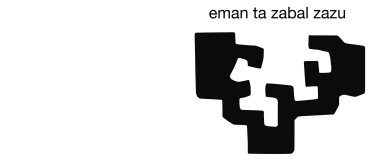


Informatika Ingeniaritzako Gradua

Software Ingeniaritza

Gradu Amaierako Proiektua

Biltegi kudeaketarako sistema baten berrikuntza integrala

Egilea

Iñigo Gonzalez Saralegui

2016

Laburpena

Azken urteetan sektore ezberdinetan lan egiten duten enpresa zein ekoizleek, euren biltegien kudeaketa on bat izatea behar-beharrezkoa ikusi dute, eskariei modu eraginkorragoan erantzun ahal izateko eta, bide batez, kostuetan aurreztu ahal izateko.

Jakina, kudeaketa hau egin ahal izateko denboran zehar hardware zein software soluzio ezberdinak erabili dira, momentuko beharretara moldatuz eskuragarri dagoen teknologia erabiliz.

Azken puntu hau da, hain zuzen ere, proiektu honen helburu nagusia. Aurretik jasotako informazio eta baliabideak erabiliz, biltegi kudeaketarako sistema baten berrikuntza integrala burutzea, gaur egungo enpresa txiki-ertainen beharrei erantzuna emateko teknologia berriak erabiliz eta, bezeroen sistema eragileekiko dependentzia ahaztuz.

Mahaigaineko ordenagailu edo terminaletan instalatzen diren aplikazioak gainditu eta nabigatzaile arrunt batetik, kudeaketa ezberdinak egin ahalko dituzte erabiltze mota ezberdinek, hala nola, biltegiaren egituraren kudeaketa, produktu eta formatuen kudeaketa, sarrera zein irteera eskarien kudeaketa... orohar, biltegi baten gestioa egiteko beharrezko atazak.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Proiektuaren eraketa

1.1 Sarrera

Dokumentu hau Euskal Herriko Unibertsitatean, Informatika Fakultatean egindako Gradu Amaierako proiektuaren memoria da. Dokumentu honetan proiektuaren nondik norako garrantzitsuenak azaltzen dira.

1.2 Motibazioa

Gradu-Amaierako Proiektua gradua amaitzeko betebehar soil bat bezala ikusi beharrean, graduan ikusi eta ikasitako gaitasunak proban jartzeko aukera bezala ikusi da. Honetaz gain, aurretik lantzeko aukera izan ez den ingurune eta tresnak erabiltzeko aukera bikaina da.

Honengatik, MVC proiektu bat lantzeko erabakia hartu da, ASP.NET eta C# lengoaiak muin bezala hartu eta beste hainbat tresna eta teknologiekin uztartuz. Gaiari dagokionez, lagunartean eta familian aurrez lantzen ari zen gai bat izanik, aukera hau baliatu nahi izan da.

1.3 Proposatutako soluzioa

Proiektu honetan proposatzen den biltegiak kudeatzeko sistema baten berrikuntza integrala burutzeko, aplikazioari dagokionean bi lerro nagusi ezberdintzen dira, alde batetik mahagaineko erabiltzaileentzat pentsatuta dagoen atala eta bestetik, terminaletatik biltegiko langileek lan egiteko erabiliko dutena.

Web-aplikazio berdinean integratuko dira biak, baina, erabiltzaile motaren arabera ezberdinduko dira eta, honetaz gain, interfaze ezberdina izango dute, nahiz eta web aplikazioa osoa *responsive*[1] izango den.

Mahaigaineko aplikazioan biltegiaren konfigurazioa eta produktu zein eskariei buruzko informazioa kudeatu ahal izango da, hala nola, erabiltzaileek egiten dituzten eragiketei buruzko informazioa kontsultatu. Aukera hau bakarrik kudeatzaileei emango zaie.

Terminaleko aplikazioari dagokionez, langileek eskariak zerbitzatu eta kudeatu ahal izateko aukerak soilik emango zaizkie, materialaren birkokateak egiteko gaitasunaz gain.

1.4 Dokumentuaren egitura

Dokumentua modu honetan egituratua dago:

- Proiektuaren eraketa
- Proiektuaren kudeaketa plana
- Aurrekarien azterketa
- Sistemaren funtzionalitateen analisia
- Softwarearen diseinua
- Softwarearen inplementazioa
- Sistemaren ebaluazioa eta hobekuntzak
- Jarraipena eta kontrola

[1] Responsive deritze eduki zein egiturak bistaratzen ari den gailuaren pantailara egokitzeko gai diren aplikazioei

- Ondorioak
- Eranskinak
- Bibliografia

2. KAPITULUA

Proiektuaren kudeaketa plana

Kapitulu honetan proiektuaren irismenaren plangintza eta jarraipen eta kontrola zein eratan egingo den definitzen da. Hauek dira aztertzen diren atalak:

- Helburuak
- Irismena
- Lan-metodologia
- Baliabideak
- Proiektua gauzatzeko plana
- Kalitate-plana
- Arriskuak
- Parte interesatuak

2.1 Helburuak

Helburu batzuk Gradu Amaierako Proiektuaren izaerak berak definitatukoak dira, memoria idaztea esaterako. Gainontzeko helburuak proiektuaren irismena finkatzean definitu dira, eta posible da proiektu amaieran aldaketak jasatea. Ondorengo hauek dira proiektuaren helburuak:

- Helburu nagusiak
  - Biltegiak kudeatzeko web-aplikazio bat garatzea
    - Biltegiaren kudeatzailearentzat mahaigaineko aplikazioa egitea.
    - Biltegiko operarioentzat terminalentzako aplikazio mugatua egitea.
  - Proiektuaren garapenaren nondik norakoak biltzen dituen memoria idaztea.
  - Proiektuaren defentsarako aurkezpena prestatzea.
- Helburu gehigarriak
  - Erraz mantendu eta hobetzeko aukera ematen duen sistema bat garatzea.
  - Azken web teknologiak erabiltzea.
  - Objektuen bidez datubasearekin lan egiteko gai izatea.
  - C#, ASP.NET, Bootstrap... ikastea.

2.2 Irismena

Iada existitzen den datubase bat baliatuz, biltegiak kudeatzeko gai den web-aplikazio bat egin nahi da. Funtsean, proiektuaren irismena, produktu minimo bideragarri bat egitean dago zentraturik.

2.2.1 Betekizunak

Betekizunei dagokienez, bi azpiatal ezberdindu behar dira:

- Alde batetik, EHUko Informatika Fakultateak hainbat jarraibide ematen ditu Gradu Amaierako Proiektuari begira. Horretarako, fakultatearen webgunean EHU, 2016 arautegi bat dago zintzilikatua. Honako hauek dira arautegian definiturik dauden betebehar batzuk:
  - *Gradu Amaierako Proiektua defendatu eta ebaluatu ahal izateko, egiaztatu egin beharko da ikasleak ikasketa planeko gainerako gai guztiak gainditu-ta dituela eta hortaz, gradu amaierako lanari dagozkion kredituak izan ezik, graduko titulua lortzeko beharrezkoak diren gainerako kreditu guztiak dituela.*

  - *Informatikaren Ingeniaritzako Graduan, lan hau Gradu Amaierako Proiektua izeneko derrigorrezko irakasgaiari dagokio, 12 kredituko irakasgaia. 12 ETCS kreditu, gutxienez 300 ordu.*
  - *Matrikulatutako ikasleek fakultateko idazkaritzan aurkeztu beharko dute Gradu Amaierako Proiektuari dagokion memoria eta inprimakina modu digitalean, eta beharrezkoa balitz paperean ere.*
  - *Memoriaren formatuak fakultateko estandarrari jarraituko dio.*

- Betebehar horiek eta GAP araudia dokumentuan definitzen diren guztiak proiektuaren betekizunak izango dira.

- Bestetik, softwareari berari dagozkion betebeharrak:

  - Terminaletako eta mahaigaineko erabiltzaileak ezberdinduak egon behar dute eta ezingo dituzte ataza berdinak burutu, horretarako login sistema ezarri beharko da.
  - Terminaletako eta mahaigaineko erabiltzaileen interfazeak ezberdina izan behar du.
  - Biltegiaren kudeatzaileek (mahaigaina) erabiltzaileak, artikuluak, formatuak, biltegiak eta hauen kokalekuak eta sarrera/irteera eskariak kudeatzeko gaitasuna izan behar dute. Honetaz gain, aplikazioaren erabiltzaileek egiten dituzten operazioen informazioa ikusteko aukera izan behar dute.
  - Biltegiko operarioek beraiei esleitutako eskariak kudeatzeko aukera izan behar dute. Honetaz gain, euren buruei eskariak esleitzeko gaitasuna izan behar dute.
  - Mahagaineko aplikazioak gutxienez, hainbat hizkuntzetan konfiguratzeko aukera izan behar du.

2.2.2 Mugak

Ditugun baliabideak mugatuak izanik, garrantzizkoa da hauek modu arduratsuan erabiltzea. Horretarako, kudeaketa on bat behar-beharrezkoa da. Dena den, kudeaketa-lanak ez luke inoiz traba bat izan behar proiektuaren garatze-prozesuan, honenbestez, proiektuaren irismenean barneratzen dena soilik jasotzen da memoria honetan.

Baliabideen ildotik, erabiliko diren teknologien inguruko ikerkuntza beharrezkoa da, aurretiko ezagutza ez dugun neurrian behintzat. Hauek, garatze prozesuan behar-beharrezkoak

diren ataletan soilik lantzen dira eta, proiektu osorako 300 orduko muga izanik, irismen mailak ondo finkatzeko kontuan hartzen dira.

2.2.3 Produktuaren irismena

Web-aplikazioari dagokion irismena bai mahaigainetik bai biltegiko terminaletatik erabiltzeko aukera izango duen gutxieneko produktu bideragarri funtzionala sortzean dago zentratuta eta ez da haratago joango, mota honetako proiektuek izan dezaketen konplexutasuna kontuan izanik. Hortaz, proiektuaren irismen maila ezberdinak helburu honekin zuzenki loturik daude eta honi dagokion kalitate mailarekin.

2.2.4 Irismen-mailak

Produktuaren irismen-maila hiru mailatan banatu da. Lehenengo biak dira oinarrizko produktu bideragarria burutzeko beteko diren irismen-mailak, eta plangintza hauek burutzeko egin beharreko atazekin osatzen da. Hirugarren maila, oinarrizko produktua betetzeko betekizunetatik haratago joatea suposatuko luke, beti ere plangintzan aurreikusitako denbora baina gutxiago erabili izango balitz aurreko bi mailak betetzeko. Hortaz, hauek izango lirateke hiru mailak:

- Oinarrizko produktu bideragarria: lehen prototipo funtzionala, funtzionamendu onargarriarekin, xehetasunak zaindu gabe.
- Erroreak eta hobekuntzak: xehetasunak zaindu eta erroreak zuzendu.
- Funtzio osagarriak: sistemak behar dituen oinarrizko funtzioetatik aparte funtzio gehiago gehitzea izango da azken iterazio honen helburua.

Bigarren iterazioaren amaieran, kalitate-planak definitzen dituen adierazle gehienak ere bete beharko lirateke.

2.2.5 Irismenaren kudeaketa

Gerta liteke hasiera batean finkatutako irismena aldatu behar izatea, atzerapenak medio edota aurreikusitako denbora baina gutxiago behar izan delako. Edonola ere, proiektuaren irismena aldatzeak proiektuaren mugak edota produktuaren irismena aldatzea ekar lezake.

Beste edozeren aurretik proiektuaren arrakasta bermatu behar da eta aldaketak norabide horretan egingo dira. Horregatik, aldaketak egin behar izanez gero, plangintza berraztertuko litzateke eta beharrezko aldaketak egin irismen berrira egokitzeko.

2.2.6 Lanaren deskonposaketa egitura

LDEak proiektuaren irismena definitzen du eta lana ataza eta azpiatazetan banatzen du, era hierarkiko batean. Azpiataza bakoitza programatu eta gainbegiratu daiteke, baita bere kosteak estimatu ere. Bost azpiataletan banatzen da:

Kudeaketa Proiektuaren kudeaketak plangintza eta azpiatazen jarraipen eta kontrola biltzen ditu. Horretaz gain, arriskuen kudeaketa-plana edota kalitate-plana ere biltzen ditu, besteak beste.

Ikerkuntza Erabiliko diren teknologiak eta garapen-inguruneak hobeto ulertzeko egin beharreko lanari dagokio.

Arkitektura Aplikazioari berari eta bere funtzionalitateei dagokion analisi eta diseinua.

Garapena Denbora gehien dedikatuko zaion atala. Web aplikazioa garatzeko burutu beharreko atazak eta egin beharreko frogak.

Itxiera Proiektuaren bizi-zikloa amaitze aldera, memoria idazteak edota defentsa prestatzeak garrantzia hartuko dute. Proiektuaren garapenarekin batera hauek lantzen joango dira baina garapenaren bukaeran informazio guztia dokumentu txukun eta antolatu batean jaso beharko da.

2.1 irudian, proiektuaren LDEa ikus daiteke, eta, segidan, ataza nagusi eta erdi mailakoen laburpena 2.1 taulan.

2.3 Lan-metodologia

Proiektua garatu ahal izateko ordutegiari dagozkion berezitasunak kontuan izanda (lan-orduetan garatzeko ezintasuna, eskola-orduak...), garrantzizkoa da hasieratik metodologia

Kodea	Ataza	Azalpena
Kudeaketa		
1.1	Plangintza	Proiektuaren plangintza integrala, orduen estimazioak, egutegia, arrisku eta kalitate-planak, etab.
1.2	Jarraipena eta Kontrola	Proiektuaren egoeraren berri emateko zuzendariarekin zein interesatuekin bilerak egin behar dira eta beharrezkoa balitz plangintza aldatu. Proiektuan identifikatutako arriskuak kontrolatu eta kalitate maila bat bermatu; egin beharreko aldaketak ere definituz.
Ikerkuntza		
2.1	Ikasketa	Proiektuan erabiliko diren teknologiak aztertu, aplikazioak ulertu, eta beharrezko informazioa bildu.
2.2	Osagaien bilaketa	Liburutegiak zein baliogarriak izan daitezkeen beste baliabideak bilatu eta gorde.
Arkitektura		
3.1	Analisia	Sistemak beharrezko izango dituen osagaiak aztertu.
3.2	Diseinua	Aplikazioaren arkitektura aztertu eta definitu.
Garapena		
4.1	Web aplikazioa	Web aplikazioaren programazioa analisian eta diseinuan hartutako erabakien arabera.
4.2	Probak	Garapenarekin batera egingo diren frogak, zein garapena amaitzeko egingo direnak.
Itxiera		
5.1	Memoria	Proiektuan zehar egindako lana eta izandako gorabeherak jasotzen dituen dokumentua idaztea.
5.2	Defensa	Proiektuaren defentsarako beharrezkoak diren materialen prestaketa.

2.1 Taula: LDEaren ataza nagusi eta erdi-mailako atazen azalpena.

on bat finkatzea. Lan-metodologia honek akatsak ekidin edota identifikatzeko balio behar du, baita plangintzarekiko izandako aldea identifikatzeko ere.

Helburua oinarrizko produktu bideragarria lortzea izanik, garapena hainbat azpiataletan banatu da, denbora-galerak ekidin eta proiektuaren jarraipen eta kontrola errazteko:

1. Erabiltzaileen kudeaketa.
2. Artikuluei dagozkien atazak inplementatu.
3. Biltegiaren mapari dagozkion atazak burutu.
4. Inbentarioari dagozkien atazak burutu.
5. Lan-aginduei dagozkien atazak burutu.
6. Txostenei edo informazioei dagozkien atazak burutu.

Esan bezala, garapena azpiatalka egingo da, atal horri dagozkion funtzionalitateak amaituz hurrengoarekin hasi aurretik. Ordutegia dela medio eguneroko lan-metodologia definitzea errealista ez denez, asteroko lan-metodologia definitzen da atal honetan; aste bakoitzaren amaieran honak atazak burutu beharko lirateke eta ordena honetan jarraitu:

1. Astean egindako lanaren balorazio azkarra: finkatutako helburuak bete ziren ikusi, eta proiektuarengan nola eragiten duen hausnartu. Hurrengo pausoa egiterako garaian hutsak baztertzeko balioko du.
2. Hurrengo asterako helburuak finkatu: proiektuaren plangintzaren baitan finkatu beharko dira helburu horiek, beti ere malgutasun batekin.
3. Gauzatzea: finkatutako helburuak betetzeko egin behar dena egin.
4. Dokumentazioa: izandako eta dokumentatuak izan behar duten oharrak idatzi, proiektuaren memoriari begira batez ere.

2.4 Emangarri nagusiak

Proiektuak hiru emangarri nagusi izango ditu:

- Memoria: Dokumentu hau, Informatika Fakultateko araudiak agintzen duenari jarraiki, ADDI plataformara igo beharko da irailaren 2a baina lehen.
- Kodea: Garatutako aplikazioaren kodea, memoriarekin batera igo beharko dena.
- Aurkezpena: Proiektuaren defentsan erabiliko den aurkezpena.

2.5 Baliabideak

2.5.1 Software-baliabideak

Proiektua gauzatzeko programazio eta idazketa ugari egin beharko da; ataza bakoitzak software espezifikoa behar duelarik.

Programatzeko zein argitaratze eta depurazio lanak egiteko erabiliko den ingurunea Visual Studio Community 2015 da, Windows 10 gainean, mahagaineko zein eramangarrian. Kode aldaketen eta bertsioen kontrolerako interfaze grafikoa duen Git bezeroa erabiliko da, SourceTree izenekoa. Nabigatzaile bat ere beharrezkoa suertatzen da beharrezko frogak egin eta bilaketak egiteko. Ohitura hutsagatik, Chrome aukeratu da. Argitaratuta dagoen web-aplikazioa ikusi ahal izateko, Sophos (Windows) edo OpenVpn (Android) bitartez VPN pribatu batera konektatu beharko da.

Plangintza zein jarraipen eta kontrola burutzeko, Google Drive-eko aplikazioak erabiliko dira, dokumentu zein kalkulu orriak. Memoria idazteko aldiz, LaTeX motorra erabiltzea erabaki da (TeXstudio aplikazioarekin batera), izaera honetako dokumentuek mantendu beharreko itxura eta egitura zorrotza modu egokian mantentzeko. Dokumentu guzti hauek gordetzeko, Google Drive erabiliko da.

2.5.2 Hardware-baliabideak

Hardware-baliabide aldetik, konputagailua behar-beharrezkoa da, jakina. Xehetasunetan sartu gabe, esan, erabiliko diren konputagailuetako bat mahaigaineko PC bat dela eta bestea PC eramangarri bat.

Horretaz gain, Android gailua ere behar da terminalentzat garatuko den aplikaziorako. Edozein gailu erabili daiteke, garatzeko unean erabiliko den gailua Nexus 4 bat da.

2.5.3 Beste baliabideak

Web aplikazioa nabigatzailean exekutatu ahal izateko, beharrezkoa da zerbitzari batean publikaturik egotea. Honetarako, zerbitzariak argitarapenak egiteko kokalekua konfiguratua izan behar du eta, gainera, datuak eskuratu ahal izateko datu basea beharrezko taulekin.

Aipatzen diren beharrak interesdunetako baten ardura izango dira, beraz, ezin da xehetasun handiagorik eman. Web aplikazioa .NET Framework 4.6 bertsioarekin lan egiteko konfiguratua egonik, zerbitzariak bertsio hau instalaturik egotea beharrezkoa da.

2.6 Proiektua gauzatzeko plana

2.6.1 Egutegia

Proiektu hau, izatez, entitate handiko proiektu bat da.

300 ordu inguruko dedikazioa izan behar du proiektuak, epe luzeko estimazioak egiteak dituen zailtasunekin. Hau kontuan izanda eta, lan-metodologian aipatu den moduan, azpiatalka egingo da lan ahalik eta desbideraketa gutxien egon daitezen.

Dedikazioaren estimazioa zein dedikazioaren jarraipen eta kontrola errazteko, kalkulu orrian taula bat eraiki da, modu horretan erraz egin daitezkeelako eguneraketak, eta baita, desbideraketek proiektuaren osotasunari nola eragiten dioten ikusi ere. Estimazioak 2.2 taularen irudian ikusi daitezke.

2.6.2 Gantt diagrama

Egutegia egitean eraikitako taularekin batera Gantt diagrama bat egin DA; era horretan, denboran zehar kokaturik ikus ditzakegu egin beharreko ataza nagusiak, zein atazen arteko dependentziak. 2.3 irudian hasierako estimazioen Gantt diagrama bat ikus daiteke.

2.6.3 Lan-karga

Hasierako estimazioen taulan eta Gantt diagraman ikus daitekeen moduan, proiektuaren iraupena nahiko luzea da: 225 egun. Lan-karga ez da beti berdina izango. Proiektua ikas-

turteko beste ikasgai batekin batera egingo denez, eta honetaz gain lana dela eta hasierako hilabeteetan nahiko dedikazio-ordu gutxi izango ditugu, dedikazio gehiena maiatzetik aurrera jasoko delarik.

Ataza batzuei erreparatzen badiegu, konplexutasun handiena duena kokalekuen kudeaketa da. Kudeatu behar diren aspektu guztiak direla eta, dedikazio ordu dexente beharko dituela aurreikusten da.

Hau dela eta, maiatzetik aurrera konzentratzen da aplikazioaren garapenaren zati garrantzitsuena.

2.7 Kalitate plana

Proiektuak eginbehar asko ditu eta atal horien guztien kalitatea bermatu behar da. Horretarako, atal honetan hiru gai jorratuko ditugu:

- Kalitatearen adierazleak
- Kalitatea ziurtatzea
- Kalitatearen kontrola

2.7.1 Kalitatearen adierazleak

Kalitatearen adierazleak asko izan balitezke ere, garrantzitsuenak kontuan hartzen saiatuko gara; bi multzo nagusi ditugu adierazleen artean: kuantitatiboak eta kualitatiboak.

Adierazle kuantitatiboak

- Plangintzaren jarraipena: plangintzan definitu diren datak eta epe-mugak errespetatzea.
- Proiektuaren fitxategi kopurua: kalitate onaren seinale ez bada ere, zenbat eta fitxategi gehiago izan geroz eta garatuagoa egongo da proiektua bere osotasunean, produktua hobetuz.

Adierazle kualitatiboak

- Sistemarekiko elkarrekintza: sistema erabiltzerakoan erabiltzaileentzako erabilerraza izan behar du, geroz eta errazagoa izan erabiltzaile eta sistemaren arteko elkarrekintza, produktuaren kalitatea hobea izango baita.
- Aplikazioaren fidagarritasuna: erabiltzaileek egiten dituzten operazioen aurrean, eta batez ere, sartzen dituzten ustekabeko informazioen aurrean erreakzionatzeko duen gaitasuna.
- Kodearen argitasuna: kodearen antolaketa ona, beharrezko komentario eta argibideekin.

2.7.2 Kalitatea ziurtatzea eta kontrola

Aurreko atalean definitutako adierazleak betetzen direla edota horiek betetzeko premia dagoela detektatzeko, proiektuaren bizi-zikloan zehar etengabe egin behar dira egiaztapenak. Garapenaren azpiatal bakoitza amaitu ostean egingo da adierazle horien konprobaketa.

2.7.3 Kalitatearen kontrola

Proiektuaren arrakastari begira, kalitate-maila minimo batzuk betetzea ezinbestekoa da. Kalitate-maila minimo hori betetzen dela egiaztatzeko egiaztapen-zerrenda bat aurki daiteke, C eranskinean. Kalitate maila minimo horietatik aparte, arrakasta bermatua dagoen heinean, arrakastaren maila neurtzeko ere erabilgarria izango da egiaztapen zerrenda hori.

2.8 Arriskuak

Proiektuaren dimentsio eta izaerarengatik hainbat arrisku topatu ditzakegu. Atal honetan, arrisku nagusiak identifikatu, eta horiek ekiditeko eta konpontzeko zein neurri hartuko diren zehaztuko dira.

- **Datuen galera:** egingo den lanaren zati handiena konputagailuekin eta era digitalean egingo denez, kontuan hartu beharreko zerbait da sistema informatikoek huts

egiteko duten joera. Datuak galdu, hondatu edota ezabatu egin daitezke. Horren aurrean, beharrezkoa da segurtasun kopiak egitea. Alde batetik Git zerbitzuari esker, kodean egindako aldaketak gorde eta igo ditzakegu, eta bestetik, dokumentazioari dagozkionak Google Drive zerbitzuan gorde ditzakegu.

- **Proba zerbitzaria eskuragarri ez izatea proiektuan zehar:** Honek proiektua garatzerakoan oztopo handia suposatuko luke eta, honen aurrean, frogak zein proiektuaren defentsako demoa eramangarrian bertan egin beharko litzateke, eramangarria prest izango dela ziurtatuz. Honetarako, datubasea eta IIS zerbitzaria izan beharko genuke instalaturik (IIS Visual Studio-rekin martxan jar dezakegu, ezer konfiguratu behar izan gabe). Hala ere, hau gertatzeko aukerak ia nuluak dira.

- **Aplikazioaren garapenean arazoak:** aplikazioak garatzerako garaian ohiko arazo bat izaten da aplikazioan aldaketaren bat egin eta aplikazioak funtzionatzeari uztea. Hori gertatzean denbora asko galdu daiteke, eta, erabiliko diren teknologiak ezezagunak direnez, oraindik ere gehiago areagotzen da arriskua. Hori, neurri batean, Git zerbitzuaren erabilerari esker ekiditen da.

- **Plangintza ez jarraitzea:** plangintza jarraitzea proiektuaren arrakastarako gida da, eta plangintza hori ez jarraitzeak proiektuaren porrota ekar lezake. Horretarako, plangintzan bertan definitu diren prozesuak jarraituko dira, desbideraketak gutxitu eta detektatzeko.

2.9 Parte interesatuak

Proiektuaren interesatu nagusia proiektuaren egilea bera da. Proiektuaren erantzukizuna harena da, eta harena da proiektuaren arrakastarako bidean eman beharreko pauso guztiak ematearen ardura.

Proiektuaren interesatu nagusia da ere, lortuko den produktua erabiliko duen pertsona, Aitor Crespo. Bere ardura izango da produktuaren argitarapenak jasoko dituen zerbitzaria kudeatzea eta lan egiteko moduan egotea. Berak definitutako beharrak kontuan hartzen dira produktuaren irismena definitzeko.

Beste maila batean, baina proiektuaren interesatu nagusiak izango da proiektuaren zuzendaria ere, kasu honetan, Imanol Usandizaga. Honen ardura izango da proiektuaren egilea gidatu eta aholkatzea eta baita egin beharreko zuzenketak adieraztea ere. Zuzendarien ardura izango da, memoria entregatzen denean, behar den txostena egitea ere.

Hurrengo interesatua proiektua aztertu eta ebaluatuko duen epaimahai taldea da. Gradu amaierako proiektuaren arautegiaren arabera epaimahaia ikasi den espezialitateko irakasleez osatua egongo da, Software Ingeniaritzako irakasleez kasu honetan.

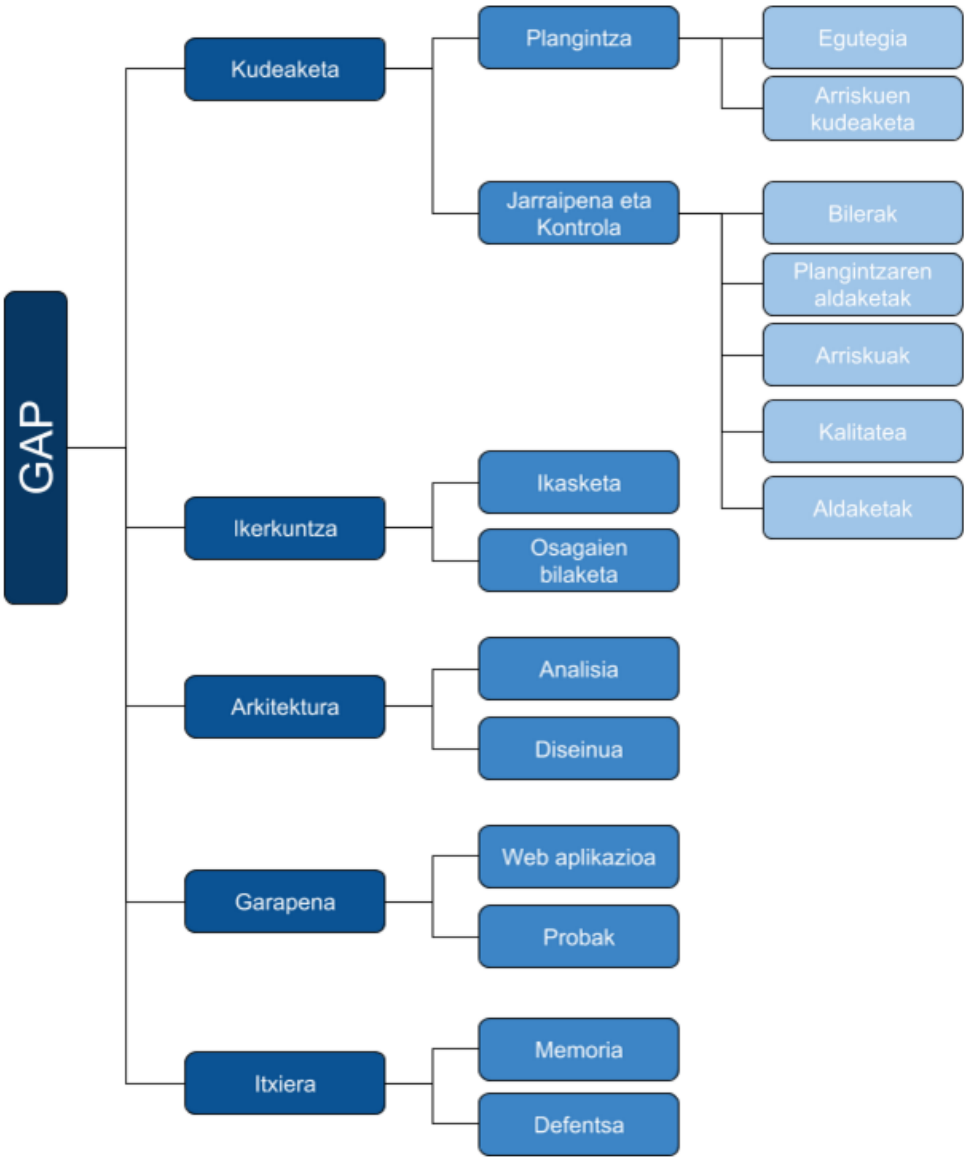
Azkenik, oraindik ezagunak ez diren interesatuak daude. Interesatu horiek proiektuan interesa dutelako edota proiektuaren helburuak betetzeko beren laguntza behar dugulako azalduko dira.

2.9.1 Komunikazio plana

Zuzendariarekin zein beste interesatuekin komunikazioa aurrez aurre egitea lehenetsiko da, telefonoz edota emailez egitea baino, betiere kasuaren arabera noski.

Epaimahaiarekin ez da komunikaziorik izango defentsaren eguna arte.

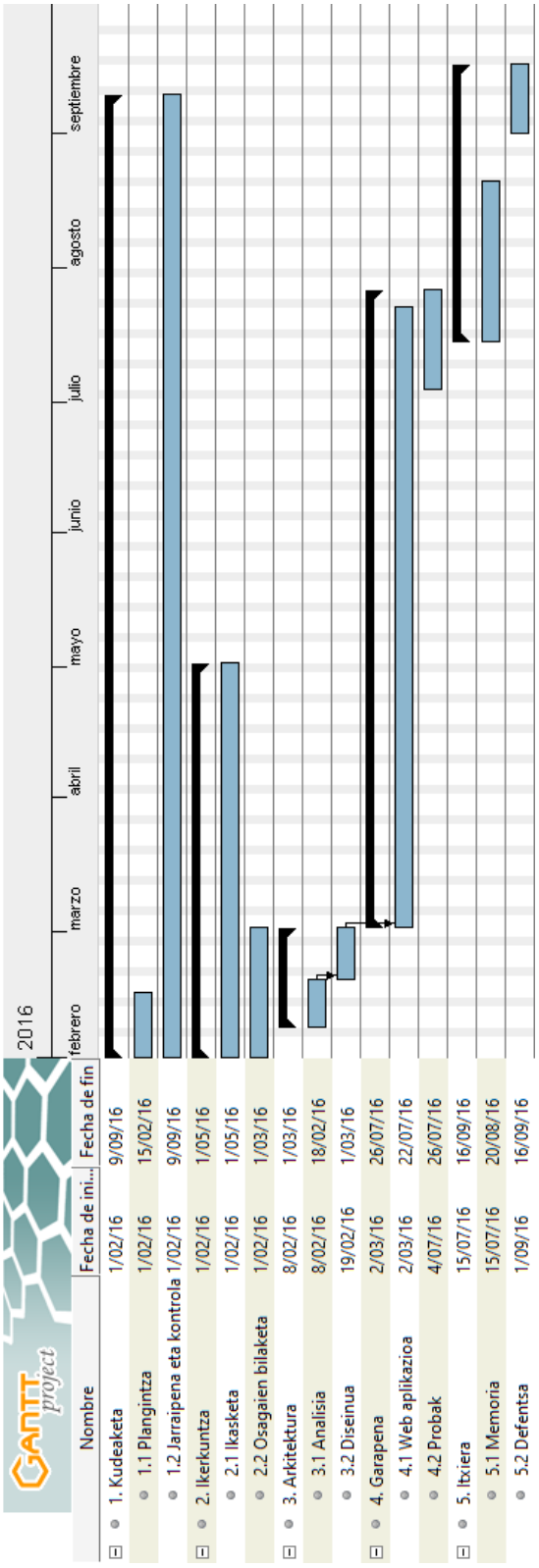
Oraindik ezagutzen ez diren interesatuak azaltzean, interesatu bakoitzarekin adostuko dira komunikazio moduak.



2.1 Irudia: Proiektuaren LDEa

Kodea	Atazaren deskribapena	Estimazioa	Hasiera	Amaiera	Iraupena (egun)
1	Kudeaketa	35	1/02	9/09	218
1.1	Plangintza	14	1/02	15/02	14
	Egutegia definitu	12	1/02	15/02	14
	Arriskuen kudeaketa plana	2	1/02	15/02	14
1.2	Jarraipena eta kontrola	21	1/02	9/09	218
	Bilerak	5	1/02	22/07	171
	Plangintzaren aldaketak	3	15/02	9/09	204
	Arriskuak eguneratu	3	1/02	9/09	218
	Kalitatea kontrolatu	5	1/02	9/09	218
	Aldaketak kudeatu	5	1/02	9/09	218
2	Ikerkuntza	40	1/02	1/05	90
2.1	Ikasketa	30	1/02	1/05	90
2.2	Osagaien bilaketa	10	1/02	1/03	30
3	Arkitektura	20	8/02	1/03	23
3.1	Analisia	10	8/02	18/02	10
3.2	Diseinua	10	19/02	1/03	12
4	Garapena	125	2/03	26/07	144
4.1	Web Aplikazioa	115	2/03	22/07	140
	Erabiltzaileen kudeaketa	20	2/03	23/03	21
	Artikuluen kudeaketa	15	4/04	22/04	18
	Kokalekuen kudeaketa	40	1/05	16/06	45
	Lan-aginduen kudeaketa	20	20/06	10/07	20
	Inbentarioaren kudeaketa	10	11/07	17/07	6
	Txostenak	10	18/07	22/07	4
4.2	Probak	10	4/07	26/07	22
5	Itxiera	90	15/07	16/09	61
5.1	Memoria	80	15/07	20/08	35
5.2	Defentsa	10	1/09	16/09	15
		310	1/02	16/09	225

2.2 Irudia: Dedikazioaren estimazioa



2.3 Irudia: Hasierako estimazioen Gantt diagrama

3. KAPITULUA

Aurrekarien azterketa

Kapitulu honetan proiekturi bere osotasuna emango dioten osagaien informazioa bildu da; batetik erabiliko diren teknologien inguruko ikerlanak eta, bestetik, aurretik antzeko kudeaketak egiteko erabili izan den soluzioen aipamena.

Bildutako informazioak, proiektuaren ingurunea argiago izateko baliozkoa izango da.

3.1 Aurreko sistemak

Merkatua aztertzen hasi ezkero, mota eta tamaina ezberdinetako soluzioak aurki ditzakegu. Hala ere, hauek orokorrean tamaina erdi edota tamaina handiko enpresei zuzenduta egoten dira eta, horren neurrian noski, izaten dira euren prezioak ere.

Gure kasuan, tamaina txiki eta erdiko enpresei zuzendu nahi da produktua eta, honengatik, sortuko den produktuaren erabiltzailea izango denak aurretik erabili izandako soluzioaren nondik norakoak aztertuko dira.

Soluzio honen funtsa datubase zentrala da. Bezero bezala lan egiten duten terminal guztiak datubase horren kontra operazioak egiten dituzte, betiere eskari eta stock-ari lotutakoak.

Honek aukera mugatuak eskeintzen zituen bai inplementazio aldetik (terminalek egin ditzaketen operazio urriak) bai plataforma konkretu bati lotutako sistema delako (Windows Mobile).

Aplikazioa soilik terminal konkretu batzuetarako egoteak suposatzen duen limitazioak eta, kudeatzaile nagusiak ordenagailu batetik kudeaketa egiteko aukera ez izateak gaur egun sistema hauen bideragarritasuna ia ezinezkoa egiten dute. Hainbat terminaletarako aplikazioak prestatu behar izateak bakarrik sekulako kostua suposatzen du, gero azken produktuaren prezioan zuzenki zerikusia izango duena.

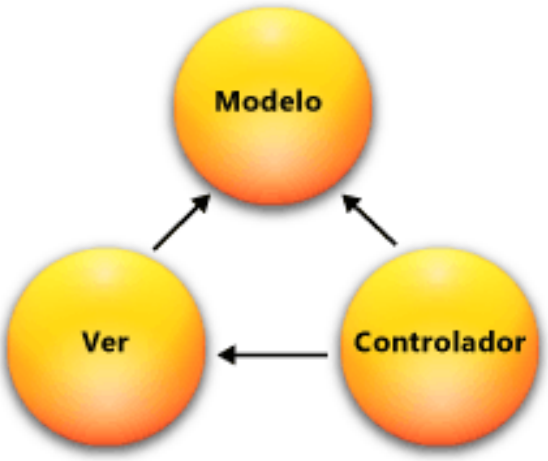
Bi arrazoi hauek dira, hain zuzen ere, produktu berri bat teknologia berriekin egitearen arrazoi nagusiak.

3.2 Teknologiak

Dokumentazio honen lehen bi ataletan iada aipatu dira proiektuaren garapenean erabiliko diren teknologia batzuk. Horiek eta sistema gauzatzeko erabiliko diren beste teknologia batzuk ere atal honetan aurkezten dira.

3.2.1 ASP.NET MVC

Model-View-Controller (MVC) arkitektura-ereduak aplikazio bat hiru geruzatan banatzen du: eredua (model), aurkezpena (view) eta kontrolagailua (controller). 3.1 irudian ikus daitezke hauek eta erlazionatzeko modua. ASP.NET MVC-k ASP.NET-en Web Form teknologia ez den beste aukera bat ematen dien garatzailei eta, gainera, iada existitzen diren ASP.NET-en ezaugarriekin integratzen da.



3.1 Irudia: ASP.NET MVC arkitektura

Esan bezala, MVC disenu eredu estandar bat da, garatzaile askok ezagutzen dutena. Jarrian, eredu hau osatzen duten osagaien xehetasunak:

- **Ereduak.** Eredu objetuak dira aplikazioaren datu domeinuaren logika inplementatzen dutenak. Sarritan, informazioa datubasetik eskuratzen dute eta ereduaren aldaketak bertan gordetzen. Adibidez, Produktua izeneko objektu batek datubasetik informazioa eskura lezake, informazio horrekin lan egiten eta eguneratutako informazioa datubasean gorde.

  Aplikazio txikietan, eredua gauza kontzeptual bat soilik izan ohi da, fisikoki eraginik izan gabe. Hau da, aplikazio batek soilik informazio bat irakurri eta aurkezpen geruzara bidaltzen badu, aplikazio horrek ez du izango definitutako eredurik. Kasu horretan, datu multzo horrek hartuko luke eredu objektu baten portaera.

- **Aurkezpena.** Aurkezpenak dira erabiltzaile-interfazea bistaratzen duten osagaiak. Interfaze hauek ereduen datuen arabera sortu ohi dira. Aurreko adibideari jarraituz, erabiltzaile-interfaze honek Produktua ereduaren eremuak bistaratuko lituzke.

- **Kontrolagailuak.** Hauek erabiltzailearen elkarrekintzak kontrolatzen dituzte, eta ereduekin lan egin, amaieran aurkezpen bat aukeratuz erabiltzaileari informazioa bistaratzeko. MVC aplikazio batena, aurkezpen edo bistak, soilik informazioa aurkezten du; kontrolagailuak erabiltzaileen aginduei erantzuten die.

MVC ereduak aplikazioaren aspektu ezberdinak banantzen ditu (sarreren logika, negozio-logika eta erabiltzaile-interfazeen logika). Ereduak logika bakoitza aplikazioan non aurki daitekeen definitzen du. Interfazeen logika aurkezpenari dagokio. Sarreren logika kontrolagailuari. Negozio-logika ereduari. Banaketa honek aplikazio bati konplexutasun handia eman ahal izatea ahalbidetzen du, momentu bakoitzean inplementazioaren atal konkretu batean soilik zentratuz.

Hiru elementu hauen arteko lotura ahulak paraleloan egiten diren garapenak errazten ditu. Garatzaile batek aurkezpenean lan egin dezake, beste bi garatzailek kontrolagailu eta ereduetan lan egiten duten bitartean (bakoitzak bere aldetik) [Tom Dykstra, 2015].

3.2.2 C#

C# objektuetara orientaturiko lengoaia seguru eta elegantea da, garatzaileei .NET Framework gainean exekutatzen diren aplikazio trinko eta seguruak garatzea ahalbidetzen

diena. Windows-en exekutatuko diren aplikazioak zein, XML web zerbitzuak, banatutako osagaiak, bezero-zerbitzari aplikazioak, datubase aplikazioak... sor daitezke.

Bere sintaxia sinplea eta ikasteko erraza da. Giltzetan oinarrituta dagoenez, aurretik Java zein C ala C++ erabili dutenei ezaguna egingo zaie. Hau honela izanik, epe laburrean C#-ekin modu produktiboan lan egiteko gai izango dira. Bere sintaxiak C++-ko konplexutasun askori aurre egiten die, aukera berriak ahalbidetuz: NULL balioak, enumerazioak, delegatuak *lambda expresioak* [1] zein Javan existitzen ez den memoriaren zuzeneko atzipena.

Prozedura eta mota generikoak erabiltzea ahalbidetzen du, mota ezbedin eta iteratzaileetan errendimentu eta segurtasun handiagoa eskeintzeaz gainera. Kapsulaketa, herentzia eta polimorfismo kontzeptuak ere onartzen ditu. Oinarrizko klasean definitutako prozedurak hauek jasotzen dituztenek birdefinitu ditzakete overrides teknika erabiliz. Gainera override klausula honek nahi gabe gauzak berdifinitzea galerazten du.

C#-k osagai ezberdinen sorketa errazten du, eraiketa teknika berritzaileak erabiliz, esaterako:

- Enkapsulatutako metodoen firmak (delegatu izenekoak).
- Propietateak, aldagai pribatuen atzipena egiteko erabili ohi direnak.
- Language-Integrated Query (LINQ) datu eredu askoren gainean kontsulta operazio ezberdinak egiteko funtzioak eskuragarri jartzen dituena
- Eta beste zenbait berrikuntza.

3.2.3 ASP.NET Core API

.NET Core, .NET Framework-en bertsio modular bat da, plataforma ezberdinen artean erabiltzeko sortu dena. Gainera, kode irekikoa da eta garatzaile komunitatearen aportazioak onartzen ditu.

.NET Framework-en azpibertsio bat izanik, bertan jasotzen diren funtzionalitateak beste plataformetan berrerabili ahalko dira, aurretik existitzen ez zena (plataforma bakoitzak bere funtzionalitateak zituen). Abantaila hauetaz gozatu ahalko duten plataformak mahaigaineko Windows eta mugikorretako Windows dira.

[1] Funtzio lokalak funtzioen deietan argumentu bezala erabili ahal izatea ahalbidetzen duten funtzio anonimoak dira lambda expresioak. Adibidez x => x * x expresioak x-en karratua itzuliko luke.

Baina, honetaz gain, *Xamarin* [2] bezalako erramintak erabiltzen direnean IOS eta Android-entzat ere eskuragarri egon behar dute funtzionalitate hauek. Hau dela eta garrantzitsua da .NET Core-ren agerpena. Honetaz gain laister Linux eta MAC-en ere erabili ahalko da.

.NET Core modularra izanik, azpipakete txikiagoetan argitaratzen da *NuGet* [3] paketeetan. Modu honetan bere eboluzioa azkarra da eta gainera, garatzaileek bakarrik behar dituzten funtzioak jaitsi eta erabiliko dituzte [Microsoft, 2016].

3.2.4 Bootstrap

Boostrap *css* [4] *framework* [5] bat da. Beste hitzetan, css fitxategi bilduma bat da, web gune batean txertatu eta css koderik ikutu behar izan gabe web orri bat minututan maketatu ahal izateko aukera ematen duena.

Honetaz gain, beste aukera batzuk ere ematen ditu, web-gunean erabili ditzakegun ikono eta beste hainbat elementu, JQuery liburutegiaren laguntzaz, esaterako taulak, listak, blokeak, oharrak, irudiak, formularioetako formatu, botoi eta eremu bereziak... Gainera, abantaila argi bat du, webguneak responsive egiteak ez du inolako zailtasunik suposatzen.

Twitterrek garatu eta mantentzen du eta hedagarria da. Oinarri bezala dituen estiloak osa edo alda ditzakegu nahi izanez gero. Hori bai, honetarako eta framework-a erabili ahal izateko aurretiko css ezagutzak behar-beharrezkoak dira. Elementuei soilik estiloak aplikatzeko html elementuen *class* elementuak erabili behar izango ditugulako.

Gainera, javascript desaktibaturik duten orrialde gehienetan ez da ibiliko, hortaz, kontuan izan beharreko zerbait da.

3.2.5 JQuery

JQuery Javascript liburutegi bat da, HTML dokumentuetako elementuekin elkarrekintza errazten duena. Hauek manipulatu, animatu, AJAX interakzioak gehitu eta abar egin dai-

[2] Xamarin, dispositibo mugikorrentzat aplikazioak sortzeko framework bat da, plataforma ezberdinetarako (Android, iOS..) garapen bakar bat egitea ahalbidetzen duena. Azken urtean Microsoft-ek eskuratu eta Visual Studio-n integratu du.

[3] NuGet, Visuak Studio-n liburutegiak eguneratu zein erraminta berriak instalatzeko balio duen pakete kudeatzaile bat da.

[4] Cascade Style Sheet, edo estilo-orriak deiturikoak, web orrialde baten elementu ezberdinen itxura aldatzeko erabiltzen diren fitxategiak dira.

[5] Framework deritzo iada egindako liburutegi edo funtzionalitate ezberdinak batzen dituen liburutegi edo sasi-aplikazio bati. Hauek lana aurreztu edo garapenak errazteko erabilgarriak izan ohi dira.

teke honi esker, webgunea birkargatu behar izan gabe. Kode irekiko software librea da edozein proiektutan erabili daitekena.

Azken finean, liburutegi honetan jasotzen diren funtzionalitateak javacript kodean eginda daude eta honi esker funtzionalitate hauek erabili ahal izateko beharrezko kodea idaztea aurrezten digu.

Erabili ahal izateko soilik fitxategi bat txertatu behar zaio html web orriari. Funtzioak erabili ahal izateko **$()** edo **jQuery()** alias-ak erabiltzen dira.

```
$(document).ready(function(){})
```

Goiko kodea erabili behar da web-orrialdea iada kargaturik dagoela jakiteko, behin prest dagoela jakinda beste funtzionalitateak exekutatu ahal izango ditugu. Adibidez,

```
$(".activo").slideToggle("slow")
```

kodeak *activo* klasea duten elementu guztiak animatzen ditu.

3.2.6 JSON

JSON (*JavaScript Object Notation*) objektuen letrazko adierazpidean oinarritutako datuen trukerako formatu arina da. Gizakientzako irakurtzeko erraza da, eta makinentzat sinplea hau sortu eta interpretatzea. Horrez gain, duen laburtasunak bezeroaren eta zerbitzariaren arteko datu-fluxuaren tamaina murrizten du.

Bi egitura erabiltzen ditu:

- Objektua: izena-balio bikoteak.
- Array-a: Balioen zerrenda ordenatua

Adibidea:

```
1 {
2   "produktuak":
3   [
4     { "izena":"Bananak" , "kategoria":"Janaria" },
```

```
5       { "izena":"Aulkia" , "kategoria":"Altzariak" },
6       { "izena":"Kamiseta" , "kategoria":"Arropa" }
7     ]
8   }
```

3.2.7 Entity Framework

Entity Framework datuetara bideratutako software aplikazioak garatzea ahalbidetzen duen ADO.NET teknologia-multzo bat da. Arkitektu eta garatzaileek bi helburu oso ezberdini aurre egin behar izan diete orain arte: Eredu, erlazio eta logikak definitu eta modelatu behar dituzte eta, horretaz gain, datuak gordeko dituzten motore edo sistemekin lan egin. Datuak gordeko dituzten sistema hauek oso ezberdinak izan daitezke euren artean eta, bakoitzak bere xehetasunak izango ditu noski.

Entity Framework-ek aukera ematen die garatzaileei datuekin lan egiteko datubaseko tauletaz eta hauen arteko erlazioetaz arduratu behar izan gabe. Hau, domeinuko objektu eta propietateak erabiliz lortzen da. Lortzen den abstrakzio-maila altua da, eta honi esker aplikazio aberatsagoak sor daitezke kode lerro gutxiagorekin, aplikazio tradizionaletan baino. Entity Framework .NET Framework-eko osagai bat izanik, hau instalaturik duen edozein konputagailutan exekuta daiteke 3.5 SP1 bertsiotik aurrera.

Nagusiki lan egiteko bi modu daude: datu-ereduei bizia ematea alde batetik eta, objektuei datuak esleitzea bestetik.

Datu-ereduei bizia emateak, aplikazioaren objektu edo klaseek eta hauek dituzten propietateak zein beste klaseekiko erlazioak, erabiliko dugun datu basean gordetzeko edo eguneratzeko beharrezko kontsultak exekutatzeaz arduratuko dela esan nahi du. Noski, aurretik, eredu hauen taulak sortuko ditu.

Objektuei datuak esleitzeak, aplikazioan definiturik ditugun objektuen instantzia egin eta hauen bitartez datubaseko datuak eskuratzea esan nahi du, beharrezko *LINQ* kontsulta erabiliz, SQL agindurik idatzi behar izan gabe.

Orokorrean, proiektu handietan bi aukerak erabiltzen dira, baina, erabilgarria suertatzen da aurretik datubaseak definituta dituzten aplikazioetan, Entity Framework erabiliz aplikaziorako datu-eredu eta hauen erlazioak eskura ditzakegulako, hauek eskuz idatzi behar izatea aurreztuz [Tom Dykstra, 2015].

4. KAPITULUA

Funtzionalitateak

Kapitulu honetan garatuko den produktuaren funtzioen azterketa sakona egiten da, diseinuari ekin baino lehen kontuan hartu beharrekoak zehazteko. Lehendabizi, sistemak izango dituen funtzionalitateen analisia egiten da, produktuaren irismenarekin bat eta, ondoren, sistema osatzen duten zatien analisi zehatzagoa.

4.1 Funtzionalitateen analisia

Funtzionalitateen analisia egiteko aurretik, interesdunen artean aurkitzen den Aitor Cresporekin bildu eta beharren araberako funtzionalitateen aukeraketa egin da. Osagarriak izan daitezkeen funtzionalitateei buruz ere hitzegin da baina hauek soilik oinarrizko produktu bideragarritik kanpo geratu dira eta funtzionalitate osagarri bezala sailkatu dira.

Oinarrizko produktu bideragarria:

- Erabiltzaileen kudeaketa.
- Artikuluei dagozkien atazak inplementatu.
- Biltegiaren mapari dagozkion atazak burutu.
- Inbentarioari dagozkien atazak burutu.
- Lan-aginduei dagozkien atazak burutu.

- Txostenei edo informazioei dagozkien atazak burutu.
- Hizkuntza aldaketa.

Funtzionalitate osagarriak:

- Kokalekuen mapa bisuala eta kokalekuen kudeaketa honen bitartez.

Irismenetik kanpo:

- Inbentarioen kudeaketa konplexua.
- Inbentarioen birkokaketak.
- Picking bideak.
- Lizentzien kudeaketa.

Aurretik aipatu bezala, oinarrizko produktu bideragarria garatzea da proiektuaren honen helburua.

4.2 Softwarea

Nahiz eta proiektu berdinaren parte izan, erabiltzaile moten arabera bi aplikazio ezberdindu behar ditugu: mahaigaineko aplikazioa eta aplikazio mugikorra.

4.2.1 Mahaigaineko aplikazioa

Ordenagailuetan erabiliko da, eta sistemaren kudeatzaileek erabiliko dute. Honen helburua biltegi baten eta honi dagokion funtzionamenduaren kudeaketa egin ahal izatea izango da.

Horregatik, irismenean aipatu bezala aplikazioak hainbat funtzionalitate izango ditu:

Erabiltzaileen kudeaketa

Aplikazioa erabiliko duten erabiltzaileek hau erabili ahal izateko erabiltzaileak izan beharko dituzte. Bi erabiltzaile mota izango dira: mahaigainekoa (Admin) eta mugikorretakoa (Terminal). Hortaz, erabiltzaile hauek sortu zein editatzeko ala ezabatzeko aukera izan behar du aplikazioak. Erabiltzaileak sortzean, beteko duten rola eta interfazea zein hizkuntzetan ikusiko duten aukeratu ahalko da.

Hemen barneratzen da ere, erabiltzaileek egin beharreko saio hasiera zein bukaera atazak.

Artikulu eta euren formatuen kudeaketa

Biltegi bat artikuluak biltegiratzeko erabiltzen da noski, eta artikulu hauek hainbat formatutan biltegiratuta eduki ditzakegu biltegi batean. Horretarako, artikuluak sortu zein aldatu ala ezabatzeko aukera izan behar dugu, eta honetaz gain, behin produktua sortuta, produktu horrentzat formatuak sortzeko aukera izan behar dugu. Noski, formatu hauek editatu zein ezabatzeko aukera ere izan behar da.

Biltegi maparen kudeaketa

Enpresa batek biltegi bat baina gehiago izan ditzake, eta aplikazioa honetarako prest izan behar da. Honetaz gain, biltegi bat hainbat eremutan banatzen da, eta eremu horiek ordenazio ezberdinak izan ditzakete: apalen arabera ala pasabideen arabera. Honen arabera, apalategien zenbakitzea ezberdina izango da inplementazioan aipatuko den moduan. Honetaz gain, biltegiratze eremu soilak ere badaude eta hauek ez dute apalategirik izango. Apalategiek altura zein sakonera ezberdinak izango dituzte eta honen arabera kokaleku gehiago ala gutxiago izango ditugu produktuak gordetzeko. Aipatutako konfigurazio guzti hauek modu erraz eta intuitibo batean sortzeko aukera izan behar dute kudeatzaileek.

Gainera biltegiratzeko kokalekuetatik at, beste kokaleku batzuk kudeatzeko aukera ere izan behar du aplikazioak: sarrera/irteera eremuak. Hauek, kamioiek eskaerak kargatu edo deskargatu ahal izateko kokalekuak dira. Hauek sortu ala ezabatzeko aukera izan behar du aplikazioak.

Lan-aginduei dagozkien atazen kudeaketa

Lan aginduen kudeaketak, biltegi baten sortzen diren sarrera zein irteera eskariak kudeatzeko funtzionalitateak izatean datza. Gainera, eskari hauetan produktuak sartzeko zein kentzeko aukera eskaini behar da. Terminalentzat hauekin lotutako ataza gehiago aurkitzen dira.

Hizkuntzen kudeaketa

Aplikazioak hainbat hizkuntzetan erabili ahal izateko aukera izan behar du. Honen ildotik, erabiltzaileek saioa hastean interfazea konfiguratuta duten hizkuntzara aldatuko zaie automatikoki. Dena den, edozeinek edozein momentutan hizkuntza aldatzeko aukera izango du.

Txostenei dagozkien atazak

Terminaletako erabiltzaileek egindako eskaeren kudeaketari dagokion informazioa ikusi ahalko dute kudeatzaileek mahaigaineko aplikazioaren orrialde nagusian. Esleiturik dauden sarrerak zein, iada prozesatu direnei buruzko informazioa ikusi ahal izango da ataza hauekin.

4.2.2 Mugikorrentzako aplikazioa

Gailu mugikorretan erabiliko da, eta biltegiko teknikariek erabiliko dute. Honen helburua biltegiaren eguneroko atazak burutu ahal izatea.

Horregatik, irismenean aipatu bezala aplikazioak hainbat funtzionalitate izango ditu:

Inbentarioaren kudeaketa

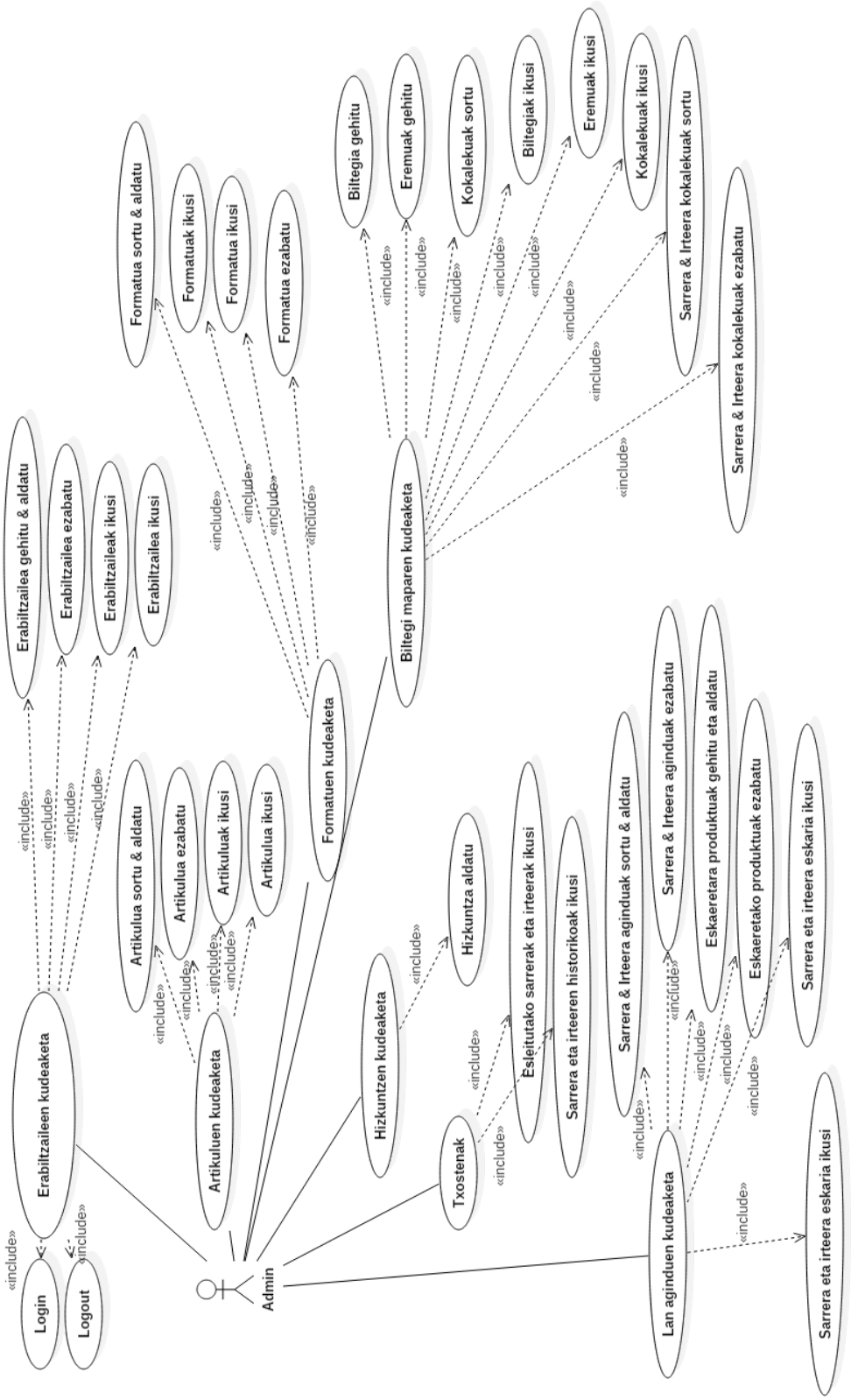
Mahaigaineko aplikazioan sortutako kokalekuetan artikuluak gordetzeko aukera izan behar dute operarioek. Honetarako, inbentario sarrerak sortzeko aukera izan behar du aplikazioak, zein hauek editatu edo ezabatzeko aukera.

Lan-aginduei dagozkien atazak

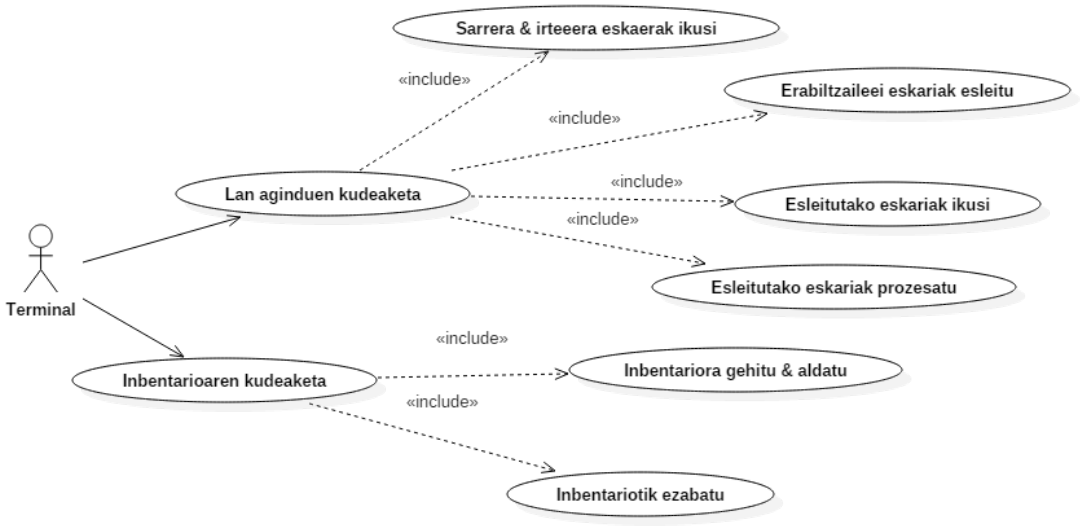
Bi ataza nagusi biltzen dira honen baitan: alde batetik sarrera eta irteera erabiltzaileei esleitzeko aukera izatea, bestetik esleitutako eskaera hauek prozesatzeko aukera izatea. Operazio hauek sortzen dituzten mugimenduak, mahaigaineko aplikaziotik kontsultatu ahalko dira (txostenak).

4.3 Erabilpen kasuak

Aurreko puntuan definitutakoaren ildotik, bi erabiltzaile motei dagozkien erabilpen kasuak 4.1 eta 4.2 irudietan jasotzen dira.



4.1 Irudia: Administratzaileen erabilpen kasuak



4.2 Irudia: Terminaletako erabiltzaileen erabilpen kasuak

5. KAPITULUA

Softwarearen diseinua

5.1 Ikuspegi orokorra

Aurretik aipatu den bezala, aplikazioak MVC arkitektura jarraitzen du. Hau dela eta, hiru geruza ezberdinetan egituratzen da aplikazioa. ASP.NET MVC software proiektuak berak hiru geruza horien kontrola eramaten du eta ereduak zein kontrolagailu edo bistak (aurkezpenak) gehitzean automatikoki sortzen ditu fitxategiak proiektu barnean dagozkien karpetetan.

Hortaz, aplikazioaren egitura nagusia *Models*, *Controllers* eta *Views* karpetetan jasotzen da.

5.2 SGA Web-aplikazioa

Aurreko puntuan aipatu bezala MVC arkitektura jarraitzen du aplikazio honek eta proiektuaren izaera beragatik, egitura eta konfigurazio jakin batzuk jarraitu behar dira aplikazioa martxan jartzeko.

Lehenik eta behin proiektuaren *App_Start* karpetan *RouteConfig.cs* fitxategia kokatzen da.

`RouteConfig`

Klase honetan aplikazioaren parte izango diren helbideak konfiguratzen dira, hau da, adibidez ezabaketa bat egin nahi dugunean, botoi bat sakatuko dugu eta honek aplikazioaren erabilpen-kasu bat exekutatuko du, beharrezko parametroak pasaz funtzio horri.

Akzio hauekin lotuta dauden helbideak eta euren parametroak, fitxategi honetan konfiguratzen dira, helbide jakin bat idazterakoan nabigatzaileak jakin dezan aplikazioaren zein kontrolagailu erabili behar duen, eta bertako zein funtzio edo prozedurari deitu behar dion zein parametrorekin.

Adibidez:

```
routes.MapRoute(
name: "Default",
url: "{controller}/{action}",
defaults: new { controller = "Home", action = "Index" }
);
```

Kode honek, besterik ezeko helbideetan, hau da, gure kasuan zerbitzariaren ip-a idazterakoan nabigatzailean, `HomeController` klaseko *Index* akzioa exekutatuko du parametrorik gabe. Gainera, hau beharrezkoa da aplikazioa martxan jarri ahal izateko. Parametroak pasaz honako adibide hau izan genezake:

```
routes.MapRoute(
name: "CreateProduct",
url: "Product/Crud/{prodData}",
defaults: new { controller = "Product", action = "Crud", prodData = UrlParameter.
    Optional }
);
```

Honek produktuaren datuak propietateetan jasota dituen eredu bat url helbidean pasaz, `ProductController` klaseko *Crud* eragiketa exekutatuko luke.

`Controller` hitza itsatsita duten klase horiek, noski, aplikazioaren kontrolagailuak dira izenean antzeman daitekeen modura. Aplikazio honetan, kontrolagailu guztiek BaseController klasea heredatzen dute *Controller* klase estandarra erabili beharrean. BaseController klasea da, Controller mota heredatuko duena. Honen helburua bakarra da soilik, hizkuntzari dagokionean, aplikazioaren portaera aldatzea prozedura bat birdefinituz eta, modu honetan gure hizkuntza sistema martxan jartzeko.

Kontrolagailuak beharko ditugu, esan bezala, aplikazioan zehar beharko diren eragiketak

definitzeko. Hauek honako hauek izango dira: `ArticulosController`, `EntradasSalidasController`, `FormatosController`, `HomeController`, `LocalesController`, `TerminalesController`, `UbicacionesController`, `UserController`, `ZonasController`.

Hauetan, erabilpen kasuetan definitutako atazak inplementatuko dira, beharrezko ereduen erabilpenarekin eta, exekuzioaren emaitzak bista edo aurkezpenen bitartez erakutsiko dira. Honetaz, gain, prozesamendu horretarako beharrezkoak diren funtzio edo prozedurak ere bertan definituko dira. Bakoitzaren edukiak, erabilpen kasuak ikusiz erraz antzeman daiteke. Argitzekoa izan daiteke `ZonasController`-en kasua. `ZonasController`, sarrera/irteera eremuak kudeatzeko kontrolagailua izango da.

Ereduak, datubaseko taulen izen berak izango dituzte, *Entity Framework* baliatuz lortuko baitira eta datubasearekiko operazioak egiteko sorketa, aldaketa eta ezabaketa operazioak idatzi beharko dira beharrezkoa duten ereduetan. Eredu hauek, interesdun den Aitor Crespok eskuragarri jarriko duen datubasetik eskuratuko dira eta hortaz, inplementazioan aipatuko dira. *Models* karpetaren barruan jasoko dira. Bertan izango da ere `SGAContext` klasea, eta honek *Entity Framework*-ari dagokion funtzionamendu guztia kontrolatuko du.

Bista edo aurkezpenak zuzenki lotuta daude kontrolagailuen atazekin eta automatikoki sortzen dira karpeta eta fitxategiak, atazari eskuin klik egin eta bista gehitzea soilik esanda. Barruko inplementazioa noski, geuk egin beharko dugu html erabiliz eta boostrap-en laguntzarekin estiloak emanez. Bistak gordetzen dituzten karpetak, kontrolagailuen izenen arabera sortzen dira eta barruan, fitxategi bakoitza ataza batekin lotua dago. Esaterako, *Home* karpeta bat izango dugu eta bertan *Index.cshtml* fitxategia.

Bista hauek, bista guztietan komuna izango den *_Layout.cshtml* delako fitxategia exekutatuko dute bere baitan eta hemen egongo dira beharrezko script-ak itsatsita, hala nola, aplikazioaren nabigazio menua eta beste. Proiektuak berak konfiguratzen du sistema hau, guk soilik barruan doan egitura idatzi beharko dugu geure gustora.

Azkenik, aipatzea `LocalesController` kontrolagailuak hizkuntzen sistema eramango duela, cookietan beharrezko balioak gordez.

5.3 Sekuentzia-diagramak

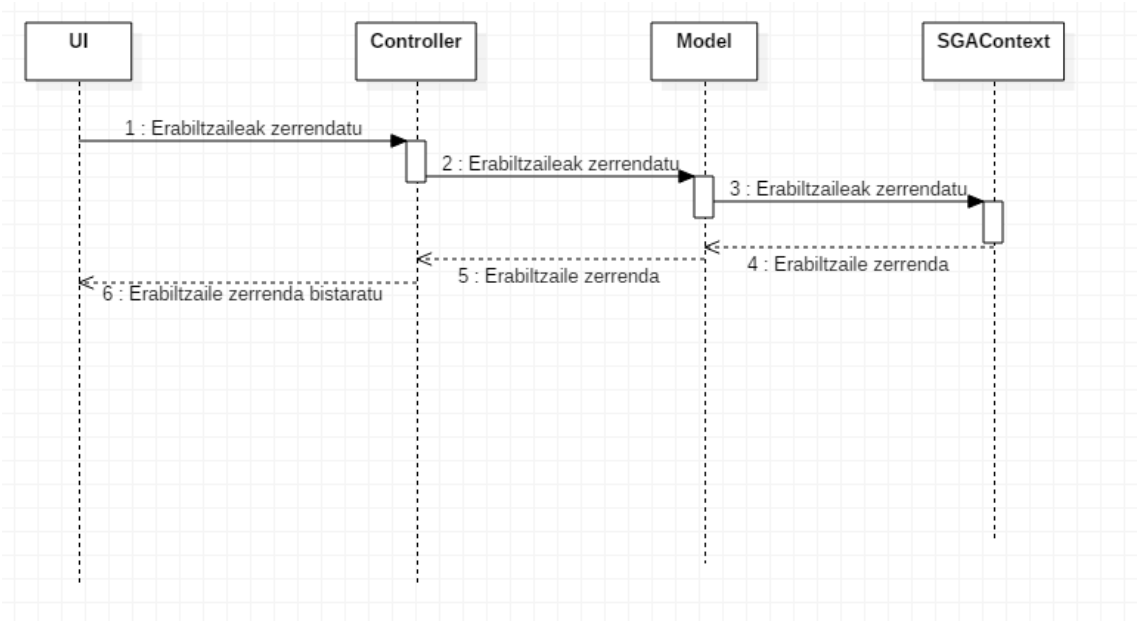
Aplikazioaren diseinua eta klaseen artean dagoen informazio-fluxua argi ikusteko, sekuentzia-diagramak osatu ditugu; ez da erabilpen-kasu bakoitzarentzako sekuentzia diagrama bat

egin, erabilpen-kasu batzuk besteen oso antzekoak dira eta proiektu osoan zehar errepikatzen dira. Diagrama hauek programatzerako garaian erabiliko ditugun gidalerroak dira, eta, honenbestez, berezienak iruditu zaizkigun sekuentzia-diagramak egin ditugu. Egindako diagramekin, aplikazioak beharko dituen informazio-fluxu egitura guztiak azaltzen dira, eta, azaltzen ez diren kasuak, hauei aldaketa txikiak aplikatuz definituko lirateke.

5.3.1 Erabiltzaileak ikusi

Erabiltzaileen zerrenda lortu nahi da datubasetik, eta ondoren bistaratu. Horretarako, erabilpen-kasuari dagokion funtzioa exekutatzen da erabiltzaileen kontrolagailuan eta, honek erabiltzaileen ereduan dagoen funtzioari esker datubaseari eskaera egiten dio. Erantzuna jasotakoan, bista eguneratzen da (5.1 irudia).

Erabilpen kasu honen antzekoak behin eta berriz azaltzen dira proiektuan zehar.



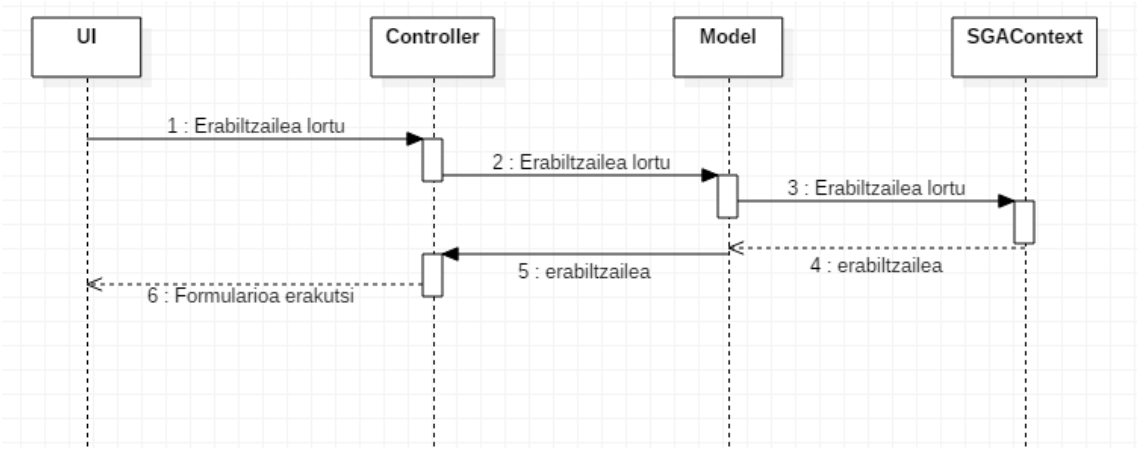
5.1 Irudia: Erabiltzaileak ikusi sekuentzi-diagrama

5.3.2 Erabiltzailea ikusi

Erabiltzailearen informazioa lortu nahi da datubasetik, eta ondoren bistaratu. Horretarako, erabilpen-kasuari dagokion funtzioa exekutatzen da erabiltzaileen kontrolagailuan

eta, honek erabiltzaileen ereduan dagoen funtzioari esker datubaseari eskaera egiten dio. Erantzuna jasotakoan, bista eguneratzen da (5.2 irudia).

Erabilpen kasu honen antzekoak behin eta berriz azaltzen dira proiektuan zehar.



5.2 Irudia: Erabiltzailea ikusi sekuentzi-diagrama

5.3.3 Erabiltzailea sortu

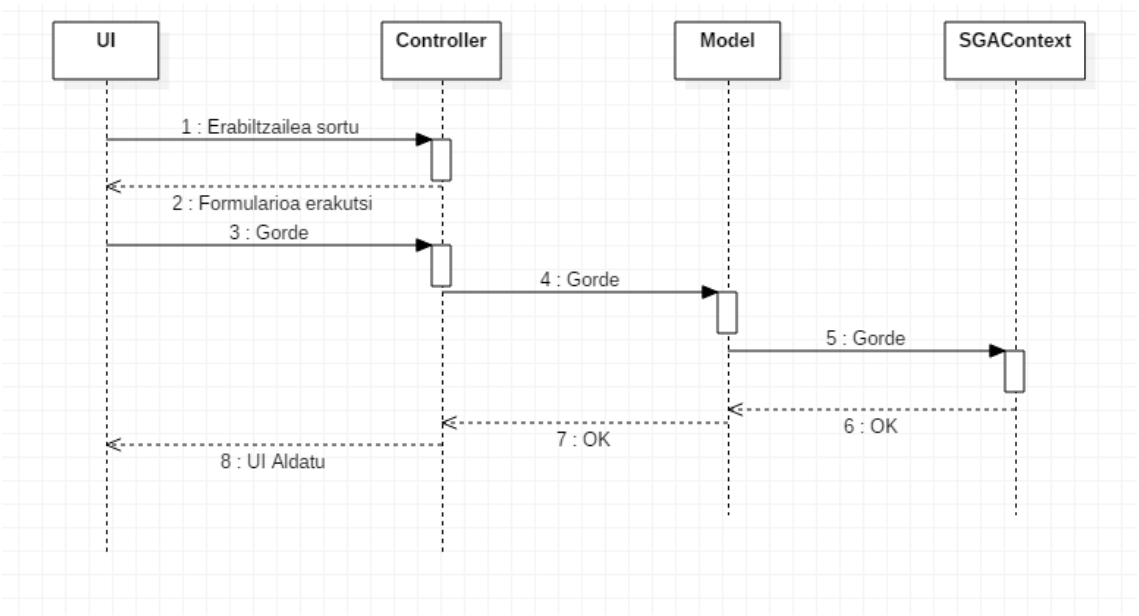
Erabiltzaile berri bat sortu nahi da. Horretarako, erabilpen-kasuari dagokion funtzioa exekutatzen da erabiltzaileen kontrolagailuan eta, honek erabiltzaileen ereduan dagoen funtzioari esker datubaseari eskaera egiten dio. Erantzuna jasotakoan, bista eguneratzen da (5.3 irudia).

Erabilpen kasu honen antzekoak behin eta berriz azaltzen dira proiektuan zehar.

5.3.4 Erabiltzailea aldatu

Erabiltzaile baten adatuak aldatu nahi dira da. Leheneik erabiltzailearen jatorrizko datuak lortu behar dira. Horretarako, erabilpen-kasuari dagokion funtzioa exekutatzen da erabiltzaileen kontrolagailuan eta, honek erabiltzaileen ereduan dagoen funtzioari esker datubaseari eskaera egiten dio. Erantzuna jasotakoan, bista eguneratzen da (5.4 irudia). Behin datuak jasota, eta erabiltzaileak gordetzeko aukera egiten duenean, berriz ere sekuentzia berdina jarraitzen da, datubasean gordetzeko eta, amaieran, bista eguneratzen da.

Erabilpen kasu honen antzekoak behin eta berriz azaltzen dira proiektuan zehar.



5.3 Irudia: Erabiltzailea sortu sekuentzi-diagrama

5.3.5 Erabiltzailea ezabatu

Erabiltzaile bat ezabatu nahi da. Horretarako, erabilpen-kasuari dagokion funtzioa exekutatzen da erabiltzaileen kontrolagailuan eta, honek erabiltzaileen ereduan dagoen funtzioari esker datubaseari eskaera egiten dio. Erantzuna jasotakoan, bista eguneratzen da (5.5 irudia).

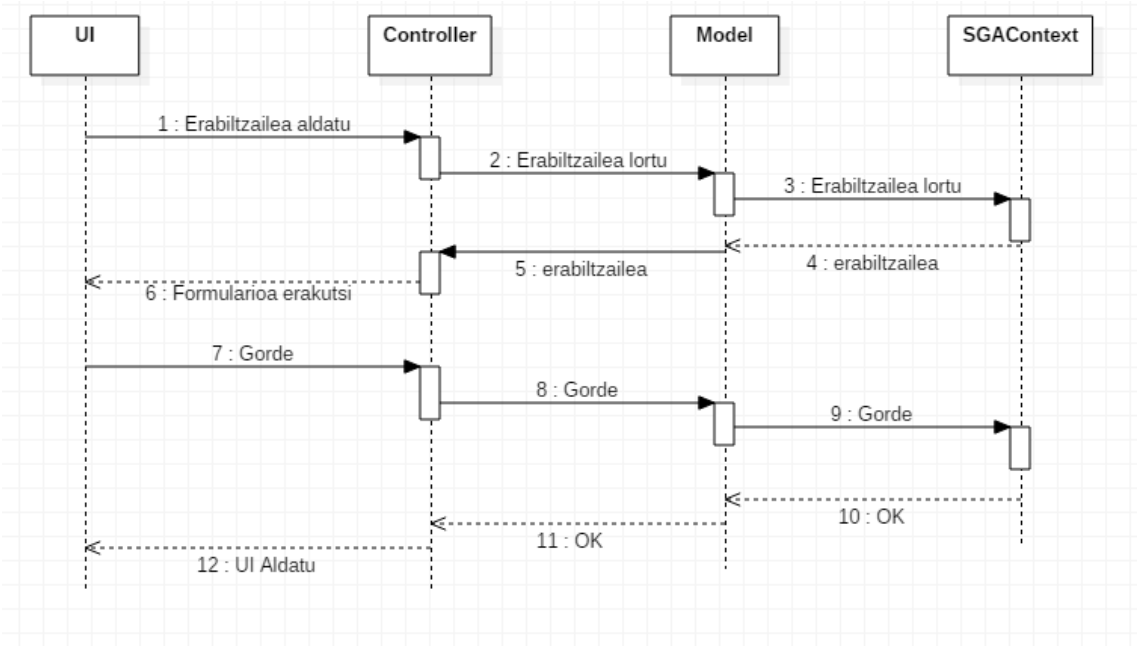
Erabilpen kasu honen antzekoak behin eta berriz azaltzen dira proiektuan zehar.

5.3.6 Eskaria prozesatu

Eskari bat prozesatzeko, lehenik eskariaren historikoa sortu behar da, eskariaren jatorrizko datu guztiekin, eta datubasean gorde. Horretarako kontrolagailuak ereduaren instantzia sortu eta datubasean gordetzen du. Hau egindakoan, kontrolagailuak eskaria ezabatzeko agindua ematen du. (5.6 irudia).

5.3.7 Hizkuntza aldatu

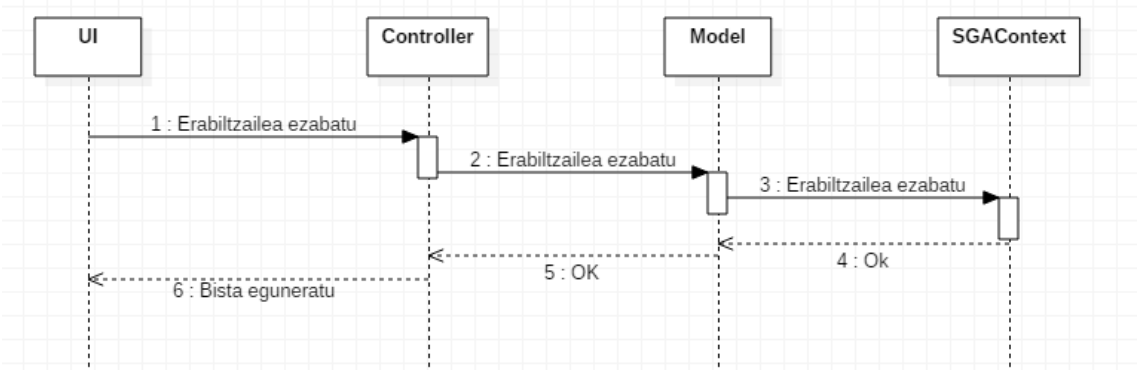
Hizkuntza aldatzea oso sinplea da, dagokion kontrologailuari eskaera egin eta bista eguneratzen da (5.7 irudia).



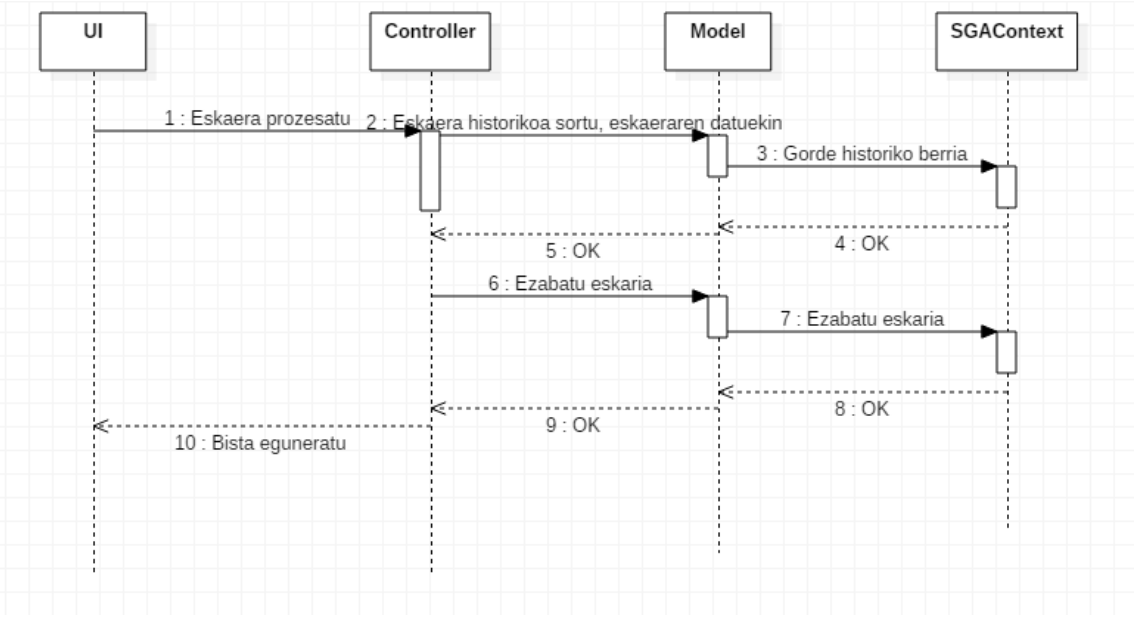
5.4 Irudia: Erabiltzailea aldatu sekuentzi-diagrama

5.3.8 Saioa hasi

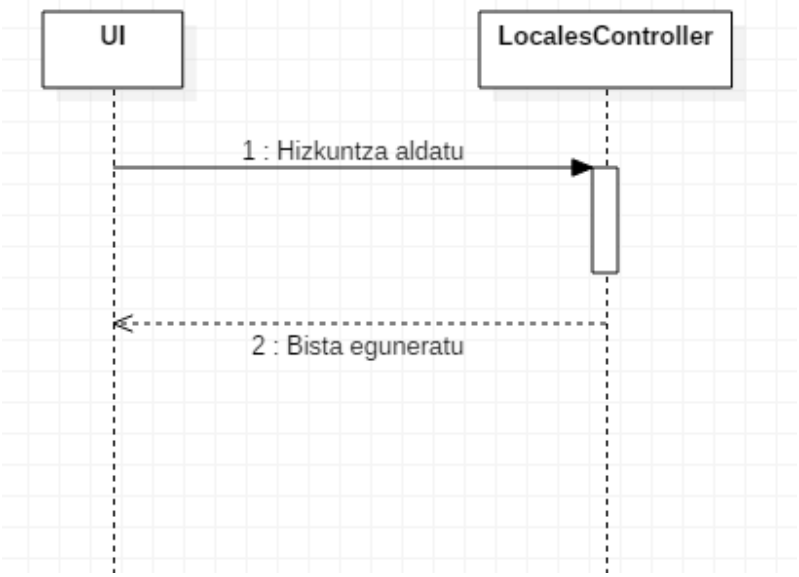
Saioa hasteko, lehenik eta behin datubasetik erabiltzailea eskuratu behar da. Existitzen bada, erabiltzaile horri bueltatzen zaio kontrolagailuari eta erabiltzaileak saioa hasiko du (5.8 irudia).



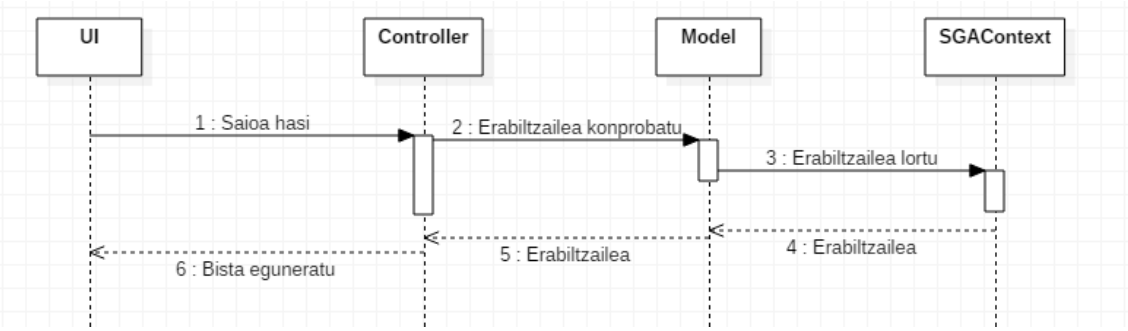
5.5 Irudia: Erabiltzailea ezabatu sekuentzi-diagrama



5.6 Irudia: Eskaria prozesatu sekuentzi-diagrama



5.7 Irudia: Hizkuntza aldatu sekuentzi-diagrama



5.8 Irudia: Saioa hasi sekuentzi-diagrama

6. KAPITULUA

Softwarearen inplementazioa

Aplikazioaren garapen guztia nola egin den azaltzen ez genuke sekula bukatuko; beraz, aplikazioaren puntu kritikoenak aztertuko ditugu, funtzionamendu orokorra ulertzeko. Hala ere, aplikazioaren iturburu-kodean beharrezko ohar eta komentario guztiak aurkitu daitezke aplikazioaren zati bakoitza ulertu ahal izateko.

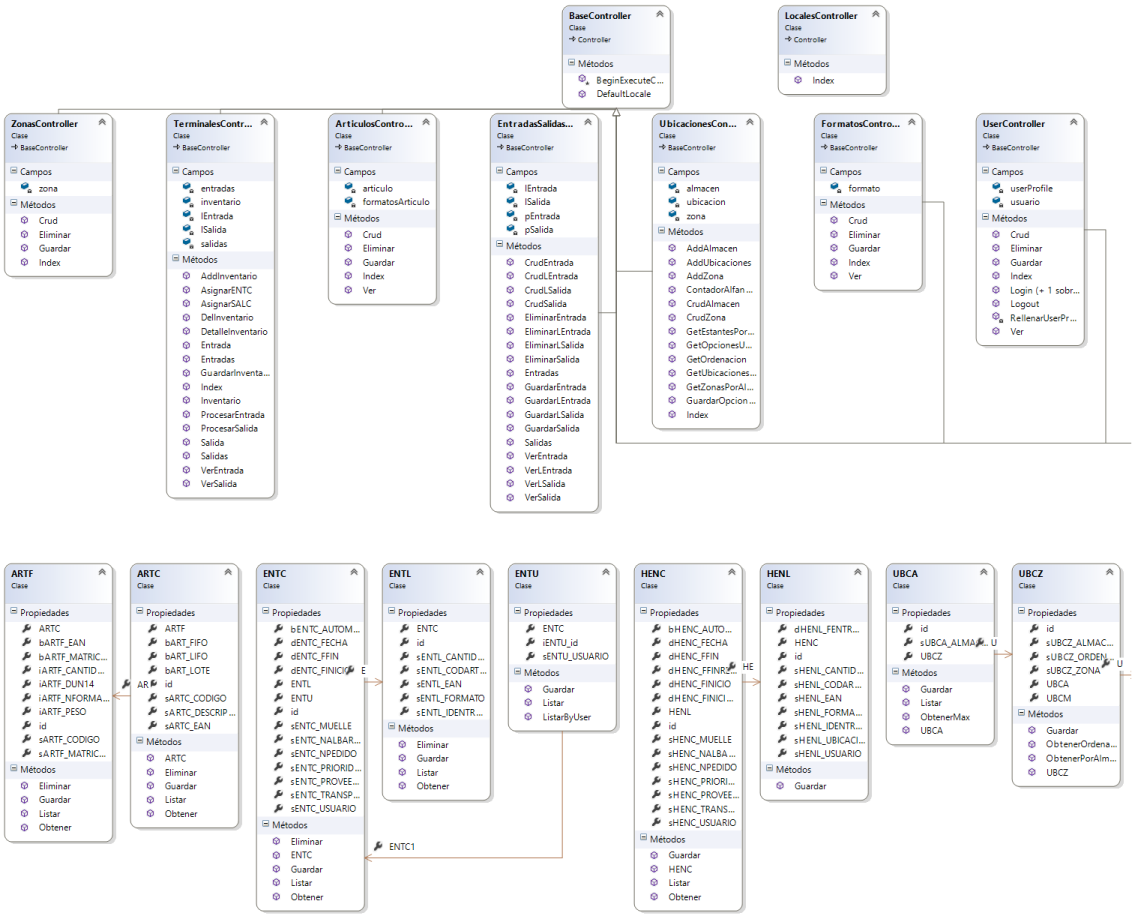
6.1 eta 6.2 irudietan ikus daiteke klase diagrama.

Garatutako aplikazioaren kodea azaltzeko, klase bakoitza banan-banan deskribatu beharrean, aplikazioaren azpiatalen arabera ekingo diogu azalpenari, behar den kasuetan funtzionamendua lortzeko beharrezkoa izan den kodea azalduz.

Hau egiten hasi aurretik, azpiatal bakoitzaren egitura zein den azaltzea beharrezkoa da, azpiatal guztiek antzeko egitura dutelako. Aurretik aipatu den bezala, aplikazioak MVC egitura dauka. Honengatik azpiatala bakoitzak kontrolagailu bat dauka, azpiatalei dagokien erabilpen kasuen logika garatzen duena eta, noski, erabilpen kasu hauetan beharrezko eredu ezberdinen erabilera egiten da. Erabilpen kasuak, aipatu bezala bista edo aurkezpen ezberdinekin lotuak daude, MVC patroia jarraituz.

Aurkezpen hauek kargatzeko, aplikazioak *_ViewStart.cshtml* fitxategia kargatzen du eta bertan definiturik dago, *Shared* karpetako *_Layout* txantiloia erabili behar duela bistak erakusteko. *_Layout* honek, aplikazioaren goiko nabigazio barra eta abar ditu, eta bere barnean kargatzen da, momentuan erabili nahi dugun bista bakoitza.

Ereduetan (model) da, datubasearekin operazioak definituta dauden tokia, arrazoi sinple bategatik: kontsulta hauek ere kontrolagailuan sartu ezkero, eredu berdina erabili nahi



6.1 Irudia: Klase diagramaren lehen zatia

bada hainbat kontrolagailutan, operazio hauek birritan definitu beharko ditugu kontrolagailu ezberdinetan eta honek mantentze-kostuak areagotuko lituzke, noski. Ereduetan daude operazio hauek, datubasean dauden erregistroen zerrendatzea, zein erregistro konkretu baten bilaketa, erregistro bat (berria ala ez) gordetzea, erregistro bat ezabatzea... dira.

Hauek `SGAContext` klasearen erabilera egiten dute, honen instantzia egitean datubaseko taulei dagozkien *DbSet* motako zerrendak automatikoki betetzen dira eta hauek kontsulta ditzakegu eta *LINQ* bitartez beharrezko operazioak egin (aurreko paragrafoan aipatu direnak). Zerrenda hauek ereduen erabilera egiten dute eta hauek euren artean dituzten erlazioak ere automatikoki betetzen dira. Esaterako, artikulu batek hainbat formatu baditu, artikuluen ereduan formatuen propietatea beterik egongo da.

`SGAContext` klaseari buruz azaldu dena hobeago ulertzeko, klaseari dagokion kode zati

bat:

```
public partial class SGAContext : DbContext
{
    public SGAContext()
    : base("name=SGAContext")
    {
    }

    public virtual DbSet<ARTC> ARTC { get; set; }
    public virtual DbSet<ARTF> ARTF { get; set; }
    public virtual DbSet<USRS> USRS { get; set; }
    ...
    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
    modelBuilder.Entity<ARTC>()
    .Property(e => e.sARTC_EAN)
    .HasPrecision(18, 0);

    modelBuilder.Entity<ARTC>()
    .HasMany(e => e.ARTF)
    .WithRequired(e => e.ARTC)
    .HasForeignKey(e => e.sARTF_CODIGO);
    ...
    }
}
```

Kode hau, automatikoki generatzen du *Entity Framework*-ek datubaseari dagozkion taula eta euren arteko erlazioen errepresentazioa sortuz. Ereduak ere (`ARTC`, `ARTF`...) automatikoki sortzen ditu, guk beharrezko aldaketak egin ditugu, goian aipatu den ildotik.

Ereduetan definitutako datubasearekiko operazioak errepikakorrak dira eredu ezberdinetan zehar eta horregatik ez dira birritan errepikatuko. Lehenengo azpiatalean azalduko dira eta gero gainetik aipatu dira gainontzeko puntuetan.

6.1 Hizkuntzen kudeaketa

Hizkuntzen kudeaketa, `LocalesController` izeneko kontrolagailuan egiten da. Klase hau osatzen duen kodea oso sinplea da, jarri nahi den hizkuntza gordetzen du eta saioa

hasita badago, rolaren arabera hasierako orrialde ezberdinera berbideraketa egiten du, bestela gauden orrialdea hizkuntza berrira pasatzen Hauxe da bere kodea:

```
1  public class LocalesController : Controller
2  {
3    public ActionResult Index(string lang = "es_ES")
4    {
5      Response.Cookies["CacheLang"].Value = lang;
6
7      if (Session["Role"]!=null)
8      {
9        if(Session["Role"].ToString()=="1")
10       {
11         Response.Redirect("~/home/index");
12       }
13       else
14       {
15         Response.Redirect("~/home/indexT");
16       }
17     }
18
19     var message = Localization.Get("changedlng");
20     return Content(message);
21   }
22 }
```

Bista guztietako goiburu bezala dagoen fitxategian (aurretik aipatu dena), daude hizkuntza aldaketaren estekak eta, hauek proiektuan erabili diren *Insya.Localization* izeneko liburutegi batzuekin, LocalesController erabiliz hizkuntza aldatzeko gai da.

Aplikazioan zehar ikusten den bezala, textuak liburutegi hori erabiliz eskuratzen dira, *@Html.Localize("idiomas")* bezalako deiak erabiliz. Honek, liburutegiarei eskatzen dio textua, eta hau gauden hizkuntzari dagokion xml fitxategitik irakurtzen da. Hauek, *Localization* karpeta barruan daude.

6.2 Erabiltzaileen kudeaketa

Azpiatal honetan erabiltzaileak sortu, aldatu, ikusi eta ezabatzeaz gain, saio hastea eta ixtearen kontrola ere azaltzen dira.

6.2.1 CRUD atazak

Crud atazak deritze, sortu, aldatu eta ezabatzeari. Honetaz gain, erregistro konkretu baten datuak ere eska ditzakegu.

Erabiltzaileei dagozkien erabilpen kasuen kontrola, `UserController` kontrolagailuan egiten da. Bertan hainbat funtzio definitzen dira, erabilpen kasuekin zuzenean loturikoak. *Crud*, *Ver*, *Guardar* eta *Eliminar*. Hauek dira euren kodeak, hurrenez-hurren:

```
1  public ActionResult Crud(string user = null)
2  {
3    USRS usuarioObtenido = new USRS();
4    usuarioObtenido= (user != null) ? usuarioObtenido=usuario.Obtener(user) : usuario;
5    userProfile=RellenarUserProfile(usuarioObtenido);
6
7    return View(userProfile);
8  }
9
10 public ActionResult Ver(string user)
11 {
12   return View(usuario.Obtener(user));
13 }
14
15 public ActionResult Guardar(UserProfileModel userProfile)
16 {
17   if (userProfile.USRS.sUSRS_PASSWORD != null) { userProfile.USRS.
       SetPasswordHashAndSalt(); }
18   if (TryValidateModel(userProfile.USRS))
19   {
20     try
21     {
22       userProfile.USRS.Guardar();
23       return Redirect("~/user/ver/" + userProfile.USRS.sUSRS_USUARIO);
24     }
25     catch (Exception)
26     {
27       ModelState.AddModelError("", Localization.Localize("excepcionCreacion"));
28       userProfile = RellenarUserProfile(userProfile.USRS);
29       return View("~/Views/User/Crud.cshtml", userProfile);
30     }
31
```

```
32      }
33      else
34      {
35          userProfile= RellenarUserProfile(userProfile.USRS);
36          return View("~/Views/User/Crud.cshtml", userProfile);
37      }
38
39  }
40
41  public ActionResult Eliminar(string user)
42  {
43    usuario.Eliminar(user);
44    return Redirect("~/user");
45  }
```

ActionResult mota itzultzen duten funtzioak dira, hauek berbideraketa zuzena ala bista bat ikusteko berbideraketak egiten dituzte eta.

Crud erabiltzaile berriak sortzeko zein existitzen direnen datuak eskuratu eta bistaratzeko erabiltzen da, honek egiten duena da, erabiltzaile izena jasotzen badu bilatu eta datuak formularioan bistaratu, bestela, formularioa hutsik ikusiko da.

Horretarako, lehenik USRS motako objektu bi erabiltzen dira (bietako bat, bistara lortutako informazioa edo, lortu ez bada hutsik pasatzeko). Bietako objektu bat baliatuz (*usuario*) *Obtener* funtzioa exekutatzen da. Honek, datubasetik erabiltzaile konkretu baten datuak eskuratzen ditu.

Kontrolagailuan erabiltzen diren objektu orokor batzuk definitzen dira klasearen hasieran:

```
1  private USRS usuario = new USRS();
2  private UserProfileModel userProfile = new UserProfileModel();
```

USRS ereduak, aurretik azaldu den bezala, eredu guztiek bezala datubasearekiko operazioak egiten dituztenak, hauek behin eta berriz errepikatzen dira eta berdinak ala oso antzekoak suertatzen dira, beraz ez dira behin eta berriz errepikatuko dokumentazio honetan, behar bezala dokumentaturik daude entregatzen den kodean. USRS ereduan azaltzen dira honako eragiketa hauek:

```
1  public List<USRS> Listar()
2  {
```

```
3     var usuarios = new List<USRS>();
4     try
5     {
6       using (var context = new SGAContext())
7       {
8         usuarios = context.USRS.ToList();
9       }
10    }
11    catch (Exception e)
12    {
13      throw new Exception(e.Message);
14    }
15    return usuarios;
16  }
17
18  public USRS Obtener(string username)
19  {
20    var usuario = new USRS();
21    try
22    {
23      using (var context = new SGAContext())
24      {
25        usuario = context.USRS
26        .Where(x => x.sUSRS_USUARIO == username)
27        .Single();
28      }
29    }
30    catch (Exception e)
31    {
32      throw new Exception(e.Message);
33    }
34    return usuario;
35  }
36
37  public void Guardar()
38  {
39    try
40    {
41      using (var context = new SGAContext())
42      {
43        if (this.id == 0)
```

```
44          {
45              context.Entry(this).State = System.Data.Entity.EntityState.Added;
46          }
47          else
48          {
49              context.Entry(this).State = System.Data.Entity.EntityState.Modified;
50          }
51
52          context.SaveChanges();
53      }
54    }
55    catch (Exception e)
56    {
57      throw new Exception(e.Message);
58    }
59  }
60
61  public void Eliminar(string username)
62  {
63    try
64    {
65      using (var context = new SGAContext())
66      {
67        context.Entry(new USRS { sUSRS_USUARIO = username }).State = System.Data.Entity
      .EntityState.Deleted;
68        context.SaveChanges();
69      }
70    }
71    catch (Exception e)
72    {
73      throw new Exception(e.Message);
74    }
75  }
```

Aurretik aipatu da SGAContext klasearen papera aplikazioaren egituran, eta horren arabera ikus daitezke eragiketa bakoitzak zer egiten duen, kodea ulertzeko oso erraza baita.

UserProfileModel objektuari dagokionez, hainbat propietate dituen mota bat eta bistara hainbat eredu aldiberean pasa ahal izateko erabiltzen da bakarrik, bestela defektuz bakarrik eredu mota bat bakarrik pasa diezaiokegu bistari. Kasu honetan, formularioetako ListBox-ak betetzeko informazioa zein erabiltzailearen informazioa aldiberean pasa ahal

izateko erabiltzen da. Honelako bizpahiru objektu daude proiektuan zehar.

Bista guztien egiturak antzekoak dira, bertan informazioa erakusten edo sartzen baita. Eredua zein den definitzen da, eta hau baliatuz informazioa bistaratzen joaten da, ala formulario baten eremuak eredu honen propietateetara lotzen dira, formularioa bidaltzean, kontrolagailuak jasotzen duen eredua beterik jaso dezan.

Sortu/aldatu bistari dagokion kodea hau litzatzeke:

```
1  @model SGA.Models.UserProfileModel
2
3  @{
4    ViewBag.Title = (Model.USRS.sUSRS_USUARIO != null ? Model.USRS.sUSRS_NOMBRE + "  " +
       Model.USRS.sUSRS_APELLIDO1 + "  " + Model.USRS.sUSRS_APELLIDO2 : @Html.Localize("
       nuevoRegistro").ToString());
5    ViewBag.Current = "Usuarios";
6  }
7
8  <div>
9  <ol class="breadcrumb">
10 <li><a href="~/">@Html.Localize("homepage")</a></li>
11 <li><a href="~/user">@Html.Localize("usuario")</a></li>
12 <li class="active">@(Model.USRS.sUSRS_USUARIO != null ? Model.USRS.sUSRS_NOMBRE + "  " +
       Model.USRS.sUSRS_APELLIDO1 + "  " + Model.USRS.sUSRS_APELLIDO2 : @Html.Localize("
       nuevoRegistro").ToString())</li>
13 </ol>
14 </div>
15
16 @using (Html.BeginForm("Guardar", "User", FormMethod.Post, new { id = "frm-usuario" }))
17 {
18   @Html.HiddenFor(x => x.USRS.id)
19
20   <div>
21   @Html.ValidationSummary(true, Localization.Localize("excepcionValidacion"), new {
       @class = "alert alert-danger" })
22   </div>
23
24   <div class="panel panel-default">
25     <div class="panel-heading">@Html.Localize("infoPersonal")</div>
26     <div class="panel-body">
27       <div class="form-group">
28       <label>@Html.Localize("usuario")</label>
```

```
29
30      @if (Model.USRS.sUSRS_USUARIO == null)
31      {
32        @Html.TextBoxFor(model => model.USRS.sUSRS_USUARIO, new { @class = "form-
    control" })
33        @Html.ValidationMessageFor(u => u.USRS.sUSRS_USUARIO, null, new { @class = "
    label label-danger" })
34      }
35      else
36      {
37        @Html.TextBoxFor(model => model.USRS.sUSRS_USUARIO, new { @class = "form-
    control", disabled = "disabled" })
38        @Html.HiddenFor(x => x.USRS.sUSRS_USUARIO)
39      }
40      </div>
41
42    <div class="form-group">
43      @Html.Localize("password")
44      @Html.PasswordFor(model => model.USRS.sUSRS_PASSWORD, new { @class = "form-
    control" })
45      @Html.ValidationMessageFor(u => u.USRS.sUSRS_PASSWORD, null, new { @class = "
    label label-danger" })
46    </div>
47    ....
48
49    <div class="text-right">
50      <button type="submit" class="btn btn-primary">@Html.Localize("guardar")</
    button>
51    </div>
```

Ez dira eremu guztiak erakusten, helburua bista baten egitura orokorra ikustea baita, eta aipatu diren xehetasunak antzematea.

Guardar funtzioarekin, aldaketak gordetzen dira, behar diren balidazioak egin ostean. Arazoak badaude bista berdinean esaten zaion erabiltzaileari zein arazo egon diren, zuzen ditzan. Goiko kodean ikus daiteke, kontrolagailuren akzio honi deitzen zaiola

Ver funtzioarekin, erabiltzaile konkretu baten informazioa eskuratzen da, eta pantailaratzen da.

Eliminar funtzioarekin, botoi bat sakatuta, eta konfirmazioa eskatu ondoren, datubasetik

ezabatzen da.

6.2.2 Debekatutako guneak

Aplikazioak bi erabiltzaile-mota ditu, eta erabiltzaile-mota bakoitzak soilik aplikazioaren zati bat ikusiko du. Honetaz gain, saioa hasi ez duen erabiltzaileak, ezingo du aplikazioa ikusi.

`UserController` kontrolagailuan egiten da hau eta aplikazioak bertara berbideratzen du erabiltzailea behar ez den orrialde bat ikusten saiatzen bada saioa hasi gabe. *Login*

Bi funtzio daude izen berarekin, eskaera egiten den moduaren arabera bata ala bestea exekutatuko delarik. Erabilpen-kasu honetara berbideraketa egiten ari bagara HttpGet motako eskaera bat izango da, eta login saiakera egin ezkero HttpPost motakoa. Honats kodea:

```
1  [HttpGet]
2  public ActionResult Login()
3  {
4    return View();
5  }
6
7  [HttpPost]
8  public ActionResult Login(USRS loginUser)
9  {
10   USRS user =null;
11   if (ModelState.IsValidField("sUSRS\_USUARIO") \&\& ModelState.IsValidField("sUSRS
       \_PASSWORD"))
12   {
13
14     try
15     {
16       user = usuario.Obtener(loginUser.sUSRS\_USUARIO);
17     }
18     catch(Exception ex)
19     {
20
21       ModelState.AddModelError("", Localization.Localize("datosIncorrectos"));
22       return View(loginUser);
23     }
24
```

```
25      if (user.sUSRS\_PASSWORD == user.HashPassword(loginUser.sUSRS\_PASSWORD, user.
        sUSRS\_SALT))
26      {
27
28        if(user.sUSRS\_ACTIVO)
29        {
30          Session["LogedUserID"] = user.id;
31          Session["LogedUserFullname"] = user.sUSRS\_USUARIO;
32          Session["Role"] = user.sUSRS\_ROL;
33
34          Response.Cookies["CacheLang"].Value = user.sUSRS\_CULTURA;
35          if (user.sUSRS\_ROL == 1)
36          {
37            return RedirectToAction("Index", "Home");
38          }
39          else
40          {
41            return RedirectToAction("IndexT", "Home");
42          }
43
44        }
45        else
46        {
47          ModelState.AddModelError("", Localization.Localize("usuarioInactivo"));
48        }
49
50      }
51      else
52      {
53        ModelState.AddModelError("", Localization.Localize("datosIncorrectos"));
54      }
55
56      return View(loginUser);
57    }
58    else
59    {
60      return View("~/Views/User/Login.cshtml", loginUser);
61    }
62
63  }
```

Hainbat konprobazio egiten dira, eta horren arabera saio informazioa gordetzen da, aplikazioan zehar konprobazioak egiteko, zein behar den kasuetan erabiltzaile-izena eskuratzeko.

*Logout*en kasuan soilik saioa garbitzen eta login egiteko orrialdera berbideratzen da.

6.3 Artikuluen kudeaketa

`ArticulosController` kontrolagailuak egiten du artikuluen erabilpen kasuen kontrola, eta hauek dira definitutako operazioen izenak *Index, Crud, Ver, Guardar, Eliminar*. Index-a dauden erregistro guztiak zerrendatzeko erabiltzen da, beste erabilpen-kasuen funtzionamentua, kontrolagailu guztietan antzekoa da. Kontrolagailu honetan *ARTC* eredua eta *List<ARTF>* zerrenda erabiltzen dira, bista ezberdinetan beharren arabera artikuluak eta bere formatuak ikusteko.

6.4 Formatuen kudeaketa

Aurreko atalarekin lotuta dagoen arren, banatuta egin da kontrol hau hedapen arrazoiengatik. `FormatosController` kontrolagailuak eramaten ditu hauen kontrola eta *Index, Crud, Ver, Guardar, Eliminar* erabilpen-kasuak definitzen dira. Bertan ARTF ereduaren erabilera egiten da.

6.5 Kokalekuen kudeaketa

Bi azpiatal ezberdin ezberdindu behar dira hemen. Alde batetik biltegiaren kokaleku guztiak kudeatzeko operazioak eta bestetik, eskarien irteera/sarrera kokalekuak kudeatzeko (*muelle de carga y descarga* bezala bezala ezagunak) operazioak.

`UbicacionesController` kontrolagailuan kontrolatzen da biltegiko kokalekuak identifikatzeko erabiltzen diren eredu guztiak. Kontuan izan behar da eredu hauek datubaseko tauletatik "mapeatuta"daudela eta hauek dituzten erlazioak dituztela euren artean. Horregatik, erabilpen kasu askota hauek zerrendatzea zein hauetako bat hartu eta berarekin erlazionaturiko erregistro guztiak lortzea dea beraien eginkizuna.

Kontrolagailu honen konplexutasuna dela eta, pena merezi du, kontrolagailuaren kodea itsasi eta erabilpen-kasu esanguratsuenak azaltzea.

```
1  public class UbicacionesController : BaseController
2  {
3    private UBCA almacen = new UBCA();
4    private UBCZ zona = new UBCZ();
5    private UBCM ubicacion = new UBCM();
6
7    public List<string> ContadorAlfanumerico(int numCaracteres, string strOriginal, int
       numMovimientos)
8    {
9    ...
10   }
11
12   public ActionResult Index()
13   {
14     return View(almacen.Listar());
15   }
16
17   public ActionResult CrudAlmacen(int cantidad)
18   {
19     var MaxId = Convert.ToInt32(almacen.ObtenerMax().Substring(1));
20     return View();
21   }
22
23   public ActionResult CrudZona(int cantidad)
24   {
25     return View();
26   }
27
28   [ActionName("ZonasPorAlmacen")]
29   public JsonResult GetZonasPorAlmacen(string almacen)
30   {
31     List<UBCZ> zonas = zona.ObtenerPorAlmacen(almacen);
32     return Json(zonas, JsonRequestBehavior.AllowGet) ;
33   }
34
35   [ActionName("EstantesPorZona")]
36   public JsonResult GetEstantesPorZona(UbicacionesModel model)
37   {
```

```
38      List<UBCM> estanterias = ubicacion.ObtenerEstanteriasPorZona(model.Zona, model.
        Almacen);
39      return Json(estanterias, JsonRequestBehavior.AllowGet);
40    }
41
42    [ActionName("UbicacionesPorEstante")]
43    public JsonResult GetUbicacionesPorEstante(UbicacionesModel model)
44    {
45      var ordenacion = zona.ObtenerOrdenacion(model.Almacen, model.Zona);
46      List<UBCM> ubicaciones = ubicacion.ObtenerUbicacionesPorEstante(model.Zona,
        model.Estanteria, model.Almacen);
47
48      try
49      {
50        List<UBCM> izquierda = new List<UBCM>();
51        List<UBCM> derecha = new List<UBCM>();
52
53
54        switch ((OrdenacionZonas)ordenacion)
55        {
56          \/Si es por pasillo, tenemos ubicaciones a izquierda y derecha
57          case OrdenacionZonas.Pasillo:
58          for(int i = 0; i <= ubicaciones.Count \!1 ; i++)
59          {
60            if((i+1) %2 ==0)
61            {
62              derecha.Add(ubicaciones[i]);
63            }
64            else
65            {
66              izquierda.Add(ubicaciones[i]);
67            }
68          }
69          ubicaciones.Clear();
70          ubicaciones.AddRange(izquierda);
71          ubicaciones.AddRange(derecha);
72          break;
73        }
74
75      }
76      catch (Exception ex)
```

```
77      {
78
79      }
80
81      return Json(ubicaciones, JsonRequestBehavior.AllowGet);
82    }
83
84    [HttpPost]
85    [ActionName("AddAlmacen")]
86    public JsonResult AddAlmacen(UBCA model)
87    {
88      var response = new ResponseModel();
89
90      if (model != null \&\& model.sUBCA\_ALMACEN != null)
91      {
92        if (model.sUBCA\_ALMACEN.Length == 1)
93        {
94          model.sUBCA\_ALMACEN = string.Concat("0", model.sUBCA\_ALMACEN);
95        }
96
97      }
98
99      try
100     {
101
102       if(model.sUBCA\_ALMACEN == string.Empty)
103       {
104         throw new Exception();
105       }
106
107       model.Guardar();
108     }
109     catch (Exception ex)
110     {
111       response.Error = true;
112       response.Mensaje = Localization.Localize("excepcionValidacion");
113     }
114
115     return Json(response);
116   }
117
```

```
118  [HttpPost]
119  [ActionName("AddZona")]
120  public JsonResult AddZona(UBCZ model)
121  {
122      var response = new ResponseModel();
123
124      if (model != null \&\& model.sUBCZ\_ZONA != null)
125      {
126          if (model.sUBCZ\_ZONA.Length == 1)
127          {
128              model.sUBCZ\_ZONA = string.Concat("0", model.sUBCZ\_ZONA);
129          }
130
131      }
132
133      try
134      {
135
136          if (model.sUBCZ\_ZONA == string.Empty)
137          {
138              throw new Exception();
139          }
140
141          model.Guardar();
142      }
143      catch (Exception ex)
144      {
145          response.Error = true;
146          response.Mensaje = Localization.Localize("excepcionValidacion");
147      }
148
149      return Json(response);
150  }
151
152  [HttpPost]
153  [ActionName("GetOrdenacion")]
154  public JsonResult GetOrdenacion(UBCZ model)
155  {
156      var response = new ResponseModel();
157      try
158      {
```

```
159         response.Mensaje = zona.ObtenerOrdenacion(model.sUBCZ\_ALMACEN, model.sUBCZ\
            _ZONA).ToString();
160       }
161       catch (Exception ex)
162       {
163         response.Error = true;
164         response.Mensaje = Localization.Localize("excepcionValidacion");
165       }
166
167       return Json(response);
168     }
169
170     [HttpPost]
171     [ActionName("GetOpcionesUbicacion")]
172     public JsonResult GetOpcionesUbicacion(UBCM model)
173     {
174       UBCM response;
175       response = ubicacion.ObtenerOpcionesUbicacion(model.sUBCM\_ALMACEN, model.sUBCM
            \_ZONA, model.sUBCM\_ESTANTERIA, model.sUBCM\_PROFUNDIDAD, model.sUBCM\_ALTURA);
176       return Json(response);
177     }
178
179     [HttpPost]
180     [ActionName("GuardarOpcionesUbicacion")]
181     public JsonResult GuardarOpcionesUbicacion(UBCM model)
182     {
183       var response = new ResponseModel();
184       try
185       {
186         model.Guardar();
187       }
188       catch (Exception ex)
189       {
190         response.Error = true;
191         response.Mensaje = Localization.Localize("excepcionValidacion");
192       }
193
194       return Json(response);
195     }
196
197     [HttpPost]
```

```
198 [ActionName("AddUbicaciones")]
199 public JsonResult AddUbicaciones(UBCM model)
200 {
201   var response = new ResponseModel();
202   try
203   {
204     var idMax= ubicacion.ObtenerMax(model.sUBCM\_ALMACEN, model.sUBCM\_ZONA);
205     if (idMax==null)
206     {
207       model.sUBCM\_ESTANTERIA = "001";
208     }
209     else
210     {
211       model.sUBCM\_ESTANTERIA = ContadorAlfanumerico(3, idMax, 1).First();
212     }
213
214     OrdenacionZonas ordenacion = (OrdenacionZonas)zona.ObtenerOrdenacion(model.
    sUBCM\_ALMACEN, model.sUBCM\_ZONA);
215     List<string> profundidades;
216     List<string> alturas;
217     switch (ordenacion)
218     {
219       case OrdenacionZonas.Estanteria:
220       profundidades = ContadorAlfanumerico(3,"0", Convert.ToInt32(model.sUBCM\
    _PROFUNDIDAD));
221       alturas = ContadorAlfanumerico(2,"0", Convert.ToInt32(model.sUBCM\_ALTURA));
222
223       foreach (string altura in alturas)
224       {
225         foreach (string profundidad in profundidades)
226         {
227           model.id = 0; \Para poder reutilizar en adicion
228           model.sUBCM\_PROFUNDIDAD = profundidad;
229           model.sUBCM\_ALTURA = altura;
230           model.Guardar();
231         }
232       }
233
234       break;
235
236       case OrdenacionZonas.Pasillo:
```

```
237         \Las ubicaciones en pasillo, se consideran como una estanteria doble, ya que
    hay que añadir las mismas ubicaciones
238         \a izquierda y derecha del pasillo, por tanto, la profundidad será doble =
    doble numero de ubicaciones.
239         profundidades = ContadorAlfanumerico(3, "0", Convert.ToInt32(model.sUBCM\
    _PROFUNDIDAD)*2);
240         alturas = ContadorAlfanumerico(2, "0", Convert.ToInt32(model.sUBCM\_ALTURA));
241
242         foreach (string altura in alturas)
243         {
244           foreach (string profundidad in profundidades)
245           {
246             model.id = 0; \Para poder reutilizar en adicion
247             model.sUBCM\_PROFUNDIDAD = profundidad;
248             model.sUBCM\_ALTURA = altura;
249             model.Guardar();
250           }
251         }
252
253         break;
254       }
255
256     }
257     catch (Exception ex)
258     {
259       response.Error = true;
260       response.Mensaje = Localization.Localize("excepcionGeneracionUbicaciones");
261     }
262
263     return Json(response);
264   }
265 }
```

ContadorAlfanumerico funtzioaren kodea ez da kopiatu, honek egiten duena 0-tik Z-rako kontaketa egiten du, esaten zaion karaktere kopurukoa eta esaten zaion kontaketa kopuruen araberako lista bat bueltatzen du, balio osoekin. Hau erabilgarria da, adibidez, apal bati dagozkion kokaleku guztiak sortzeko automatikoki, goiko kodean ikus daitekeen bezala.

JsonResult itzultzen duten funtzio asko ditugu. Hauek *JSON* formstuan dauden datu eta zerrendak itzultzen ditu. Funtzio hauek *JQuery* bitartez exekutatzen dira *$ajax* zein *$getJ-*

son erabiliz, eta gero Json hau prozesatzen da bistan informazioa kargatu eta egiturak eguneratzeko.

Kudeaketa guztia soilik bista batean egiten da, hau da, eskaera guztiak *Index.cshtml* fitxategitik egiten dira *Jquery* bitartez eta bista eta bere egiturak eguneratzen dira. Modu honetan, biltegi bat edo hainbatetan eremu berriak, zein apalak eta euren kokalekuak sortzeko kudeaketa hori asko azkartzen da era erabiltzaileari erraztasuna ematen zaio eta intuitiboagoa egiten du aplikazioa.

UBCA, *UBCZ* eta *UBCM* ereduak erabiltzen dira; lehenak biltegien informazioa du, bigarrenak, eremuen informazioa biltegi kodearekin, eta azkenak, kokalekuen informazioa du (biltegi kodea eta eremu kodearekin). Hau da, hiruen artean erlazioa dago.

Hortaz, biltegiak, zein eremuak, zein ubikazioak sor ditzakegu, ordenazioen arabera. Badaude, esan bezala, eremu bereziak apalik ez dutenak. Hauetaz aparte bi ordenazio daude: apalen arabera eta pasilloen arabera. Apalen arabera kokalekuen zenbakitzea normala da, baina pasilloaren arabera ordenatu ezkero, pasillo bakoitza nabigatzen goatzen norantzaren arabera, ezkerreko apalek zenbakitze inparea izango dute eta eskubikoek parea. Hau *AddUbicaciones* erabilpen-kasuan ikusten da eta bistaratzerako orduan *GetUbicaciones-PorEstante*-n ere antzeman daiteke.

Esan bezala, JQuery erabiltzen da exekuzioak egin eta informazioa bistaratzeko, 6.3 irudian ikus daiteke adibide bat.

Bestalde, `ZonasController` kontrolagailuan egiten da eskarien irteera/sarrera kokalekuen kudeaketa. *Index, Crud, Guardar, Eliminar* operazioak daude bertan eta *Index* erabilpen-kasuan mota guztietakoak (sarrera, irteera eta sarrera/irteera) bistaratzen dira *ZonasModel* klasearen bitartez hiru zerrenda ezberdin bistara pasaz.

6.6 Sarreren eta irteera eskarien kudeaketa

`EntradasSalidasController` kontrolagailuak egiten du sarrera zein irteera eskarien erabilpen kasuen kontrola. Aurreko bi kasuetan ez bezala, hontan eskariak eta hauekin erlazionaturiko produktuen errenkadekin loturiko erabilpen-kasu guztiak kontrolatzen dira. Index beharrean, ezberdintzeko Entradas eta Salidas erabilpen-kasuak ditugu, eta hauetaz gain *VerEntrada, CrudEntrada, VerSalida, CrudSalida, GuardarEntrada, GuardarSalida, EliminarEntrada, EliminarSalida, VerLEntrada, CrudLEntrada, VerLSalida, CrudLSali-*

da, GuardarLEntrada, GuardarLSalida, EliminarLEntrada eta EliminarLSalida; L hizkia dutenak produktuen errenkadei dagozkien kasuak izanik.

Erabiltzen diren ereduak *ENTC*, *ENTL*, *SALC* eta *SALL* dira.

6.7 Inbentarioaren kudeaketa

Terminalen aplikazioari dagozkion erabilpen-kasu guztiak `TerminalesController` kontrolagailuan zentralizatu dira. Inbentarioaren kudeaketari dagozkion eragiketak dira *Inventario, DetalleInventario, AddInventario, DelInventario, GuardarInventario*. Gordetzerakoan konprobazio eta balidazioak egiten dira, sartutako kokalekua baliozkoa dela konprobatzeko adibidez.

Honetarako *UBCI* ereduaren erabilera egiten da.

6.8 Lan-aginduen kudeaketa

Lan-aginduen kudeaketari dagokionez, hainbat erabilpen-kasu jasotzen dira `TerminalesController` kontrolagailuan. Alde batetik, sarrera eta irteera eskariak bistaratu behar dira, erabiltzaileei esleitzeko eta gero prozesatu ahal izateko. Hortaz, eskariak ikusteko *Entradas* eta *Salida* funtzioak.

Hauek ikustean erabiltzaile batek bere buruari eslei diezaioke. Momentu horretan *AsignarENTC* ala *AsignarSALC* operazioak egiten ditu, motaren arabera. *ENTU* eta *SALU* ereduak erabiliz, erregistro berri bat gordetzen da, eskari eta erabiltzaile arteko lotura hori duena.

Terminaletako aplikazioaren orrialde nagusian esleitutako eskari hauek ikus daitezke. Hau `HomeController` kontrolagailuak kontrolatzen du.

```
1  public ActionResult IndexT()
2  {
3    if (Session["LogedUserID"] != null)
4    {
5      PedidosModel pm = new PedidosModel();
6      pm.EntradasAsignadas = entradasUsuario.ListarByUser(Session["
       LogedUserFullname"].ToString());
```

```
7       pm.SalidasAsignadas = salidasUsuario.ListarByUser(Session["LogedUserFullname"
        ].ToString());
8       return View(pm);
9     }
10    else
11    {
12      return RedirectToAction("Login", "User");
13    }
14
15  }
```

PedidosModel klaseari esker, bistari hainbat eredu batera pasa ahal dizkiogu.

Behin hau eginda, erabiltzaile batek sarrera edo irteera eskaria prozesa dezake. Honek hainbat inplikazio ditu, alde batetik sarrera (*ENTC*) edo irteera (*SALC*) tauletatik informazioa ezabatu behar da, prozesatutako eskaria erregistratu aurretik (informazio hau txostenetan erakusten da, besteak beste).

Prozesatzeko pantailan eskarien informazioa erakutsi behar da, horretarako *VerEntrada* eta *VerSalida* eragiketak daude. Hauetan informazioa eskuratzen da eta hasiera data aldatzen, prozesatzen ari garela adierazteko. Behin eskaria prozesatzerakoan aipatutako operazioak egin behar dira, eta gainera informazio guztia, txostenen tauletara kopiatu. Hau, honako kodearekin egiten da (bakarrik sarrerena azaltzen da laburtzearren):

```
1   public ActionResult ProcesarSalida(int codigo)
2   {
3     SALC salida = salidas.Obtener(codigo);
4     salida.dSALC_FFIN = DateTime.Now;
5     HSAC hSalida = new HSAC();
6     hSalida.bHSAC_AUTOMATICO = salida.bSALC_AUTOMATICO;
7     hSalida.dHSAC_FECHA = salida.dSALC_FECHA;
8     hSalida.dHSAC_FFIN = (DateTime) salida.dSALC_FFIN;
9     hSalida.dHSAC_FFINREAL = (DateTime) salida.dSALC_FFIN;
10    hSalida.dHSAC_FINICIO = (DateTime) salida.dSALC_FINICIO;
11    hSalida.dHSAC_FINICIOREAL = (DateTime) salida.dSALC_FINICIO;
12    hSalida.id = salida.id;
13    hSalida.sHSAC_CLIENTE = salida.sSALC_CLIENTE;
14    hSalida.sHSAC_MUELLEPLAYA = salida.sSALC_MUELLEPLAYA;
15    hSalida.sHSAC_NALBARAN = salida.sSALC_NALBARAN;
16    hSalida.sHSAC_NPEDIDO = salida.sSALC_NPEDIDO;
17    hSalida.sHSAC_PRIORIDAD = salida.sSALC_PRIORIDAD;
```

```
18    hSalida.sHSAC_TRANSPORTISTA = salida.sSALC_TRANSPORTISTA;
19    hSalida.sHSAC_USUARIO = Session["LogedUserFullname"].ToString();
20
21    if (TryValidateModel(hSalida))
22    {
23      hSalida.Guardar();
24    }
25
26    foreach(var item in salida.SALL)
27    {
28      HSAL lhSalida = new HSAL();
29      lhSalida.dHSAL_FSALIDA= DateTime.Now;
30      lhSalida.id = item.id;
31      lhSalida.sHSAL_CANTIDAD = item.sSALL_CANTIDAD;
32      lhSalida.sHSAL_CODARTICULO = item.sSALL_CODARTICULO;
33      lhSalida.sHSAL_EAN = item.sSALL_EAN;
34      lhSalida.sHSAL_FORMATO = item.sSALL_FORMATO;
35      lhSalida.sHSAL_IDSALIDA = item.sSALL_IDSALIDA;
36      lhSalida.sHSAL_USUARIO = Session["LogedUserFullname"].ToString();
37      lhSalida.Guardar();
38    }
39
40    salida.Eliminar(salida.id);
41
42    return RedirectToAction("IndexT", "Home");
43  }
```

Ikus daitekeen moduan, eredu ezberdinen artean informazioa trukatzen da, eta aldaketak gorde, kasu honetan irteera eskaria ezabatu aurretik. Honela prozesatutako eskariaren informazio guztia erregistraturik geratzen da.

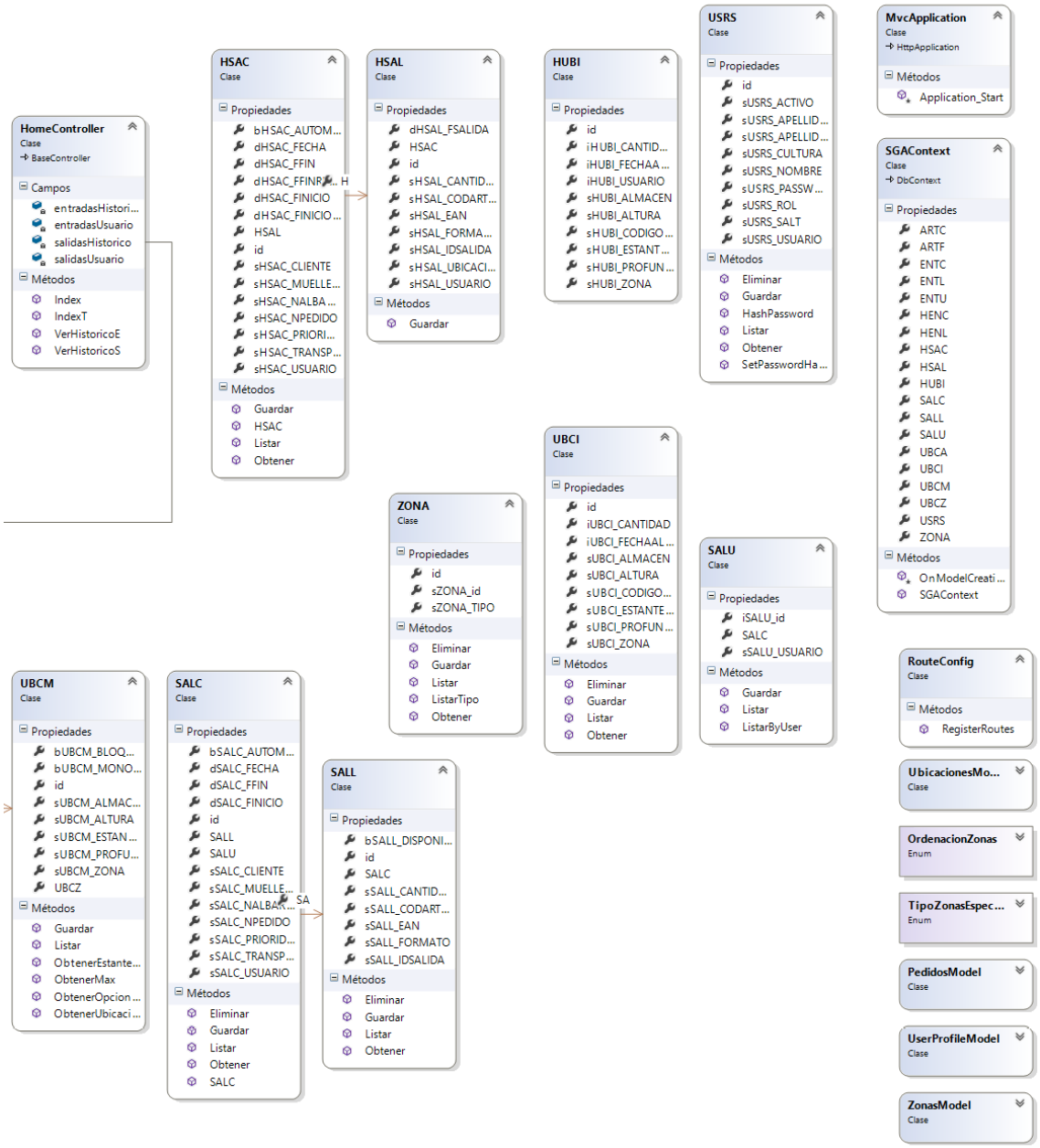
6.9 Txostenen kudeaketa

Aurreko puntuan aipatzen den bezala betetzen da txostenetan azaltzen den informazioa. Informazio honetaz gain, pendiente daude sarrera eta irteera eskariak ere erakusten dira. Hau `HomeController` kontrolagailuan kontrolatzen da, azken finean, mahaigaineko aplikazioaren orrialde nagusian bistaratzen baita informazio guzti hau.

Hau da bertan erabiltzen kodea eginkizun honetarako:

```
private ENTU entradasUsuario = new ENTU();
private SALU salidasUsuario = new SALU();
private HSAC salidasHistorico = new HSAC();
private HENC entradasHistorico = new HENC();

public ActionResult Index()
{
  if (Session["LogedUserID"] != null)
  {
    PedidosModel pm = new PedidosModel();
    pm.EntradasAsignadas = entradasUsuario.Listar();
    pm.SalidasAsignadas = salidasUsuario.Listar();
    pm.EntradasHistorico = entradasHistorico.Listar();
    pm.SalidasHistorico = salidasHistorico.Listar();
    return View(pm);
  }
  else
  {
    return RedirectToAction("Login", "User");
  }

}

public ActionResult VerHistoricoE(int codigo)
{
  return View(entradasHistorico.Obtener(codigo));
}

public ActionResult VerHistoricoS(int codigo)
{
  return View(salidasHistorico.Obtener(codigo));
}
```

Ikus daitekeen moduan hainbat eredu erabiltzen dira datubaseari kontsulta egiteko. Gainera, mahaigaineko orrialde nagusian informazioa guztia bistaratzen da, eta aukera daukatu prozesatutako sarrera zein irteeren informazioa ikusteko *VerHistoricoE* eta *VerHistoricoS* eragiketei esker.



6.2 Irudia: Klase diagramaren bigarren zatia

```
    $("#ubicaciones-body").on('click', '#butEditarUbicacion', function () {
        getOpcionesUbicacion();
        $("#opcionesUbicacion").toggle();
    });

    $("#butGuardarOpcionesUbicacion").click(function () {
        guardarOpcionesUbicacion();
    });
});

//Funciones de obtención e inclusión en el DOM de información, se obtiene la información en formato JSON

function rellenarZonas(value) {
    var url = '@Url.Action("ZonasPorAlmacen")';
    $.getJSON(url + "/" + value,
        function(data) {
            $("#zonas").hide();
            $("#estanterias").hide();
            $("#ubicaciones").hide();
            $("#zonas-body").empty();
            $("#zonas").show();
            $.each(data, function(i, item) {
                $("#zonas-body").append("<a class='btn btn-default btn-sm zona' role='button' id='" + item.sUBCZ_ZONA + "'>"
                    + item.sUBCZ_ZONA + "</a> ");
            });
            $("#idAlmacenSel").attr("class", value);
            $("#zonas-body").append("<a class='btn btn-primary btn-sm' role='button' href='#' id='butNuevaZona' >"
                + '@Html.Localize("nuevaZona")' + "</a> ");
        });
}
```

6.3 Irudia: ZonasPorAlmacen funtzioari deia JQuery bitartez

7. KAPITULUA

Sistemaren ebaluazio eta hobekuntzak

Kapitulu honetan garatutako sistemaren gaitasunak ikusteko egindako probak azalduko dira, baita proba horiek egindakoan detektatutako arazoak eta emandako konponbideak ere.

Sistemaren garapenaren hasiera-hasieratik egin ditugu probak, osagai berri bat edota funtzionalitate berri bakoitza inplementatzean, frogatu egin da esperotako erantzuna lortzen dela.

Modu horretan, funtzionalitate berriak garatzean aurretik detektatutako soluzioak aplikatu dira eta honela garatzeko orduan eraginkortasuna areagotu da.

Sistema bere osotasunean garatua egondakoan, proba sakonagoak egin dira, sistemaren funtzionamendu normalean aurrekusita ez dauden egoerak behartuz.

Egindako probak bi multzo nagusitan banatu ditzakegu: sistemaren funtzionamendu egokia bermatzeko probak eta erabiltzaileari behar den informazioa ematen zaiola bermatzeko probak; hau da, erabiltzaileak behar ez den informazioa sartzean errore pantailak ez azaltzea eta beharrezko mezu identifikagarriak erakustea edo, gutxienez errore hauek kasu oso arraroetan emango direla bermatzea, erabilpen-esperientzia ez kutsatzeko.

Sistemak behar bezala funtzionatzen duela frogatzeko nahiko proba sinpleak egin dira, erabilpen-kasu ezberdinei dagozkien formularioetan datuak sartu, zein aplikaziotik nabigatu eta informazioa bistaratzea. Bistaratzeekin orokorrean ez da arazo handiegirik izan, baina formularioekin dexente arazo izan dira, `RouteConfig`-ari dagokion konfigurazioak egitea ez delako oso erraza hasieran behintzat eta, honetaz gain, formularioetan eredueta-

ko datu batzuk formatu konkretu batean idatzi behar direlako. Ez idazteak, aplikazioaren errore bat bistaratzen zuen kasu askotan.

Hau konpontzeko, balidazioak erabili dira, kontrolagailuetan beharrezko ereduetan informazioa zuzen dagoela konprobatuz, eta okerreko kasuetan zein *exception* kasuetan hauek kontrolatu eta beharrezko mezuak bistaratuz erabilpen kasu horren bistan.

Honetarako, kontrolagailuan zein bistetan idatzi behar dira kode zati batzuk. Zuzendutako kasu konkretu bat hau izan da:

```
1  public ActionResult Guardar(ARTC article, int orden)
2  {
3
4    if(ModelState.IsValid)
5    {
6      switch (orden)
7      {
8        case 1:
9        article.bART_FIFO = true;
10       break;
11       case 2:
12       article.bART_LIFO = true;
13       break;
14       case 3:
15       article.bART_LOTE = true;
16       break;
17     }
18     try
19     {
20       article.Guardar();
21       return Redirect("~/articulos/ver/" + article.sARTC_CODIGO);
22     }
23     catch (Exception)
24     {
25       ModelState.AddModelError("", Localization.Localize("excepcionCreacion"));
26       return View("~/Views/articulos/Crud.cshtml", article);
27     }
28
29     }
30   else
31   {
```

```
32      return View("~/Views/articulos/Crud.cshtml", article);
33      }
34
35  }
36
```

Ikus daitekeen moduan, aldaketak gordetzerako orduan, arazoren bat gertatzen bada, zein ereduaren datuak baliozkoak ez badira, berriz ere formulario berdina erakusten da, eta bertan errore mezu bat bistaratuko da, 7.1 irudian ikus daitekeen moduan.

7.1 Irudia: Artikulua sortzerakoan balidazioa

Hau posible egiteko, beharrezkoa da bistan ere balidazio kodea sartzea. Alde batetik, goian azaltzen den errore mezu orokorrerako (*ValidationSummary*) eta, bestetik eremu bakoitzean jasotzen den erroreetarako (*ValidationMessageFor*). Eremuetarako, aurretik derrigorrezko bezala definiturik egon behar dute eremu horiek, zorionez, *Entity Framework*-ak automatikoki egiten du hau.

Dagokion bistako kode zati honetan ikus daiteke, balidazioa egiteko beharrezko kodearen zati bat:

```
1   ...
2
3   @using (Html.BeginForm("Guardar", "Articulos", FormMethod.Post, new { id = "frm-
        articulo" }))
```

```
4  {
5    @Html.HiddenFor(x => x.id)
6    <div>
7    @Html.ValidationSummary(true, Localization.Localize("excepcionValidacion"), new {
       @class = "alert alert-danger" })
8    </div>
9    <div class="panel panel-default">
10   <div class="panel-heading">@Html.Localize("infoArticulo")</div>
11   <div class="panel-body">
12   <div class="form-group">
13   <label>@Html.Localize("codigo")</label>
14   @if(Model.sARTC_CODIGO == null)
15   {
16     @Html.TextBoxFor(model => model.sARTC_CODIGO, new { @class = "form-control"})
17     @Html.ValidationMessageFor(model => model.sARTC_CODIGO, null, new { @class = "
       label label-danger" })
18 ...
19
```

Aipatutako bi balidazioak ikus daitezke adibide honetan eta, honetaz gain, balidazio horiei *Bootstrap*-en estiloak aplikatu zaizkio ikusgarriago egiteko.

Erabiltzaileak espero ez den informazioa sartzen duen kasuetan ere balidazioak egin dira arazoa konpontzeko, detektaturiko kasu guztien beharrezko tratamendua eginez.

8. KAPITULUA

Jarraipena eta kontrola

Kapitulu honetan proiektuaren garapenean izan ditugun gorabeherak azaltzen dira, proiektuaren garapenean izandako desbideraketak, kalitatearen kontrola eta baita identifikatutako arriskuek izan duten eragina ere.

8.1 Proiektuaren garapena

Proiektuaren garapena hasiera batean definitutakoarekin bat eginez eramaten saiatu gara; gauza batzuk egitea uste baino gutxiago kostatu da, eta beste batzuk, berriz, uste baino gehiago.

Proiektuaren hasieran, produktuaren beraren hasieraketa izanik eta garatu behar den lehenengo azpiatalarekin jartzean konfigurazio eta lotura guztiak egin behar dira lehen aldiz, honek suposatzen duen zailtasunarekin. Lehenengo urrats hau nahiz eta sinplea izan kostatu egin da esandako arrazoiengatik baina nahiko ondo dimentsionatuta egon da ordu aldetik.

Gero artikuluen garapenak eremuek dituzten xehetasunetatik haratago ez dute arazo handirik suposatu eta esfortzu handiena eremuen balidazioetan eman da.

Kokalekuen kudeaketa egitea pisutsua eta konplikatua izan da, biltegi baten funtzionamendua ongi ezagutzea beharrezkoa delako garapena ondo egiteko eta horretarako Aitor Crespo interesdunaren laguntza ezinbestekoa izan da garapena aurrera eraman ahal izateko denbora gehiegirik galdu gabe.

Azken hiru atazek konplexutasun aldetik ez dute arazo handiegirik izan, terminalen bertsioa egiteak suposatzen duenetik haratago. Hori bai, beharrezko datubaseko ereduak prest ez egoteak atal hauen epeak atzeratzea suposatu du eta zertxobait denboran pasatzea atzetik datozen gainerako konpromezuak atzeratu behar izateaz gain.

Kontuan izan behar da, azpiatal bakoitzean itzulpen-testuen kudeaketa egin behar izan da, testuak dagokion hizkuntzan ikusi ahal izateko, hau azpiatal bakoitzaren dimentsionaketan aurreikusita zegoen gauza bat izan da.

8.1.1 Desbideraketak

Proiektuaren garapenean, hasierako plangintzatik alderatuz, desbideraketa batzuk izan dira, aurreko atalean azaldu den bezala, baina desbideraketa garrantzitsuenen arrazoiak azalduko ditugu jarraian. Behar izan den denbora erreala, teorikoarekin batera, gutxibeherako orduak dira, ezin baitira zehatz-mehatz neurtu. Halaber, errealitatearen ahalik eta erreflexu onena izan daitezen saiatu gara. Horretarako *Kanban* [1] metodologiaz lagundu gara.

Desbideraketa positiboak ere izan dira eta honek, nolabait desbideraketa orokorra hain handia ez izatea ekarri du %6,49, hortaz, proiektuan zehar izandako gorabeherak nahiko txukun kudeatu ahal izan dira. 8.1 irudian ikus daitezke ataza bakoitzean izandako desbiderapenak.

Kudeaketa

Plangintzan aldaketa batzuk eman dira, aurreko puntuan azaldu den bezala datubaseko ereduekin izandako arazoengatik atzerapena egon delako, eta honekin lotuta, denbora garela bat. Hau kudeatu egin behar izan da nolabait, eta horregatik hor erreflejatzen da desbideraketa hori.

Garapena

Garapenari dagokionean, aipatu da iada izandako arazoa. Honek bi atalen garapenari eragin dio (Lan-aginduak eta Inbentarioa) eta, bi kasu hauetan nahiz eta ordu kopuruan asko

[1] Kanban metodologia txarteletan oinarritzen da, eta txartel hauekin egin beharreko lana identifikatu, deskribatu, mugatu eta kuantifikatu daiteke.

ez izan, atazetara bideratutako denbora teorikoarekin konparatuz %50-eko hazkundea suposatzen du bi kasuetan.

Bestalde, denbora gehixeago erabili da ere proba eta balidazioak egiteko eta kasu honetan ere ordu asko izan ez arren ehuneko potoloa azaltzen zaigu taulan.

Kokalekuen kudeaketak duen konplexutasunak denbora apur bat gehiago behar izatea suposatu du, %12,50-eko desbideraketak adierazten duen moduan.

8.1.2 Kalitatea

Gure proiektuarekin lortu dugun kalitatea aztertzen da atal honetan. Jarraian, 2.7.1 atalean definitutako kalitate-adierazleak aztertu dira.

Adierazle kuantitatiboak

- Plangintzaren jarraipena: plangintzan definitu diren datak eta epe-mugak errespetatzea. Ahalik eta kontrol hoberena eraman da, kontuan izanik zuzenean gure esku ez dauden ezustekoak gertatu direla. Proiektuaren osotasunari begira desbiraketa hauek ez dira oso handiak izan eta, egindako kudeaketa txukuna izan da.

- Proiektuaren fitxategi kopurua: fitxategi asko sortu dira, erabilpen kasuekin zuzenki lotutakoak eta MVC patroia jarraituz.

Adierazle kualitatiboak

- Sistemarekiko elkarrekintza: sistema erabiltzerakoan erabiltzaileentzako erabilerraza izan behar du, geroz eta errazagoa izan erabiltzaile eta sistemaren arteko elkarrekintza, produktuaren kalitatea hobea izango baita. Hau, *Boostrap* erabiliz lortu da, eta balidazioek eskeintzen duten informazioari esker aplikazioa erabiltzeko errazago egitea lortu da.

- Aplikazioaren fidagarritasuna: Balidazioen bitartez, erabiltzaileek behar ez den informazioa sartzen dutenean aplikazioak huts ez egitea kontrolatu da.

- Kodearen argitasuna: kodearen antolaketa ona, beharrezko komentario eta argibideekin. MVC arkitektura erabili denez antolaketa ona bermatua dago nolabait, eta

beharrezko kasuetan komentarioak idatzi dira, ahalik eta ulergarrien egiteko idatzitako kodea.

8.1.3 Arriskuak

2.8 atalean proiektuaren arriskuak identifikatu dira. Arriskuak garapenean izandako eragina aztertuko da segidan.

- **Datuen galera:** *Drive* zein *Git* erabili dira helburu hauetarako eta kodea zein dokumentazioa hainbat tokitan egon da aldi berean, beraz datuen galera posiblea guztiz ekidin da.
- **Proba zerbitzaria eskuragarri ez izatea proiektuan zehar:** Hasieratik eskuragarri izan dugu eta inolako momentuan ez dugu arazorik izan garapenarekin aurrera jarraitu ahal izateko zerbitzariaren funtzionamenduari dagokionean.
- **Aplikazioaren garapenean arazoak:** garapenean arazoak egon diren kasuetan *Git* erabiliz denez, egindako aldaketak behar ezkero atzera botatzea posible izan da, edo konparaketak egitea eta arazoaren arrazoia zein zen errazago ikustea. Honek, garapenean denbora aurreztu du.
- **Plangintza ez jarraitzea:** plangintza behar bezala jarraitu da, nahiz eta bidean desbideraketak egon, egindako kudeaketari esker hasieran finkatutako produktua lortzea posible izan da. Gainera, ikerketari dedikatutako denbora ere oso aberasgarria izan da, garapenarekin lagundu duen informazioa eskuratu baita.

Kodea	Atazaren deskribapena	Estimazioa	Erreala	Desbideraketa	
1	Kudeaketa	35	37	2	5,71%
1.1	Plangintza	14	14	0	0,00%
	Egutegia definitu	12	12	0	0,00%
	Arriskuen kudeaketa plana	2	2	0	0,00%
1.2	Jarraipena eta kontrola	21	23	2	9,52%
	Bilerak	5	4	-1	-20,00%
	Plangintzaren aldaketak	3	4	1	33,33%
	Arriskuak eguneratu	3	2	-1	-33,33%
	Kalitatea kontrolatu	5	4	-1	-20,00%
	Aldaketak kudeatu	5	4	-1	-20,00%
2	Ikerkuntza	40	40	0	0,00%
2.1	Ikasketa	30	25	-5	-16,67%
2.2	Osagaien bilaketa	10	15	5	50,00%
3	Arkitektura	20	20	0	0,00%
3.1	Analisia	10	10	0	0,00%
3.2	Diseinua	10	10	0	0,00%
4	Garapena	125	148	23	18,40%
4.1	Web Aplikazioa	115	133	18	15,65%
	Erabiltzaileen kudeaketa	20	20	0	0,00%
	Artikuluen kudeaketa	15	15	0	0,00%
	Kokalekuen kudeaketa	40	45	5	12,50%
	Lan-aginduen kudeaketa	20	30	10	50,00%
	Inbentarioaren kudeaketa	10	15	5	50,00%
	Txostenak	10	8	-2	-20,00%
4.2	Probak	10	15	5	50,00%
5	Itxiera	90	85	-5	-5,56%
5.1	Memoria	80	75	-5	-6,25%
5.2	Defentsa	10	10	0	0,00%
		310	330	20	6,45%

8.1 Irudia: Hasierako planarekiko izandako desbiderapenak.

9. KAPITULUA

Ondorioak

Kapitulu honetan proiektuarekin lortutako emaitzak eta ateratako ondorioak azalduko dira, eta azkenik, egindako lanarekin aurrera jarraituz gero zein aukera edukiko genituzkeen aztertuko da.

9.0.1 Lortutako emaitza

Oinarrizko produktuaren helburua finkatzerakoan lortu nahi zen produktua eskuratzea posible izan da. Gainera, guztiz ezberdinak diruditen bi aplikazio sortu dira proiektu bakar baten barruan, honek dakarren mantentzeko abantailekin.

Bootstrap-en laguntzaz itxura erakargarriko eta dispositibo ezberdinetara egokitzen den web-aplikazio bat lortu da biltegi baten oinarrizko atazak burutzea posible egingo duena, bai biltegiko teknikari zein biltegi horretako administratzaile edo buruzagientzat.

Gainera, datuak oker sartzerakoan erabiltzaileari jakin beharreko informazioa itzultzeko gai da egin diren balidazioen kontrolarekin eta horrek aplikazioaren kalitatea areagotzen du.

Noski, soluzio profesional bat izatetik oso urrun dago oraindik produktu hau baina proiektu luze baten hasiera izan daitekeen produktua eskuratu da, beraz, emaitza egokia lortu dela esan genezake.

9.0.2 Ebaluazio pertsonala eta ikasitako lezioak

Proiektu honetan gradu osoan zehar eskuratutako gaitasunak frogan jarri ditugu. Ez soilik diseinuan edota programazioan ikasitakoa, proiektu bat aurrera eramaterakoan gerta litezkeen arazoei ere aurre egin behar izan diegu. Proiektu honen izaeragatik bagenekien buruhausteak eman zitzakeen proiektu bat zela, beharrezko kudeaketa ona egiten ez bazen.

Azkenean, nahiz eta hainbat gorabehera eta arazo izan garapenean, proiektua aurrera ateratzea lortu dugu eta oso gustura gaude egindako lanarekin.

Proiektuaren bizi-zikloan zehar badira ikusi eta ikasi ditugun zenbait gauza, eta hauetatik eratorrita honats interesgarriak izan litezkeen zenbait lezio:

- Ordu asko sartu behar izan da proiektuan zehar eta zenbait unetan lan egiteko ordutegi zein klase-orduengatik proiektuari dedikatzeko orduak nahiko mugatuak izan dira, hortaz ordu horiek klaseak amaitu ostean pilatu dira gehienbat. Hau nolabait ekiditeko, astean helburuak jartzea zein, irismena azpiataletan banatzea oso erabilgarria izan da proiektuaren garapenaren kontrola ez galtzeko.

- Kodearen bertsioetarako kontrola erabiltzea oso erabilgarria suertatzen da, kode horren segurtasun kopiak izateko eta, horretaz gain, egindako aldaketen gainean kontrol handia izateko. Honek, egindako lanaren jarraipen ona egiteko aukera ematen du eta arazoak badaude aldaketak erraz atzera bota daitezke.

- Sareko zerbitzuetan informazia gordetzea oso baliagarria da. Honek eskaintzen duen segurtasunaz gain, edozein momentutan edonon informazio guztia eskura edukitzea oso baliagarria da lana aurreratu ahal izateko.

- Proiektuaren bizi-zikloa nahiko luzea izan da, eta hasieratik gauzak dokumentatzea dirudiena baina garrantzitsuagoa da, gauzak behar bezala dokumentatzen ez badira ahaztu edo galdu egin daitezke eta. Hasieratik saiatu gara dena dokumentatuta eramaten, eta bai proiektuaren garapen-prozesuan eta bai memoria idazterako garaian oso baliagarria izan zaigu hau.

- Proiektuaren memoria dimentsio handiko memoria da eta mota honetako memorietan esperientza eza nabaritu da. Garrantzitsua da hasieratik memoriaren edukiak argi izatea eta ondo egituratzea, osatzerakoan edukiekin arazoak ez izateko.

9.0.3 Etorkizuneko aukerak

Lortutako produktua, esan bezala, produktu minimo bideragarri bat izan da. Erabilpen-kasu asko ditu inplementaturik, baina industriarako soluzio profesional batetik urrun dago oraindik.

Dauden erabilpen-kasuetan erabiltzaileari laguntza zein balidazio gehiago eskaintzeaz gain (bilaketak, informazioaren auto-betetzea...) erabilpen kasu gehiago eta teknologia gehiago sartu beharko litzateke produktuaren potentziala handiagotzeko. Horietako bat *three.js* bezalako *framework* bat erabiltzea izango litzateke, *javascript* erabiliz irudi geometrikoak sortu eta biltegiaren mapa bat osatzeko.

Eranskinak

A. ERANSKINA

Bilera-aktak

A.0.1 Bilera akta

Data: 2016/02/03

Lekua: Donostia

Partaideak: Aitor Crespo (Interesduna), Iñigo González

Bilerako gai zerrenda: Betekizunen analisia, funtzionalitateen definizioa.

Betekizunen analisia

Proiektuak izan behar dituen betekizunei buruz hitzegiten da, proiektuak berak bere izaeragatik dituen mugak kontuan izanik eta garatzeko egongo den denboraren estimazio azkar bat eginez.

Interesdunak duen ideietatik batzuk baztertu egiten dira denbora faltagatik eta bukaeran proiektuaren betekizunak zein izaera definitzen dira.

Funtzionalitateen definizioa

Betekizunak definitu ondoren, irismen errealista bat zein izango litzatekeen hitzegiten da, zein funtzionalitate garatu behar diren eta hauen definizio bat egiten da, proiektuaren ikuspegi bat izateko.

Bileraren bukaeran proiektuaren planifikazioan lanean hasteko nahikoa informazio biltzen da.

A.0.2 Bilera akta

Data: 2016/05/16

Lekua: Donostia

Partaideak: Aitor Crespo (Interesduna), Iñigo González

Bilerako gai zerrenda: Proiektuaren jarraipena.

Garatutako funtzionalitateen berri ematen zaio interesdunari, *feedback*-a jasotzeko asmoz. Bezeroak bere zerbitzarian instalaturik duen web-aplikazioan frogak egiten ditu eta behar diren iradokizunak egiten ditu, bai interfazeari bai funtzionalitateari berari buruz.

Ikusitakoarekin gustora azaltzen da eta falta diren datubaseko taulak aurreratuko dituen konpromezua hartzen du, garapena oztopatu ez dezan.

B. ERANSKINA

Kalitatearen egiaztapen zerrenda

Proiektuaren kalitatearen egiaztapen zerrenda	Bai/ez
Proiektuaren irismena bat dator egindako lanarekin?	
Proiektuaren garapena plangintzatutakoarekin bat dator?	
Egindako estimazioak zuzenak edo onargarriak dira?	
Proiektuaren arriskuak kontrolatuak daude?	
Proiektuaren exekuzioaren gora-berak behar bezala dokumentatu dira?	

Produktuaren kalitatearen egiaztapen zerrenda	Bai/ez
Beharrezko balidazioek zuzen funtzionatzen dute?	
Aplikazioan errore-mezuak kontrolaturik daude?	
Aplikazioaren itxura zein portaera espero dena da?	
Testu guztiak ondo itzulita daude?	

C. ERANSKINA

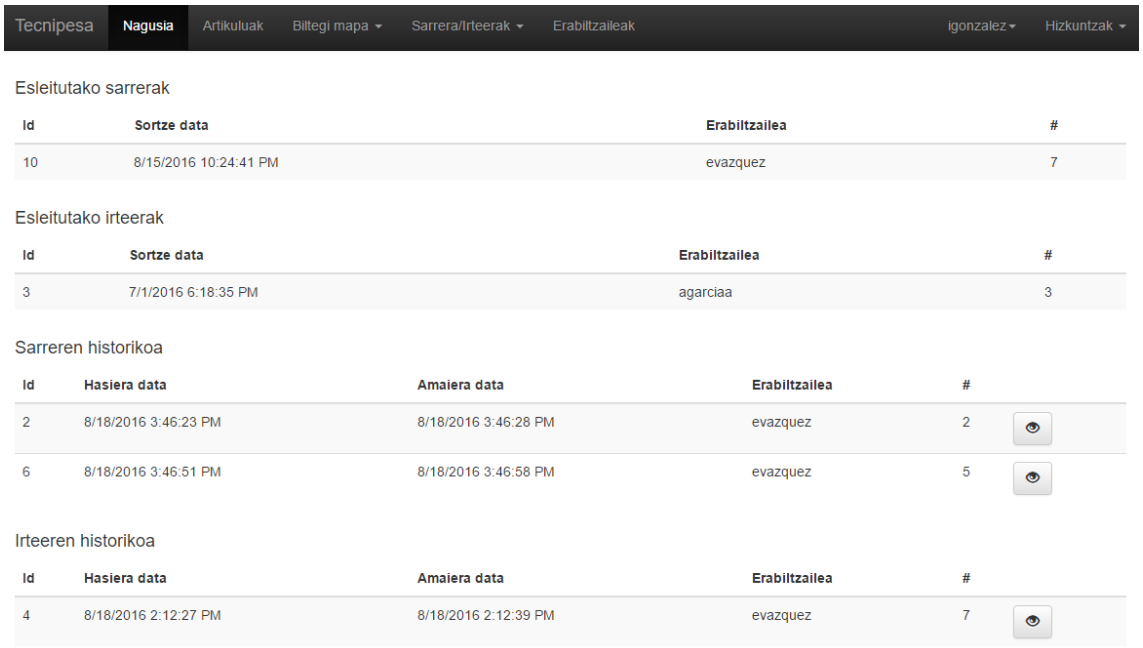
Erabilpen-gida

Saioa hasteko, erabiltzaile eta pasahitza sartu behar duzu, zaren erabiltzai-motaren arabera, terminalentzako aplikazioan ala mahaigaineko aplikazioan sartuko zara.

C.1 Irudia: Login pantaila

C.1 Mahaigaineko bertsioa

C.1.1 Hasierako orrialdea



C.2 Irudia: Login pantaila

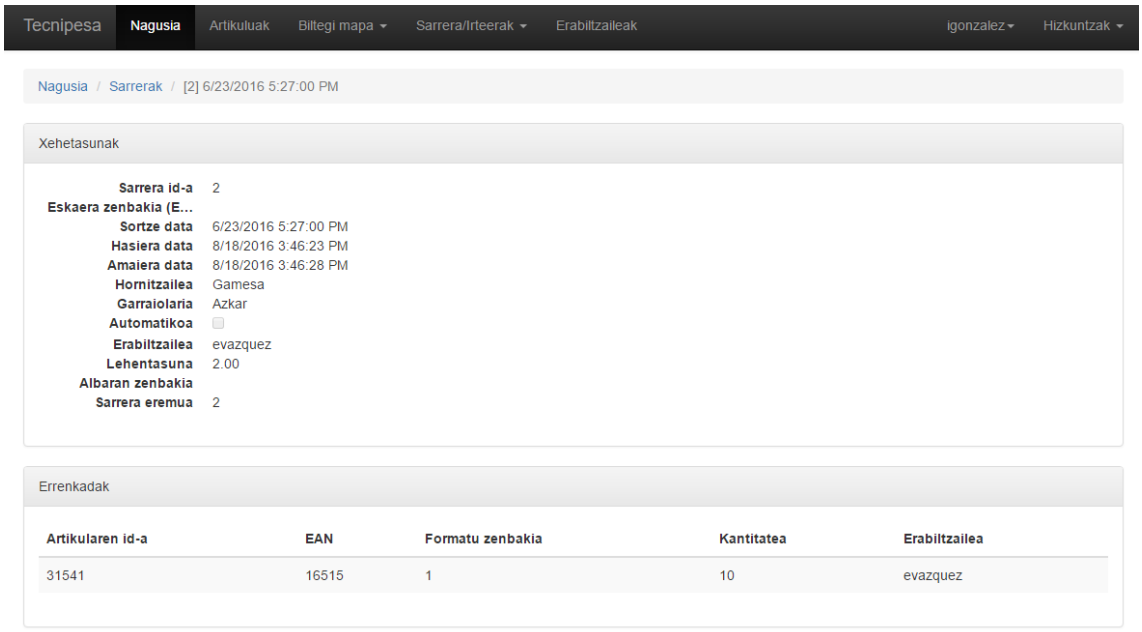
Goian ikusten duzuna, mahaigaineko aplikazioaren orrialde nagusia da. Bertan, operarioei esleitutako eskariak zein hauek prozesatutako eskariak ikusi ahalko dira.

Historikoen ondoan dagoen begiaren botoia zapaldu ezkero, eskari horren xehetasunak ikusi ahalko dira (C.3 irudia).

C.1.2 Artikuluak

Artikuluen sekzioan sartu ezkero, ikusiko den lehen gauza sortutako artikuluan zerrenda izango da. Bertan editatzeko, ikusteko zein ezabatzeko aukerak daude erabilgarri (C.4 irudia).

Artikulua gehitzeko aukeran, datuak sartu behar dira, eta datuak ondo ez badaude errore mezuak azalduko zaizkigu, informazioa ondo bete dezagun (C.5 irudia).

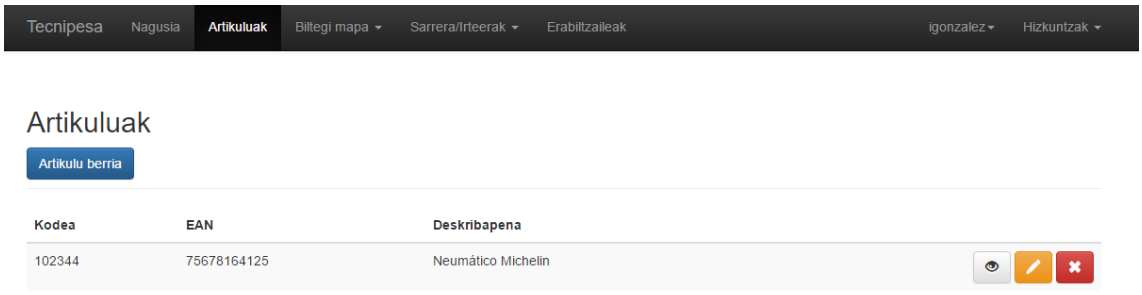


C.3 Irudia: Eskariaren xehetasunak

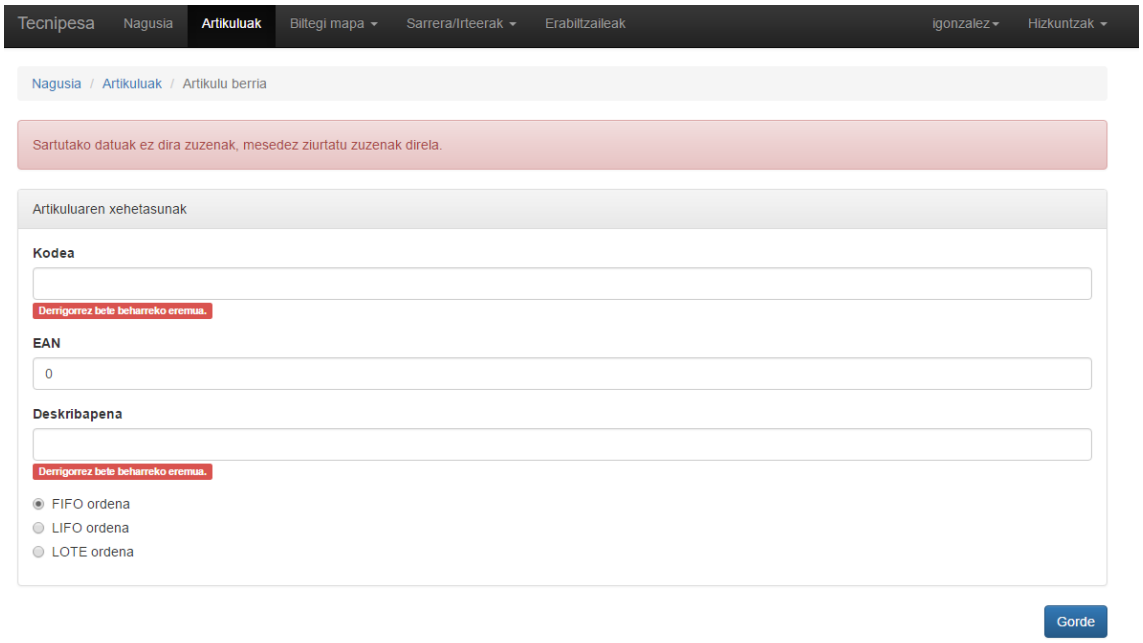
Artikuluaren xehetasunak begiratzen baditugu, artikulua bera eta bere formatuak ikusi ahalko ditugu. Formatu hauen gaineko operazioak ere erabilgarri daude (C.6 irudia).

C.1.3 Biltegi mapa - Kokalekuak

Kokalekuen kudeaketa osoa orrialde beretik egiten da, C.7 irudian ikus daitekeen moduan, operazio guztiak botoi urdinak klikatuz egin daitezke, identifikagailua sartu eta sorketa burutu. Biltegi ezberdin ala eremu ezberdin baten identifikagailua klikatzean, informazioa



C.4 Irudia: Artikuluen zerrenda



C.5 Irudia: Artikuluak gehitu

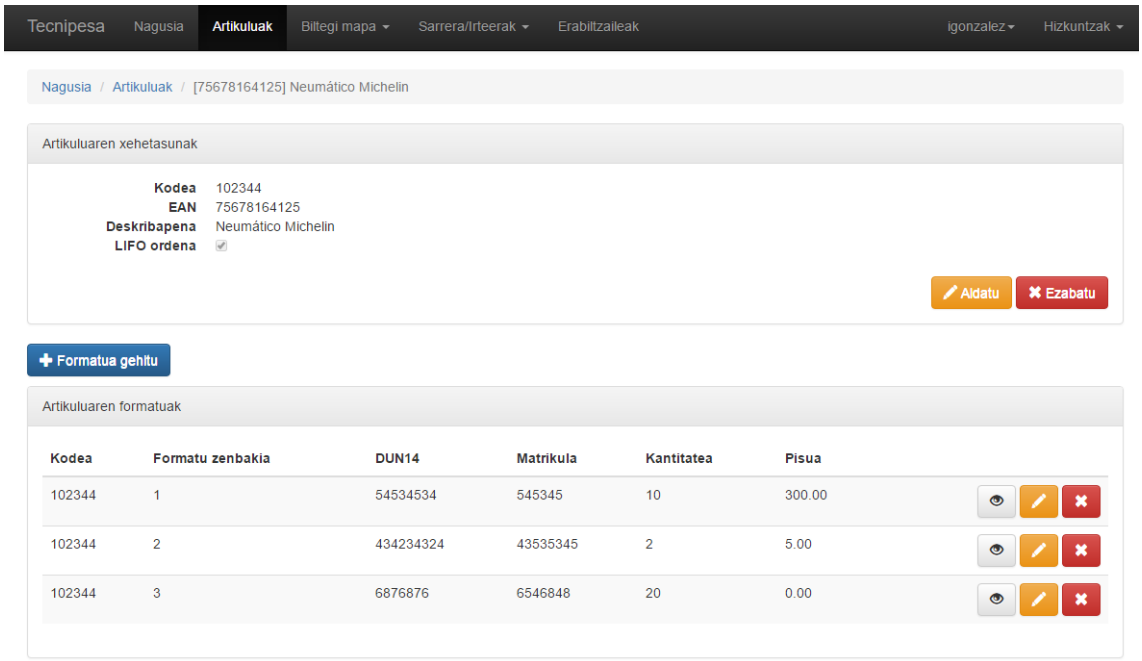
eguneratzen da.

C.1.4 Biltegi mapa - Sarrera/irteera eremuak

Sarrera/irteera eremuen zerrendapena ikusten da C.8 irudian, aukera dugu hauek ezabatu/editatu ala berriak sartzeko, ikus daitekeen moduan.

C.1.5 Eskarien kudeaketa

Sarrera/irteerak menuan klik egin ezkero, sarrera edo irteeren zerrenda ikus dezakegu C.9 irudian ikusten da, eskari baten informazioa, eskari horri erlazionaturiko artikuluekin. Eskaera berri bat sartu nahi badugu (kasu honetan sarrera), berari lotutako informazioa sartu dezakegu sorketan (C.10 irudia), ala ez badakigu, aurrerago editatu eta sartzeko aukera dugu.



C.6 Irudia: Artikulua ikusi

C.1.6 Erabiltzaileen kudeaketa

Erabiltzaileen menuan klik egin ezkero, erabiltzaileen zerrenda ikusiko dugu, bere aukera guztiekin (C.11 irudia). Erabiltzaile berria sortu nahi badugu, bere informazioa eta aukerak sartu beharko ditugu (C.12 irudia).

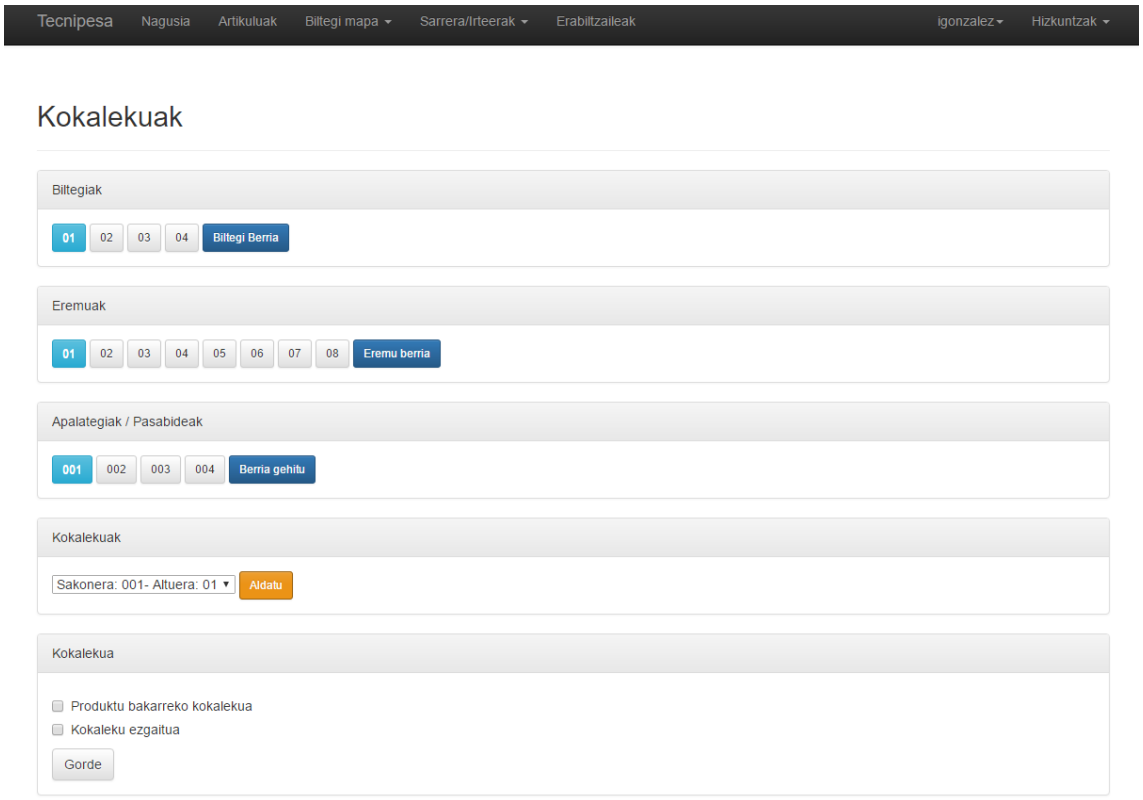
C.1.7 Hizkuntza aldaketa

Edozein momentutan aplikazioaren hizkuntza alda dezakegu, goi-eskubian dugun aukera zapalduz, C.13 irudian ikusten den moduan.

C.2 Terminaletako bertsioa

C.2.1 Hasiera

C.14 irudian ikus ditzakegu erabiltzaileari esleitutako eskarien zerrenta, eta bertan menura ala atzera joateko aukera zein eskari bakoitza prozesatzeko aukera.



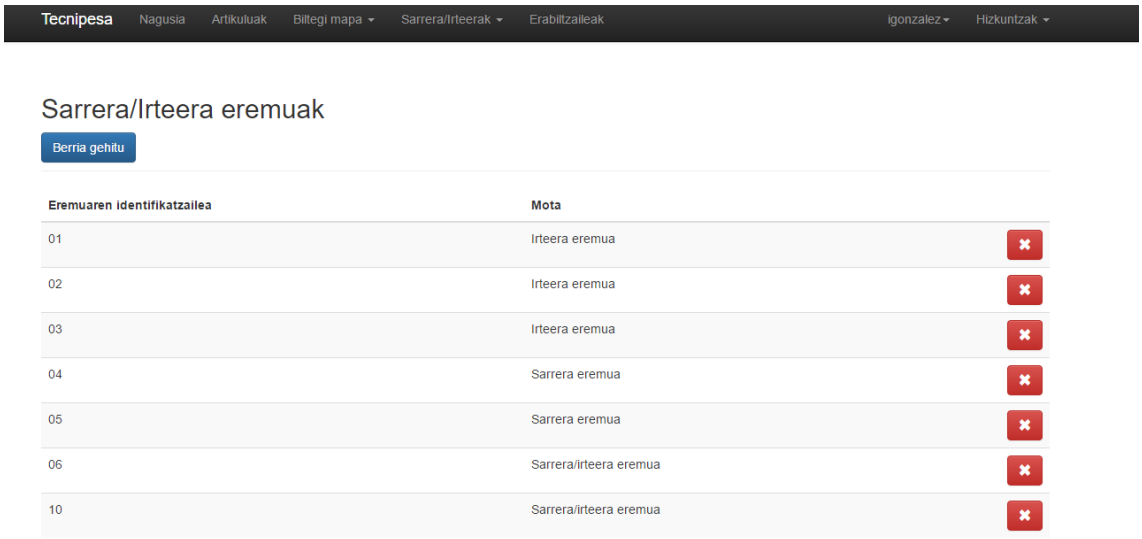
C.7 Irudia: Kokalekuen kudeaketa

C.2.2 Sarrerak

Menuan sarrerak ikusteko aukera egiten badugu, C.15 irudiko informazioa ikusiko dugu, hemen aukera izango dugu geure buruari eskari bat esleitzeko. Ondoren hasierako orrian azalduko zaigu.

C.2.3 Irteerak

Menuan irteerak ikusteko aukera egiten badugu, C.16 irudiko informazioa ikusiko dugu, hemen aukera izango dugu geure buruari eskari bat esleitzeko. Ondoren hasierako orrian azalduko zaigu.



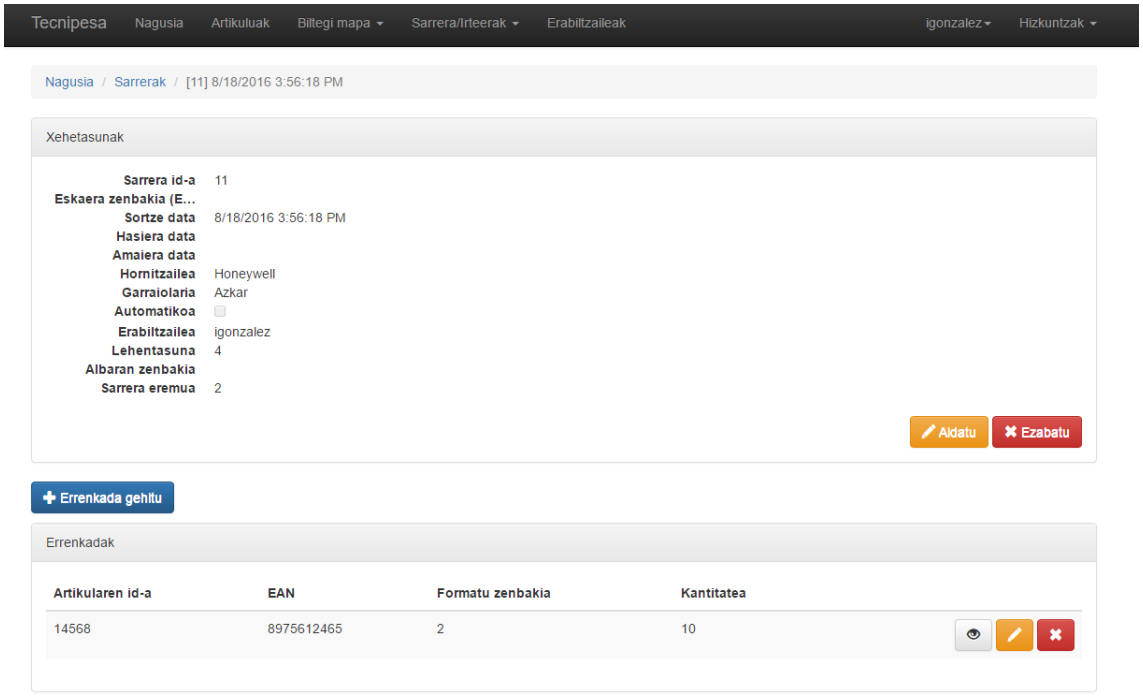
C.8 Irudia: Sarrera/irteera eremuak

C.2.4 Eskaria prozesatu

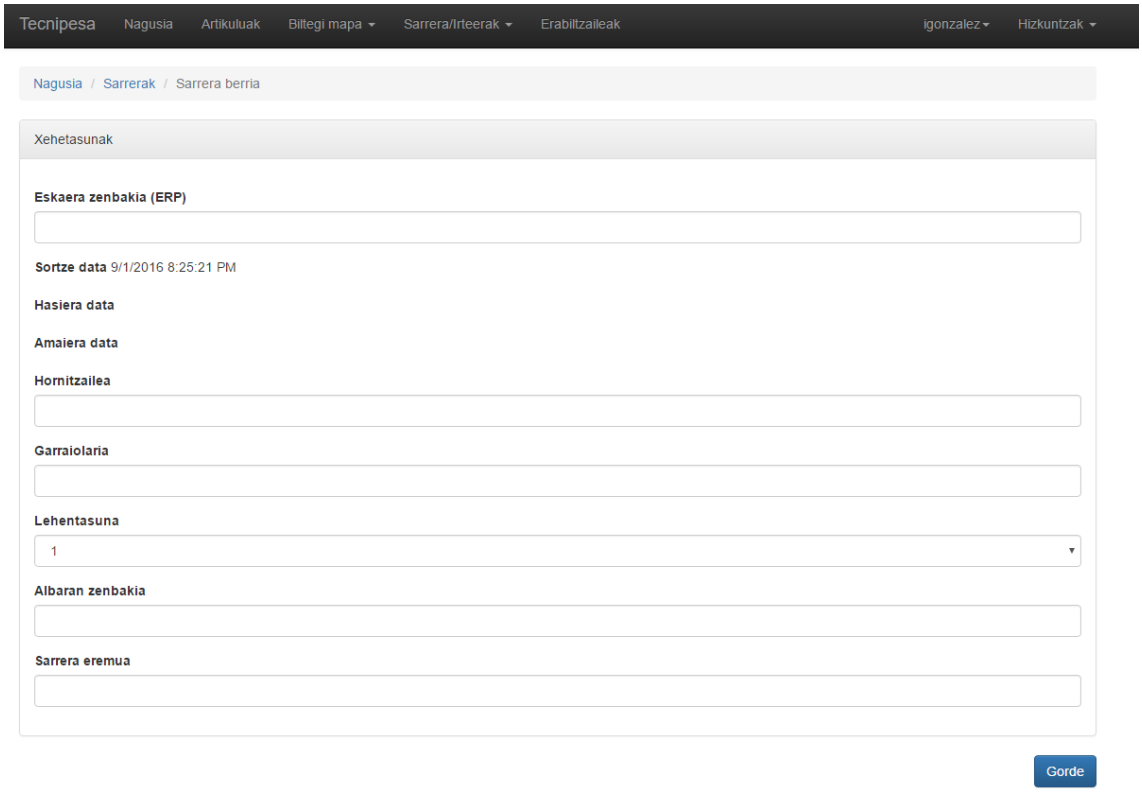
Eskaria prozesatzeko aukera klikatzen badugu, hasierako orriko eskari batean, eskari horren xehetasuna ikusiko dugu eta atzera egiteko ala prozesatzeko aukera (C.17 irudia).

C.2.5 Inbentarioa

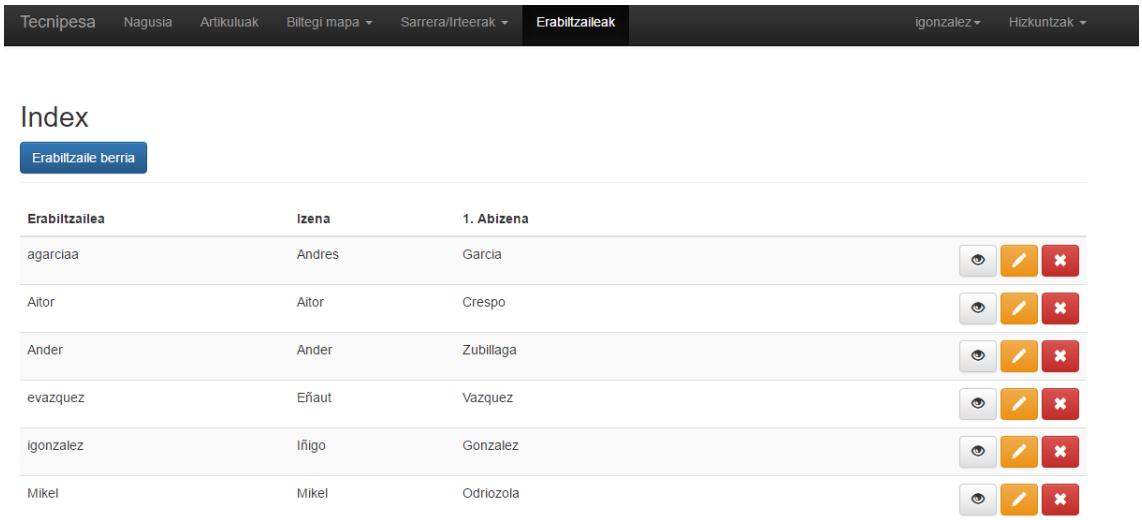
Menuan inbentarioa botoia sakatzen badugu, biltegien inbentarioak ikusiko ditugu (C.18 irudia). Pantaila horretan xehetasunak ikusi ditzakegu ala erregistro berria gehitu, erregistro berria gehitu ezkero, informazio guztia bete beharko da (C.19 irudia), eta kokalekuaren kodea zuzen, bestela ez da sarrera gehituko.



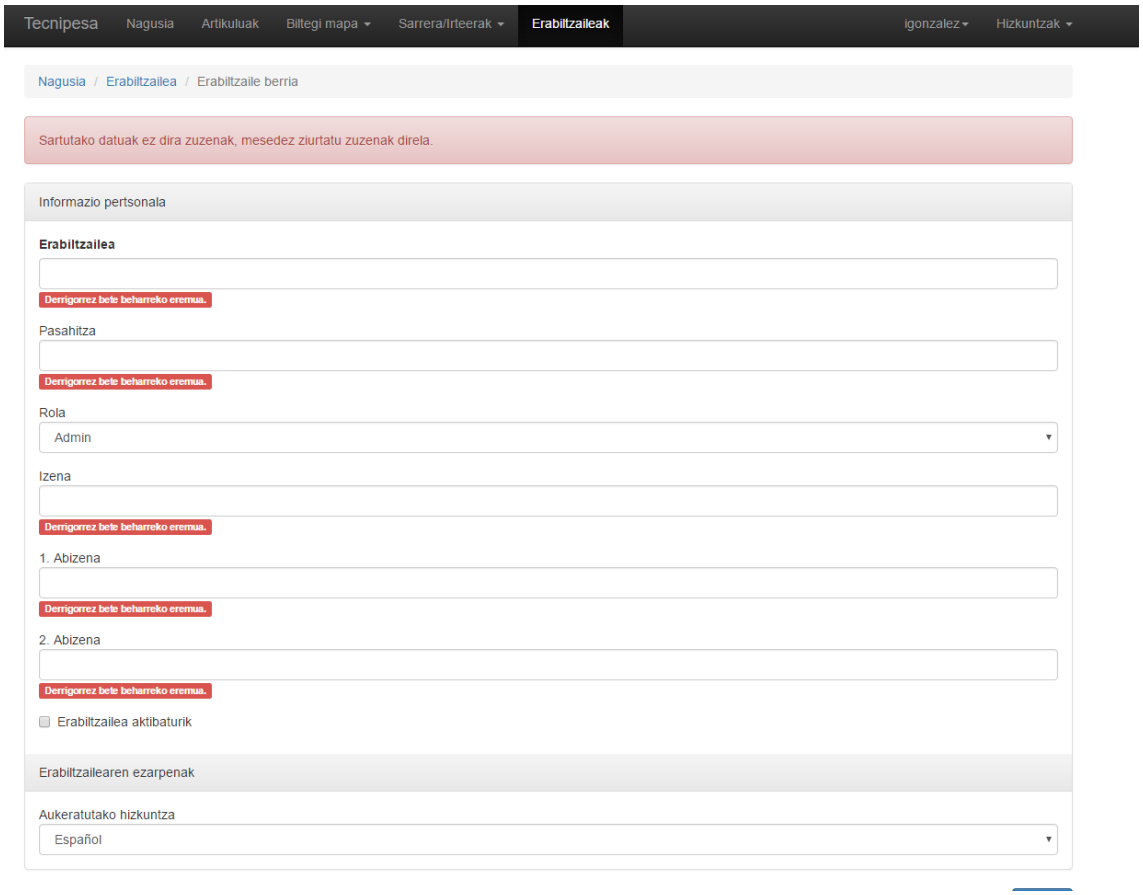
C.9 Irudia: Eskariak



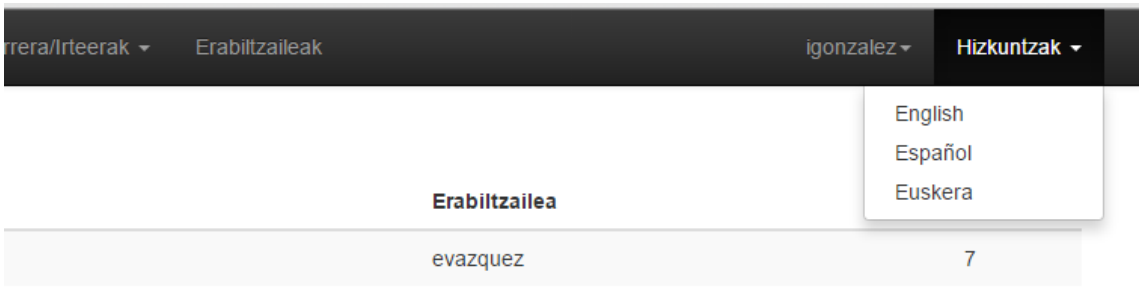
C.10 Irudia: Sarrera berria



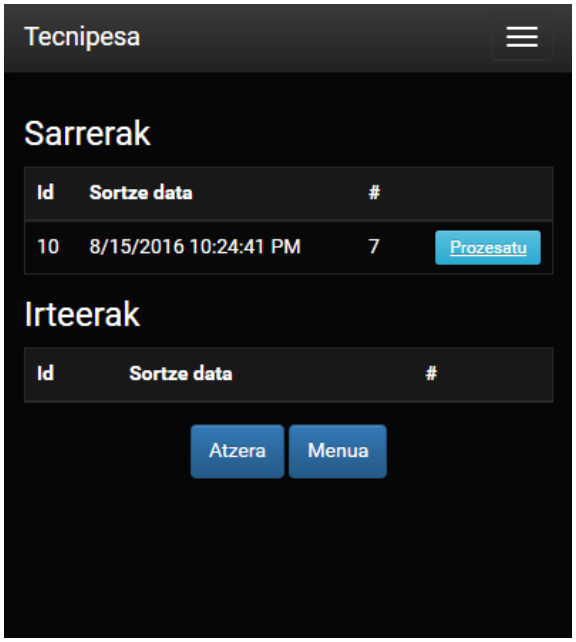
C.11 Irudia: Erabiltzaileen zerrenda



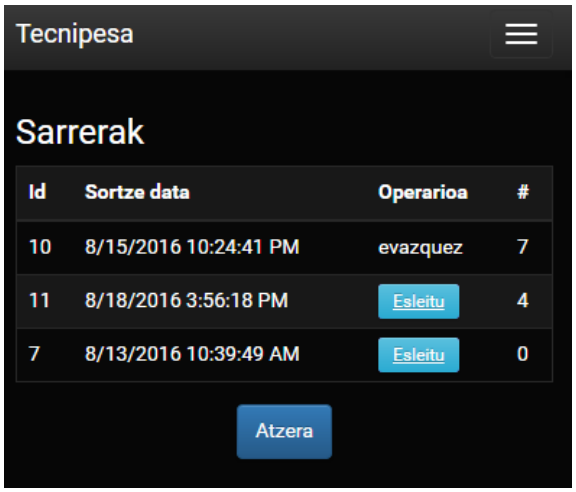
C.12 Irudia: Erabiltzailea sortu



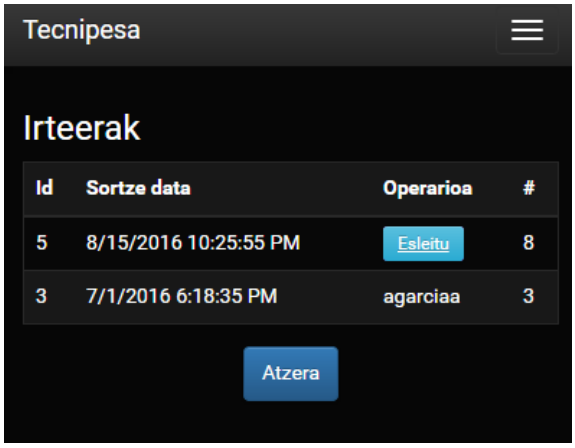
C.13 Irudia: Hizkuntza aldatu



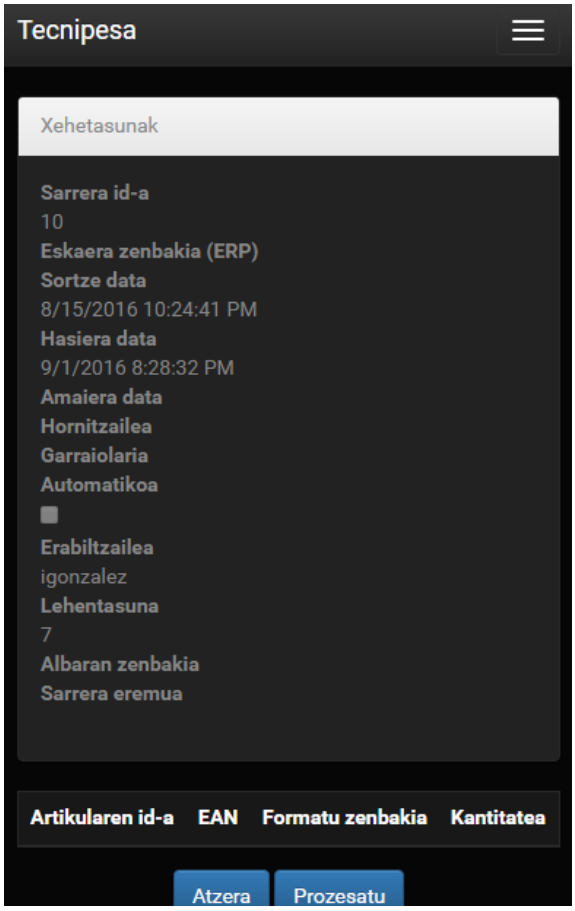
C.14 Irudia: Terminal hasiera



C.15 Irudia: Sarrerak

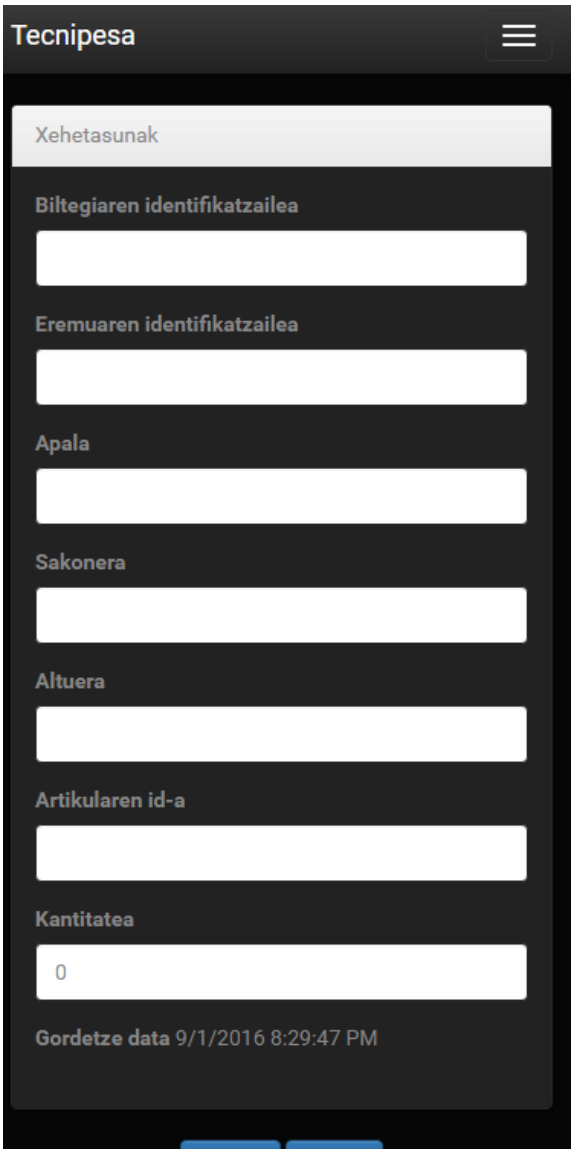


C.16 Irudia: Irteerak



C.17 Irudia: Prozesatu

C.18 Irudia: Inbentarioa ikusi



C.19 Irudia: Inbentariora gehitu

Bibliografia

[Microsoft, 2016] Microsoft (2016). Asp.net core. http://www.asp.net/core.

[Tom Dykstra, 2015] Tom Dykstra, R. A. (2015). *Getting Started with Entity Framework 6 Code First using MVC 5*. Microsoft. http://www.asp.net/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application.