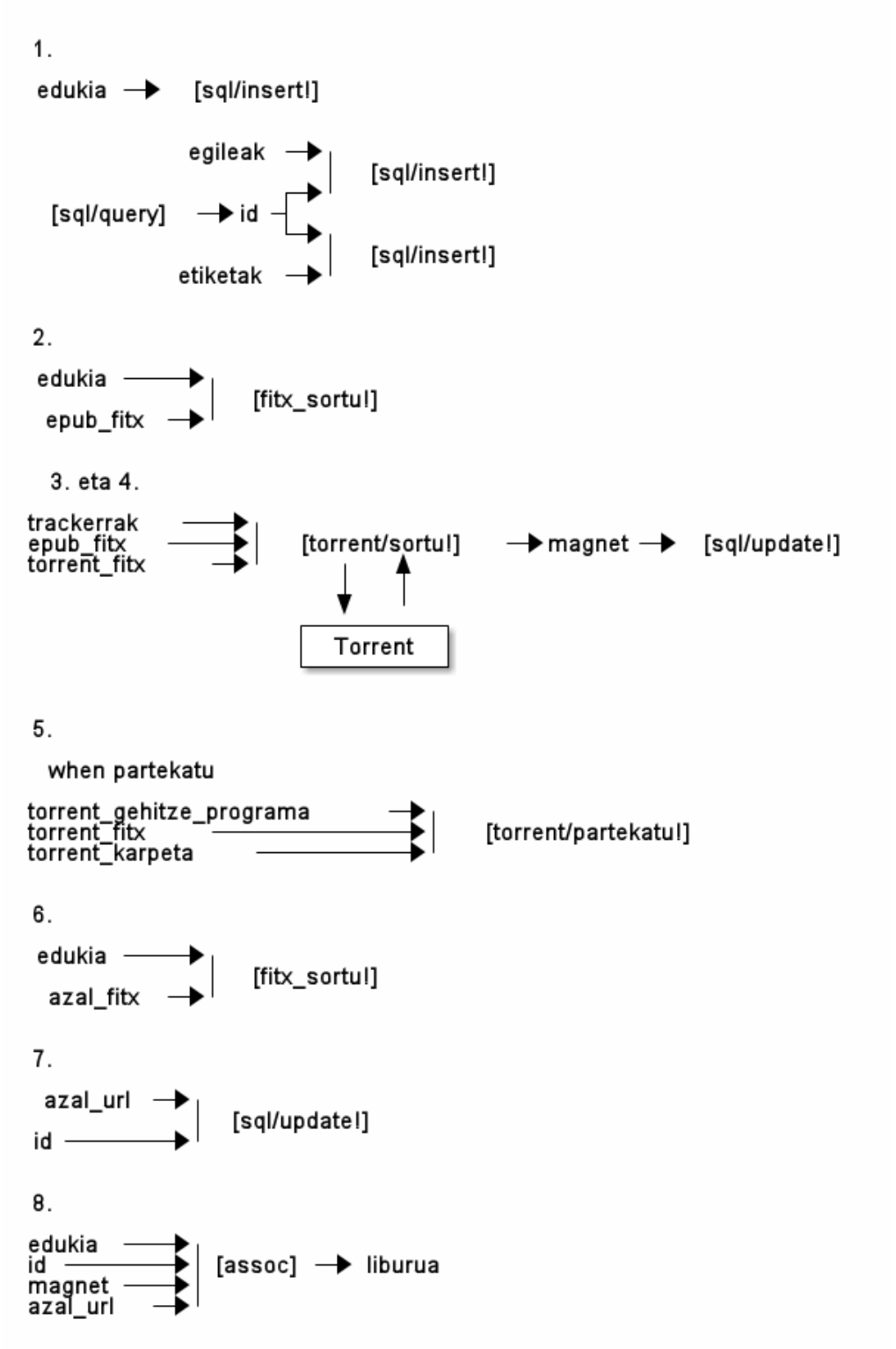
izan daiteke). Horregatik, osagaiak alferrik ez errendatzeagatik, modu horretan antolatu da.

2. **Bezeroaren garapena**

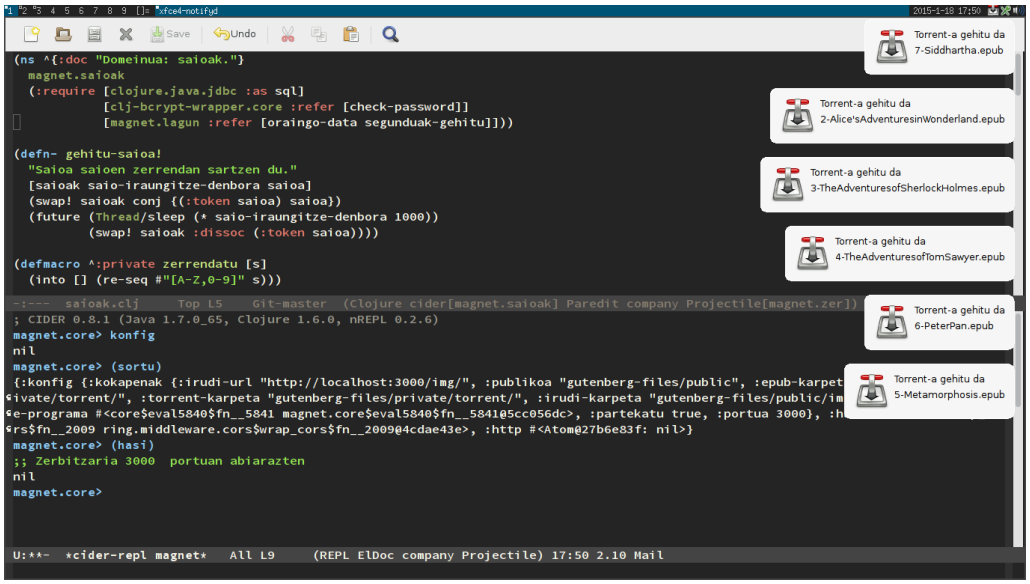
   Interfazearen garapena errazteko Foundation frameworka erabili da. Era horretan, interfazearen HTML egitura bakarrik idatzi da eta ez da CSSrik idazteko beharrik eduki. Foundation, gainera, *responsive* motako webguneak garatzeko pentsatua dago. Horri esker, erraza da tamaina ezberdinetako pantailetara moldatzea.

   Garapen prozesuan eta banaketan bezeroaren erabilera ezberdina izan da. Zerbitzariarekin bezala, garatzerakoan, lagungarriak diren tresnak erabili dira. Alde batetik, CSS eta JS dependentzia trinkotu gabeak erabili dira (banaketan foundation.min.css edo react-0.9.0.min.js gisako fitxategiak daude), arazketa lanak errazteko helburuarekin. Bestetik, fitxategiak zuzenean edo web-zerbitzari baten bidez kargatu ordez figwheel tresna erabili da.

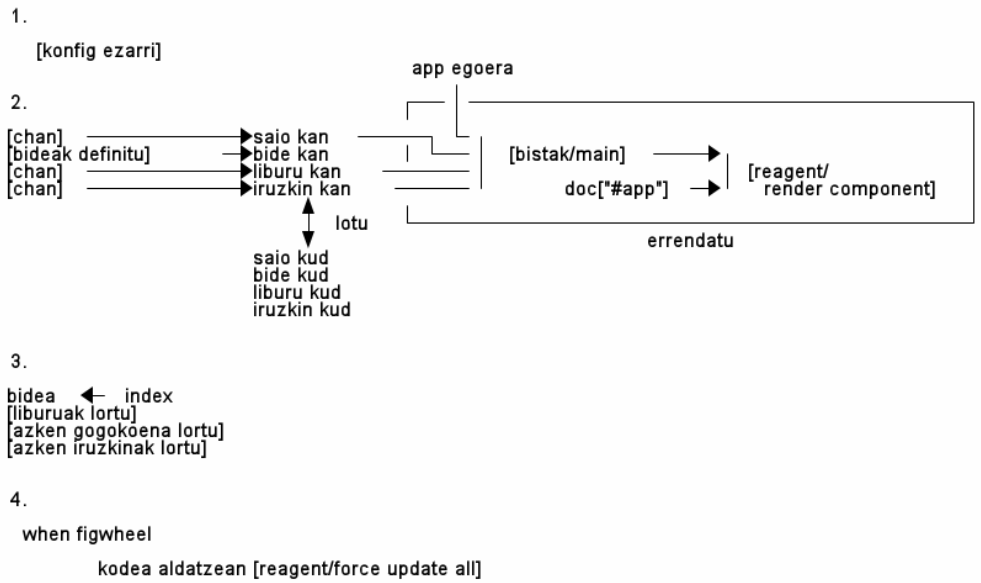
   Figwheelek kodea eta CSS fitxategiak automatikoki kargatzen ditu, orria berritu behar izan gabe. Hau da, kodean aldaketa bat burutuz gero, aldaketa hori segituan nabigatzaileari pasatzen zaio. 5.8 irudian fighweel martxan ikus daiteke kodean egindako aldaketa nabigatzaileari pasatzen. Horretarako, aplikazioaren egoera duten aldagaiak *defonce*-rekin definitu dira (normalean *def* erabiltzen da). *defonce*-k aldagaia behin bakarrik definitzen du, eta, aplikazioa martxan dagoela kodea berriz ebaluatzen bada, ez du aldagaiaren balioa berriz definitzen. Horri esker, modu interaktiboan programatu daiteke -zerbitzarian REPLrekin egiten den antzera- aldaketak unean ikusiz. Garapena modu horretan egiteak interfazea eraikitzea asko erraztu du. Izan ere, aldaketak berehala ikusi dira, orria berriz kargatzean denbora eta datuak galdu gabe.



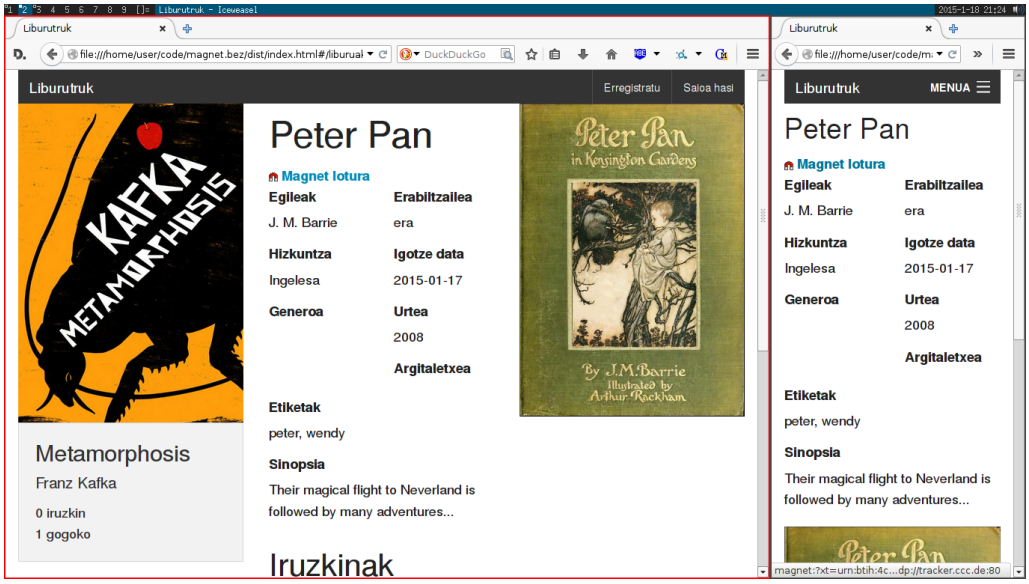
5.3 Irudia: *liburua-gehitu!* funtzioak jarraitzen dituen pausoak.



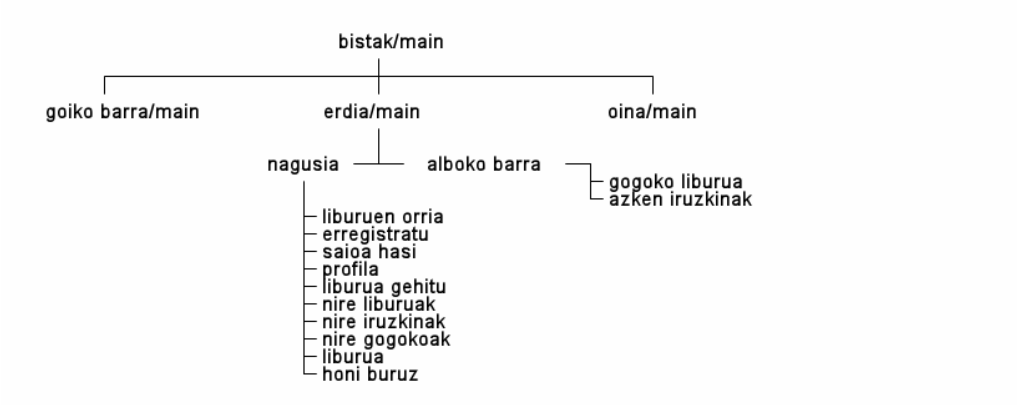
5.4 Irudia: Garapeneko irudi bat.



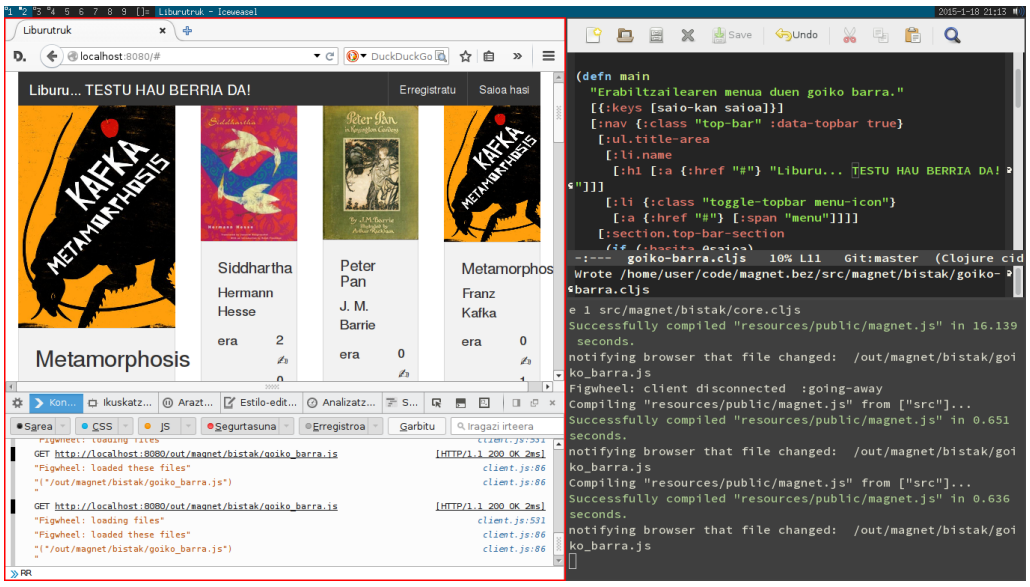
5.5 Irudia: Bezeroa abiarazteko pausoak.



5.6 Irudia: Bezeroa: tamaina ezberdineko bi leiho.



5.7 Irudia: Bezeroaren interfazearen banaketa.



5.8 Irudia: Bezeroa garatzen.

3. **Kodearen antolaketa**

   Bezeroa osatzen duten fitxategiak modu honetan daude antolatuta:

   - dist/: banaketarako prestatutako fitxategiak.
   - resources/public/: garapenerako prestatutako fitxategiak.
   - src/magnet/: ClojureScript kodea
     - bistak/: aplikazioaren bistak, modu honetan banatuta:
       * erdia/: erdiko zatia, eduki nagusia:
         · alboko-barra.cljs: alboko barra, gogoko liburua eta azken iruzkinak.
         · liburua-gehitu.cljs: liburua gehitzeko formularioa.
         · liburua.cljs: liburuaren datuak eta iruzkinak.
       * core.cljs: nagusia, sarrera.
       * erdia.cljs: erdiko zatia.
       * goiko-barra.cljs: goiko barra, menua duena.
       * lagun.cljs: funtzio laguntzaileak.
       * oina.cljs: webgunearen oina.
     - bideak.cljs: bideak definitzen ditu.
     - core.cljs: aplikazioaren muina.
     - lagun.cljs: funtzio laguntzaileak (*levenshtein*)
     - makroak.clj: bideak eraikitzeko makroak.
     - makroak.cljs: makroen modulua definitzeko fitxategia.
   - test/magnet/core_test.clj: probak (ez da erabili).
   - .gitignore: bertsio-kontrolean sartu behar ez diren fitxategiak adierazten ditu.
   - LICENSE: kodearen lizentzia.
   - README.md: dokumentazioa.
   - konfig.js: konfigurazio adibidea.
   - project.clj: proiektuaren konfigurazioa, Leiningen-ek erabiltzen duena.

6. KAPITULUA

Kudeaketa eta jarraipena

6.1 Burututako lana

Proiektu hau gauzatzeko 300 ordu erabiltzea pentsatu zen eta ataza bakoitzari eskaini beharreko orduak estimatu ziren. Ondorengo taulan estimazioak, benetan erabili diren orduak eta bien arteko aldea azaltzen dira.

6.1 Taula: Estimatutako eta erabilitako orduak.

Ataza	Estimatutako orduak	Erabilitako orduak	Aldea (orduak)
Planifikazioa	15	14	-1
Kudeaketa	15	12	-3
Prestakuntza	20	45	25
Diseinua	50	43	-7
Garapena	87	141	54
Probak	13	7	-6
Memoria	100	104	4
Guztira	300	366	66

Taulan ikusten den bezala, estimatutakoa baino 66 ordu gehiago erabili dira. Desbideratze horretan (%22), pisu handiena prestakuntzak eta garapenak dute.

Alde batetik, prestakuntzarako estimatutako ordu kopurua txikia izan da. Proiektu honen izaerak (teknologia eta tresna berriak, horietako hainbat hasiera batean ezezagunak), ordea, prestakuntzarako ordu gehiago erabiltzea eskatu du. Estimazioak egiterako orduak hori kontuan ez hartzeak ekarri du ezberdintasun hori.

Bestetik, garapenean ere desbideratze nabarmena egon da. Horren barnean, zerbitzariaren garapenerako erabili dira ordu gehienak. Ataza horretarako 40 ordu erabiltzea estimatu zen, baina 86 erabili dira; hau da, garapenerako erabili beharreko ordu guztiak. Igoera horren atzean bi arrazoi nagusi egon dira:

1. Magnet loturak automatikoki sortu eta fitxategiak partekatzea. Hasieran atzemandako arriskuetako bat zen. Inplementatzea posible dela ikusi da, eta inplementazio horrek kostu dezente eduki du. Proiektuarentzako funtzionalitate erakargarria zenez, ez da albo batera utzi nahi izan.

2. Probek bideratuta garatzea. Hasiera batean, inplementazioaren kodea idatzi aurretik probak prestatzeak garapena moteldu du. Behin prozesu horretara ohituta, ordea, lagungarria izan da eta garapena ez du moteldu.

Bezeroarentzako, berriz, 44 ordu erabili dira; estimatutako balioaren pare. Bezeroa hobetzeko ordu gehiago erabiltzeko tentazioa egon da, baina, zerbitzariaren garapenean erabilitako orduak ikusirik, estimatutako balioa gainditu aurretik garapenari amaiera eman zaio.

Hala ere, desbideratze negatiboa izan duten atazak ere egon dira: planifikazioa, kudeaketa, diseinua eta probak. Horietatik, alde handiena probekin egon da (-%46). Kontraesana dirudi, zerbitzariaren garapena probek moteldu dutela aipatu ostean. Probetarako kontatutako orduak zerbitzariaren probak, hasieran, prestatzeko eta eskuz egindako zerbitzariaren edo bezeroaren proba-saioenak dira. Behin proben bidezko garapen prozesua martxan jarrita, zerbitzariaren probak idaztea garapenaren barne sartu da, proba bat eta horren inplementazioa idaztea bata bestearen segidan egin baitira.

Memoriari ordu asko eskaini zaizkio eta estimatutako baliotik hurbil geratu da.

6.2 Komunikazioa

Komunikazio gehiena posta elektronikoz burutu da. Bilerak batez ere mugarrietara iristerakoan egin dira; egindakoa begiratzeko eta nola jarraitu aztertzeko. Orokorrean, komunikatzeko arazorik ez da egon.

6.3 Kalitatea

Kudeaketari dagokionez, aurreko atalean azaldu den moduan, komunikazio aldetik ez da orokorrean arazorik egon.

Produktuaren kalitatearen inguruan, zerbitzariak eskatutakoa betetzen du. Probetan 190 egiaztapen egiten dira eta guztiak pasatzen ditu -OpenJDK 7, OracleJDK 7 eta OracleJDK8 plataformetan. Gainera, aplikazioak 963 SLOC edo LOC[1] ditu eta probek 895. Clojure kodea bakarrik kontuan hartuz gero -torrentak sortzearen zatia Javaz dago-, 824 eta 895 balioak lortzen dira; hau da, probetarako kode gehiago eta guzti dago. Java kodea ez kontatzeak bere arrazoia du; izan ere, torrent fitxategiak eta magnet loturak sortzeko prozesua ez da probetan kontutan hartu. Probak API mailan egiten direnez, kanpotik, ezin dugu konprobatu torrent fitxategi egokia sortzen dela -gogoratu fitxategi horiek ez direla publikoak, zerbitzariak bakarrik ikusten ditu-, eta, itzuliko den magnet loturaren balioa aldez aurretik ezin dugu jakin. Hori horrela izanik, fitxategiekin lotutako probak eskuz burutu dira.

Bezeroaren kalitatea, berriz, ezin izan da modu kuantitatibo batean neurtu. Probak eskuz burutu dira, behin eta berriz, eskatutakoa ongi betetzen duela jakiteko. Halere, zerbitzariak proba guztiak pasatzen dituela jakiteak konfiantza eman du APIa erabiltzeko orduan.

Hau idazterako orduan, memoriak 95 orri inguru zituen, 130 eranskinarekin. 100 orritik hurbil geratu da, luzapenetan ibili gabe. Beraz, luzera egokia dela uste da. Memoria tutoreak berrikusteaz gain, proiektuaren gaia ezagutzen ez duen hirugarren pertsona batek zati batzuk irakurri ditu eta ulergarritasunari buruzko bere iritzia eman du.

6.4 Arriskuak

Arriskuei dagokienez, magnet loturak automatikoki sortzea, kosta arren, lortu da. Metadatuak automatikoki sortzea, aldiz, kostu handia edukitzeagatik ez da egin. JavaScript edo ClojureScript bidez epub fitxategiak irakurri eta bertatik datuak lortu behar dira. Gainera, automatizazio horrek ez ditu horrenbeste onura ekartzen, fitxategiek agian ez baitituzte nahi adina metadatu -kasu horretan eskuz editatu beharko lirateke.

[1] Source Lines Of Code edo Lines Of Code, kalkulatzeko CLOC programa erabili da: `http://cloc.sourceforge.net/`

Planifikazioari dagokionez, lehen aipatu den bezala, prestakuntzan eta garapenean egon da desbideratze handiena. Prestakuntzaren desbideraketak ez du proiektuan eraginik eduki, gehienbat udan, epeetan jirabiran sartu aurretik, egin baita. Garapenean, zerbitzariaren inplementazioak denbora asko hartu du, baina, hala ere, ezarritako mugarrien epeak errespetatzea lortu da.

7. KAPITULUA

Ondorioak eta etorkizuneko lanak

7.1 Ondorioak

Ondorioak bi azpiataletan banatu dira: batetik, formakuntzari begirako ondorioak, eta, bestetik, garapen teknikoari buruzkoak.

1. **Formakuntza**

   Formakuntzari dagokionez, proiektua garapen profesionalerako oso lagungarria izan da. Planifikazioak duen garrantzia ikusi da, helburuak eta epeak betetzeko ezinbestekoa izanik. Arriskuak aurrez identifikatuta edukitzea oso lagungarria izan da arrisku batekin topo egindakoan zer egin jakiteko eta blokeatuta ez geratzeko. Magnet loturak automatikoki sortzearekin jarraitu egin da, zekarren kostua jakinda, eta metadatuak automatikoki sortzea baztertu egin da. Memoria idaztearekin, proiektua nola dokumentatu ikasi da. Orokorrean, proiektu informatiko baten kudeaketan eta garapenean esperientzia lortu da.

   APIa definitzerako orduan asko ikasi da, gai horren inguruan informazio asko jaso baita eta buruari buelta ugari eman baitzaizkio. Behin deskribapen osoa edukita, TDD (Test-Driven Development) bidez zerbitzaria inplementatzeak lan asko eskatu du, baina uneoro aurrera egiten zela ikusi da, mantso izan arren. Inplementazioa zatika egiteak, mugarri txikiak ipiniz, motibazioaren aldetik asko lagundu du. Halere, elkar lotuta dauden kasuak aldi berean ez

inplementatzegatik, batzutan kodea berregituratu behar izan da. Berregituraketa lan horietarako, probak euskarri egokiak izan dira.

Magnet lotura eta BitTorrenten inguruan, kontzeptu berriak ez ziren arren, ezagutza berriak lortu dira. Nola funtzionatzen duten eta nola erabili daitezkeen ikusi da.

JavaScript ekosistema oso aldakorra da. Tresna eta joera berriak perretxikoak bezala ateratzen dira, eta asko berehala baztertzen edo desagertzen dira. Egonkortasuna lortzea zaila dela jakinda, React erabiltzea erabaki da. Tresna erakargarria dela iruditu da eta, berria izan arren, etorkizuna duena -Facebook barnean erabiltzen dute. Beraz, erabaki egokia izan dela uste da; betiere jakinda urte batetik bestera panorama zeharo alda daitekeela. Gainera, aukerak aztertzerako orduan, orri bakarreko aplikazioak eraikitzeko modu ezberdinak ezagutu dira.

Clojure -eta ClojureScript- erabiltzea ez da aukera ohikoena. Beste lengoaia ezagunago bat erabil zitekeen, Java bera adibidez. Unibertsitatean objektuei orientatutako programazioa erakutsi ohi da; eta Clojurek programazio funtzionala erabiltzera bultzatzen du, eredu horretatik aldenduz. Paradigma ezberdinak probatzea eta erabiltzea ondo dator eta, programazio funtzionala erabiliz idatzitako proiektu handiena izanik, asko ikasi da.

2. **Garapen teknikoa**

   Proiektua, amaitu ostean, bide onetik joan dela eta helburuak bete dituela esan daiteke. Hasieran ikusitako hutsunea -torrent fitxategia eta magnet loturak sortu, eta indize gisa jokatzen duen web API baten beharra- betetzea lortu da. Gainera, Web Sistemak ikasgaiko proiektuaren kontzeptua sakontzea lortu da. Garapena amaitutakoan, berriz, sakondu edo gehitzeko eginkizun berriak aurkitu dira -gehiago "Etorkizuneko lanak"atalean.

   Egin diren aukera teknikoei esker, Liburutruk aplikazioa instalatzeko eta erabiltzeko erraza da. Gainera, aplikazioa publikoa da eta edonork eskura dezake. Beraz, proiektu honi esker aukera ematen da komunitate mugatu eta kontrolatu batean baliabideak partekatzeko.

   Kodearen lizentzia librea denez, eta GitHub bidez eskuragarri dagoenez, edonork kopiatu, aldatu edo banatu dezake. Erabaki horri esker aplikazioa gehiago garatzeko aukera irekita uzten da. Bestalde, kodea editatuz, aplikazioa norberaren beharretara egokitu daiteke. Liburuei lotutako komunitate bat izan

beharrean, beste domeinu batera egokitu daiteke. Edo magnet loturak erabiltzen dituen beste P2P sare bat erabil daiteke[1].

7.2 Etorkizuneko lanak

Aurrera begira, hainbat lan burutu daitezke. Hasteko, APIaren inplementazioa amaitu daiteke, liburuak filtratzea eta administratzaileen zatia idaztea falta baita.

APIarekin lanean jarraitzea erabakiz gero, diseinua zabaldu eta findu daiteke. Diseinatutako APIa orokorra da, bezeroa baino lehenago zehaztu dena. Baina erabilpen espezifikoetarako egokitu eta doitu daiteke hala nahi izanez gero. Komunitate kutsu gehiago emateko, erabiltzaileen arteko erlazioak gehitu daitezke: lagunak, beraien arteko mezuak, taldeak... Liburuak sailkatuta lortzeko aukera eskaintzea ere ongi legoke: gogokoen arabera ordenatuta, iruzkin kopuruen arabera, edo, egindako bisiten arabera, adibidez. APIa fintzerako orduan, saioei begirada bat ematea izango litzateke egokiena. Saioen kudeaketa zerotik diseinatu da -tokenak erabiliz- eta, segurtasunak garrantzi handia duen atala izanik, agian hobe litzateke OAuth gisako estandar bat erabiltzea. Horrek konplexutasuna gehituko lukeenez, aldaketa onuragarria izan daitekeen aztertu beharko litzateke[2].

Bestalde, APIaren erabilera ez dago mugatuta; edonork erabil dezake, nahi duen guztietan. Mugatu nahi bada, aplikazio gisa erregistratzeko eskatu daiteke. Hau da, APIa erabiltzen duen aplikazioak, hasieran, sisteman izena emango du eta API-dei guztiak aplikazioaren izenean egingo ditu. Modu horretan, aplikazio bakoitza kontrolatu daiteke. Minutuko gehienez API-dei kopuru jakin bat onartu daiteke; edo erabilera baldintzak betetzen ez dituzten aplikazioak baztertu. Horretarako OAuth erabili daiteke -Web API batzuk modu horretan funtzionatzen dute.

Bezeroari dagokionez, proiektuko inplementazioak APIaren zati bat bakarrik erabiltzen du. Beraz, garapenerako aukera handia dago oraindik. Ez da derrigorrezkoa API guztia erabiltzea, noski, baina bezeroari funtzionalitatea gehitzeko aukera asko dago oraindik.

[1] Posible litzateke aplikazioak P2P sare bat baino gehiago aldi berean erabiltzea. Baliabide baten magnet lotura sare batetik bestera aldatu egiten da, baina formatu berdina du azken batean. Adibidez, posible da baliabide bat bi magnet loturen bidez erabiltzea: batean *BitTorrent Info Hash* erabiliz eta bestean *ED2K hash* (EDonkey sarerako).

[2] OAuthi egindako kritikak ez dira gutxi; OAuth 2.0-ren egilea bera ere ez dago estandarrarekin gustura: `http://hueniverse.com/2012/07/26/oauth-2-0-and-the-road-to-hell/`

Bezeroarekin jarraituz, epub fitxategietatik metadatuak ateratzea ez da inplementatu, kostu handia baitu. Kostua onargarria izanez gero, erabiltzaileak aukeratutako fitxategia bezeroan irakurri eta analizatuz metadatuak atera litezke. Era berean, azalaren dimentsioak ere mugatzea ongi legoke. Bestalde, bezeroa orri bakarreko aplikazioa da, baina erabilera aldetik webgune normal batetik ez da hainbeste aldentzen. Formatua egokia bada, orri bakarreko aplikazioetan baizik eraiki ezin daitekeen interfaze bat eduki lezake, orrien interfaze klasikotik aldenduz.

8. KAPITULUA

Bibliografia

Structure and Interpretation of Computer Programs
Harold Abelson & Gerald Jay Sussman, with Julie Sussman, 1996 (2n edition)
ISBN 0-262-51087-1

Web API Design
Brian Mulloy, 2012
`https://pages.apigee.com/web-api-design-ebook.html`

The Joy of Clojure
Michael Fogus & Chris Houser, 2011

Apigee-ko baliabideak
`http://apigee.com/about/resources`

API ereduak
`https://dev.twitter.com/overview/api`
`https://developer.github.com/v3/`

Clojureren webgunea
`http://clojure.org`

Clojure for the brave and true
`http://www.braveclojure.com/`

Clojure Documentation. Komunitateak garatutako dokumentazioa
`http://clojure-doc.org/`

A brief overview of the Clojure web stack. Clojure bidezko web-garapena ulertzeko lagungarria

https://brehaut.net/blog/2011/ring_introduction

React dokumentazioa

https://facebook.github.io/react/docs/getting-started.html

David Nolen-en bloga. ClojureScripten inguruko artikuluak ditu

https://swannodette.github.io/

The Twelve-Factor App

http://12factor.net/

Nginx dokumentazioa

http://nginx.org/en/docs/

Eranskinak

A. ERANSKINA

Erabilitako tresnak

Proiektu honetan tresna ugari erabili dira. Atal honen helburua horiek guztiak biltzea da.

- Sistema eragileak
  - Debian GNU/Linux

    `https://www.debian.org/`
  - Windows 7
- Emacs

  `https://www.gnu.org/software/emacs/`
  - org-mode

    `http://orgmode.org`
  - Clojure Mode

    `https://github.com/clojure-emacs/clojure-mode/`
  - CIDER

    `https://github.com/clojure-emacs/cider`
  - ParEdit

    `http://www.emacswiki.org/emacs/ParEdit`

  - Magit

    `https://magit.github.io/`

- IntelliJ IDEA

  `http://www.jetbrains.com/idea/`

- Git

  `http://www.git-scm.com/`

- git-annex

  `http://git-annex.branchable.com/`

- Firefox

  `https://www.mozilla.org/en-US/firefox/new/`

  - JSONView

    `https://addons.mozilla.org/en-US/firefox/addon/jsonview/`

- Chromium

  `http://www.chromium.org/`

- Zerbitzaria eta bezeroa:

  - OpenJDK

    `http://openjdk.java.net/`

  - Clojure

    `http://clojure.org`

  - Leiningen

    `http://leiningen.org/`

    * Liburutegi eta pluginak (izena eta bertsioa[1]):
      · org.clojure/tools.namespace 0.2.7
      · io.aviso/pretty 0.1.13
      · im.chit/vinyasa 0.2.2
      · spyscope 0.1.5

[1] Group ID, Artifact ID, Version. Informazio gehiago hemen: `https://github.com/technomancy/leiningen/blob/stable/doc/TUTORIAL.md#dependencies`

        - cider/cider-nrepl 0.8.1

- Zerbitzaria:
  - H2

    `http://h2database.com/html/main.html`
  - ttorrent

    `https://github.com/mpetazzoni/ttorrent`
  - Liburutegiak (izena eta bertsioa):
    - compojure 1.1.8
    - ring-cors 0.1.4
    - clj-json 0.3.2
    - org.clojure/java.jdbc 0.3.5
    - com.h2database/h2 1.3.170
    - clj-time 0.8.0
    - clj-bcrypt-wrapper 0.1.0
    - javax.servlet/servlet-api 2.5
    - http-kit 2.1.16
    - org.clojure/data.codec 0.1.0
    - sanitize-filename 0.1.0
    - org.clojure/tools.namespace 0.2.7
    - midje 1.6.3
  - Leiningen pluginak:
    - lein-midje 3.1.3
    - codox 0.8.10

- Bezeroa
  - ClojureScript

    `https://github.com/clojure/clojurescript`
  - React

    `http://reactjs.com/`
  - Foundation

    `http://foundation.zurb.com/`

  - Liburutegiak (izena eta bertsioa):
    * org.clojure/core.async 0.1.346.0-17112a-alpha
    * cljs-ajax 0.3.3
    * com.andrewmcveigh/cljs-time 0.2.4
    * reagent 0.4.3
    * secretary 1.2.1
    * figwheel 0.1.5-SNAPSHOT
  - Leiningen pluginak:
    * com.cemerick/austin 0.1.5
    * codox 0.8.10
    * lein-cljsbuild 1.0.3
    * lein-figwheel 0.1.5-SNAPSHOT

- Tranmission-GTK

  `https://www.transmissionbt.com/`

- Nginx

  `http://nginx.org/`

- OpenSSL

  `https://www.openssl.org/`

- TEX Live

  `https://www.tug.org/texlive/`

- reveal.js

  `http://lab.hakim.se/reveal-js/`

- PHP 5.4

  `http://php.net/`

  - Composer

    `https://getcomposer.org/`

  - Liburutegiak
    * vlucas/phpdotenv: ~1.0

    * guzzlehttp/guzzle: ~5.0
    * pimple/pimple: ~3.0
    * klein/klein: ~2.0
    * mustache/mustache: ~2.0

- OpenTracker

  `https://erdgeist.org/arts/software/opentracker/`

- FrootVPN

  `https://www.frootvpn.com/`

- OpenVPN

  `http://openvpn.net/`

- GitHub

  `https://github.com/`

- Travis CI

  `https://travis-ci.org`

- PlantUML

  `http://plantuml.com/`

- Ditaa

  `http://ditaa.sourceforge.net/`

- Dia

  `https://wiki.gnome.org/Apps/Dia`

- GanttProject

  `http://www.ganttproject.biz/`

- GIMP

  `http://www.gimp.org/`

B. ERANSKINA

Instalazio-eskuliburua

B.1 Aplikazioa eskuratzea

Fitxategi honekin batera, liburutruk-aplikazioa.zip izeneko fitxategi batek egon beharko luke. Horrela ez bada, GitHub-etik[1] jaitsi daiteke. Fitxategi hori deskonprimituz gero, liburutruk-aplikazioa izeneko karpeta bat agertuko da. Horren barnean beste bi karpeta egongo dira: zerbitzaria eta bezeroa. Horiek irakurri.txt izeneko fitxategi bana dute, instalatzeko azalpenak dituztenak. Hurrengo bi ataletako edukia dute, hain zuzen ere.

B.2 Zerbitzaria

Zerbitzaria erabiltzeko baldintza minimoa sisteman Java (JVM, makina birtuala) instalaturik edukitzea da. OpenJDK[2] edo Oracle[3] erabili daiteke, baina 7 bertsioa edo berriagoa izan behar da.

liburutruk-aplikazioa.zip deskonprimituz gero, *zerbitzaria* izeneko karpetan ondorengo fitxategiak egongo dira:

[1] `https://github.com/lnmnd/liburutruk/releases/download/v1.0.0/liburutruk-aplikazioa.zip`

[2] `http://openjdk.java.net/`

[3] `https://www.java.com/en/download/`

- irakurri.txt: laguntza-testu hau.
- magnet.jar: exekutatu beharreko aplikazioa.
- konfig.adb.clj: konfigurazio-adibidea.
- magnet.h2.db: hasieratutako datu-basea, konfig.adb.clj-ko konfigurazioarekin erabiltzeko.
- files: fitxategientzako karpetak, hutsak, konfig.adb.clj-ko konfigurazioarekin erabiltzeko.
- konfig.gutenberg.clj: Gutenberg proiektuko [4] liburu batzuk dituen konfigurazioa.
- gutenberg.h2: liburu batzuekin hasieratutako datu-basea, konfig.gutenberg.clj-ko konfigurazioarekin erabiltzeko.
- gutenberg-files: fitxategientzako karpetak, Gutenberg-eko liburuekin, konfig.gutenberg.clj-ko konfigurazioarekin erabiltzeko.

Konfiguratzeko, konfig.adb.clj fitxategiko edukia konfig.clj izeneko fitxategi batean gorde eta parametroak egokitu, beharrik izanez gero. Konfigurazioa Clojure lengoaian idatzita dago, baina formatu sinplea du orokorrean (":gakoa balioa"moduko pareak). Parametroak ondorengoak dira:

- portua: Erabiliko den portu zenbakia. 1023tik gorako zenbakia erabili root baimenak behar ez izateko.
- db-kon: Datu-base konexioa. magnet.h2.db erabiltzeko konfiguratuta dago, bertako subname aldatu datu-basea beste izen batekin gorde bada.
- muga: Bilduma batean agertuko diren elementu kopurua.
- saio-iraungitze-denbora: Saioaren iraungitze-denbora, segundotan. Saio bat hasten denetik denbora hori pasaz gero, baliogabetu egingo da.
- partekatu: Torrentak partekatu nahi diren edo ez. true bada *torrent-gehitze-programa* erabiliko da.
- kokapenak: Fitxategientzat erabiliko diren hainbat kokapen. Gogoratu bertan aipatutako karpetak sortzeaz.

[4] Gutenberg proiektuak liburuak digitalizatzen eta artxibatzen ditu, gehienak domeinu publikokoak izanik.

- trackerrak: Torrent trackerren zerrenda. Lehenengoa torrentaren tracker gisa ezarriko da eta gainontzekoak magnet loturan agertuko dira. pi:6969 sare lokalean erabilitako trackerra da, kendu izen bereko trackerrik ez bada existitzen.
- torrent-gehitze-programa: Torrent fitxategia eta katalogoa emanda, torrenta partekatze duen programa. Funtzio anonimo bat da, ondorengo argumentuekin:
  - sh: Shell komandoa exekutatuko duen funtzioa.
  - torrent: Torrent fitxategia.
  - katalogoa: Deskargatzeko erabiliko den karpeta.

Fitxategiak partekatzeko Transmission aplikazioa erabili da, GTK+ bertsioa hain zuzen ere. Hobespenetan *Web* atalean web bezeroa gaitu da (*Enable web client*) kanpoko komandoak jasotzeko aukera edukitzeko. Gero *transmission-remote* erabili da torrent fitxategiak gehitzeko. Hala ere, komando baten bidez torrentak gehitzea ahalbideratzen duen edozein aplikazio erabili daiteke, behar bezala konfiguratuz gero.

Behin konfiguratuta, zerbitzaria martxan jarri jar fitxategia exekutatuz:

```
java -jar magnet.jar
```

Konfigurazioan adierazitako portuan entzuten geratuko da. 3000 portua erabili bada, adibidez, joan `http://localhost:3000/v1/liburuak` helbidera. JSON formatuko erantzun bat agertuko da.

Adibideko konfigurazioarekin, hasieran datu-basean daturik egongo ez denez, konfig.gutenberg.clj konfigurazioa gehitu da. Fitxategi hori konfig.clj gisa kopiatuz gero, Gutenberg proiektuko liburu batzuekin hasieratutako datu-basea eta fitxategiak erabiliko dira. Horiekin batera, "era"erabiltzaile izena eta "1234"pasahitza duen erabiltzaile bat ere gehituta dago. Liburuak erabiltzaile horrek gehitutakoak dira eta hainbat iruzkin eta gogoko ere baditu.

Konfigurazioan beste datu-base bat erabiltzea zehaztu bada, hori hasieratu beharko da. Horretarako, zerbitzaria lehen aldiz martxan jarri aurretik, aplikazioari hasieratu argumentua pasa:

```
java -jar magnet.jar hasieratu
```

Orain arte azaldu den moduan konfiguratu eta abiarazitako zerbitzaria egokia da garapenerako edo modu lokalean erabiltzeko, baina pare bat eragozpen ditu:

1. Ohikoa ez den portu zenbakia erabiltzen du.
2. HTTP soila erabiltzen denez konexioa ez da segurua.

APIa kanpotik erabiltzeko, beraz, alderantzizko proxy[5] bat erabiltzea gomendatzen da. Jarraian Nginx horretarako nola konfiguratu azalduko da.

1. **Alderantzizko proxy baten konfigurazioa: Nginx**

   Nginx[6] kode irekiko alderantzizko proxy bat da. Hainbat ezaugarri ditu eta modu askotara konfiguratu daiteke, baina konexioak portu batean jaso eta gure zerbitzariari bidaltzeko bakarrik konfiguratuko da.

   Demagun APIarentzako erreserbatutako domeinua `magnet` dela, eta, lehen aipatu gisa, zerbitzaria 3000 portuan martxan jarri dela. UNIX motako makina batean gaudela ere suposatuko da; beraz, kokapenak horren araberakoak izango dira. Horrez gain, fitxategiak aldatzeko eta zerbitzaria martxan jartzeko root baimenak beharko dira.

   Lehenik, zerbitzari birtual bat konfiguratuko da. Horretarako `/etc/nginx/sites-available` karpetan magnet izeneko fitxategi bat sortu behar da, ondorengo edukiarekin:

```
server {
  listen 80;
  server_name magnet;

  location / {
    proxy_pass http://localhost:3000;
  }
}
```

   Konfigurazio horren bidez, magnet izeneko zerbitzarira 80 portutik iristen diren konexioak `proxy_pass`-en aipatzen den helbidera bideratuko dira (aplikazioa

[5] Reverse proxy: `https://en.wikipedia.org/wiki/Reverse_proxy`
[6] `http://nginx.org/`

martxan dagoen helbidea). Horrekin, `http://magnet` helbidea erabili ahalko da.

Zerbitzari hori aktibatzeko `/etc/nginx/sites-enabled` karpetan lotura bat gehitu behar da. Komando lerrotik:

```
ln -s /etc/nginx/sites-available/magnet /etc/nginx/sites-enabled/magnet
```

Nginx berrabiaraztea besterik ez da falta. Hori egiteko modua sistemaren araberakoa da. Debian batean, adibidez:

```
service nginx restart
```

Konfigurazio horrekin lehen aipatutako arazo bat konpontzen da, baina konexioa oraindik ez da segurua. Bezero eta zerbitzariaren arteko komunikazio guztia ikusgai dago. Komunikazio hori seguruagoa egiteko TLS/SSL[7] erabiliko da. Horretarako ziurtagiria sortuko da. Ziurtagiri jaulkitzaile batek sinatutakoa izango ez denez (norberak sinatutakoa izango da), nabigatzaileek ez segurutzat hartuko dute. APIa publiko egin nahi bada ziurtagiri egoki bat lortzea izango litzateke egokiena, baina konfigurazioa probatzeko norberak sortzea aski da.

Ziurtagiria sortzeko:

```
mkdir -p /etc/ssl/localcerts
openssl req -new -x509 -days 365 -nodes \
-out /etc/ssl/localcerts/magnet.crt \
 -keyout /etc/ssl/localcerts/magnet.key
```

Eskatzen diren datuak bete ondoren, urte bat iraungo duen ziurtagiria sortuko da `/etc/ssl/localcerts` karpetan.

Zerbitzari birtuala konfiguratzeko, aurrez sortutako `magnet` fitxategia aldatuko da:

```
server {
  listen 443 ssl;
  server_name magnet;
  ssl_certificate     /etc/ssl/localcerts/magnet.crt;
```

[7] Transport Layer Security (TLS) eta Secure Sockets Layer (SSL)

```
    ssl_certificate_key /etc/ssl/localcerts/magnet.key;

    location / {
      proxy_pass http://localhost:3000;
    }
  }
```

80 portuan entzun ordez 443 portuan entzungo duela jarri da, hau da, ssl moduko konexioa izango dela. `ssl_certificate` eta `ssl_certificate_key` bidez ziurtagiria non dagoen adierazten da (lehen sortutakoa) eta `server_name` eta `proxy_pass` ez dira aldatu.

Horrela, informazio guztia zifratuta pasako da. `https://magnet/v1/liburuak` helbidera joanez gero, ziurtagiria ez dela baliozkoa agertuko da, baina onartuz gero orria https gisa kargatuko da, hau da, konexioa zifratuarekin.

B.3 Bezeroa

Bezeroa HTML, CSS eta JavaScript-ez osatuta dago, eta nahiko berria den eta estandarrak betetzen dituen web-nabigatzaile batekin funtzionatzen du (Firefox eta Chromerekin probatu da). Exekutatzeko ez du dependentzia gehiagorik.

Konfiguratzeko, liburutruk-aplikazioa.zip fitxategia deskonprimitu ondoren, *bezeroa* karpetara joan. Bertan konfig.js izeneko fitxategia egongo da. Fitxategia ireki eta bertan dauden parametroak aldatu, behar izanez gero:

- figwheel: false utzi, garapenerako bakarrik erabiltzen da.
- api_oinarria: API-aren oinarri helbidea, ez ahaztu amaierako / ipintzeaz.
- azken_gogoko_kopurua: Azken liburuen artean gogokoena aukeratzeko kontuan hartuko den liburu kopurua.
- azken_iruzkin_kopurua: Azken iruzkinetan agertuko den iruzkin kopurua.
- liburu_orriko: Liburuak erakustean, orri bakoitzean agertuko den liburu kopurua.

Erabiltzeko aski da bertako index.html web-nabigatzaile batekin irekitzea. Ondo konfiguratuta egonez gero eta zerbitzaria martxan badago funtzionatuko du. Gerta liteke, fitxategi lokala izateagatik, nabigatzaileak API deiak egiten ez uztea. Horrelakorik ez gertatzeko hoberena web-zerbitzari baten bidez zerbitzatzea da. Horretarako nahikoa da *bezeroa* karpeta zerbitzatzea (ez da derrigorrezkoa zerbitzariaren erroan kokatuta egotea).

B.4 Kodea eskuratzea

Fitxategi honekin batera, liburutruk-aplikazio-kodea.zip izeneko fitxategi batek egon beharko luke. Horrela ez bada, GitHub-etik[8] jaitsi daiteke. Fitxategi hori deskonprimituz gero liburutruk-aplikazio-kodea izeneko karpeta bat agertuko da. Horren barnean beste hiru karpeta egongo dira: zerbitzaria, bezeroa eta bezeroa-php. Bakoitzak README.md izeneko fitxategi bat du, kodea erabiltzeko argibideak dituena (markdown formatuan). Horrez gain, zerbitzaria eta bezeroak doc izeneko karpeta bat ere badute. Karpeta horretako index.html fitxategia irekiz gero kodearen dokumentazioa irakurri daiteke.

Kode guztia GitHubera igota dagoenez, bertatik ere eskuratu daiteke:

- Zerbitzaria: `https://github.com/lnmnd/magnet.zer`
- Bezeroa: `https://github.com/lnmnd/magnet.bez`

Helbide horiek Git bidez klonatu daitezke.

Helburuetako bat izan ez arren, APIa erabiltzen duen beste webgune bat, oso sinplea, ere eraiki da. Orri bakarra du eta bertan liburu, egile eta erabiltzaile guztiak agertzen dira. APIa beste modu batera erabiltzeko ariketa eta adibide gisa idatzi da. Zerbitzarian exekutatzen den PHP lengoaia erabili da, JavaScript bidez nabigatzailean exekutatu ordez. liburutruk-aplikazio-kodea.zip fitxategiko bezeroa-php karpetan edo ondorengo helbidean aurkitu daiteke:

`https://github.com/lnmnd/magnet.php`

[8] `https://github.com/lnmnd/liburutruk/releases/download/v1.0.0/liburutruk-kodea.zip`

C. ERANSKINA

Liburutruk aplikazioa erabiltzeko eskuliburua

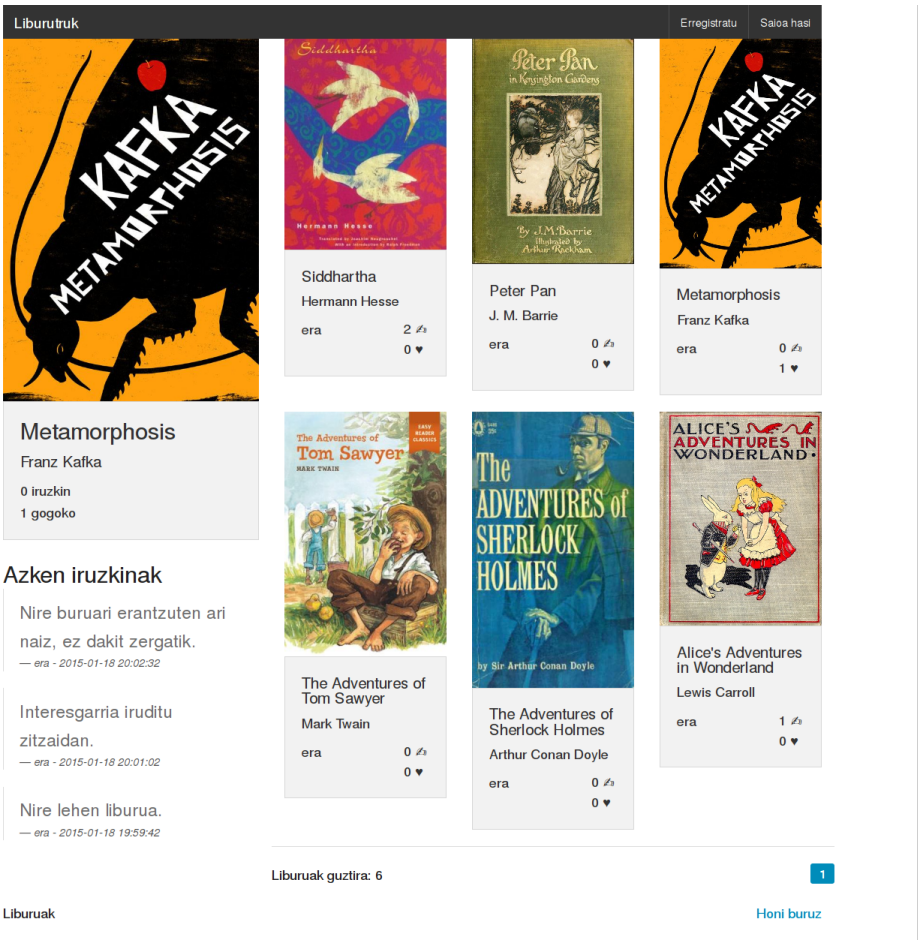
Atal honetan web-nabigatzailearen bidez aplikazioa nola erabiltzen den azalduko da.

Orriak, kargatutakoan, C.1 irudiko itxura edukiko du (C.2 irudikoa pantaila txikian, bi zatitan banatu da).

Goian menua ageri da, erregistratzeko eta saioa hasteko aukerekin. Ezkerrean zutabe mehar bat ondorengo edukiarekin: liburu berrietatik gogokoena eta azken iruzkinak. Eduki nagusia azken liburuak dira, berrienetik hasita. Liburu bakoitzari lotuta ondorengo informazioa doa: titulua, egileak, erabiltzailea (liburua igo duena), iruzkin kopurua eta gogoko kopurua. Liburuen azpian liburu kopurua eta nabigazioa dago, liburu zaharragoak ikusi ahal izateko.

Liburu batean, gogoko liburuan edo azken iruzkinetako batean klikatuz gero dagokion liburua irekitzen da. Orri horretan (C.3 irudia) informazio garatuagoa aurkezten da, garrantzitsuena iman batekin adierazitako *Magnet lotura* izanik. Behean iruzkinen zerrenda agertzen da, aurrerago aipatuko dena.

Web-nabigatzaileak *magnet* protokoloa ezagutzen badu, *Magnet lotura*-n klikatu eta BitTorrent bezeroaren bidez irekiko da. Erabiltzen den lehen aldia bada, irekitzeko orduan, erabili nahi den aplikazioa aukeratu beharko da segur aski (C.4 irudia). Nabigatzaileak protokoloa ulertzen ez badu, lotura kopiatu eta BitTorrent bezeroari lotura irekitzea eskatu. Liburua partekatzen ari diren kideak aurkitu arte denbora pixka bat pasa daiteke. Behin aurkituta, fitxategia azkar deskargatuko da. Esan beharrik ez dago BitTorrent bezeroak magnet loturentzako euskarria izan behar duela.



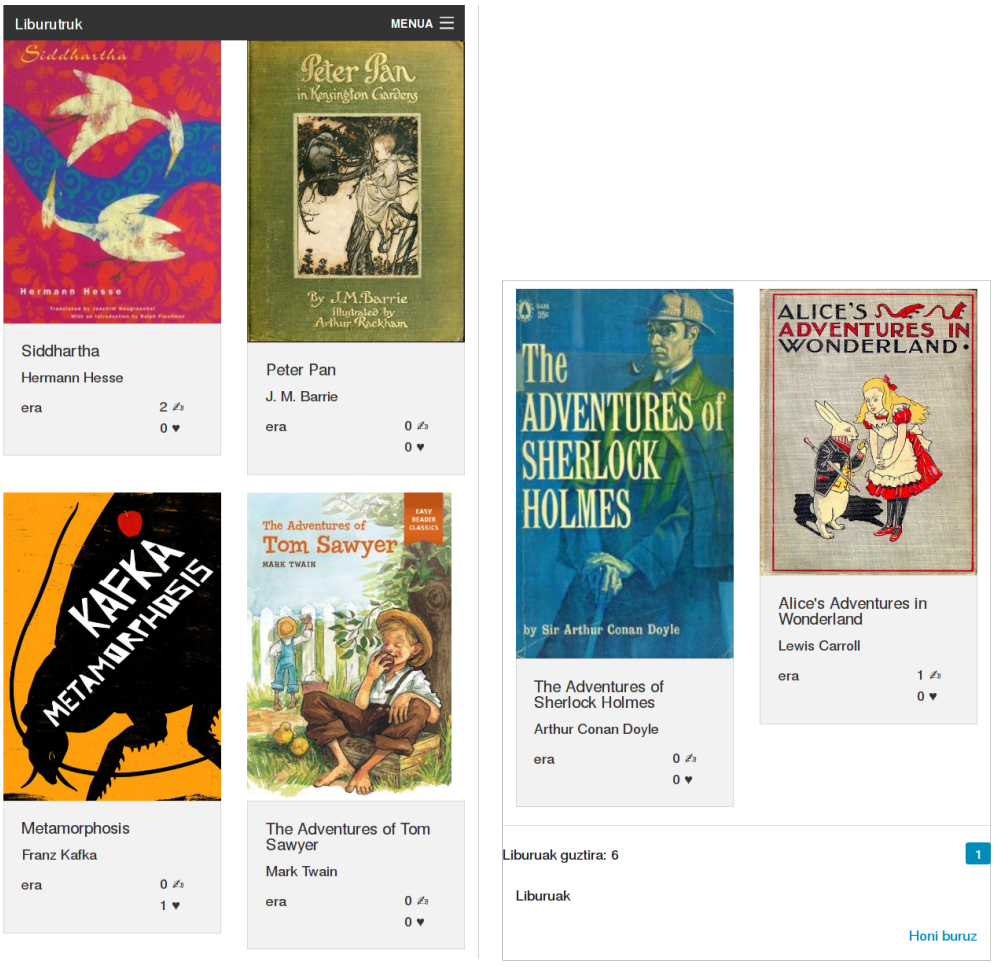
C.1 Irudia: Hasierako orria.

Aplikazioari zuku guztia ateratzeko erabiltzaile kontu bat edukitzea da onena. Horretarako erregistratu orrira joan behar da (C.5 irudia). Jarraian, erabiltzaile izen bat (aurrez existitzen bada abisatuko da), pasahitza eta izena (erakutsiko dena) bete behar dira. Deskribapena aukerakoa da. Erregistratu botoiari klikatzean erabiltzaile berriaren profila agertuko da (C.6 irudia). Profilean uneko datuak agertzen dira, aldatzeko aukerarekin. Kontua ere ezabatu daiteke bertatik.

Erregistratzean automatikoki saioa hasten da, goiko menuan gertatutako aldaketetan atzeman daitekeen bezala. "[Erabiltzailea]-ren saioa amaitu"klikatuta saioa amaitzen da, sarrerara joanez eta menua lehengoratuz. Behin erabiltzaile bat lortu eta gero menuko *Saioa hasi*-ren bidez sar daiteke erabiltzaile gisa (C.7 irudia).

Saioa hasita duen erabiltzaile batek hainbat aukera ditu menuan:

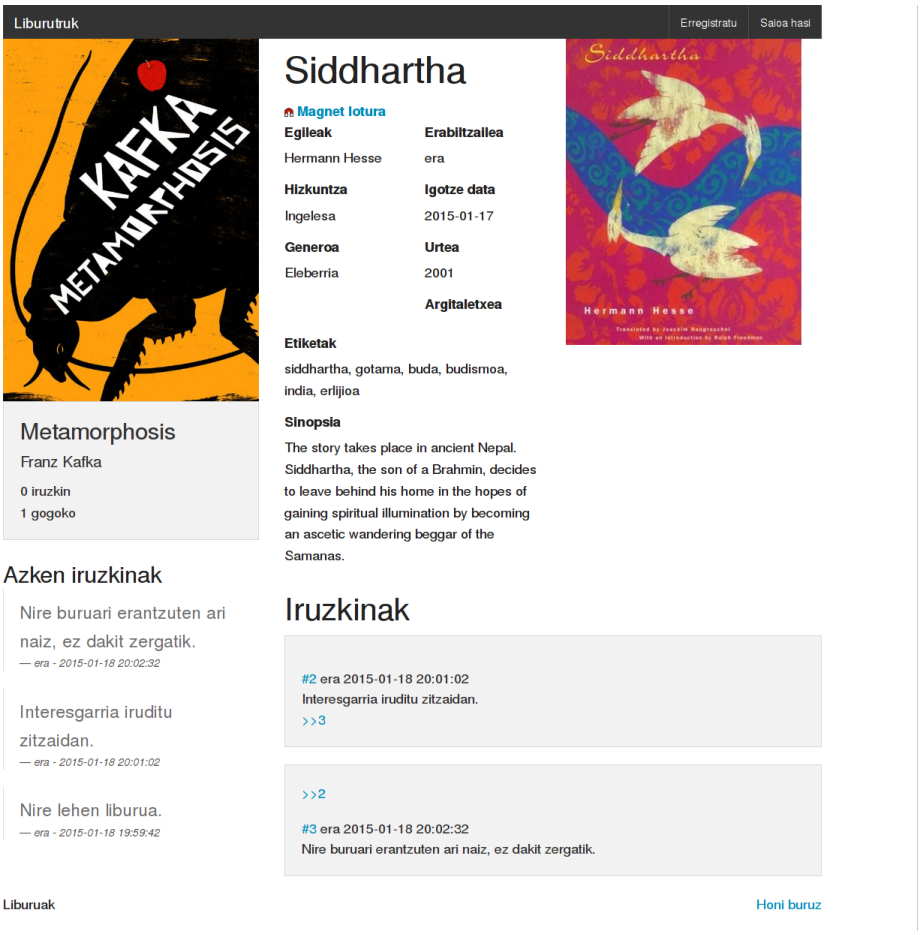
- Liburua gehitu: liburu berri bat gehitzeko.



C.2 Irudia: Hasierako orria pantaila txikian.

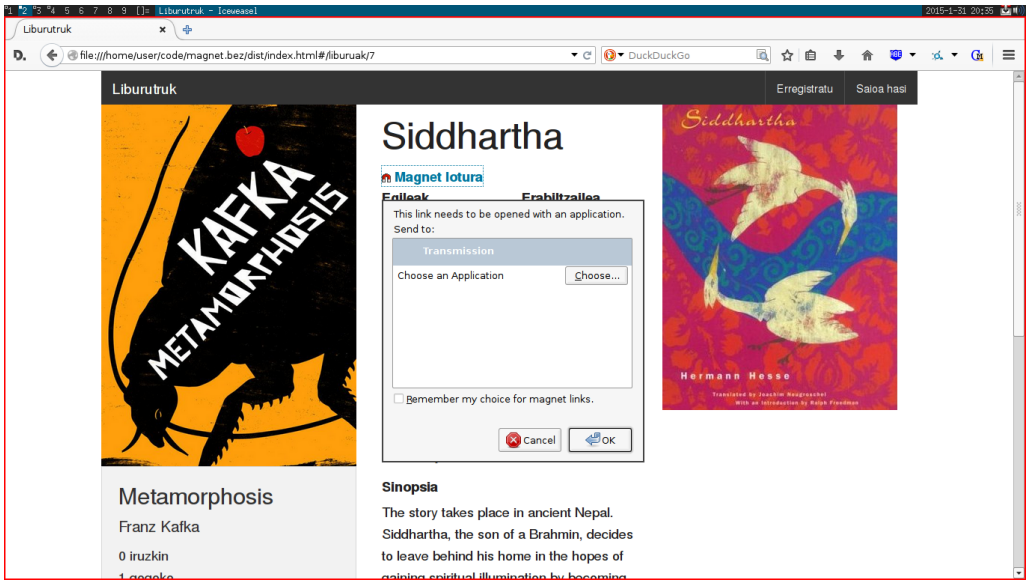
- Nire liburuak: liburuak kudeatzeko.
- Nire iruzkinak: iruzkinak kudeatzeko.
- Nire gogokoak: gogoko liburuak kudeatzeko.
- Nire profila: aurrez aipatutako profil orria.
- *Erabiltzailea*-ren saioa amaitu: aurrez aipatutako berdina.

Liburua gehitzea aukeratuz gero, formulario handi bat agertuko da (C.8 irudia). Bertan, lehenik, fitxategi bat igotzeko aukera dago. Fitxategi horrek epub motakoa izan behar du eta liburua gehitu ondoren partekatuko den fitxategia izango da. Jarraian ondorengo eremuak datoz:



C.3 Irudia: Liburu baten orria.

- Titulua: titulu berdina edo antzerakoa duen liburu bat existitzen bada abisatu egiten da.
- Egileak: "Gehitu"botoiaren bidez gehitzen dira, gutxienez egile batek egon behar du, eta existitzen diren egileen zerrenda erakusten da, laguntza gisa.
- Hizkuntza: aukera ohikoak agertzen dira.
- Sinopsia: testu kate luzeagoa idazteko tokia du.
- Argitaletxea: existitzen diren aukerak agertzen dira.
- Urtea: argitaratu den urtea, zenbaki bat.
- Generoa: zer generori dagokion, aukerazko eremua.
- Etiketak: etiketak gehitu daitezke, aukeran. Egileak-en funtzionamendu berdina du.



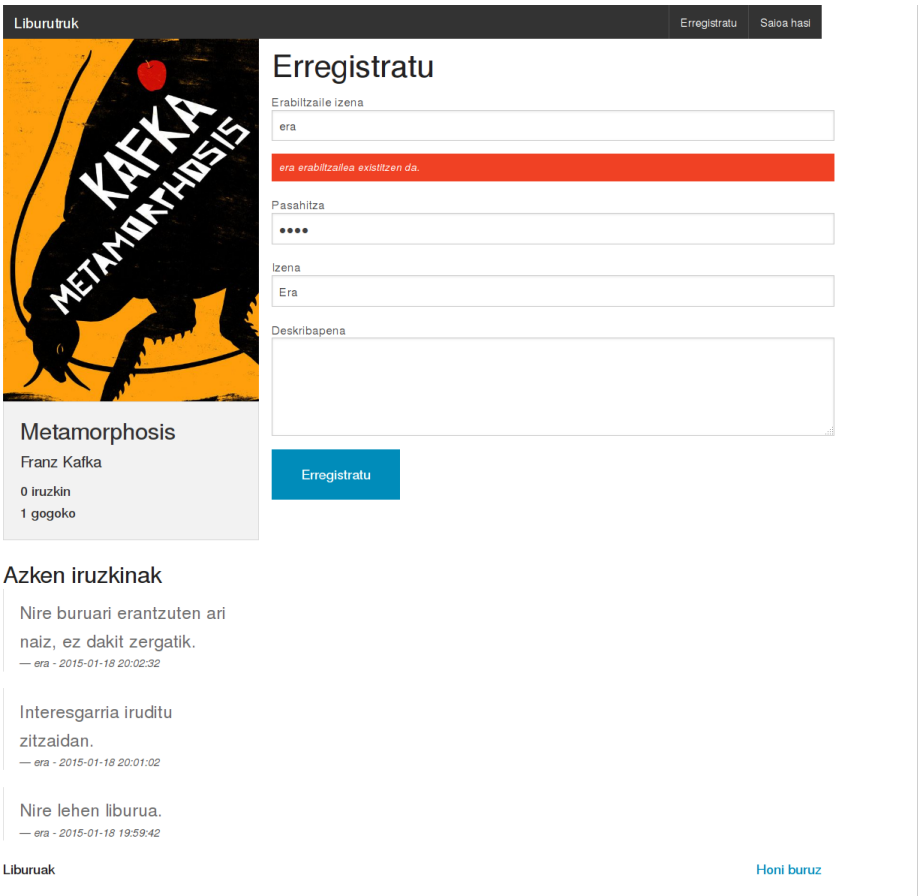
C.4 Irudia: Magnet lotura bat irekitzen.

- Azala: JPG formatuko irudia, aukeratzean bertan aurreikusten da.

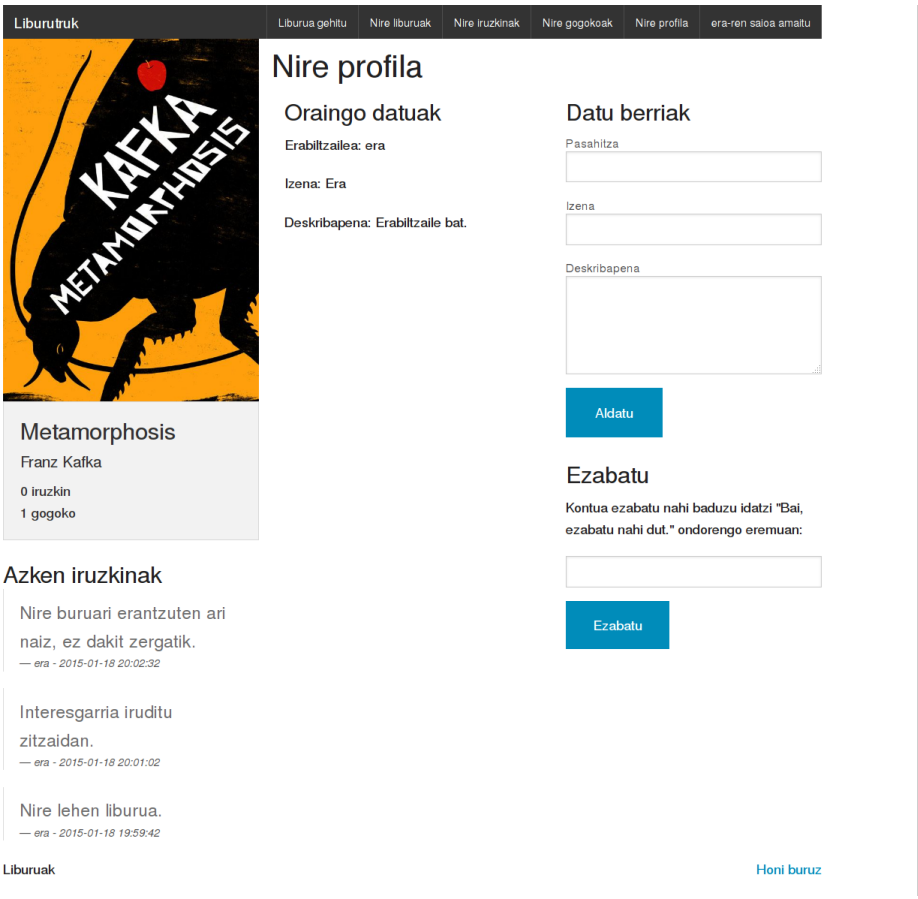
"Gehitu"klikatzean, dena ongi joan bada, liburu berriaren orria agertuko da. Aukeratutako epub fitxategia automatikoki partekatzen hasten da eta dagokion magnet lotura jada erabilgarri geratzen da, fitxategia eskuratzeko.

Saioa hasita liburu bat bisitatuz gero, hainbat gauza berri agertzen dira (C.9 irudia). Batetik, azalaren azpian liburua gogoko gisa markatzeko aukera dago (jada gogokoetan edukiz gero, bertan adieraziko da). Bestetik, iruzkinen azpian iruzkin berria gehitzeko aukera dago. Iruzkin bakoitzak identifikatzaile bat du (zenbaki bat) eta horiei erreferentzia egiteko aukera dago. Iruzkinak grafo moduan daude antolaturik: bakoitzak hainbat guraso eta erantzun ditu.

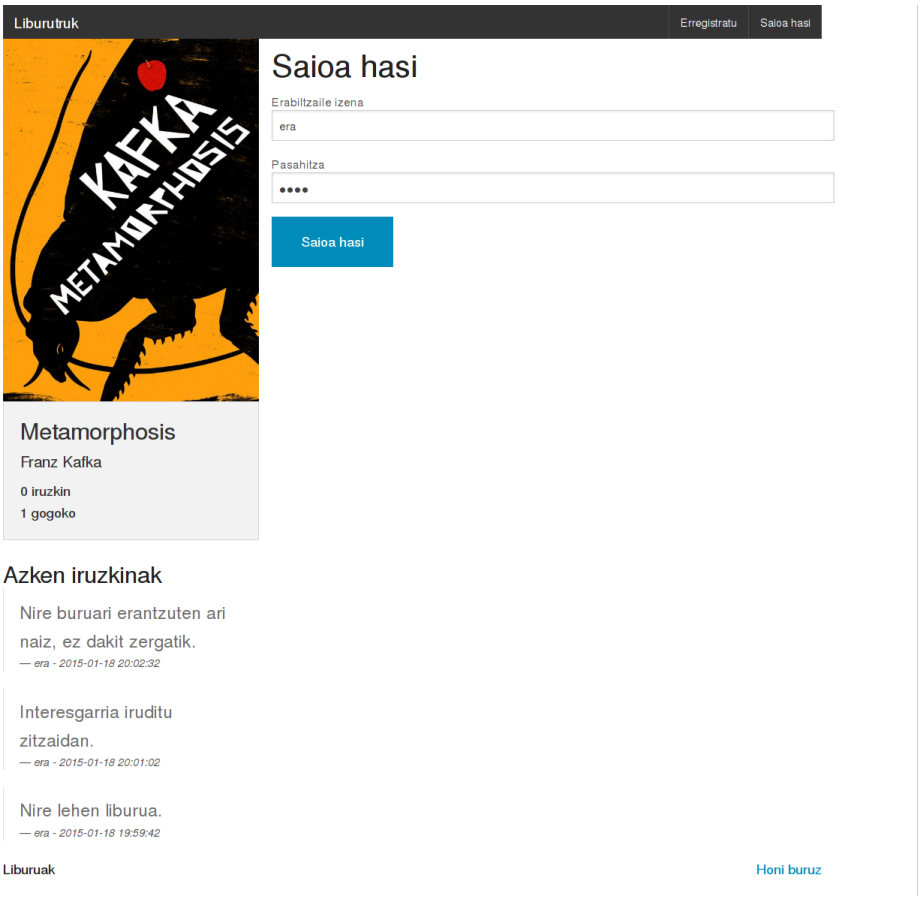
Menuko bigarren aukerara joanez gero ("Nire liburuak"), norberak gehitutako liburuen zerrenda agertzen da. Liburu horiek ikusteko eta ezabatzeko aukera ere badago. Azkenik, "Nire iruzkinak"eta "Nire gogokoak"atalek ere antzerako moduan funtzionatzen dute, iruzkin eta gogokoekin hurrenez hurren.



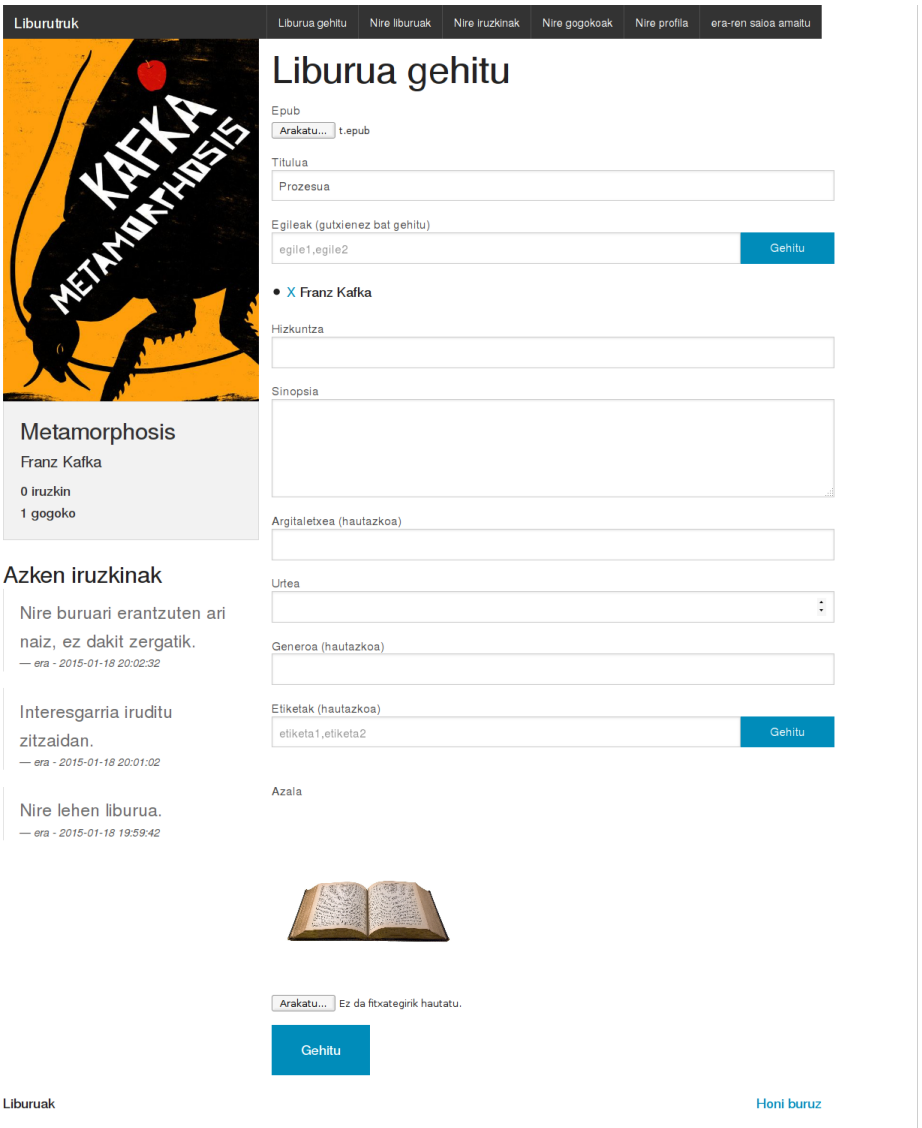
C.5 Irudia: Erregistratu orria.



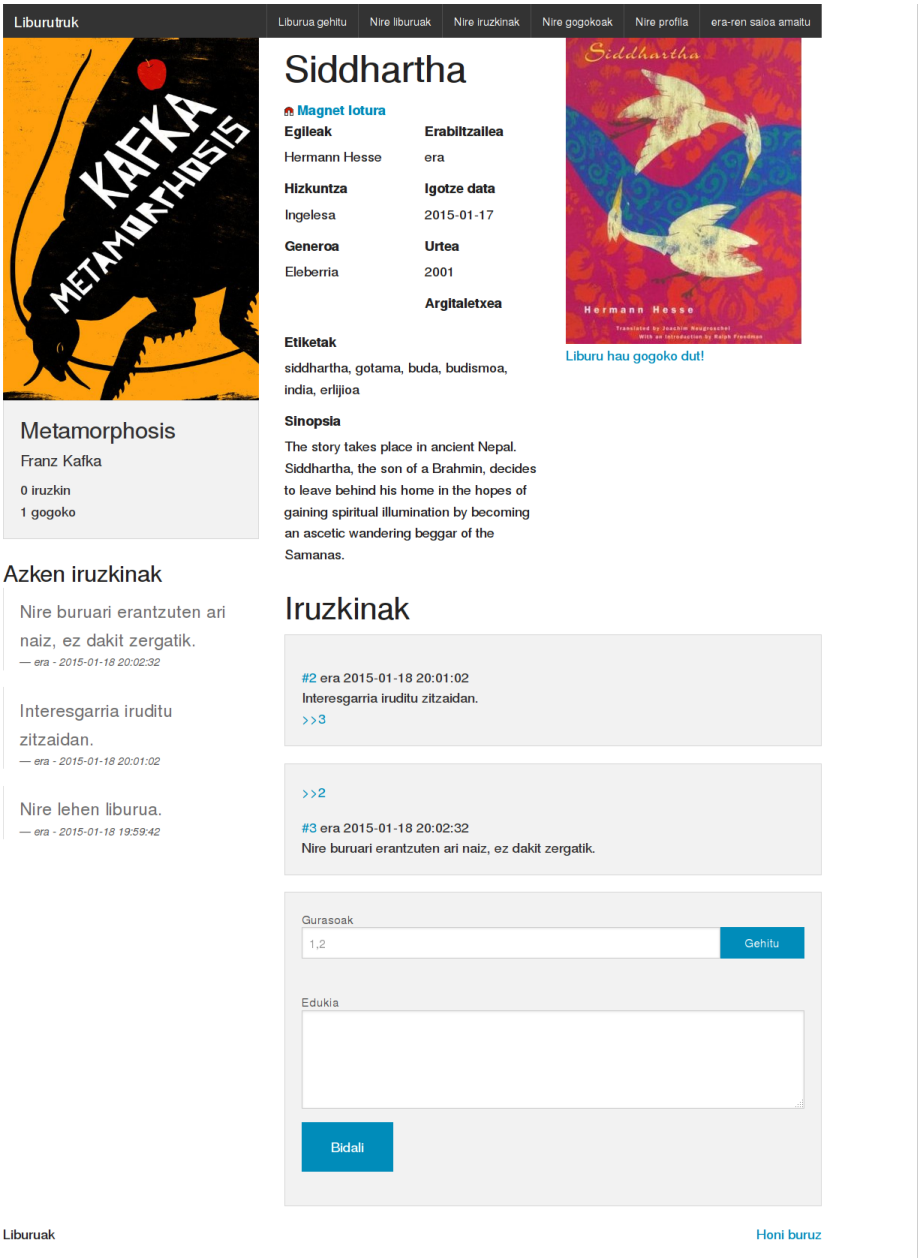
C.6 Irudia: Erabiltzailearen profila.



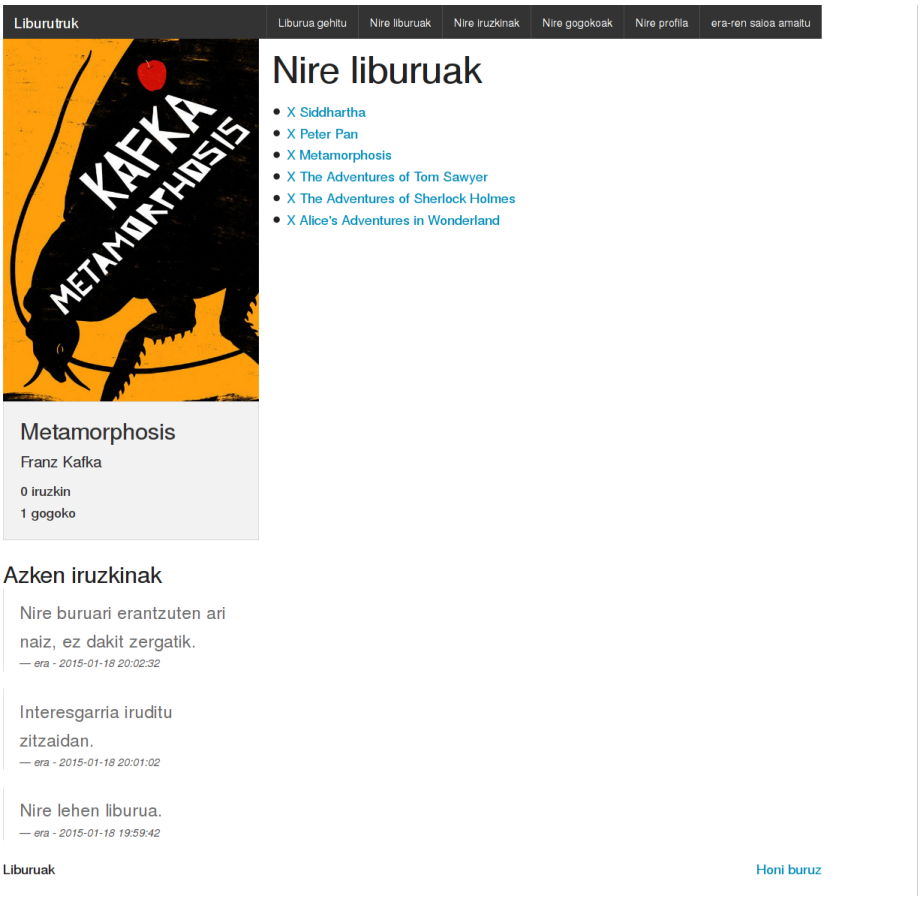
C.7 Irudia: Saioa hasten.



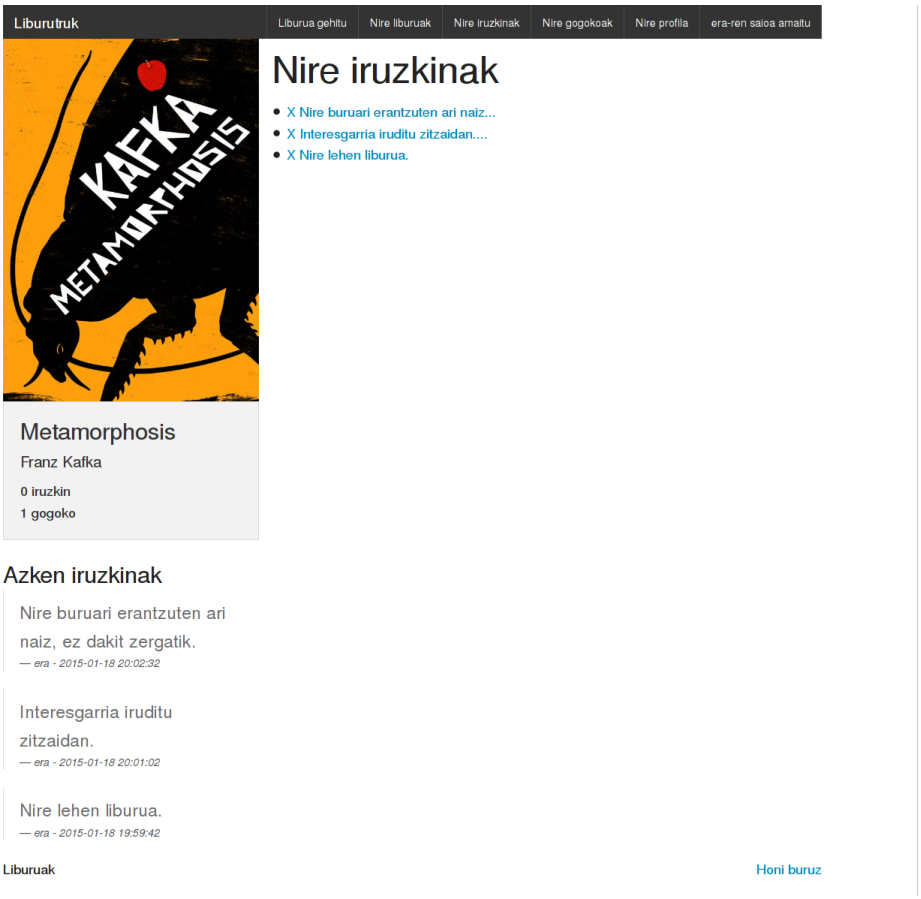
C.8 Irudia: Liburu bat gehitzen.



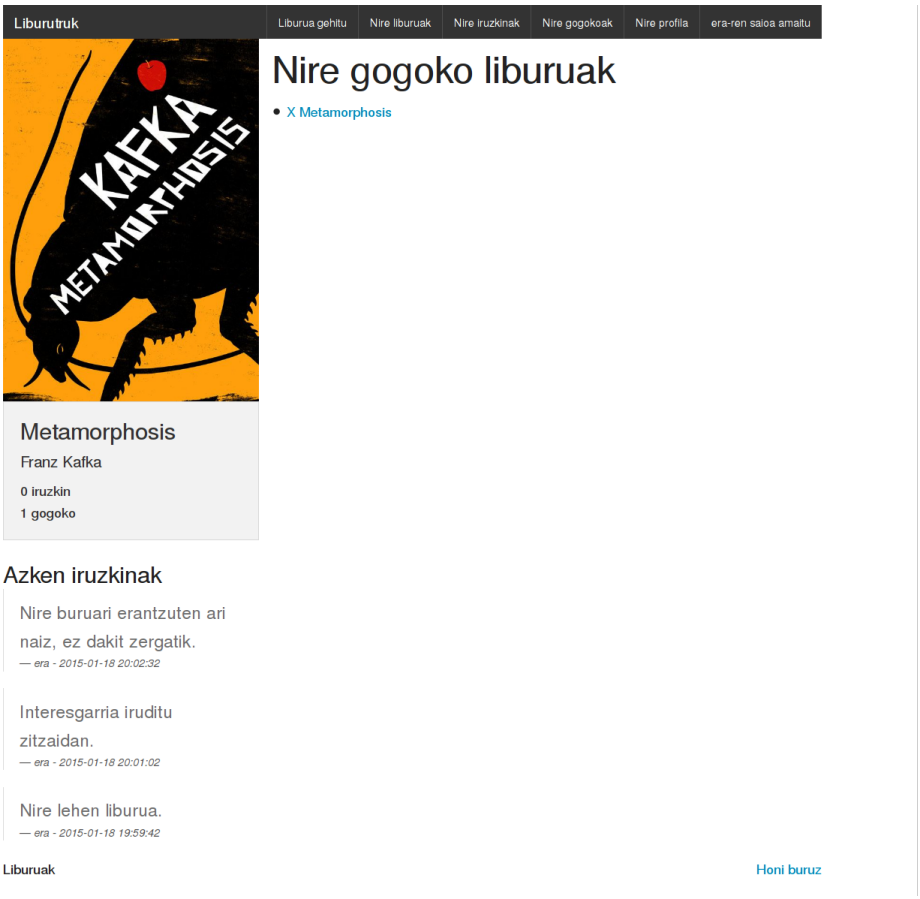
C.9 Irudia: Liburu bat saioa hasita ikusten.



C.10 Irudia: Erabiltzailearen liburuak.



C.11 Irudia: Erabiltzailearen iruzkinak.



C.12 Irudia: Erabiltzailearen gogoko liburuak.