Informatika Fakultatea

Informatika **I**ngeniaritzako **G**radua

■ **G**radu **A**maierako **P**roiektua ■

Software Ingeniaritza

StudentGrades: Ebaluazio jarraituko irakasgai bateko notak ikusteko eta kudeatzeko aplikazioa

---

Mikel Aranburu

2016 - uztaila

Laburpena

Ingurune akademikoetan, ebaluazio jarraituaren erabilera gero eta zabalduagoa dago gaur egun. Ebaluazio jarraituak ikasleei irakasgaietan parte-hartze handiagoa eskatzen dio, dena azken azterketa edo proiektu batean jokatu ordez notak pixkanaka lortzearen abantailaren truke.

Ikasle eta irakasleei nota hauen kudeaketa eta batez ere bisualizazio aukera interesgarria eskaintzeko asmoarekin sortu da proposatzen dugun web aplikazio hau.

Aplikazioa garatzeko Ruby on Rails, Google Charts eta Bootstrap bezalako puntako teknologiak erabili ditugu, web aplikazio lokalizatu, erabilgarri eta *responsive* bat lortuz. Azken produktuaren kalitatea ziurtatzea helburu nagusietako bat izan da. Horretarako, gaur egungo garapen eta kudeaketarako metodologia eta teknikak erabili dira. Probek bideratuko garapena (TDD) eraman da aurrera Ruby on Railsen, eta proiektuan zehar erabiltzaileekin probak egin dira. Kudeaketari dagokionez, pertsona bakarreko talde batentzako moldatutako Scrum metodolgia arina erabili da, hobekuntza etengabeko eredua jarrituz.

Amaitutako aplikazioa https://student-grades.herokuapp.com helbidean dago eskuragarri, ondorengo erabiltzaileak erabil daitezkeelarik probak egiteko:

- Ikaslea:

  jonz@ikasle.ehu.eus

  pasahitza: 12345678
- Irakaslea:

  jamesc@irakasle.ehu.eus

  pasahitza: 12345678
- Administratzailea:

  mikelf@irakasle.ehu.eus

  pasahitza: 12345678

Gaien Aurkibidea

Kode-zati, Irudi eta Taulen zerrenda

KODE-ZATIAK

IRUDIAK

TAULAK

1

Sarrera

Atal honetan gure aplikazioa zein testuingurutan kokatzen den azalduko dugu. Ondoren, dokumentu honetan aipatuko diren hainbat kontzeptu argituko ditugu, eta gurearen antzeko funtzionalitatea duten aplikazio batzuk aipatuko ditugu. Azkenik, gure proposamena azalduko dugu, eta dokumentu honen antolamendua zein izango den zehaztu.

1.1. TESTUINGURUA

Proiektu honetan garatutako StudentGrades aplikazioaren funtzioa irakasle eta ikasleek beraien irakasgaietako notak ikustea eta kudeatzea da. Beraz, aplikazio honen testuingurua unibertsitate edo antzeko erakunde bat izango da. Ikasleek beraien zereginen notak eta bereziki noten joera ezagutu ahalko dute, gaur egun mugikorren bidezko web nabigazioaren garrantzia kontuan hartuz (aplikazioa mugikorrean ikusteko moldatuta egongo da). Joera hori kalkulatzerakoan kontuan hartuko da ebaluazio formatiboaren eta akumulatiboaren arteko ezberdintasuna, hau da, orain arteko notak irakasgaiko joeran eragina izango duen ala ez.

1.2. DEFINIZIOAK

1.2.1. Zer dira ebaluazio akumulatiboa eta ebaluazio formatiboa?

Ebaluazio **akumulatiboaren** filosofian, ikaskuntza helburu jakin bat betetzeko prozesua bezala ulertzen da. Beraz, une jakin batean ikaskuntzaren joera kalkulatzeko orduan, ordura arte betetako zereginen noten batezbestekoa hartuko dugu kontuan. Kasu honetan, guk proposatutako formula ondokoa da:

$$J_n = (N_1*P_1 + N_2*P_2 + ... + N_n*P_n) / (P_1 + P_2 + ... + P_n)$$

J_i = Notaren Joera Igarren Zereginean
N_i = Igarren Zereginaren Nota
P_i = Igarren Zereginaren Pisua

Ebaluazio **formatiboaren** kasuan berriz, ikaskuntza prozesu bezala ulertzen da. Beraz, une jakin batean ikaskuntzaren joera kalkulatzeko orduan, azken zereginaren nota eta ordura arte egindako beste zereginen joera kontuan hartuko ditugu, ordura arte egindako beste zereginen joerari pisu jakin bat emanez. Kasu honetan, guk proposatutako formula ondokoa da:

$$J_n = (100 - r) * (N_n) + r * (N_1*P_1 + N_2*P_2 + ... + N_{n-1}*P_{n-1}) / (P_1 + P_2 + ... + P_{n-1})$$

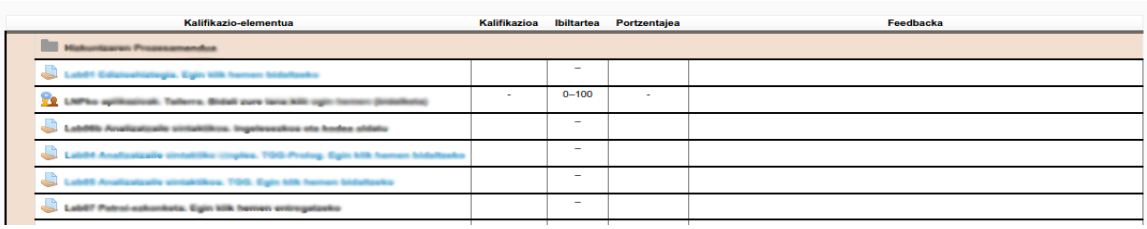
J_i = Notaren Joera Igarren Zereginean
r = Aurreko Joerari emango zaion Pisua
N_i = Igarren Zereginaren Nota
P_i = Igarren Zereginaren Pisua

1.2.2. Aplikazioaren hizkuntza

Aplikazioaren kodea beste hizkuntza bateko norbaiti erakutsi behar izateko aukera kontuan hartuz, aplikazioa ingelesez programatu dut, hau da, klaseen, metodoen, aldagaien, proben... izenak ingelesez idatzi ditut. Beraz, adibide bezala, irakasgaiei dagokien klasea Course deitzen da, eta dokumentu honetan bai irakasgai eta bai Course erabiliko ditut, egoerak eskatzen duenaren arabera.

1.3. AURREKARIAK

Mota honetako aplikazioei dagokienez (*LMS, Learning Management System*, bezala ezagunak), software libreari dagokionez proposamen ezagunena eta zabalduena Moodle da. Hala ere, Moodle-ek ez du datuak modu grafikoan erraz ikusteko aukerarik ematen.



1.1. Irudia: eGela (moodle)

Gainera, gure aplikazioak ebaluazio mota ezberdinak aukeratzeko aukera emango du (Moodle-ek ere ematen du aukera bertsio berrietan kalkulu-orri moduan antzeko zerbait, baina ez da hain erraz eta intuitiboa).

LMS ez-libreei dagokienez, Blackboard Learn da merkatuko nagusia, baina prezio altua eta konplexutasuna kontuan hartuta, beste behar batzuk betetzen dituen proposamen bat dela esango nuke.

1.4. PROPOSAMENA

Proiektu honetan proposatzen den aplikazioa Moodle baino sinpleagoa izango da, eta datuen erakusketa bisualean oinarrituko da. Mugikorrerako bertsioari garrantzia emango zaio, *Responsive* filosofia jarraituz. Gainera, EHU bezalako unibertsitate eleaniztun baten testuinguruan kokatzen denez, aplikazioaren lokalizazio egokia egingo da, ingelesez, gazteleraz eta euskaraz.

1.5. DOKUMENTUAREN ANTOLAMENDUA

Dokumentu hau ondoko ataletan banatuko da:

- **1. Atala: Sarrera**
  Atal honetan proiektuaren testuingurua, definizioak, aurrekariak eta gure proposamena aurkeztuko dira. Azkenik, dokumentuaren antolamendua argituko da.

- **2. Atala: Proiektuaren helburuak**
  Atal honetan proiektuarekin zer lortu nahi dugun azalduko da. Lehenik eta behin, proiektuaren irismena zehaztuko da, gero kalitatearen inguruko erabakiak zehaztuko dira eta azkenik arriskuen analisia eta bideragarritasun-analisia egingo dira.

- **3. Atala: Proiektuaren planifikazioa**
  Atal honetan proiektuaren hasierako planifikazioa azalduko da, hau da, Scrum iterazioak martxan jarri aurretik zehaztutako zereginak.

- **4. Atala: Teknologia eta tresnak**
  Atal honetan aplikazioa garatzeko erabili ditugun teknologiak (frameworkak, lengoaiak, liburutegiak, APIak...) eta tresnak (nabigatzailea, testu-editorea...) zein izan diren eta zergatik erabili ditugun azalduko da.

- **5. Atala: Arkitektura**
  Atal honetan aplikazioaren arkitektura azalduko da, bai aplikazioari berari dagokionez eta bai datubaseari dagokionez.

- **6. Atala: Metodologiak**
  Atal honetan proiektua aurrera eramateko erabili ditugun metodologiak azalduko ditugu

- **7. Atala: Probak**
  Atal honetan aplikazioaren funtzionamendua egokia dela ziurtatzeko idatzitako software probak azalduko ditugu, proiektuaren zein betekizunekin erlazionatzen diren erakutsiz, taula moduan.

- **8. Atala: Iterazioak**
  Atal honetan proiektua osatu duten iterazio bakoitzean zer egin dugun azalduko dugu, zereginen taularen aldaketak erakutsiz.

- **9. Atala: Ondoriak**
  Atal honetan proiektua amaitu ondoren atera ditugun ondorioak azalduko dira.

- **A Eranskina: Interfazeen Prototipoak**
  Eranskin honetan hasierako diseinua islatzen duten interfazeen prototipoak erkutsiko dira.

- **B Eranskina: Erabiltzaileekin egindako proben emaitzak**
  Eranskin honetan erabiltzaileekin egindako probetan jasotako feedback-a zein izan zen azalduko dugu.

- **C Eranskina: UML Diagramak**
  Eranskin honetan hasierako diseinua islatzen duten interfazeen prototipoak erkutsiko dira.

2

Proiektuaren helburuak

Atal honetan gure proiektuaren helburuak azalduko ditugu, hau da, zer funtzionalitate bete nahi genituen eta bete ditugun. Gainera, proiektuaren kalitatearen inguruan hitz egingo dugu, arriskuen analisia egingo dugu eta bideragarritasun-analisiaren emaitzak aurkeztuko ditugu.

2.1. IRISMENA

Proiektuan zehar erabili dugun metodologia jarraituz, iterazio bakoitzean proiektuaren betekizunak moldatuz joan dira, uneko egoera kontuan hartuta (erabiltzaileekin egindako probetan jasotako *feedback*-a, proposatutako hobekuntzak...). Oro har, hasierako betekizunen zerrenda zabaldu egin da, funtzionalitateak inplementatu ahala hobekuntzak eta funtzionalitate berriak gehituz.

Betekizunak bi multzotan banatu dira: **funtzionalak** eta **ez-funtzionalak**. Betekizun funtzionalek softwarearen funtzio jakin bat deskribatzen dute, hala nola erabiltzailearen profila ikusteko aukera, eta ez-funtzionalek berriz softwarearen ezaugarri bat, hala nola erabiltzeko erraztasuna edo segurtasuna.

Hona hemen **lehen iterazioan** definitu genuen betekizun funtzionalen multzoa:

- Irakasleak:
    - Irakasten dituen irakasgaien zerrenda ikusi.
    - Irakaslearen profila ikusi.
    - Irakasgai batean ikasleek izan dituzten notak ikusi, taula bezala.
    - Irakasgai batean ikasleek izan dituzten notak eta joerak ikusi, grafiko bezala, hainbat ikasle eta klaseko batezbestekoa konparatzeko aukerarekin.
    - Irakasgai batean ikasleek izan dituzten notak gehitu edo aldatu.
    - Irakasgai jakin batean zeregin berri bat sortu (adibidez, azterketa bat).
- Ikasleak:
    - Ikasten dituen irakasgaien zerrenda ikusi.
    - Ikaslearen profila ikusi.
    - Irakasgai batean izan dituen notak ikusi, taula bezala.
    - Irakasgai batean izan dituen notak eta joera ikusi, grafiko bezala.
- Denak:
    - Sisteman login egin.
    - Hasierako orrialdea ikusi.

Proiektuaren **amaieran**, ondorengo betekizun funtzionalak bete ditugu:

- Irakasleak:
    - **Irakasten dituen irakasgaien zerrenda ikusi**: Nabigazio barratik irakasgaien zerrendara joan daiteke, eta bertan taula bat ikusiko da, errenkada bakoitzean irakasgairen orrialderako esteka, hizkuntza, hasiera- eta amaiera-datak eta irakasgaia konfiguratzeko esteka.
    - **Irakaslearen profila ikusi eta mezuen hizkuntza aldatu**: Nabigazio barratik irakasleak bere profila ikus dezake. Bertan, bere inguruko informazioa ikusiko du, eta mezuak jasotzeko hizkuntza aukeratu ahalko du.
    - **Irakasgai batean ikasleek izan dituzten notak ikusi, taula bezala:** Irakasgai baten orrialdean, taula bat erakutsiko da, zeregin bakoitzean ikasle bakoitzak izan duen nota eta (zeregineko) batezbesteko nota erakutsiz. Zutabeak birordenatzeko aukera emango da. Gainera, zeregina notaduna, notagabea edo etorkizunekoa den zehaztuko da.
    - **Irakasgai batean ikasleek izan dituzten notak eta joerak ikusi, grafiko bezala, hainbat ikasle eta klaseko batezbestekoa konparatzeko aukerarekin**: Irakasgai baten orrialdean, ikasleen notak erakusten dituzten grafikoak erakutsiko dira: hasieran, ikasleen batezbesteko notak erakusten dituen barra-grafikoa ikusiko da. Ikasle jakin bati dagokion barran klik eginda edota taulan ikasleari dagokion errenkadako checkbox-a hautatuta, ikasle horrek zereginetan izan duen notak agertuko dira barra-grafiko bezala, gainetik klaseko batezbestekoa adierazten duen lerro batekin. Taulan ikasle gehiago hautatuz gero, euren notei dagozkien barrak ere erakutsiko dira, kolore ezberdinetan. Nota/Joera botoietan Joera hautatuz gero, grafikoa joera-grafikoa bihurtuko da, irakasgaiaren mota kontuan hartuta kalkulatuko dena (akumulatiboa/formatiboa). Joera-grafikoa azalera-grafikoa da, eta noten grafikoan bezala, hainbat ikasleren datuak konpara daitezke. Ikaslerik hautatzen ez bada, klasearen joera erakutsiko da.
    - **Irakasgai batean ikasleek izan dituzten notak gehitu edo aldatu**: Noten taulan, irakasleak nota batean klik eginda nota sartzeko/aldatzeko popup bat zabalduko da.
    - **Irakasgai bat konfiguratu: joera mota, tasa, hasiera eta amaiera datak aldatu**: Irakasgai bakoitzak joera mota, tasa eta hasiera/amaiera datak aldatzeko formulario bat izango du, irakasgaiaren orrialdetik edo zerrendatik eskura daitekeena.
    - **Irakasgai jakin batean zeregin berri bat sortu (adibidez, azterketa bat)**: Irakasgai beten orrialdean zeregin bat gehitzeko botoi bat egongo da, dagokion formularioa zabalduko duena.

- **Irakasgai bateko zereginen zerrenda ikusi**: Irakasgai baten orrialdean zereginen zerrenda ikusteko botoi bat egongo da, irakasgaiko zereginen zerrenda erakutsiko duena, zeregin bakoitzaren izena, mota, lekua, data eta pisuarekin.

- **Zeregin bat ezabatu**: Irakasgaiko zereginen zerrendan zeregin bakoitza ezabatzeko esteka erakutsiko da.

- **Zeregin bat editatu**: Irakasgaiko zereginen zerrendan zeregin bakoitza editatzeko formulariora daraman esteka erakutsiko da.

- **Irakasgai bateko ikasleen estatistikak ikusi, irakasgaitik kentzeko aukerarekin**: Irakasgaiko orrialdean, ikasleak kudeatzeko botoiari emanda taula bat erakutsiko da, errenkada bakoitzean ikaslearen izena, orain arteko nota, ebaluatutako ehunekoa eta notaren joera erakutsiz. Gainera, ikasle hori irakasgaitik kentzeko aukera emango da.

- Ikasleak:

  - **Ikasten dituen irakasgaien zerrenda ikusi**: Nabigazio barratik irakasgaien zerrendara joan daiteke, eta bertan taula bat ikusiko da, errenkada bakoitzean irakasgairen orrialderako esteka, hizkuntza eta hasiera- eta amaiera-datak.

  - **Ikaslearen profila ikusi eta mezuen hizkuntza aldatu**: Nabigazio barratik ikasleak bere profila ikus dezake. Bertan, bere inguruko informazioa ikusiko du, eta mezuak jasotzeko hizkuntza aukeratu ahalko du.

  - **Irakasgai batean izan dituen notak ikusi, taula bezala**: Irakasgai baten orrialdean, taula bat erakutsiko da, zeregin bakoitzaren izena, mota, pisua eta ikasleak izan duen nota erakutsiz. Zutabeak birordenatzeko aukera emango da. Gainera, zeregina notaduna, notagabea edo etorkizunekoa den zehaztuko da.

  - **Irakasgai batean izan dituen notak eta joera ikusi, grafiko bezala**: Irakasgai baten orrialdean, ikaslearen notak erakusten dituzten grafikoak erakutsiko dira: hasieran, ikasleak zereginetan izan ditu notak agertuko dira barra-grafiko bezala, gainetik klaseko batezbestekoa adierazten duen lerro batekin. Nota/Joera botoietan Joera hautatuz gero, grafikoa joera-grafikoa bihurtuko da, irakasgaiaren mota kontuan hartuta kalkulatuko dena (akumulatiboa/formatiboa). Joera-grafikoa azalera-grafikoa da, eta ikaslearen eta klasearen noten joera erakutsiko ditu.

  - **Etorkizunean dituen zereginak ikusi**: Nabigazio barratik ikasleak etorkizunean dituen zereginak ikus ditzake. Zereginen

taulako errenkada bakoitzean zereginaren izena, irakasgairako esteka eta zereginaren data erakutsiko dira.

- **Mezu bat jaso nota berri bat duen bakoitzean**: Irakasleak nota gabeko zeregin bati lehen aldiz nota ezartzen dionean, nota horri dagokion ikasleak bere posta elektronikoan abisua jasoko du, bere profilean hautatu duen hizkuntzan.

- Administratzaileak:

  - **Irakasgai guztien zerrenda ikusi**: Nabigazio barratik irakasgaien zerrendara joan daiteke, eta bertan taula bat ikusiko da, errenkada bakoitzean irakasgairen orrialderako esteka, hizkuntza, hasiera- eta amaiera-datak eta irakasgaia ezabatzeko aukera agertuko dira.

  - **Irakasgai berri bat sortu**: Irakasgaien zerrendaren orrialdean, irakasgai berri bat gehitzeko botoia egongo da. Botoiari emanda, irakasgai berria sortzeko formularioa erakutsiko da, dagozkion eremuekin.

  - **Irakasgai bat ezabatu**: Irakasgaien zerrendako errenkada bakoitzean irakasgaia ezabatzeko aukera erakutsiko da.

  - **Irakasgai bateko irakasle / ikasleak gehitu / ezabatu:** Irakasgai baten orrialdean, irakasgaiko irakasleen eta ikasleen taulak erakutsiko dira. Taula bakoitzaren gainean ikasle edo irakasle berri bat gehitzeko combobox eta botoia egongo dira.
    Taulako errenkada bakoitzean dagokion ikasle edo irakaslea ezabatzeko aukera emango da.

- Denak:

  - **Sisteman login egin**: Login egin gabeko erabiltzaileak nabigazio barratik login formularioa atzi dezake, pasahitza gogoratzeko eta berreskuratzeko aukerekin.

  - **Hasierako orrialdea ikusi**: Gunean lehen aldiz sartzean hasierako orrialdea erakutsiko da, ongietorria emanez eta aplikazioaren funtzionalitate batzuen irudiak erakutsiz. Erabiltzaileak login egiten duenean ongietorria eta egin ditzakeen gauzen zerrenda erakutsiko dira.

Ikus dezakegunez, erabiltzaile mota berri bat sortzea erabaki dugu garapenean aurrera joan ahala (Administratzailea), dagozkion betekizunekin. Gainera, Irakasle eta Ikasleek eskura dituzten funtzionalitateak zabaldu ditugu, egoera erreal batean erabil daitekeen aplikazio bat sortuz. Erabilpen kasu guzti hauek UML formatuan ikus daitzeke C Eranskineko bigarren atalean.

Betekizun ez-funtzionalei dagokienez, ondoko hauek hartu behar izan ditugu kontutan:

- **Erabilgarritasuna**: Aplikazioa erabilerraza eta intuitiboa izango da (interfazeen prototipoekin eta beranduago benetako interfazeekin probak egingo dira benetako ikasle eta irakasleekin hobekuntzak egiteko). Aplikazioa mugikorrean modu egokian ikusteko moldatuta egongo da.

- **Lokalizazioa**: Aplikazioaren hizkuntza bezala ingelesa, euskara eta gaztelera ezartzeko aukera izango da, nabigazio-barratik. Kontuan hartu behar da irakasgai baten orrialdera sartzean automatikoki irakasgai horren hizkuntza hautatuko dela eta orrialdean lehen aldiz sartzean erabiltzailearen mezuen hizkuntza hautatuko dela.

- **Segurtasuna eta pribatutasuna**: Mota honetako aplikazio batek informazio sentikorra erabiltzen duenez (pasahitzak, notak) informazio hau modu seguruan bakarrik eskura daitekeela ziurtatu beharko da.

2.2. KALITATEA

Kalitatezko produktu bat sortu nahi dugunez, gutxienez lortu nahi dugun kalitatea (kalitate minimoa) eta gure helburua den kalitatea (helburuko kalitatea) definituko ditugu. Kalitate maila hau lortzen dela ziurtatzeko, kalitatearen kontrola nola egingo den ere zehaztuko da.

2.2.1. Kalitate minimoa

- **Kudeaketa:** Scrum metodologia moldatua erabiliko da (taldekide bakarrekoa). Bi astetik behin iterazio amaierako bilera egingo da proiektuko irakaslearekin, egindako lana kontuan hartuta hurrengo iterazioan zer egin erabakitzeko.

- **Komunikazioa:** Proiektuko irakaslearekin email bidezko komunikazioa mantenduko da, iterazio amaierako bilerez gain.

- **Produktua:** Hasierako planifikazioan definitutako atazak aurrera eramango dira, produktuaren oinarrizko bertsioa lortuz.

- **Memoria:** EHUk eskaintzen duen eredua jarraituko da.

2.2.2. Helburuko kalitatea

- **Kudeaketa:** Scrum metodologia moldatua erabiliko da (taldekide bakarrekoa). Astean behin (kasuren batean bi asteko tartea utz liteke,

beste irakasgaietan karga handia izan delako lan gutxiago egin delako) iterazio amaierako bilerak egingo ditugu proiektuko irakaslearekin, egindako lanari oniritzia eman eta hurrengo iterazioko atazak aukeratuz eta lehentasunak zehaztuz (zereginen taula erabiliz). Bilera bakoitzaren ondoren akta bat sortuko da, bileran zer eztabaidatu den jasotzeko.

- **Komunikazioa:** Proiektuko irakaslearekin email bidezko komunikazioa mantenduko da, iterazio amaierako bilerez gain. Emaila egunero begiratu beharko da, azken orduko aldaketen berri izateko.

- **Produktua:** Hasierako planifikazioko atazak betetzeaz gain, garapenean zehar agertzen diren funtzionalitate berriak inplementatuz joango gara. TDD metodologia jarraituz, kodea idatzi ahala probak inplementatuko ditugu, kodeak espero dugun funtzionamentua duela ziurtatuz. Gainera, ikasle eta irakasleekin probak egingo ditugu produktua hasi aurretik (interfazeen prototipoak ulergarriak direla ziurtatzeko) eta ondoren (produktua erabilgarria dela ziurtatzeko). Azkenik, dokumentazio bezala UML diagramak egingo dira.

- **Memoria:** EHUk eskaintzen duen eredua jarraituko da.

2.2.3. Kalitatearen kontrola

- **Kudeaketa:** Iterazio amaierako bilerak modu egokian egiten diren bitartean, kudeaketaren kalitatea ziurtatuta egongo da.

- **Komunikazioa:** Komunikazio arazoren bat sortzen bada hurrengo bileran eztabaidatuko da.

- **Produktua:** Irakasle eta ikasleekin probak egingo dira, garapena hasi aurretik interfazeen prototipoak erabiliz eta produktua garatutakoan benetako produktuarekin. Jasotako feedbacka produktua hobetzeko erabiliko da.

- **Memoria:** Iterazio amaierako bileretan berrikusteaz gain, memoria bidali baino lehen irakasleari erakutsiko zaio, azken uneko zuzenketak edo hobekuntzak egiteko.

2.3. ARRISKUEN ANALISIA

Proiektu honi aurre egiterakoan ikusten ditudan arrisku handienak tamaina honetako proiektu bat aurrera eramaten esperientziarik ez izatea eta erabiliko dudan teknologiekin esperientzia txikia izatea dira. Beste hainbat arrisku ere kontutan hartu behar izan dira, nahiz eta arrisku-maila

txikiagoa izan. Taulan erakusten diren arriskuetan ondorengo ezaugarriak zehaztuko dira:

- **Mota**, hau da, arriskuaren sailkapena.
- **Arrazoia**, hau da, arriskuaren deskribapena.
- **Arrisku-maila** (Altua/Ertaina/Baxua), hau da, zein den arriskua gertatzeko probabilitatea.
- **Eragina** (Altua/Ertaina/Baxua), hau da, arriskuak proiektuan zenbateraino eragingo duen.
- **Prebentzio-plana**, hau da, zer egingo den arriskua ekiditeko
- **Kontingentzia-plana**, hau da, zer egingo den arriskua gertatuz gero.

Mota	Arrazoia	Arrisku-maila	Eragina	Prebentzio-plana	Kontingentzia-plana
Planifikazio okerra	Esperientzia txikia proiektu handiak aurrera eramaten	Altua	Ertaina	Gaiaren inguruko formazioa egin	Ordu gehiago sartu edo helburuak aldatu
	Esperientzia txikia Ruby on Rails erabiltzen	Ertaina	Ertaina	Ruby eta Rails-i buruzko formazioa egin	Ordu gehiago sartu formazioan
	Esperientzia txikia proiektuen planifikazioan	Ertaina	Ertaina	Metodologia arinarekin planifikazioa moldatuz joan	Ordu gehiago sartu edo helburuak aldatu
Gaixotasuna edo ezbeharra	Baliteke familiako kideren batek osasun-arazoak edo beste ezbeharren bat izatea, eta horrek proiektuari emandako dedikazioa arriskuan jar dezake	Ertaina	Ertaina	-	Ordu gehiago sartu

	Baliteke neu gaixotzea	Baxua	Altua	-	Ordu gehiago sartu
Beste irakasgaietan egin beharreko lana	Apirilaren erdialdera arte, beste irakasgaietako lanak eta azterketak izango ditudanez, kontuan hartu behar da proiektuan lan egiteko denbora gutxiago izango dudala	Ertaina	Ertaina	Beste irakasgaietako lana ahalik eta hoberen planifikatu	Hasieran lan karga gutxiago izan
Informazio-galera	Ordenagailuak arazo teknikoren bat izan dezake	Baxua	Altua	Bertsio-kontrola erabili (Git + Github)	-
Estimatutako datan proiektua oraindik ez amaitzea	Baliteke arrazoi ezberdinak direla medio amaitzea espero den datan oraindik lan gehiago egin behar izatea	Ertaina	Altua	Planifikazioa moldatu	Proiektua hurrengo deialdian aurkeztu beharko da, eta ordu gehiago sartu

2.1. Taula: Arriskuen analisia

2.4. BIDERAGARRITASUN ANALISIA

Arriskuen analisian identifikatutako arriskuak aztertu ondoren, ondoriozta dezakegu proiektua aurrera eramateko baliabideak baditugula. Nahiz eta Ruby on Rails erabiltzen esperientzia gutxi izan, formakuntzarako planifikatutako denborarekin nahikoa izango dela esango nuke.

3

Proiektuaren planifikazioa

Atal honetan proiektuaren planifikazioa aurkeztuko dugu, hau da, Scrum iterazioen hasieran nola definitu genituen bete beharreko zereginak.

3.1. HASIERAKO ZEREGINAK

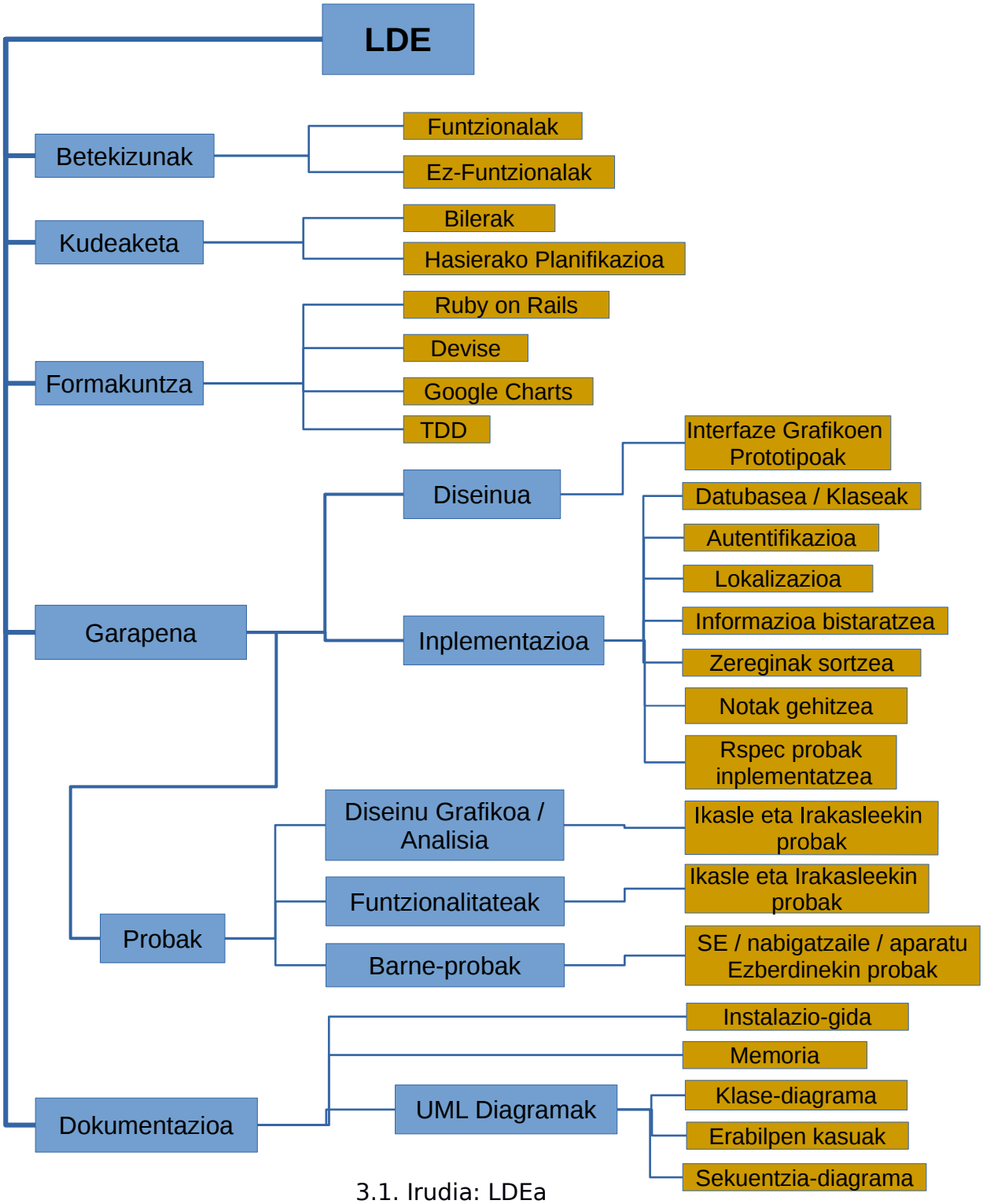
Faseak			Zereginak		Estimazioa
Betekizunak			Funtzionalak		3
			Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna...)		2
Kudeaketa			Bilerak		15
			Hasierako planifikazioa		8
Formakuntza			Ruby on Rails		80
			Devise		10
			Google Charts		8
			TDD (Ruby)		6
Garapena	Diseinua		Interfaze Grafikoen Prototipoak		4
	Inplementazioa		Datu basea / Klaseak		10
			Autentifikazioa (Devise, Rolify, CanCanCan)		8
			Lokalizazioa		4
			Informazioa bistaratzea		20
			Zereginak sortzea		2
			Notak gehitzea		4
			Rspec probak inplementatzea		20
			Beste funtzionalitate batzuk		20
	Probak	Diseinu grafikoa eta analisia	Ikasle eta irakasleekin probak		10
		Funtzionalitateak	Ikasle eta irakasleekin probak		4
		Barne-probak	Sistema eragile eta nabigatzaile eta aparatu ezberdinekin probak egin		6
Dokumentazioa			Instalazio-gida		1
			Memoria		40
			UML Diagramak	Klase-diagrama	5
				Erabilpen kasuak	5
				Sekuentzia-diagrama	5
				GUZTIRA:	300

3.1. Taula: Hasierako Planifikazioa

Proiektuaren hasieran, zereginen *backlog* bat osatu genuen, Scrum metodologia jarraituz. Taula bezala antolatuta, zeregin bakoitzaren fasea eta estimazioa (ordutan) zehaztu genituen. Backlog hau, iterazioak aurrera joan ahala aldatuz joan da: adibidez, Rolify eta CanCanCan ez erabiltzeko erabakia hartu zen, eta Beste Funtzionalitateentzat gordetako orduak definituz joan ziren funtzionalitate berriak agertu ahala.

3.2. LDEa

Hasierako zeregin hauen Lanaren Deskonposaketa Eskema 2. Taulan aurkeztutako zereginak modu grafikoan erakusteko modua da.



3.1. Irudia: LDEa

4

Teknologiak eta Tresnak

Atal honetan proiektuan erabili ditugun teknologiak eta tresnak aurkeztuko ditugu, zertarako eta zergatik hautatu ditugun azalduz.

4.1. TEKNOLOGIAK

Proiektu honetan, puntako teknologiak erabiltzen saiatu naiz, ahalik eta emaitza onenak lortzeko ahalik eta modu sinple eta ulergarrienean. Tresnei dagokienez, Railsen garapen filosofiak ez du IDE konplexu bat erabiltzea eskatzen, beraz, nahiko tresna sinpleak erabili ditut.

4.1.1. Ruby eta Ruby on Rails

Motibazioa

Ruby on Railsekin izan nuen lehen harremana SGTA irakasgaian izan zen, klasean eman ez genuen teknologiaren bat ikasteko baldintza zuen zeregin batean. Emandako aukeren artean Rails hautatu nuen, Interneten berari buruz zerbait irakurria nuelako. Zeregina txikia zen, ez nuen asko sakondu baina Rails oso interesgarria iruditu zitzaidan; esperientzia horren ondoren ez nuen berriz Rails erabili. Proiektua egiterako orduan argi nuen web aplikazio bat egin nahi nuela, eta beraz framework bat hautatzeko beharra izan nuen. ASP.NET eta PHP ezagutzen nituen, baina ez nuen nahi bezain esperientzia ona izan eurekin, eta ondorioz Rails erabiltzea erabaki nuen, formakuntzan nahikoa denbora eman beharko nuela jakin arren.

Ruby

Ruby helburu orokorreko lengoaia dinamiko, erreflexibo eta objektu-orientatua da. Dinamikotasunari dagokionez, esan nahi duena da Ruby-k lengoaia estatikoek (adibidez, C-k) konpilazioan egiten dituzten hainbat ekintza exekuzioan aurrera eraman ditzakeela eta lengoaia erreflexiboa izateak esan nahi du Ruby programa batek bere burua eralda dezakeela exekuzioan. Adibidez, `define_method()` erabiliz objektu baten instantzia jakin batean metodo berri bat defini dezakegu exekuzio-garaian. Objektu-orientazioari dagokionez, Ruby OO lengoaia "purua" da, hau da, Ruby-n kontrol egitura ez den den-dena da objektua da. Adibide bezala, ondoko kode zatia:

```
puts 1.class
```

4.1. Kodea: 1-en klasea

Kode hau exekutatzean `Fixnum` jasoko dugu, Ruby-n ez baitago primitiborik.

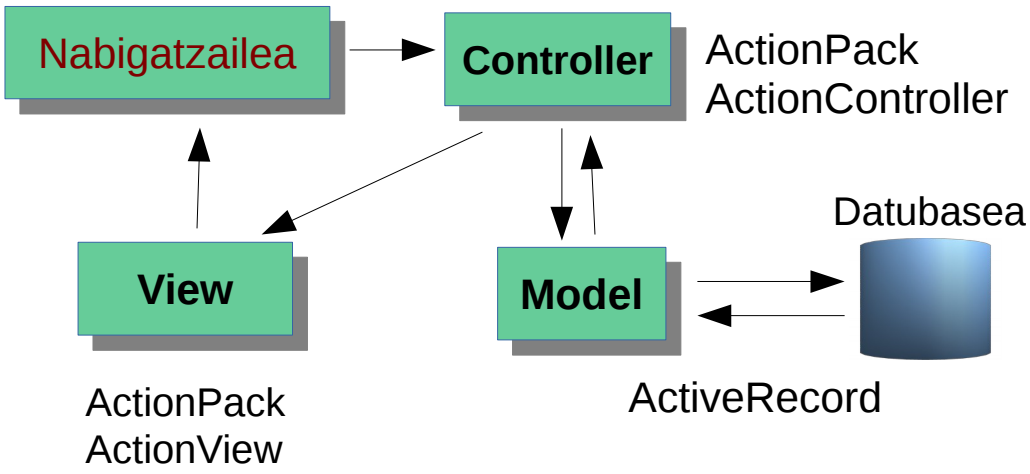
Ruby on Rails

Ruby on Rails (Rails bakarrik bezala ere ezaguna) web aplikazioen garapenerako framework bat da, Ruby lengoaian idatzia. Rails-ek, web garapenerako framework garaikide garrantzitsuenetako askok bezala, MVC eredua jarraitzen du, hau da, hiru geruzatan banatzen du aplikazioa, zeregin ezberdinak betetzen dituztelarik:

- **Model**: Datuen manipulazioa eta negozio logika egiten du
- **Controller**: Erabiltzaileak sistemarekin duen elkarrekintza jaso eta Model edota View-ri aginduak ematen dizkio
- **View**: Datuen errepresentazio grafikoa egiten du

Framework-aren egitura ondoko paketeean banatzen da:

- **ActiveRecord**: ORM (object-relational mapper) sistema, datu-basearekiko harremanak maneiatzen dituena.
- **ActiveResource**: REST web zerbitzuak eta negozio-objektuak lotzen dituen sistema.
- **ActionPack**: Web-eskariak maneiatzen dituen sistema, routing-a, Controller-en REST ekintzak eta View-ak erakustea barne hartuz.
- **ActiveSupport**: Rails-ek erabiltzen dituen hainbat utilitate eta liburutegi.
- **ActionMailer**: Emailak bidaltzeko sistema.



4.1. Irudia: MVC eredua eta Rails-en paketeak

Railsen filosofiaren bi ezaugarri nagusiak DRY (Don't Repeat Yourself) eta CoC (Convention over Configuration) printzipioak dira. Lehenak informazioa leku bakar eta konkretuan gordetzea defendatzen du; Railsen kasuan, adibidez, Model baten eremuak ez dira Ruby klasean bertan definitzen, ActiveRecord-ek klasearen izena kontuan hartuta datubaseko zutabeetatik bertatik hartzen ditu. CoC-ri dagokionez, aplikazioetan aspektu ezohikoak bakarrik zehaztu behar izatea defendatzen du, modu honetan garatzaileak ahalik eta erabaki gutxien hartu behar izan ditzan; Rails-en kasuan, adibidez, aplikazioen dirketorio-egitura estandarra da eta Railsek fitxaegiak direktorio jakinetan egotea espero du, konfigurazioa sinplifikatuz eta besteen aplikazioen egitura ulertzea erraztuz.

Rubyren inguruko formakuntza egiteko *Practical Object-Oriented Design in Ruby* eta *Eloquent Ruby* liburuak erabili ditut, eta Railsen inguruko formaziorako *Ruby on Rails Tutorial, Third Edition* erabili dut.

Alternatibak

Gaur egun, Ruby on Rails sareko hirugarren web framework erabiliena dela estimatzen da, ASP.NET eta AngularJS-k bakarrik gainditurik.

Rankings

Framework	Github Score	Stack Overflow Score	Overall Score
ASP.NET		100	100
AngularJS	100	96	98
Ruby on Rails	95	98	96
ASP.NET MVC		93	93
Django	90	93	91

4.1. Taula: hotframeworks.com-en arabera 5 web framework erabilienak

Badira Ruby erabiltzen duten beste hainbat web framework, horien artean ezagunena Sinatra delarik, baina Rails-en dokumentazio eta hedadura askoz handiagoak dira. Web aplikazioak garatzeko MVC motako frameworkei dagokienez, oso eremu zabala da eta alternatiba asko daude; ezagunenenetako batzuk 4.1.Taulan agertzen diren AngularJS (Javascript), ASP.NET (C#) eta Django (Python) dira.

4.1.1.1. Devise

Devise Railserako autentikazio plugin (Railsen, *Gem*) bat bat da. Aplikazioaren login sistema inplementatzeko erabili dut, eta hasieran konfigurazioa zail xamarra izan den arren, martxan jartzea lortu ondoren ez dit arazorik eman.

4.1.1.2. Rspec, Capybara, FactoryGirl

Software probak egitea Railsen filosofiaren parte da. Probak egiteko Rspec erabiltzea erabaki dut, *Rails 4 Test Prescriptions* liburuak gomendaturik (proben inguruko formakuntza liburu horrekin egin dut). Datuak simulatzeko FactoryGirl plugina (*Gem*) erabili dut. Hona hemen Rspec eta FactoryGirl erabiltzen dituen proba baten adibidea:

```
RSpec.describe Teacher, type: :model do
  let!(:teacher){FactoryGirl.create(:teacher)}
  let!(:course){FactoryGirl.create(:analisis)}

  it "checks if it teaches a course" do
    expect(teacher.teaches(course)).to eq(false)
    teacher.courses << course
    expect(teacher.teaches(course)).to eq(true)
  end

  it "gets teacher not teaching course" do
    expect(Teacher.not_teaching(course)).to eq([teacher])
    course.teachers << teacher
    expect(Teacher.not_teaching(course)).to eq([])
```

```
  end
end
```

4.2. Kodea: teacher_spec.rb

Ikus daitekeenez, Rspec probak nahiko irakurgarriak dira lengoaia ezagutu gabe ere. Izan ere, Rspec-en helburua da ezagutza tekniko gutxi dituzten pertsonek uler ditzaketen sintaxia lortzea. Integrazio (*Feature*) probak egiteko Capybara erabili dut, Javascript simulatzeko beharra izan dudanean Selenium driverrarekin. Hona hemen Capybara eta Selenium erabiltzen dituen integrazio proba baten adibidea:

```
scenario "selects student", js: true do
   login_page.visit_page
   login_page.login(teacher)

   visit '/'
   first(:link, 'Courses').click
   click_link(calculo.title)

   page.find('label', text: 'Grades').click
   page.find(:xpath, "//*[@id=\"student-data-
table\"]/tbody/tr[1]/td[1]").click

   expect(page).to have_selector('#google-chart text', text:
student.sortable_name)
end
```

4.3. Kodea: course_data_spec.rb

Proba honek irakaslearen irakasgai baten orrialdeko taulan ikasle bat hautatzen du eta grafikoan ikasle hori agertzen dela ziurtatzen du.
`js: true` -rekin Selenium driverra martxan jartzea aukeratzen dugu, izan ere taulan ikasle bat aukeratu eta grafikoa eguneratzeko Javascript erabiltzen dugu, eta lehenetsitako driverrak ez du Javascript kodea exekutatzen. Seleniumek nabigatzailea irekitzen du eta pausoz pauso definitutako aginduak exekutatzen ditu (login orrialdera joan, login egin, irakasgaien orrialdera sartu etab.) eta azken lerroan grafikoan hautatu dugun ikaslea agertzen dela ziurtatzen dugu. Aurreko adibidearen kasuan bezala, Capybara-ren sintaxia nahiko ulergarria da, eta ezagutza tekniko gutxikoek ulertzeko modukoa izatea du helburu.

4.1.2. CSS

Aplikazioaren *frontend*-ean CSS3 erabili dut, une honetan HTML dokumentuen aurkezpena definitzeko erabiltzen den estandarra.

4.1.2.1. Bootstrap

Bootstrap HTML, CSS eta Javascript liburutegi bat da, web aplikazioen *frontend*-a garatzeko erabiltzen dena. Bootstrap-ek *Responsive* filosofia inplementatzen du, hau da, aparatu ezberdinetatik modu egokian ikus daitezkeen web aplikazioak sortzeko erremintak ematen dizkigu. Gure

aplikazioa bai mahaigaineko ordenagailuetan eta bai mugikorretan ikusiko dela espero dugunez, Bootstrap erabili dugu *frontend*-ean.

4.1.2.2. SASS

SASS CSS-ra itzultzen den lengoaia da, CSS-k ez dituen hainbat funtzio osatzen dituena, hala nola aldagaiak eta habiaratzea:

- Aldagaiak:

```
$info-color: #5bc0de;
$warning-color: #f0ad4e;
$danger-color: #d9534f;

...

.legend-info {
      background: $info-color;
}

.legend-warning {
      background: $warning-color;
}

.legend-danger {
      background: $danger-color;
}
```

4.4. Kodea: courses.scss – aldagaiak

- Selektoreak habiaratzea:

```
#progress-div{
       margin-left: 0px;
       margin-right: 0px;
       & .progress-bar {
           -webkit-transition: none !important;
           transition: none !important;
       }
       & h4#percentage-header{
           margin-top: 0px;
       }
}
```

4.5. Kodea: courses.scss - habiaratzea

4.1.3. Javascript

Javascript-en ezagutza minimo bat banuen arren, errepasatzeko eta gehiago sakontzeko beharra ikusten nuen. *Javascript: The Good Parts* liburuak asko lagundu dit Javascript ondo ulertzen eta praktika onak barneratzen. Javascript web aplikazioetan bezero aldeko funtzionalitateak inplementatzeko erabiltzen da bereziki.

4.1.3.1. jQuery

jQuery Javascript liburutegi ezagun bat da, bezeroaren aldeko *scripting*-a egiteko erabiltzen dena. jQuery-rekin bezeroak orrialdearekin duen elkarrekintza inplementatu dut (checkbox bat aukeratzean zer gertatzen den, *tooltip*-ak, orrialdearen *layout*-a aldatzea...).

4.1.3.2. Paloma

Paloma Railserako plugin (*Gem*) bat da, Railsen Controller-en eta Javascriptez inplementatutako Controller-en arteko lotura egiten duena. Adibidez, tasks_controller.rb fitxategiko TasksController-eko *new* ekintza tasks.js fitxategiko TasksController-eko *new* ekintzarekin lotzen da kasu honetan:

```
class TasksController < ApplicationController
  ...
  def new
    @course = Course.friendly.find(params[:course_id])
    @task = Task.new
    js :starts => @course.starts, :ends => @course.ends
  end
```

4.6. Kodea: tasks_controller.rb

```
var TasksController = Paloma.controller('Tasks', {
  new: function(){
    $('.datetimepicker').datetimepicker({ format: 'YYYY-MM-DD' });
        $
('.datetimepicker').data("DateTimePicker").minDate(this.params.starts)
;
        $
('.datetimepicker').data("DateTimePicker").maxDate(this.params.ends);
  },
...
```

4.7. Kodea: paloma/tasks.js

Ikus dezakegunez, Palomak automatikoki Railseko Controller-aren ekintza exekutatzean Javascript Controller-ean izen bereko ekintza exekutatu behar duela detektatzen du. Gainera, Railseko Controller-ak `starts` eta `ends` parametroak pasatzen dizkio, besteak `this.params` aldagaitik jasotzen dituelarik. Ondorioz, Task berri bat sortzeko formularioan, Task horren data aukeratzeko DateTimePicker-ean Task-ari dagokion Course-aren hasieraren eta bukaeraren arteko datak bakarrik izango dira aukeragai.

4.2. TRESNAK

4.2.1. Sublime Text

Kodea idazteko Sublime Text 3 erabili dut, nahiko aplikazio sinple eta arina. Lagungarriak izan diren funtzionalitateak proiektuaren direktorio-egitura ikusteko aukera, kode-nabarmentzea eta proiektuko fitxategi guztietan testua bilatzeko aukera izan dira.

4.2.2. Google Chrome

Aplikazioa probatzerakoan (bai eskuzko probetan eta bai integrazio proba batzuetan) Google Chrome nabigatzailea erabili dut. *Inspect* funtzionalitatea gakoa izan da eskuzko probak egiterakoan, Javascript kontsola (*logging*-a egiteko eta aginduak exekutatzeko aukera), DOM zuhaitza (orrialdearen egitura, CSS klaseak etab. ikusteko) eta CSS arauak aldatzeko aukerak eskaitzen baititu. *Inspect* funtzionalitatea ondo ezagutzeak asko errazten du web aplikazio baten garapena.

4.2.3. Terminala

Terminala Rails aplikazio baten garapenean oinarrizko tresna da. Terminaletik Model/View/Controller-ak sor eta ezaba daitezke, datubase-migrazioak sortu, exekutatu eta desegin, zerbitzaria martxan jarri, probak exekutatu...

4.2.4. Git eta GitHub

Proiektuaren bertsio-kontrola egiteko, Git erabili dut, GitHub-en ostalaritzarekin.

4.2.5. Heroku

Heroku doako *hosting* zerbitzu bat da, Rails aplikazioekin oso ondo bat datorrena. Hasierako konfigurazioa egin ondoren (oso ondo dokumentatua hemen), eta GitHub-eko errepositorioarekin lotu ondoren, GitHub-en *commit* egiten dudan bakoitzean aplikazioa Herokun automatikoki eguneratzea lortu dut. Beraz, esfortzurik gabe, edonondik eskuragarri dagoen web aplikazio bat martxan izatea lortu dut Herokuri esker.

5

Arkitektura

Atal honetan gure aplikazioaren arkitektura azalduko dugu, irakasgai eta zereginei dagokien adibide bat jarraituz.

5.1. APLIKAZIOA

Teknologien atalean azaldu bezala, Rails-ek MVC arkitektura jarraitzen du. Beraz, aplikazioaren atal nagusiak Model-ak, View-ak eta Controller-ak izango dira.

Aplikazioaren arkitektura azaltzeko, aplikazioaren ereduaren zati baten funtzionamendua (irakasgaiak, zereginak barne) azalduko dugu. Beraz, kode zatietan azalpen honetatik kanpo geratzen diren zatiak ez dira erakutsiko, eta azalpenetik kanpo geratzen diren zatiak ez dira azalduko. Aplikazioaren klase-diagrama C Eranskineko lehen atalean topa daiteke.

5.1.1. *Routing*-a

Rails-ek REST motako bideak (*route*) erabiltzea hobesten du, baliabideak (adibidez, irakasgaiak) manipulatzeko.

```
resources :courses, only:
[:new, :create, :edit, :update, :index, :show, :destroy] do
  resources :tasks, only:
[:edit, :update, :new, :create, :index, :destroy]
  ...
end
```

5.1. Kodea: routes.rb

5.1. kode zatiak aplikazioaren bideak (*route*) definitzen dituen routes.rb fitxategiaren zati bat erakusten du. Lehenik eta behin Courses (irakasgaiak) baliabidearentzako bideak nahi ditugula definitu dugu, kasu honetan Railsek lehenesten dituen *index, new, create, show, edit, update eta destroy* ekintzei dagozkienak.

HTTP aditza	Path-a	Controller#action	Zer egiten du?
GET	/courses	courses#**index**	Irakasgai zerrenda erakutsi
GET	/courses/new	courses#**new**	Irakasgai berri bat sortzeko HTML orrialdera bidali
POST	/courses	courses#**create**	Irakasgai berri bat sortu
GET	/courses/:id	courses#**show**	Irakasgai jakin bat erakutsi
GET	/courses/:id/edit	courses#**edit**	Irakasgai bat editatzeko HTML orrialdera bidali
PATCH/PUT	/courses/:id	courses#**update**	Irakasgai bat eguneratu
DELETE	/courses/:id	courses#**destroy**	Irakasgai bat ezabatu

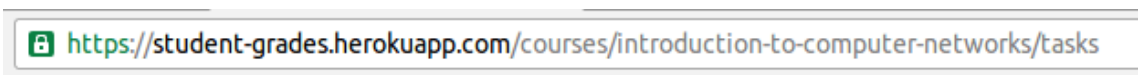
5.1. Taula: Course-ren bideak

5.1. Taulan ikus dezakegu *routes.rb*-n irakasgaiei dagokien lerroak zein bide sortzen dituen, erabiltzen duten HTTP aditza eta zein Controller-Action-ekin lotzen diren. Task-ei (zereginak) dagokien lerroak berriz, ondorengo bideak sortuko ditu:

HTTP aditza	Path-a	Controller#action	Zer egiten du?
GET	/courses/:course_id/tasks	tasks#**index**	Irakasgai baten zereginen erakutsi
GET	/courses/:course_id/tasks/new	tasks#**new**	Irakasgai batean zeregin berri bat sortzeko HTML orrialdera bidali
POST	/courses/:course_id/tasks	tasks#**create**	Irakasgai batean zeregin berri bat sortu
GET	/courses/:course_id/tasks/:id/edit	tasks#**edit**	Irakasgai bateko zeregin bat editatzeko HTML orrialdera bidali
PATCH / PUT	/courses/:course_id/tasks/:id	tasks#**update**	Irakasgai bateko zeregin bat eguneratu
DELETE	/courses/:course_id/tasks/:id	tasks#**destroy**	Irakasgai bateko zeregin bat ezabatu

5.2. Taula: Task-en bideak

routes.rb-n egin dugun habiatzeari esker, Task bakoitza Course jakin batekin lotuta dagoela adierazi dugu, bideetan :course_id-k identifikatzen duen irakasgaiarekin hain zuzen. Adibide, bezala, *Introduction to Computer Networks* irakasgaiako zereginak ikusi nahi ditugunean, ondorengo URL-ra jaongo gara nabigatzailean:



5.1. Irudia: Zereginen URL-a

5.1.2. Controller-ak

Adibide honi dagozkion Controller-ak irakasgaiei dagokien CoursesController eta TasksController dira. Rails-eko Controller-en zeregina orokorrean honela deskriba daiteke: Nabigatzailetik datorren eskariari dagokion ekintza exekutatu, ekintza horretan instantzia aldagaiei balioak eman eta View jakin bat erakutsi. View honen izena, bestelakorik zehaztu ezean, ekintzaren izen bera duen *html.erb* motako fitxategia izango da.

CoursesController-ekin hasiko gara, routes.rb-n sortu ditugun bideei dagozkien ekintzak erakutsiz.

```
def index
    if current_person.is_a? Admin
      redirect_to :action => :index_admin
      return
    end
    @courses = current_person.courses
    ...
end
```

5.2. Kodea: courses_controller.rb – index

Index metodoak `@courses` instantzia aldagaiean (aldagai lokalak ez bezala, klaseko metoko guztietatik eskura daitekeen aldagai mota, @ sinboloarekin hasten dena Ruby-n) une honetako erabiltzailearen irakasgaiak (current_person Devise-k login eginda dagoen erabiltzailea itzultzeko definitzen duen metodo laguntzailea da, gure kasuan Devise erabiltzen duen klasea Person dela kontuan hartuta) gordetzen ditu erabiltzailea Admin motakoa ez bada, hau da, Student motakoa bada ikasten dituen irakasgaiak izango dira eta Teacher motakoa bada irakasten dituenak. Kontrakorik zehaztu ez dugunez, kasu honetan *index.html.erb* View-a erakutsiko da nabigatzailean, hau da, ekintzaren izen bera duena. Admin motakoa bada berriz, Controller honen beste ekintza batera berbideratuko gaitu, azalpen honetatik kanpo geratzen dena.

```
def show
    @course = Course.friendly.find params[:id]
    ...
    if current_person.is_a?(Teacher) &&
current_person.teaches(@course)
        helper = GoogleChartsData.new @course
        js '#setup_teacher',
            :bar_percentage => @course.progress,
            :course_data => @course.data_for_helper,
            :selected_students => selected_students,
            :selected_type => selected_type,
            :expanded => expanded,
            :average_grade_data => helper.average_grade_data,
            :full_grade_data => helper.full_grade_data,
            :average_tendency_data => helper.average_tendency_data,
            :full_tendency_data => helper.full_tendency_data
        render "show_teacher"
    elsif current_person.is_a?(Student) &&
current_person.studies(@course)
```

```
        helper = GoogleChartsData.new @course
        js '#setup_student',
            :bar_percentage => current_person.progress(@course),
            :student_grade_data =>
helper.student_grade_data(current_person),
            :student_tendency_data =>
helper.student_tendency_data(current_person),
            :course_data => @course.data_for_helper,
            :expanded => expanded,
            :selected_type => selected_type
        render "show_student"
    elsif current_person.is_a?(Admin)
        @potential_teachers = Teacher.not_teaching(@course)
        @potential_students = Student.not_studying(@course)
        js '#setup_admin'
        render "show_admin"
    else
        redirect_to courses_path
    end
end
```

5.3. Kodea: courses_controller.rb – show

Show metodoak lehenik eta behin erakutsiko dugun Course-a jasotzen du @course instantzia aldagaiean (`friendly` metodoa nabigatzaileko bideetan id-zenbakiak erakutsi ordez objektuen izenak erakusteko erabiltzen dugun FriendlyId *gem*-arengatik darabilgu). Ondoren, uneko erabiltzailearen motaren arabera ekintza ezberdinak eramango ditu aurrera.

- Irakaslea bada (betiere irakasgaia irakasten duela ziurtatuz) GoogleChartsData klaseko instantzia bat sortuko dugu (Google Charts-en APIak eskatzen duen motako objektuak sortzen dituena) eta Javascript motako Controller-ari (js '#setup_teacher'-ek *setup_teacher* ekintza exekutatzeko zehazten du) behar dituen datuak pasako dizkogu, grafikoak eta aurrerapen-barra sortzeko. Azkenik, *show_teacher.html.erb* View-a erakutsiko dugu nabigatzailean.
- Ikaslea bada, gauza berbera egingo dugu, irakasgaia ikasten duela kontuan hartuz eta grafikoentzako datu ezberdinekin, jakina, eta *show_student.html.erb* View-a erakutsiko dugu.
- Azkenik, Admin motakoa bada, irakasgaia ikasten ez duten ikasleak eta irakasten ez duten irakasleak gordeko ditugu instantzia aldagaietan (Admin-ak irakasgaiko irakasle eta ikasleak kudea ditzake), eta dagokion Javascript Controller-a eta View-a zehaztuko ditugu.

```
def new
    @course = Course.new
end
```

5.4. Kodea: courses_controller.rb - new

New metodoak Course berri bat sortuko du (lehenetsitako balioekin). Instantzia aldagai hau erakutsiko den *new.html.erb* View-eko formularioko balioekin beteko da **Create** ekintzan.

```
def create
    @course = Course.new course_params
    if @course.save
      redirect_to @course
    else
      js '#new'
      render "new"
    end
end
```

5.5. Kodea: courses_controller.rb - create

Create metodoak formulariotik jasoko ditugun parametroekin (course_params metodo laguntzaileak definitzen duen *whitelist*-etik pasa ondoren) Course berri bat sortu eta datubasean gordetzen du, eta Course berri horren **Show** ekintzara bidaltzen gaitu. Gordetzean arazorik balego (kasu ohikoena balidazioren bat ez pasatzea da) berriz ere *new.html.erb* View-a kargatuko luke, beste saiakera bat egiteko aukera emanez.

```
def edit
    @course = Course.friendly.find params[:id]
end
```

5.6. Kodea: courses_controller.rb - edit

Edit metodoak, **New**-k egiten duen antzera, Course bat gordeko du instantzia aldagaiean kasu honetan dagoeneko datubasean dagoen bat. Ondoren, edit.html.erb View-a erakutsiko du, bertako formularioko eremuak datubasetik atera dugun Course honen atributuen balioak aldatzeko erabiltzeko.

```
def update
    @course = Course.friendly.find params[:id]
    if @course.update_attributes(course_params)
      redirect_to @course
    else
      render 'edit'
    end
end
```

5.7. Kodea: courses_controller.rb - update

Update metodoak **Edit** ekintzak kargatutako formularioan egindako aldaketak gordeko ditu dagokion Course-an, **Create**-k **New**-rekin egiten duen antzera. Bi metodo bikote hauen egitura ia-ia berdina da.

```
def destroy
    course = Course.friendly.find(params[:id])
    name = course.title
    course.destroy
    flash[:success] = I18n.t('course_deleted', course: name)
    redirect_to :action => :index_admin
end
```

5.8. Kodea: courses_controller.rb - destroy

Destroy metodoak :id id-ari dagokion Course-a ezabatuko du datubasetik, *flash* notifikazio bat erakutsiko du orrialdean eta *index_admin* ekintzara berbidaliko gaitu (Administratzaileak bakarrik ezaba ditzake Course-ak).

TasksController-en ekintzek gutxigorabehera CoursesController-ekoen egitura jarraitzen dute, beraz ez dute azalpen berezirik behar.

5.1.3. Model-ak

Atal honetan adibide honi dagozkion Course eta Task Model-en edukia azalduko dugu, **Course**-rekin hasiz.

```
class Course < ActiveRecord::Base
       extend FriendlyId
       friendly_id :slug_candidates, use: :slugged

...

def slug_candidates

    [
      :title,
      [:title, :id]
    ]
end

...
```

5.9. Kodea: courses.rb – FriendlyId

5.9. Kode-zatiak klase honetan FriendlyId *gem*-a erabiliko dugula zehazten du. Lehenago aipatu dugun bezala, FriendlyId-rekin aplikazioaren URL-etan id zenbakiak izan ordez objektuen izenak izatea lortuko dugu. Kasu honetan, *slug_candidates* metodoak definitzen du izen hori zein den. Bakarra bada, *title* atributua erabiliko da, eta *title* bereko bi irakasgai badaude *title+id* erabiliko da.

```
...

has_many :teach_courses, dependent: :delete_all

has_many :teachers, through: :teach_courses

has_many :study_courses, dependent: :delete_all
has_many :students, through: :study_courses

has_many :tasks, dependent: :delete_all

...
```

5.10. Kodea: courses.rb - asoziazioak

Ondoren, Model honen asoziazioak definituko ditugu. Lehen bi lerroetan Course-k hainbat Teacher izan ditzakeela zehazten dugu, TeachCourses Model-ak lotzen dituelarik (datubasean, TeachCourses taula *teacher_id*, *course_id* bikotez osatuta egongo da). Hurrengo bi lerroek gauza berbera egiten dute Student-ekin, eta azken lerroak Course-k hainbat Task dituela definitzen du, Course jakin bat ezabatzen denean dagozkion Task-ak ere ezabatuko direla zehaztuz.

Asoziazio hauei esker, SQL erabili gabe lan egin dezakegu Model-ekin. Adibidez, Course jakin baten Task-ak lortzeko datubasearekin konexioa ireki, `"GET * FROM TASKS WHERE course_id = ?"` exekutatu eta emaitzak dagokien Model-aren formatura bihurtu beharrean `course.tasks` exekutatu eta ActiveRecord-i esker espero dugun emaitza lortuko dugu. Jakina, ActiveRecord erabil askoz gauza konplexuagoak ere egin daitezke (baldintza bat betetzen duten objektuak jaso, modu ezberdinetan ordenatu, SQLren antzera eremu guztiak jaso beharrean eremu jakin batzuk jaso, azkena sortu den objektua jaso...), nahiz eta ez den askotan suertatzen horiek erabiltzeko beharrik.

```
...

TENDENCY_TYPES = ['Formative', 'Accumulative']
LANGUAGES = ['En', 'Es', 'Eus']
MIN_RATE = 0
MAX_RATE = 100

validates_inclusion_of :tendency_type, :in => TENDENCY_TYPES
validates_inclusion_of :rate, :in => MIN_RATE..MAX_RATE
validates :title, presence: true
validate :correct_dates
validates_inclusion_of :language, :in => LANGUAGES

def correct_dates
  if ends.nil?
    errors.add(:ends, :blank)
  end
  if starts.nil?
    errors.add(:starts, :blank)
  end
  unless starts.nil? or ends.nil?
    if starts >= ends
      errors.add(:ends, :ends_before_starts)
    end
  end

  unless tasks.nil?
    if tasks.select{|task| (task.date > ends or task.date < starts)
}.any?
      errors.add(:ends, :tasks_outside_range)
    end
  end
end

...
```

5.11. Kodea: courses.rb - balidazioak

Kode zati honetan Course-ren balidazioak definitu ditugu, hau da, zein baldintza bete behar dituen Course batek datubasean sartu ahal izateko. Horretako, Course onargarri baten atributuek izan ditzaketen balioen tartean definitzen ditugu, adibidez `tendency_type`-ek zein balio har ditzakeen edo Course-ak hasiera data berriaren aurretik edo amaiera data berriaren ondoren Task-en bat baldin badu daten aldaketa ez dela onartuko. Balidazio hauek Course berri bat sortzean eta Course bat eguneratzean exekutatuko dira.

```
...

def progress
    (tasks.inject(0) {|result, element| result + element.weight} *
100).round(2)
end

def student_progress(student)
    (completed_performances(student).inject(0) {|result, element|
result + element.task.weight} * 100).round(2)
end

def task_performances(student)
    tasks.map{|task| task.task_performances.find_by student: student}
end

def completed_performances(student)
    task_performances(student).select{|tp| tp.grade >= 0}
end

def average_grade(student)
    completed_performances = completed_performances(student)
    unless completed_performances.any?
        return 0
    end
    (completed_performances.map{|tp| tp.grade *
tp.task.weight}.reduce(:+).to_f /
    completed_performances.map{|tp| tp.task.weight}.reduce(:
+)).round(2)
end

def add_teacher(teacher)
    teachers << teacher unless teachers.include? teacher
end

def remove_teacher(teacher)
    teachers.delete(teacher)
end

def add_student(student)
    students << student unless students.include? student
    tasks.each do |task|
        task.create_empty_performances_for(student)
    end
end

def remove_student(student)
    students.delete(student)
    tasks.each do |task|
```

```
        task.task_performances.where(student: student).destroy_all
    end
end

...
```

5.12. Kodea: course.rb - metodoak

Azkenik, Model honi dagozkion metodoak definitu ditugu. Rails-ek negozio logika Model-etan inplementatzea hobesten du, eta metodo argiak eta motzak idaztea. Honek probak idaztea, Rails-en filosofiaren beste ezaugarri garrantzitsu bat, errazten du. Zorionez, Ruby lengoaia labur eta zehatza da, adibide bezala ikusi `progress` metodoa, lerro bakarrean irakasgaiko zeregin guztien pisuen batura itzultzen duena.

Task Model-ari dagokionez, Course-ren kasuan bezala FriendlyId erabiliko dugu URL esanguratsuagoak lortzeko.

```
extend FriendlyId
    friendly_id :slug_candidates, use: :slugged

...

def slug_candidates
    [
      :name,
      [:name, :id]
    ]
end

...
```

5.13. Kodea: task.rb – FriendlyId

Ondoren, Task-en asoziazioak zein diren zehaztuko dugu.

```
...


belongs_to :course
has_many :task_performances, dependent: :delete_all
has_many :students, through: :task_performances

after_create :create_empty_performances

...

def create_empty_performances
    course.students.each do |s|
      TaskPerformance.create(student: s, task: self, grade: -1)
    end
end

...
```

5.14. Kodea: task.rb – asoziazioak eta after_create

Asoziazio hauek adierazten dutena da, lehenik eta behin, Task bat Course batena dela (belongs_to), hau da, *one-to-many* erlazio batean daudela (Task batek Course bat, Course batek hainbat Task). Horrez gain, Student bat eta Task bat lotzen duten TaskPerformanceak dituela adierazten da, TaskPerformane hauetako bakoitzak ikasleak zereginean izaten duen nota gordeko duelarik.

Asoziazioez gain, `after_create` ekintza bat definitu dugu hemen. Ekintza hau *trigger* moduko bat da, klase honetako objektu bat sortzean metodo jakin bat exekutatzeko aukera eskaintzen diguna. Gure kasuan, zeregin berri bat sortzean zereginaren irakasgaiko ikasle guztiek zeregin horretan TaskPerformance (zereginean ikasleak duen nota jasotzen duen klasea) huts bat sortzea nahi dugu. Kasu honetan, zereginaren nota hutsik dagoela adierazteko notari -1 balioa emango diogu, ondoren irakasleak benetako notarekin ordezkatuko duen balioa.

```
...

MIN_WEIGHT = 0.01
STEP_WEIGHT = 0.01
MAX_WEIGHT = 1
DEFAULT_WEIGHT = 0.05

SUBCLASSES = ["Exam", "ExerciseDelivery", "LabDelivery", "Other"]

validates :name, presence: true
validates :name, uniqueness: {scope: :course, case_sensitive: false}
validate :date_in_course_range
validates :place, presence: true
validates :type, presence: true
validates :course_id, presence: true
validates_inclusion_of :weight, :in => MIN_WEIGHT..MAX_WEIGHT
validate :weight_in_course_progress

def date_in_course_range
  if date.nil?
    errors.add(:date, :blank)
    return
  end
  if date < course.starts or date > course.ends
    errors.add(:date, :inclusion)
  end
end

def weight_in_course_progress
  if weight.nil?
    errors.add(:weight, :blank)
    return
  end
  if (weight * 100 + course.progress) > 100
    errors.add(:weight, :weight_progress)
  end
end

...
```

5.15. Kodea: task.rb – balidazioak

Balidazioei dagokienez, Course-ren eredu berbera jarraitzen dute, eta zeregin berri edo eguneratu batek bete beharreko baldintzak zehazten dute. Aipagarriak dira `date_in_course_range` (zereginaren data irakasgaiaren hasiera eta amaieraren artean dagoela ziurtatzeko) eta `weight_in_course_progress` (zereginaren pisua eta irakasgaian dagoeneko badauden zereginen pisuen baturak ehuneko ehuna pasatzen ez duela ziurtatzeko) metodoak.

```
...

def tooltip(student)
  student_grade = grade(student)
  tooltip_grade = student_grade < 0 ? "-" : student_grade
  tooltip = name + "\n" + I18n.t('grade') + ": " + tooltip_grade.to_s
+ "\n" + I18n.t('weight') + ": " + I18n.t('percentage', number:
(weight*100).round(2))
end

def fixed_performance(student)
  tp = task_performances.find_by(student: student)
  if tp.grade < 0
    tp.grade = 0
  end
  return tp
end

def fixed_grade(student)
  fixed_performance(student).grade.to_f.round(2)
end

def average_grade
  if completed_performances.length == 0
    return 0
  end
  (completed_performances.map{|tp| tp.grade*tp.task.weight}.reduce(:
+).to_f /
  completed_performances.map{|tp| tp.task.weight}.reduce(:+)).round(2)
end

def completed_performances
  completed_performances = task_performances.select{|tp| tp.grade >=
0}
end

...

def self.select_options
  hash = SUBCLASSES.map{|subclass| [I18n.t(subclass.downcase),
subclass] }
end

def label_color(student)
  if (date > DateTime.now)
    return "warning"
  elsif (grade(student) < 0)
    return "danger"
  else
```

```
    return "info"
  end
end

def teacher_label_color
  if (date > DateTime.now)
    return "warning"
  elsif (has_ungraded_performance)
    return "danger"
  else
    return "info"
  end
end

def has_ungraded_performance
  task_performances.each do |tp|
    if tp.grade < 0
      return true
    end
  end
  false
end

def create_empty_performances_for(student)
  TaskPerformance.create(student: student, task: self, grade: -1)
end

# http://www.alexreisner.com/code/single-table-inheritance-in-rails
def self.inherited(child)
  child.instance_eval do
    def model_name
      Task.model_name
    end
  end
  super
end

...
```

5.16. Kodea: task.rb - metodoak

Azkenik, Task-en metodoak definitu ditugu. Course-ren kasuan aipatu bezala, Rails-ek metodo motz eta zehatzak idaztea hobesten du, Unit Test motako probak modu errazean egin ahal izateko.

5.1.4. View-ak

Rails-en, View-ak html.erb amaierako fitxategiak dira. Formatu hau oinarrian HTML lengoaia da, baina Ruby zatiak txertatzeko aukera ematen du, baldintzazko egiturak, aldagaiak, formularioak etab. Sortzeko aukera emanez. View-ek dagokien Controller-eko ekintzatik instantzia aldagaiak jasotzen dituzte. Adibidez, Course bat editatu nahi badugu, courses_controller.rb-ko edit ekintzan (5.6. Kodea) @course instantzia aldagaia sortzen dugu. Course-ri dagokion edit.html.erb View-ak aldagai hori jasoko du, eta dagokion formularioa sortuko du.

```
<%= form_for(@course) do |f| %>
          ...
<% end %>
```

5.17. Kodea: courses/edit.html.erb – fomularioa

Ikus daitekeenez, html.erb formatuko View-etan Ruby kodea integratzeko <% ... %> (emaitza inprimatu nahi ez bada) eta <%= ... %> (emaitza inprimatu nahi bada) egiturak erabil daitezke. 5.17. Kode-zatiko bi etiketen artean Course-ri dagozkion eremuak (*field*) definituko ditugu, adibidez irakasgaiaren hasierari dagokion eremua:

```
<div id="course_start_field" class="col-xs-8 col-xs-offset-2 field
form-group">
        <%= f.label :starts, :class => "control-label" do %>
                <%= t('starts') %>:
        <% end %>
        <%= f.datetime_select :starts %>
</div>
```

5.18. Kodea: courses/edit.html.erb – eremua

Itxura egokia emateko Bootstrap klaseen ondoren, `field form-group` klaseekin formularioko eremua dela zehaztuko dugu, eta formularioko label-a zehaztu ondoren DateTime hautatzaile bat sortuko da `starts` parametroarentzat. Ruby kode txertatu hau eta Bootstrap klaseak erabiliz, interfaze grafikoak modu oso eraginkorrean sor daitezke.

5.2. DATUBASEA

5.2.1. Datubasearen eraikuntza

Aplikazioaren datu basea garatzeko orduan, Rails-ek arlo honetan proposatzen duen metodologia jarraitu dut. Rails-en datubasea sortzeko *migrazioak* erabiltzen dira, hau da, datubasea inkrementalki eraikitzen duten pausoak. Adibide bezala, hona hemen Course (irakasgai) klaseari dagokion taula sortzen duen migrazioa:

```
class CreateCourses < ActiveRecord::Migration
  def change
    create_table :courses do |t|
      t.string :title
      t.date :starts
      t.date :ends

      t.timestamps null: false
    end
  end
end
```

5.19. Kodea: 20160310171219_create_courses.rb

Garapenean aurrera egin ahala, konturatu nintzen irakasgaiek zein hizkuntzatan eskaintzen ziren jakin behar genuela. Ondorioz migrazio hau sortu nuen datubasean aldaketa hori egiteko, hau da, *courses* taulari *language* zutabea gehitzeko.

```
class AddLanguageToCourses < ActiveRecord::Migration
  def change
    add_column :courses, :language, :string
  end
end
```

5.20. Kodea: 20160417174522_add_language_to_courses.rb

Rails kontsola erabiliz migrazioak exekuta daitezke, datu basean aldaketak egikarituz,

```
mikel@mikelen-ordenagailua:~/Desktop/GAP/application/student-grades$ rake db:migrate
== 20160612154157 AddMessageLanguageToPeople: migrating =======================
-- add_column(:people, :message_language, :string, {:default=>"En"})
   -> 0.0005s
== 20160612154157 AddMessageLanguageToPeople: migrated (0.0006s) ==============
```

5.2. Irudia: Migrazioa exekutatzen

eta era berean migrazioak desegin egin daitzeke, datu basea aurreko egoera batera itzuliz.

```
mikel@mikelen-ordenagailua:~/Desktop/GAP/application/student-grades$ rake db:rollback
== 20160612154157 AddMessageLanguageToPeople: reverting =======================
-- remove_column(:people, :message_language, :string, {:default=>"En"})
   -> 0.0251s
== 20160612154157 AddMessageLanguageToPeople: reverted (0.0274s) ==============
```

5.3. Irudia: Migrazioa desegiten

Aplikazioa garatzerakoan, datubasean probak egiteko datuak sartu behar izaten ditugu, benetako egoera batean izango genukeen informazioa simulatuz. Gure kasuan, irakasgaiak, zereginak, irakasle eta ikasleak, notak, hauen arteko loturak... behar ditugu. Railsek *seeds.rb* fitxategia sortzen du honetarako: bertan, ruby lengoaian, sor dezakegu datubasean izatea nahi dugun informazioa. Adibide bezala, hona hemen sei irakasgaitan zeregin bakoitzean ikasle bakoitzari ausazko nota sortzeko kodea:

```
for i in 1..6
  Course.find(i).tasks.each do |task|
    Course.find(i).students.each do |student|
      TaskPerformance.create(task: task, student: student, grade:
rand(2..10))
    end
  end
end
```

5.21. Kodea: seeds.rb

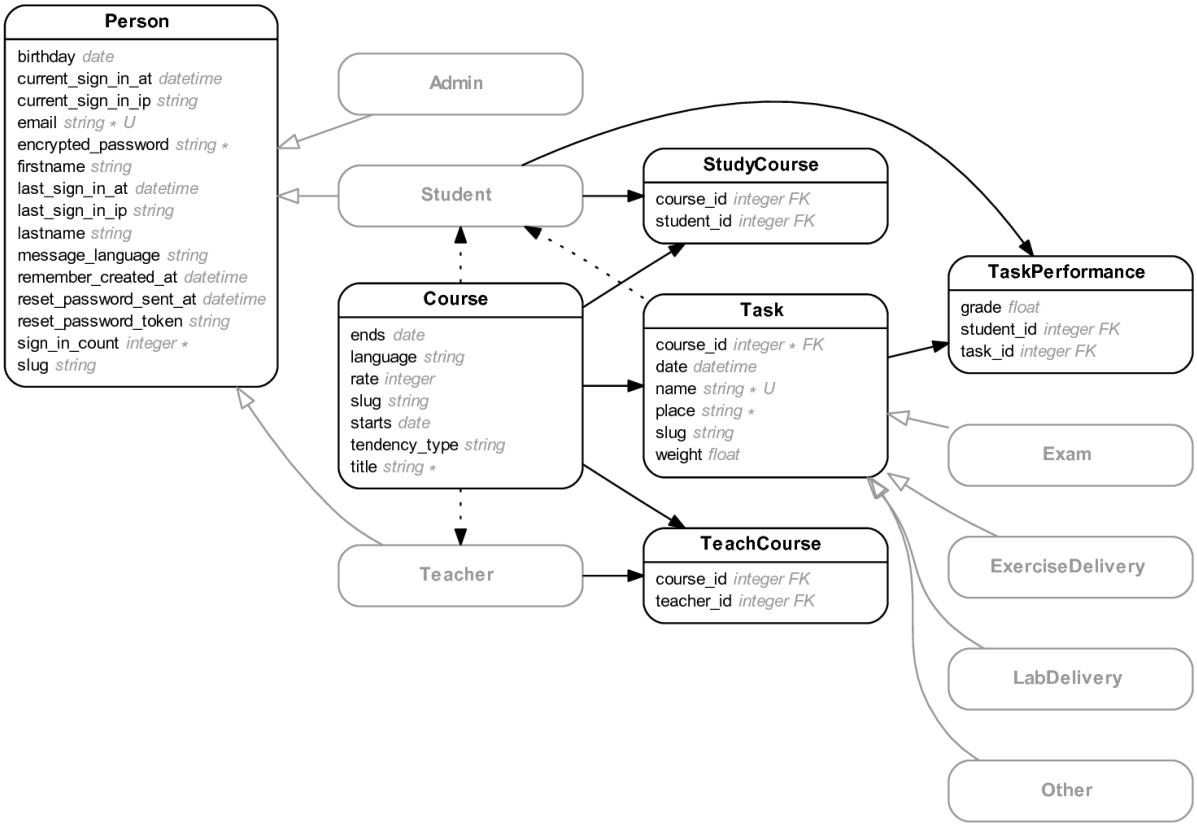
Rails kontsolan `rake db:seed` exekutatuz, datubasea *seeds.rb* fitxategiaren edukiekin beteko da.

Migrazioak eta *seeds.rb*-rekin, datu-basea sortzeko, aldatzeko eta betetzeko modu malgu eta boteretsua ematen du Rails-ek.

5.2.2. Datubaseak kudeakeatzeko sistemak

Aplikazioaren garapen-ingurunean, hau da, nire ordenagailuan, SQLite3 erabili dut, Rails-ek lehenesten duen DBKSa. Produkzio-ingurunean berriz, PostgreSQL erabili dut, Heroku-k erabiltzen duen DBKSa baita. PostgreSQL-k hainbat funtzionalitate aurreratu eskaintzen ditu (indize aurreratuak, *view*-ak), baina arlo horretan sakontzea proiektu honen irismenetik kanpo dagoela iruditu zaidanez ez dut arlo horretan lanik egin.

5.2.3. Datubasearen azken egitura



5.4. Irudia: Datubasearen azken egitura

6

Metodologiak

Atal honetan proiektuan erabilitako metodologiak azalduko dira, bai lanekoak (Scrum, TDD) eta bai kudeaketari dagozkionak.

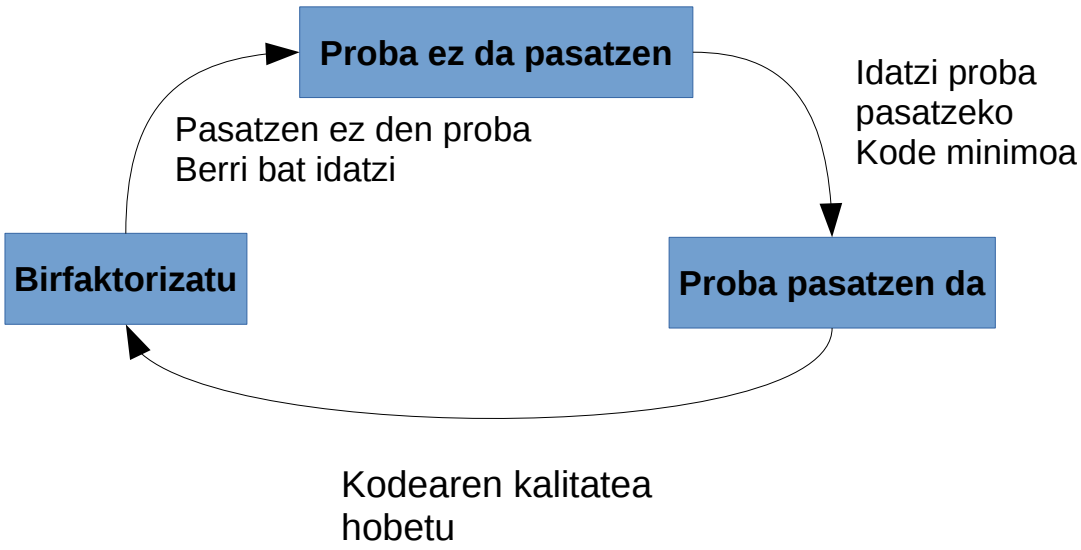
6.1. LANEKO METODOLOGIA

Proiektu hau aurrera eramateko erabili dudan metodologia **Scrum** izan da. Scrum metodologia arina da, eta ezaugarri hauek definitzen dute:

- Produktuaren planifikazio eta exekuzio osoa egin ordez, garapen inkrementala aurrera eramatea, iterazioka.
- Garapen-fase ezberdinak sekuentzialki ordez modu teilakatuan aurrera eramatea.
- Proiektuan parte hartzen duten agente ezberdinei rolak ezartzea (Product Owner, Scrum Master, Scrum Team)

Nire kasuan, Scrum-en moldaketa bat erabili dut, pertsona bakarreko talde batean lan egin dudala kontuan hartuta. Ondorioz, nire rola Scrum Team izatea izan da, hau da, produktuaren garapenean zentratzea. Beste interesatuen rolei dagokienez, proiektuko irakasleak Product Owner-aren papera bete du. Product Owner-aren betebeharra garatuko den produktuaren ideia argia izatea eta Scrum Team-a bideratzea da. Pertsona bakarreko taldea eta beste interesatu bakarra izanik, Scrum Master rola betetzeko beharrik ez dagoela ondorioztatzen da.

Scrum-ez gain, beste metodologia bat ere erabili dut proiektuaren garapenean: *Test Driven Development* paradigma, **TDD** bezala ezagunagoa.



6.1. Irudia: TDD

TDD metodologia erabiliz, 6.1. Irudian ikus daitekeen bezala, lehenik pasatzen ez den kodea idazten da, kodearen funtzionamendua zehaztuz, ondoren proba pasatzeko kode minimoa idazten da, kodearen kalitatea kontuan hartu gabe, eta azkenik kodearen kalitatea hobetzen da (birfaktorizazioa) kodeak proba pasatzen duen ziurtasuna izanik. Nire kasuan ez dut TDD hain modu hertsian aplikatu, horretara ohitzea zaila baita: batzuetan kodea pasa gabeko proba idatzi gabe inplementatu dut, baina ondoren beti proba idatzi eta pasatzen duela ziurtatu dut. Modu honetan, aplikazioak espezifikazioak betetzen dituela ziurtatzen duen proba-multzoa lortu dut, 7. Atalean erakusten dena, betekizunekin erlazionatuz.

6.2. KUDEAKETA METODOLOGIA

Proiketuan egin beharreko lana hasierako planifikazioarekin hasi da (3. Atala), baina metodologia arina erabili dugunez, atazak moldatuz joan dira produktua garatu ahala. Zeregin batzuetan espero baino denbora gehiago eman da (TDDren inguruko formakuntza) eta beste batzuk berriz ez dira bete, egiteko beharrik ikusi ez delako (Autentifikaziorako Rolify eta CanCanCan liburutegiak erabiltzea).

Egindako lana kudeatzeko erabili dudan tresnarik garrantzitsuena 8. Atalean agertzen diren taulak (Scrum-en Product Backlog deritzenak) dira. Bertan, iterazio bakoitzean, egin beharreko zereginak, zeregin horretan iterazioaren hasieran egiten zen estimazioa, benetan emandako orduak eta estimazio berria agertzen dira, eta azkenik zeregin horri hurrengo iterazioan emango zaion lehentasuna. 8. Ataleko azalpenean hobeto azalduko da taularen funtzionamendua.

7

Probak

Atal honetan proiektuan zehar egin ditugun probak azalduko dira, bai Rspec-ekin egindako software probak eta bai erabiltzaileekin egindako probak.

7.1. BETEKIZUNAK ETA PROBAK

Atal honetan **Rspec**-ekin egindako probak aplikazioaren betekizunekin erlazionatuko ditut, inplementatutako proben taula osatuz. Probak hiru talde nagusitan banatuko dira:

- **Model**: Unit Test motako probak dira, aplikazioko klaseen funtzionalitatea probatzen dute.
- **Controller**: Controller-ei egindako eskaerak eta espero diren erantzunak/ekintzak probatzen dituzte.
- **Feature**: Integration Test motako probak dira, hau da, erabiltzaileak aplikazioarekin duten elkarrekintza simulatzen dute: Estekak bisitatu, formularioak bete etab.

Hona hemen betekizunak eta dagokien zenbakia (betekizunak eta probak lotzeko erabiliko da):

- Irakasleak:
  - **1.** Irakasten dituen irakasgaien zerrenda ikusi.
  - **2.** Irakaslearen profila ikusi eta mezuen hizkuntza aldatu.
  - **3.** Irakasgai batean ikasleek izan dituzten notak ikusi, taula bezala.
  - **4.** Irakasgai batean ikasleek izan dituzten notak eta joerak ikusi, grafiko bezala, hainbat ikasle eta klaseko batezbestekoa konparatzeko aukerarekin.
  - **5.** Irakasgai batean ikasleek izan dituzten notak gehitu edo aldatu.
  - **6.** Irakasgai bat konfiguratu: joera mota, tasa, hasiera eta amaiera datak aldatu.
  - **7.** Irakasgai jakin batean zeregin berri bat sortu (adibidez, azterketa bat).
  - **8.** Irakasgai bateko zereginen zerrenda ikusi.
  - **9.** Zeregin bat ezabatu.
  - **10.** Zeregin bat editatu.
  - **23**. Irakasgai bateko ikasleen estatistikak ikusi, irakasgaitik kentzeko aukerarekin.

- Ikasleak:
  - **11.** Ikasten dituen irakasgaien zerrenda ikusi.
  - **12.** Ikaslearen profila ikusi eta mezuen hizkuntza aldatu.
  - **13.** Irakasgai batean izan dituen notak ikusi, taula bezala.
  - **14.** Irakasgai batean izan dituen notak eta joera ikusi, grafiko bezala.
  - **15.** Etorkizunean dituen zereginak ikusi.
  - **16.** Mezu bat jaso nota berri bat duen bakoitzean.

- Administratzaileak:
  - **17.** Irakasgai guztien zerrenda ikusi.
  - **18.** Irakasgai berri bat sortu.

    - **19.** Irakasgai bat ezabatu.
    - **20.** Irakasgai bateko irakasle / ikasleak gehitu / ezabatu.

- Denak:
    - **21.** Sisteman login egin.
    - **22**. Hasierako orrialde ikusi.

7.2. PROBEN TAULAK

7.2.1. Model Probak (Unit Test)

Klasea	Proba	Deskribapena	Betekizuna(k)
Student	checks if it studies a course	Ikasleak irakasgai jakin bat ikasten duen egiaztatzen du	13,14
	gets the progress in a course	Ikasleak irakasgai batean bete dituen zereginen pisuen ehunekoa kalkulatzen du	13,14
	gets the future tasks	Ikasleak etorkizunean egin beharko dituen zereginak lortzen ditu	15
	gets students not studying course	Irakasgai jakin bat ikasten ez duten ikasleak itzultzen ditu	20
Teacher	checks if it teaches a course	Irakasleak irakasgai jakin bat irakasten duen egiaztatzen du	3,4,5,6,7,8,9,10
	gets teachers not teaching course	Irakasgai jakin bat irakasten ez duten irakasleak itzultzen ditu	20
Course	gets the right language	Irakasgaiaren izena ondo itzultzen du (lokalea kontuan izanda)	1,17
	gets the right progress	Irakasgaiko zereginen pisuen batura kalkulatzen du	7,10
	gets the right progress of a student	Ikasle batek osatu dituen zereginen pisuen batura kalkulatzen du	13,14,23
	gets the task performances of a student	Ikasle batek irakasgaiko zereginetan dituen betetzeak itzultzen ditu (egindakoak eta egin gabeak)	16
	gets the completed performances of a student	Ikasle batek irakasgaiko zereginetan egin dituen betetzeak itzultzen ditu	4,14
	gets the average	Ikasle batek irakasgaiko	4,14,23

Klasea	Proba	Deskribapena	Betekizuna(k)
	grade of a student in the course	zereginetan izan duen batezbesteko nota kalkulatzen du	
	gets the total grade of a student in the course	Ikasle batek irakasgaiko zereginetan momenturaino izan duen noten batura kalkulatzen du	23
	gets the future tasks of a student	Ikasle batek irakasgaian etorkizunean egin beharko dituen zereginak lortzen ditu	15
	gets the past tasks of a course	Irakasgaian dagoeneko egindako zereginen zerrenda itzultzen du	4,14
	adds a teacher	Irakasgaiari irakasle bat gehitzen dio	20
	removes a teacher	Irakasgaiari irakasle bat kentzen dio	20
	adds a student	Irakasgaiari ikasle bat gehitzen dio	20
	removes a student	Irakasgaiari ikasle bat kentzen dio	20, 23
	Validations: can’t be edited with wrong tendency_type	Irakasgai bat ezin da editatu joera motaren balioa ezegokia bada	6,18
	Validations: can’t be edited with wrong rate	Irakasgai bat ezin da editatu tasaren balioa ezegokia bada	6,18
Task	gets the tooltip for a student	Ikasle batek grafikoan zereginean izango duen tooltip-a sortzen du	14
	gets the fixed performance of a student	Ikasle batek zereginean duen betetzea lortzen du (zeregina egin ez bada nota 0 jarriz)	3,5
	gets the fixed grade of a student	Ikasle batek zereginean duen nota lortzen du (zeregina egin ez bada 0 itzuliz)	13, 4,14
	gets the average grade of the task	Zereginean ikasleek duten batezbesteko nota kalkulatzen du	3,4,14
	gets the completed performances	Zereginean egin diren betetzeak bilatzen ditu	4,14
	gets the student hash	Ikasle baten nota eta pisua errepresentatzen duen hash-a itzultzen du	4,14

Klasea	Proba	Deskribapena	Betekizuna(k)
	gets the class hash	Zereginaren batezbesteko nota eta pisua errepresentatzen duen hash-a itzultzen du	4,14
	gets the student label colors	Ikaslearen taulan zereginari emango zaion kolorea kalkulatzen du	13
	gets the teacher label colors	Irakaslearen taulan zereginari emango zaion kolorea kalkulatzen du	3
	checks if there are ungraded performances	Zereginean notarik ezarri gabeko betetzeak badauden kalkulatzen du	3
	Validations: can be created with correct values	Zeregina sor daiteke parametro zuzenekin	7,10
	Validations: can’t be created without a name	Ezin da zeregina sortu izenik gabe	7,10
	Validations: can’t be created without a date	Ezin da zeregina sortu datarik gabe	7,10
	Validations: can’t be created with a date outside the course’s range	Ezin da zeregina sortu data irakasgaia hasi baino lehen edo amaitu ondoren bada	7,10
	Validations: can’t be created without a type	Ezin da zeregina sortu motarik gabe	7,10
	Validations: can’t be created with a weight outside the range	Ezin da zeregina sortu pisua tarte egokian ez badago (1-100)	7,10
	Validations: can’t be created without a weight	Ezin da zeregina sortu pisurik gabe	7,10
	Validations: can’t be created with a weight that exceeds the course progress	Ezin da zeregina sortu irakasgaiko zereginen pisuen baturak gehi zeregin berriaren pisuak 100 gainditzen badu	7,10
	Validations: can’t be created with a duplicate name	Ezin da zeregina sortu dagoeneko izen horretako zeregin bat existitzen bada	7,10

Klasea	Proba	Deskribapena	Betekizuna(k)
Task Performance	Validations: can be created with correct values	Zereginaren betetzea sor daiteke balio egokiekin	5
	Validations: can’t be created with wrong grade	Ezin da zereginaren betetzea sortu notaren balio okerrarekin	5
	Validations: can’t be updated for future task	Ezin da zereginaren betetzea eguneratu zereginaren data etorkizunean bada	5
Google Charts Data	Translates weights to colors	Zeregin baten pisua emanda grafikoko barrak izango duen kolorea kalkulatzen du (zenbat eta pisu handiagoa, orduan eta ilunagoa)	14
	Gets full grade data	Irakasgai bateko zereginen nota guztiak itzultzen ditu GoogleCharts-ek erakusteko formatuan	4
	Gets full tendency data	Irakasgai bateko zereginen joera guztiak itzultzen ditu GoogleCharts-ek erakusteko formatuan	4
	Gets average grade data	Irakasgai bateko zereginen batezbesteko notak itzultzen ditu GoogleCharts-ek erakusteko formatuan	4
	Gets average tendency data	Irakasgai bateko zereginen batezbesteko joerak itzultzen ditu GoogleCharts-ek erakusteko formatuan	4
	Gets student grade data	Irakasgai batean ikasle baten zereginen nota guztiak itzultzen ditu GoogleCharts-ek erakusteko formatuan	14
	Gets student tendency data	Irakasgai batean ikasle baten zereginen joera guztiak itzultzen ditu GoogleCharts-ek erakusteko formatuan	14
Tendency Calculator	Calculates formative tendencies	Zereginen pisu eta nota zerrenda bat, eta tendentzia mota eta tasa emanda joerak kalkulatzen ditu (Formatiboa, hau da, tasak aurreko joerak duen pisuan eragina du)	4,14

Klasea	Proba	Deskribapena	Betekizuna(k)
	Calculates accumulative tendencies	Zereginen pisu eta nota zerrenda bat, eta tendentzia mota eta tasa emanda joerak kalkulatzen ditu (Akumulatiboa, hau da, tasak aurreko joerak duen pisuan ez du eraginik)	4,14

7.1. Taula: Model Probak

7.2.2. Controller Probak

Klasea	Proba	Deskribapena	Betekizuna(k)
Courses Controller	Teacher lists courses	Irakasle batek irakasten dituen irakasgaiak zerrendatzen ditu	1
	Teacher shows course	Irakasle batek irakasten duen irakasgai jakin bat erakusten du	3.4
	Teacher can edit a course	Irakasle batek irakasten duen irakasgai bat editatzeko formularioa ikus dezake	6
	Teacher can update a course	Irakasle batek irakasten duen irakasgai bat alda dezake (joera mota, tasa)	6
	Teacher can’t update a course with wrong parameters	Irakasle batek ezin du irakasgaia aldatu parametro okerrekin (adibidez “Formative” edo “Accumulative” ez den mota)	6
	Teacher can’t update a course doesn’t teach	Irakasle batek ezin du irakasten ez duen irakasgai bat aldatu	6
	Teacher can save course page state	Irakasle batek irakasgai baten orrian egindako aukerak sesioan gordetzen dira	6
	Teacher can remove a student from a course it teaches	Irakasle batek irakasten duen irakasgai batetik ikasle bat ken dezake	23

Klasea	Proba	Deskribapena	Betekizuna(k)
	Student lists courses	Ikasle batek ikasten dituen irakasgaiak zerrendatzen ditu	11
	Student shows course	Ikasle batek ikasten duen irakasgai jakin bat erakusten du	13,14
	Student can’t edit a course	Ikasle batek ezin du ikasten duen irakasgai bat editatzeko formularioa ikusi	6
	Student can’t update a course	Ikasle batek ezin du ikasten duen irakasgai bat aldatu	6
	Student can save course page state	Ikasle batek irakasgai baten orrian egindako aukerak sesioan gordetzen dira	13,14
	Admin lists all courses	Administratzaileak Irakasgai guztien zerrenda ikus dezake	17
	Admin can create a new course	Administratzaileak Irakasgai berri bat sor dezake	18
	Admin can’t create a new course with wrong parameters	Administratzaileak ezin du Irakasgai berri bat sortu parametro okerrekin (adibidez, amaiera data hasiera dataren aurretik)	18
	Admin can delete a course	Administratzaileak Irakasgai bat ezaba dezake	19
	Admin can add and remove teachers	Administratzaileak Irakasgai bateko Irakasleak ken eta gehi ditzake	20
	Admin can add and remove students	Administratzaileak Irakasgai bateko Ikasleak ken eta gehi ditzake	20
Tasks Controller	Teacher can see a new task form	Irakasle batek irakasgai batean zeregin berri bat sortzeko formularioa ikus dezake	7
	Teacher can create a new task	Irakasle batek zeregin berri bat sor dezake	7

Klasea	Proba	Deskribapena	Betekizuna(k)
	Teacher can’t create a new task with wrong parameters	Irakasle batek ezin du zeregin bat sortu parametro okerrekin	7
	Teacher can’t create a new task if doesn’t teach task’s course	Irakasle batek ezin du zeregin bat sortu irakasten ez duen irakasgai batean	7
	Teacher can edit a task	Irakasleak irakasten duen Irakasgai bateko Zeregin bat editatzeko formularioa ikus dezake	10
	Teacher can update a task	Irakasleak irakasten duen Irakasgai bateko Zeregin bat edita dezake	10
	Teacher can’t update a task whose course doesn’t teach	Irakasleak ezin du irakasten ez duen Irakasgai bateko zeregin bat editatu	10
	Teacher can’t update a task with wrong parameters	Irakasleak ezin du zeregin bat editatu parametro okerrekin (adibidez, pisua = 155)	10
	Teacher can delete tasks	Irakasleak irakasten duen Irakasgai bateko Zeregin bat ezaba dezake	9
	Teacher can’t delete tasks from courses doesn’t teach	Irakasleak ezin du irakasten ez duen Irakasgai bateko zeregin bat ezabatu	9
	Student can see future tasks	Ikasle batek etorkizunean egin beharko dituen zereginen zerrenda ikus dezake	15
Task Performances Controller	Teacher can’t edit performance unless teaches course	Irakasle batek ezin du ikasle batek zeregin batean duen nota aldatu zeregin horren irakasgaia irakasten ez badu	5
	Teacher can edit performance if teaches course	Irakasle batek ikasle batek zeregin batean duen nota aldatu dezake zeregin horren irakasgaia irakasten badu	5

Klasea	Proba	Deskribapena	Betekizuna(k)
	Student can’t edit performances	Ikasle batek ezin du zereginen nota aldatu	5
Students Controller	Student shows profile	Ikasle batek bere profila erakusten du	2
	Student can update profile	Ikasle batek bere profila eguneratu dezake	2
	Student can’t update profile with wrong parameters	Ikasle batek ezin du bere profila eguneratu parametro okerrekin (adibidez, mezuen hizkuntza “XXYYZZ”)	2
Teachers Controller	Teacher shows profile	Irakasle batek bere profila erakusten du	12
	Teacher can update profile	Irakasle batek bere profila eguneratu dezake	12
	Teacher can’t update profile with wrong parameters	Irakasle batek ezin du bere profila eguneratu parametro okerrekin (adibidez, mezuen hizkuntza “XXYYZZ”)	12
MySessions Controller	Person changes page language to person’s message language	Pertsona batek (Irakasle / Ikasle / Administratzaile) hautatuko hizkuntza izango du orriak login egitean	22

7.2. Taula: Controller Probak

7.2.3. Feature Probak (Integration Test)

Klasea	Proba	Deskribapena	Betekizuna(k)
Login	Student logs in	Existitzen den ikasle batek login egin dezake	21
	Student fails login	Existitzen ez den ikasle batek ezin du login egin	21
	Teacher logs in	Existitzen den irakasle batek login egin dezake	21
	Teacher fails login	Existitzen ez den irakasle batek ezin du login egin	21
Course Data	Student sees course list of studied courses	Ikasle batek ikasten dituen irakasgaien taula ikus dezake	11

	Student sees course	Ikasle batek ikasten duen irakasgai baten orrialdea ikus dezake	13,14
	Teacher sees list of courses taught	Irakasle batek irakasten dituen irakasgaien taula ikus dezake	1
	Teacher sees course	Irakasle batek irakasten duen irakasgai baten orrialdea ikus dezake	3,4
	Teacher creates task	Irakasle batek zeregin berri bat sor dezake	7
	Teacher configures course	Irakasle batek zeregin baten joera-parametroak alda ditzake	6
	Teacher selects student	Irakasle batek taulako Ikasle bat hauta dezake bere grafikoa ikusteko	4
	Teacher sees list of course tasks	Irakasle batek Irakasgai bateko zereginen zerrenda ikus dezake	8
	Teacher deletes course task	Irakasle batek Irakasgai bateko zeregin bat ezaba dezake	9
Future Tasks	Student sees future task list	Ikasle batek etorkizunean izango dituen zereginak ikusten ditu	15
Profile	Student sees profile	Ikasle batek bere profila ikus dezake	2
	Teacher sees profile	Irakasle batek bere profila ikus dezake	12

7.3. Taula: Feature Probak

8

Iterazioak

Atal honetan proiektuan zehar aurrera eraman ditugun iterazioak erakutsiko dira, emandako orduak, estimazioak eta lehentasunak zehaztuz eta iterazioan egindako lana deskribatuz.

8.0. AZALPENA

Iterazioetan egindako lana eta estimazioak iterazio bakoitzari dagokion tauletan agertzen dira (8.1 Taula lehen iterazioan, 8.2 bigarrenean etab.) eta ondoko informazioa adierazten dute:

- **FASEAK** zutabeak zeregin bakoitza zein motatakoa den definitzen du.
- **ZEREGINAK** zutabean egin beharreko zereginak zerrendatzen dira.
- **AURREKO ESTIMAZIOA** zutabean iterazioaren hasieran zeregin hori amaitzeko beharko litzatekeen denboraren estimazioa agertzen da.
- **EMANDAKO ORDUAK** zutabean zeregin hori aurrera eramaten iterazioan eman den denbora agertzen da.
- **ESTIMAZIO BERRIA** zutabean iterazioaren amaieran zeregina amaitzeko beharko den denboraren estimazioa agertzen da.
- **LEHENTASUNA** zutabean hurrengo iterazioan zereginari emango zaion lehentasuna erakusten da, 1 (ez zaio lehentasunik emango), 2 (pixka bat lan egingo da) edo 3 (iterazio zeregin nagusietako bat izango da) balioa har dezakeelarik.

Aurreko estimazioaren, emandako orduen eta estimazio berriaren balioak kolore laranjarekin agertzen badira, estimatutako denbora baino gehiago eman dela adierazten da.

8.1. *1. ITERAZIOA* (2016/02/25 – 2016/03/10)

Lehen iterazioan aplikazioa garatzen hasi naiz. Orrialdearen indizea eta nabigazio barra inplementatu ditut, eta autentifikatze sistema ere bai.

8.1. Irudia: Autentifikatze sistema

Gainera, orrialdea hizkuntza ezberdinetara itzultzeko sistema inplementatu dut.

8.2. Irudia: Orrialdearen itzulpena

FASEAK		ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Betekizunak		Funtzionalak	3	1	2	3
		Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna...)	2	1	1	3
Kudeaketa		Bilerak eta aktak	15	1,5	13,5	3
		Hasierako planifikazioa	8	2	6	3
Formakuntza	Teknikoa	Ruby on Rails	80	5	75	3
		Devise, Authority, Rolify	10	3	7	2
		Google Charts	8	0	8	1
		TDD (Ruby)	6	0	6	1
Garapena	Diseinua	Interfaze Grafikoen Prototipoak	4	2	2	3
	Inplementazioa	Datu basea / Klaseak	10	3	7	2
		Autentifikazioa	8	2	6	2
		Lokalizazioa	4	1	3	2
		Informazioa bistaratzea	20	1	19	3
		Zereginak sortzea	2	0	2	1
		Notak gehitzea	4	0	4	1
		Rspec probak inplementatzea	20	0	20	1
		Beste funtzionalitate batzuk	20	0	20	1

FASEAK			ZEREGINAK		AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
	Probak	Diseinu grafikoa eta analisia	Ikasle eta irakasleekin probak		10	0	10	3
		Funtzionalitateak	Ikasle eta irakasleekin probak		4	0	4	1
		Barne-probak	Sistema eragile eta nabigatzaile eta aparatu ezberdinekin probak egin		6	0	6	1
Dokumentazioa			Instalazio-gida		1	0	1	1
			Memoria		40	2	38	3
			UML Diagramak	Klase-diagrama	5	0	5	1
				Erabilpen kasuak	5	0	5	1
				Sekuentzia-diagramak	5	0	5	1
GUZTIRA:					300	24,5	275,5	

8.1. Taula: 1. Iterazioa

Bileran erabaki da hurrengo iterazioa taulan zehaztutako lehentasunen arabera aurrera eramatea eta ikasleekin interfazeen prototipoen probak egitea. Memoriaren lehen atalean lana egin behar dela ere erabaki da.

8.2. *2. ITERAZIOA* (2016/03/10 – 2016/04/16)

Aste Santuko oporrak tartean izanda, eta irakasgaietako lanak entregatzeko eta azken azterketak egiteko astean proiekturako denborarik izan ez dudanez, iterazio hau oso luzea izan da. Iterazio honetan batez ere Ruby eta Railsen inguruko formakuntza egin dut *Practical Object-Oriented Design in Ruby* eta *Ruby on Rails Tutorial* liburuekin. Gainera, lehen iterazioan sortutako interfazeen prototipoen ebaluazioa egiten hasi naiz hainbat ikaskiderekin (A Eranskina). Memoriaren lehen atalean ere lana egin dut, eta aplikazioa garatzen jarraitu dut, Irakasgaiak eta Zereginak sortuz eta informazioaren bistaratzean lan eginez.

FASEAK		ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Betekizunak		Funtzionalak	2	2	4	3
		Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna ...)	1	1	4	3
Kudeaketa		Bilerak eta aktak	13,5	1	12,5	3
		Hasierako planifikazioa	6	2	4	3
Formakuntza	Teknikoa	Ruby on Rails	75	30	45	1
		Devise, Authority, Rolify	7	3	4	1
		Google Charts	8	0	8	1
		TDD (Ruby)	6	0	6	3
Garapena	Diseinua	Interfaze Grafikoen Prototipoak	2	2	0	1
	Inplementazioa	Datu basea / Klaseak	7	1	6	1
		Autentifikazioa	6	2	4	1
		Lokalizazioa	3	0,5	2,5	1

			Informazioa bistaratzea		19	4	15	1
			Zereginak sortzea		2	0	2	1
			Notak gehitzea		4	0	4	1
			Rspec probak inplementatzea		20	0	20	1
			Beste funtzionalitate batzuk		20	0	20	1
	Probak	Diseinu grafikoa eta analisia	Ikasle eta irakasleekin probak		10	3	7	3
		Funtzionalitateak	Ikasle eta irakasleekin probak		4	0	4	1
		Barne-probak	Sistema eragile eta nabigatzaile eta aparatu ezberdinekin probak egin		6	0	6	1
Dokumentazioa			Instalazio-gida		1	0	1	1
			Memoria		38	3	35	3
			UML Diagramak	Klase-diagrama	5	0	5	1
				Erabilpen kasuak	5	0	5	1
				Sekuentzia-diagramak	5	0	5	1
GUZTIRA:					275,5	54,5	229	

8.2. Taula: 2. Iterazioa

Bileran erabaki da hurrengo iterazioan ikasleekin interfazeen prototipoen probak amaitzea, TDDren inguruko formakuntza egitea eta aplikazioaren betekizunak definitiboki formalizatzea memorian.

8.3. *3. ITERAZIOA* (2016/04/16 – 2016/04/27)

Iterazio honetan interfazeen prototipoen probak amaitu ditut eta betekizunak formalizatu ditut. Gainera, TDDren inguruko formakuntza egiten hasi naiz *Rails 4 Test Prescriptions* liburuarekin. Azkenik, aplikazioan irakasgaietako noten grafikoak inplementatzen hasi naiz.

FASEAK		ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Betekizunak		Funtzionalak	4	1	3	3
		Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna...)	4	1	3	3
Kudeaketa		Bilerak eta aktak	12,5	1,5	11	3
		Hasierako planifikazioa	4	1	3	3
Formakuntza	Teknikoa	Ruby on Rails	45	10	35	2
		Devise, Authority, Rolify	4	2	2	1
		Google Charts	8	4	4	1
		TDD (Ruby)	6	2	4	3
Garapena	Diseinua	Interfaze Grafikoen Prototipoak	0	0	0	1
	Inplementazioa	Datu basea / Klaseak	6	2	4	2
		Autentifikazioa	4	2	2	1
		Lokalizazioa	2,5	0,5	2	1
		Informazioa bistaratzea	15	4	11	2
		Zereginak sortzea	2	0	2	1
		Notak gehitzea	4	0	4	1
		Rspec probak inplementatzea	20	0	20	1
		Beste funtzionalitate batzuk	20	0	20	1

FASEAK			ZEREGINAK		AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
	Probak	Diseinu grafikoa eta analisia	Ikasle eta irakasleekin probak		7	2	5	1
		Funtzionalitateak	Ikasle eta irakasleekin probak		4	0	4	1
		Barne-probak	Sistema eragile eta nabigatzaile eta aparatu ezberdinekin probak egin		6	0	6	1
Dokumentazioa			Instalazio-gida		1	0	1	1
			Memoria		35	2	33	1
			UML Diagramak	Klase-diagrama	5	0	5	1
				Erabilpen kasuak	5	0	5	1
				Sekuentzia-diagramak	5	0	5	1
				GUZTIRA:	229	35	194	

8.3. Taula: 3. Iterazioa

Bileran erabaki da hurrengo iterazioan TDDren inguruko formakuntzarekin jarraitzea, eta interfazeen prototipoekin egindako proben emaitzak berrikusi dira.

8.4. *4. ITERAZIOA* (2016/04/27 – 2016/05/04)

Iterazio honetan TDDren eta Rubyren inguruko formakuntza egiten jarraitu dut, eta probak inplementatzen hasi naiz RSpec erabiliz. Aplikazioan irakasgaietako noten grafikoak inplementatu ditut Google Charts erabiliz.

8.3. Irudia: Irakaslearen bista, lehen bertsioa

FASEAK		ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Betekizunak		Funtzionalak	3	1	2	3
		Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna...)	3	0	3	3
Kudeaketa		Bilerak eta aktak	11	1	10	3
		Hasierako planifikazioa	3	0	3	3
Formakuntza	Teknikoa	Ruby on Rails	35	10	25	2
		Devise, Authority, Rolify	2	0	2	1
		Google Charts	4	3	1	1
		TDD (Ruby)	4	0	4	3
Garapena	Diseinua	Interfaze Grafikoen Prototipoak	0	0	0	1
		Datu basea /	4	2	2	2

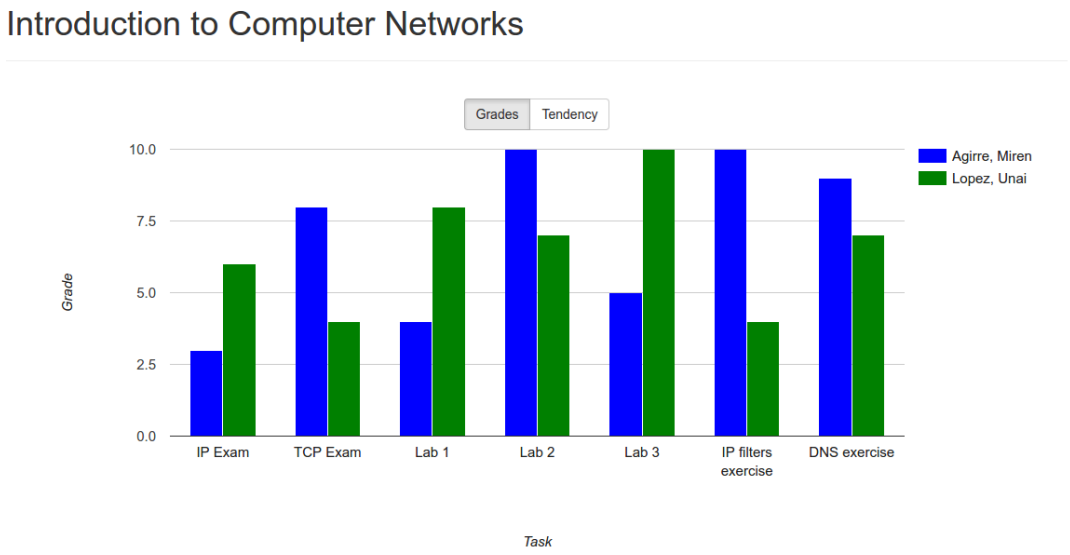
FASEAK			ZEREGINAK		AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
	Inplementazioa		Klaseak					
			Autentifikazioa		2	0	2	1
			Lokalizazioa		2	0	2	1
			Informazioa bistaratzea		11	3	8	2
			Zereginak sortzea		2	0	2	1
			Notak gehitzea		4	0	4	1
			Rspec probak inplementatzea		20	2	18	3
			Beste funtzionalitate batzuk		20	0	20	1
	Probak	Diseinu grafikoa eta analisia	Ikasle eta irakasleekin probak		5	0	5	1
		Funtzionalitateak	Ikasle eta irakasleekin probak		4	0	4	1
		Barne-probak	Sistema eragile eta nabigatzaile eta aparatu ezberdinekin probak egin		6	0	6	1
Dokumentazioa			Instalazio-gida		1	0	1	1
			Memoria		33	1	32	1
			UML Diagramak	Klase-diagrama	5	0	5	1
				Erabilpen kasuak	5	0	5	1
				Sekuentzia-diagramak	5	0	5	1
				GUZTIRA:	194	23	171	

8.4. Taula: 4. Iterazioa

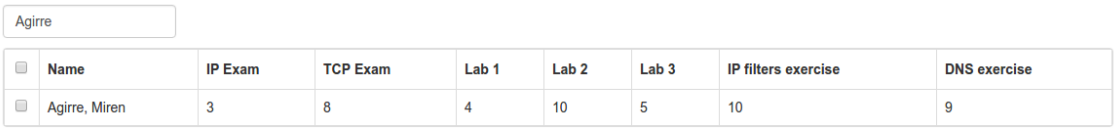
Bileran erabaki da hurrengo iterazioan noten taulan bilaketak egiteko eta notak konparatzeko aukerak inplementatzea, eta TDDren inguruko formakuntzan jeta probak inplementatzen jarraitzea.

8.5. *5. ITERAZIOA* (2016/05/04 – 2016/05/11)

Iterazio honetan TDDren inguruko formakuntza egiten jarraitu dut, eta probak inplementatzen jarraitu dut RSpec erabiliz. Aplikazioan irakasgaietako noten tendentziaren lehen bertsioa inplementatu dut Google Charts erabiliz. Noten taulari bilaketa eremua gehitu diot eta ikasle ezberdinen notak konparatzeko aukera inplementatu dut checkboxak erabiliz.



8.4. Irudia: Bi ikasleren notak konparatu



8.5. Irudia: Ikasle bat bilatu

FASEAK	ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Betekizunak	Funtzionalak	2	0	2	3
	Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna...)	3	0	3	3
Kudeaketa	Bilerak eta aktak	10	1,5	8,5	1
	Hasierako planifikazioa	3	1	2	1

FASEAK			ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Formakuntza	Teknikoa		Ruby on Rails	25	5	20	2
			Devise, Authority, Rolify	2	0	2	1
			Google Charts	1	2	3	1
			TDD (Ruby)	4	2	2	3
Garapena	Diseinua		Interfaze Grafikoen Prototipoak	0	0	0	1
	Inplementazioa		Datu basea / Klaseak	2	0	2	2
			Autentifikazioa	2	0	2	1
			Lokalizazioa	2	0	2	1
			Informazioa bistaratzea	8	4	4	2
			Zereginak sortzea	2	0	2	1
			Notak gehitzea	4	0	4	1
			Rspec probak inplementatzea	18	4	14	3
			Beste funtzionalitate batzuk	20	0	20	1
	Probak	Diseinu grafikoa eta analisia	Ikasle eta irakasleekin probak	5	0	5	1
		Funtzionalitateak	Ikasle eta irakasleekin probak	4	0	4	1
		Barne-probak	Sistema eragile eta nabigatzaile eta aparatu ezberdinekin probak egin	6	0	6	1
Dokumentazioa			Instalazio-gida	1	0	1	1
			Memoria	32	1	31	2
		UML Diagramak	Klase-diagrama	5	0	5	1
			Erabilpen kasuak	5	0	5	1

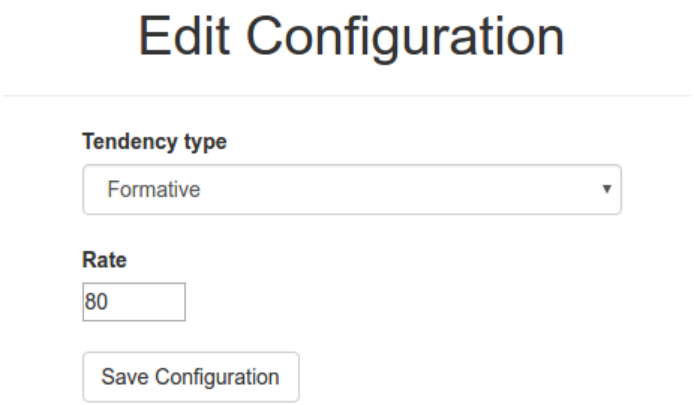
FASEAK	ZEREGINAK		AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
		Sekuentzia -diagramak	5	0	5	1
GUZTIRA:			171	20,5	154,5	

8.5. Taula: 5. Iterazioa

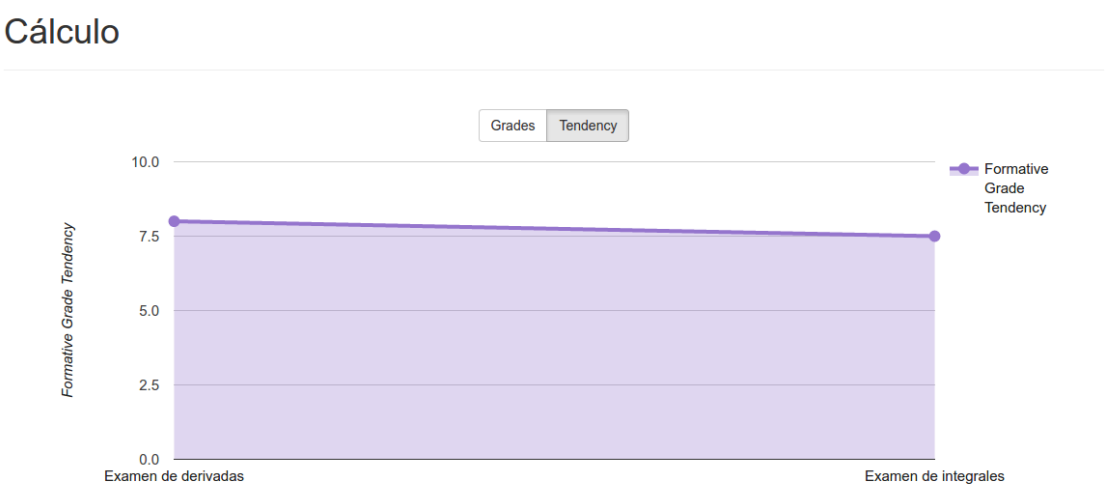
Bileran noten joera kalkulatzeko formula definitu da, eta erabaki da hurrengo iterazioan TDDn formatzen jarraitzea, probak proba-banku bezala antolatzea (betekizunekin erlazionatuz) eta joera grafikoak inplementatzea, joera kalkulatzeko modua konfiguratzeko aukerarekin.

8.6. *6. ITERAZIOA* (2016/05/11 – 2016/05/18)

Iterazio honetan TDDren inguruko formakuntza egiten jarraitu dut, eta probak inplementatzen jarraitu dut RSpec erabiliz. Irakasleek irakasgai baten tendentzia mota konfiguratzeko aukera inplementatu dut. Ikasleek irakasgai batean ikusten duten tendentzia grafikoa irakasgaiaren tendentzia motaren arabera kalkulatzea inplementatu dut, eta betetako zeregin ehunekoa erakusten duen barra bat ere bai.



8.6. Irudia: Joeraren konfigurazioa



8.7. Irudia: Ikaslearen joera irakasgai batean

FASEAK			ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Betekizunak			Funtzionalak	2	1	1	2
			Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna...)	3	1	2	2
Kudeaketa			Bilerak eta aktak	8,5	1	7,5	1
			Hasierako planifikazioa	2	0	0	1
Formakuntza	Teknikoa		Ruby on Rails	20	10	10	2
			Devise, Authority, Rolify	2	0	2	1
			Google Charts	3	2	1	1
			TDD (Ruby)	2	3	4	3
Garapena	Diseinua		Interfaze Grafikoen Prototipoak	0	0	0	1
	Inplementazioa		Datu basea / Klaseak	2	0	2	2
			Autentifikazioa	2	0	2	1
			Lokalizazioa	2	0,5	1,5	1
			Informazioa bistaratzea	4	2	2	2
			Zereginak sortzea	2	0	2	1
			Notak gehitzea	4	0	4	1
			Rspec probak inplementatzea	14	4	10	3
			Beste funtzionalitate batzuk	20	0	20	1
	Probak	Diseinu grafikoa eta analisia	Ikasle eta irakasleekin probak	5	0	5	1
		Funtzionalitateak	Ikasle eta irakasleekin probak	4	0	4	1
		Barne-probak	Sistema eragile eta nabigatzaile	6	0	6	1

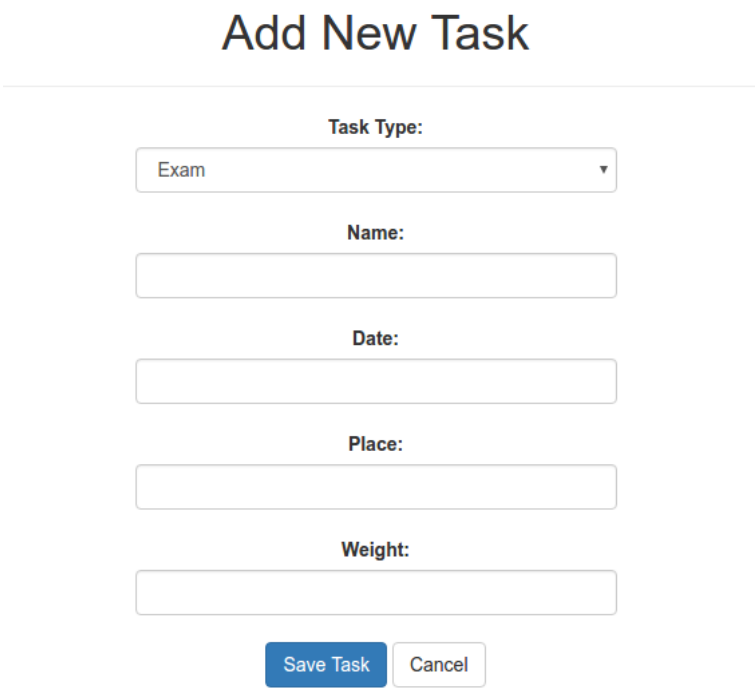
FASEAK	ZEREGINAK		AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
	eta aparatu ezberdinekin probak egin					
Dokumentazioa	Instalazio-gida		1	0	1	1
	Memoria		31	3	28	1
	UML Diagramak	Klase-diagrama	5	0	5	1
		Erabilpen kasuak	5	0	5	1
		Sekuentzia-diagramak	5	0	5	1
	GUZTIRA:		154,5	27,5	130	

8.6. Taula: 6. Iterazioa

Bileran erabaki da dokumentazioan lan egingo dela, eta joera-grafikoen konparazioa eta zeregin berrien sorrera inplementatuko direla hurrengo iterazioan.

8.7. *7. ITERAZIOA* (2016/05/18 – 2016/05/25)

Iterazio honetan, irakasleak zeregin berriak sortzeko funtzionalitatea inplementatu dut. Gainera, ikasleak irakasgai baten zereginaren joera ikustean klaseko batezbesteko joerarekin konparatzeko aukera inplementatu dut. Mugikorrean aplikazioa modu egokian ikusteko moldaketak egin ditut. Probak sortzen segi dut, betekizunekin erlazionatuz.



8.8. Irudia: Zeregin berria

FASEAK		ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Betekizunak		Funtzionalak	1	0	1	2
		Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna...)	2	1	1	2
Kudeaketa		Bilerak eta aktak	7,5	1,5	6	1
		Hasierako planifikazioa	0	0	0	1
Form	Teknikoa	Ruby on Rails	10	5	5	2

FASEAK			ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
akuntza			Devise, Authority, Rolify	2	0	2	1
			Google Charts	1	0	1	1
			TDD (Ruby)	4	3	1	3
Garapena	Diseinua		Interfaze Grafikoen Prototipoak	0	0	0	1
	Inplementazioa		Datu basea / Klaseak	2	0	2	2
			Autentifikazioa	2	0	2	1
			Lokalizazioa	1,5	1	0,5	1
			Informazioa bistaratzea	2	2	0	2
			Zereginak sortzea	2	2	0	1
			Notak gehitzea	4	0	4	1
			Rspec probak inplementatzea	10	2	8	3
			Beste funtzionalitate batzuk	20	2	18	1
	Probak	Diseinu grafikoa eta analisia	Ikasle eta irakasleekin probak	5	0	5	1
		Funtzionalitateak	Ikasle eta irakasleekin probak	4	0	4	1
		Barne-probak	Sistema eragile eta nabigatzaile eta aparatu ezberdinekin probak egin	6	0	6	1
Dokumentazioa			Instalazio-gida	1	0	1	1
			Memoria	28	2	26	1
		UML Diagramak	Klase-diagrama	5	0	5	1
			Erabilpen kasuak	5	0	5	1

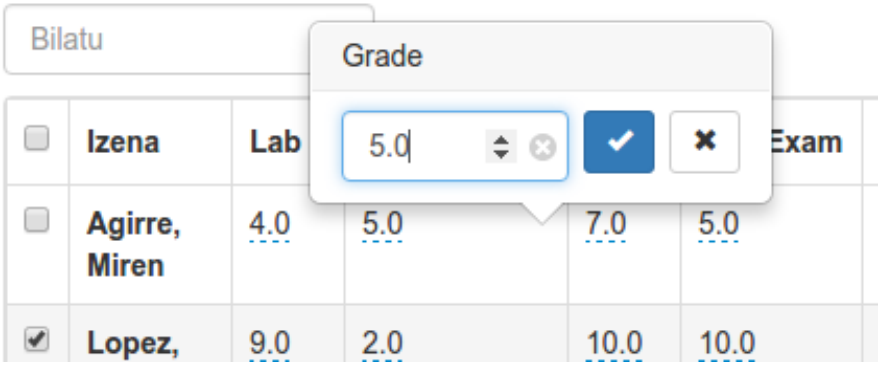
FASEAK	ZEREGINAK		AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
		Sekuentzia-diagramak	5	0	5	1
		GUZTIRA:	130	21,5	108,5	

8.7. Taula: 7. iterazioa

Bileran erabaki da hurrengo iterazioan dokumentazioan lan egingo dela eta zereginen notak gehitzeko eta aldatzeko funtzionalitatea inplementatuko dela.

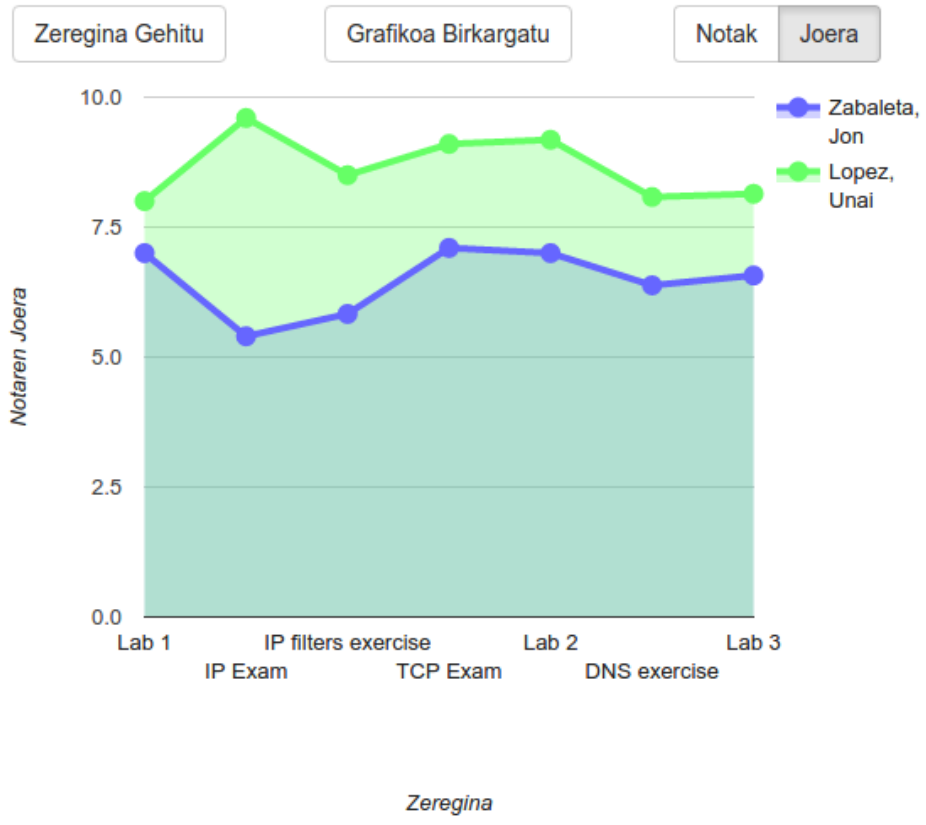
8.8. *8. ITERAZIOA* (2016/05/25 – 2016/06/01)

Iterazio honetan, irakasleak zereginen notak gehitzeko eta aldatzeko funtzionalitatea inplementatu dut.



8.9. Irudia: Nota gehitu edo aldatu

Gainera, ikasleen notak konparatzeaz gain, tendentziak ere konparatzeko aukera inplementatu dut. Grafikoa birkargatzeko (notak aldatzean grafikoan ematen den aldaketa ikusteko) eta grafikoa zabaltzeko (tamaina handiagoan ikus dadin) aukerak inplementatu ditut.



8.10. Irudia: Joeren konparazioa

Ikasle batek etorkizunean izango dituen zereginen zerrenda ikusteko aukera inplementatu dut. Azkenik, aplikazioa mugikorrean ahalik eta hoberen ikus dadin hainbat hobekuntza egin ditut eta hasierako orrialdea aldatu dut aplikazioko irudiak erakusteko.

FASEAK			ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Betekizunak			Funtzionalak	1	0	0	1
			Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna...)	1	0	0	1
Kudeaketa			Bilerak eta aktak	6	1	5	1
			Hasierako planifikazioa	0	0	0	1
Formakuntza	Teknikoa		Ruby on Rails	5	3	2	2
			Devise, Authority, Rolify	2	0	2	1
			Google Charts	1	1	0	1
			TDD (Ruby)	1	2	3	3
Garapena	Diseinua		Interfaze Grafikoen Prototipoak	0	0	0	1
	Inplementazioa		Datu basea / Klaseak	2	0	2	2
			Autentifikazioa	2	0	2	1
			Lokalizazioa	0,5	0,5	0	1
			Informazioa bistaratzea	0	2	2	2
			Zereginak sortzea	0	0	0	1
			Notak gehitzea	4	3	1	1
			Rspec probak inplementatzea	8	2	6	3
			Beste funtzionalitate batzuk	18	6	12	3
	Probak	Diseinu grafikoa eta analisia	Ikasle eta irakasleekin probak	5	0	5	1
		Funtzionalitatea	Ikasle eta irakasleekin	4	0	4	1

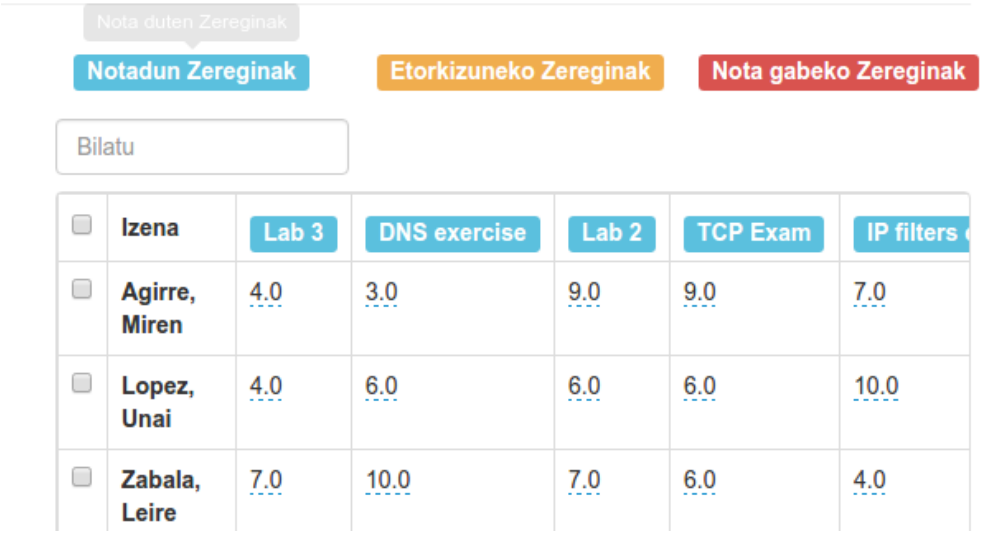
FASEAK			ZEREGINAK		AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
		k	probak					
		Barne-probak	Sistema eragile eta nabigatzaile eta aparatu ezberdinekin probak egin		6	0	6	1
Dokumentazioa			Instalazio-gida		1	0	1	1
			Memoria		26	1	25	1
			UML Diagramak	Klase-diagrama	5	0	5	1
				Erabilpen kasuak	5	0	5	1
				Sekuentzia-diagramak	5	0	5	1
				GUZTIRA:	108,5	21,5	93	

8.8. Taula: 8. Iterazioa

Bileran erabaki da hurrengo iterazioan dokumentazioan lan egiteaz gain hasierako orrialdea aldatuko dela pantaila-kapturak direla ondo uler dadin, eta nota gabe egon arren geafikoetan eragiten duten zereginen arazoa konponduko dela.

8.9. *9. ITERAZIOA* (2016/06/01 – 2016/06/09)

Iterazio honetan, etorkizuneko eta nota gabeko zereginak erakusteko modua zuzendu dut: etorkizunekoen nota ezingo da aldatu eta ez dira grafikoetan agertuko (taulan bakarrik) eta gainera zereginaren egoeraren arabera kolore bat izango du taulan (notarekin urdina, nota gabe gorria, etorkizunean laranja).



8.11. Irudia: Zereginen egoerak

Gainera, RSpec probak osatu ditut eta kodea txukundu dut ahalik eta errepikapen gutxien izateko eta ahalik eta ulergarriena izan dadin.

FASEAK	ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Betekizunak	Funtzionalak	0	0	0	1
	Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna...)	0	0	0	1
Kudeaketa	Bilerak eta aktak	5	1,5	3,5	1
	Hasierako planifikazioa	0	0	0	1

FASEAK			ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Formakuntza	Teknikoa		Ruby on Rails	2	1	1	2
			Devise, Authority, Rolify	2	0	2	1
			Google Charts	0	0	0	1
			TDD (Ruby)	3	3	1	3
Garapena	Diseinua		Interfaze Grafikoen Prototipoak	0	0	0	1
	Inplementazioa		Datu basea / Klaseak	2	0	2	1
			Autentifikazioa	2	0	2	1
			Lokalizazioa	0	1	1	1
			Informazioa bistaratzea	2	4	2	2
			Zereginak sortzea	0	0	0	1
			Notak gehitzea	1	0	1	1
			Rspec probak inplementatzea	6	4	2	3
			Beste funtzionalitate batzuk	12	2	10	3
	Probak	Diseinu grafikoa eta analisia	Ikasle eta irakasleekin probak	5	0	5	1
		Funtzionalitateak	Ikasle eta irakasleekin probak	4	0	4	1
		Barne-probak	Sistema eragile eta nabigatzaile eta aparatu ezberdinekin probak egin	6	0	6	1
Dokumentazioa			Instalazio-gida	1	0	1	1
			Memoria	25	3	22	1
			UML Diagram: Klase-diagrama	5	0	5	1
			UML Diagram: Erabilpen	5	0	5	1

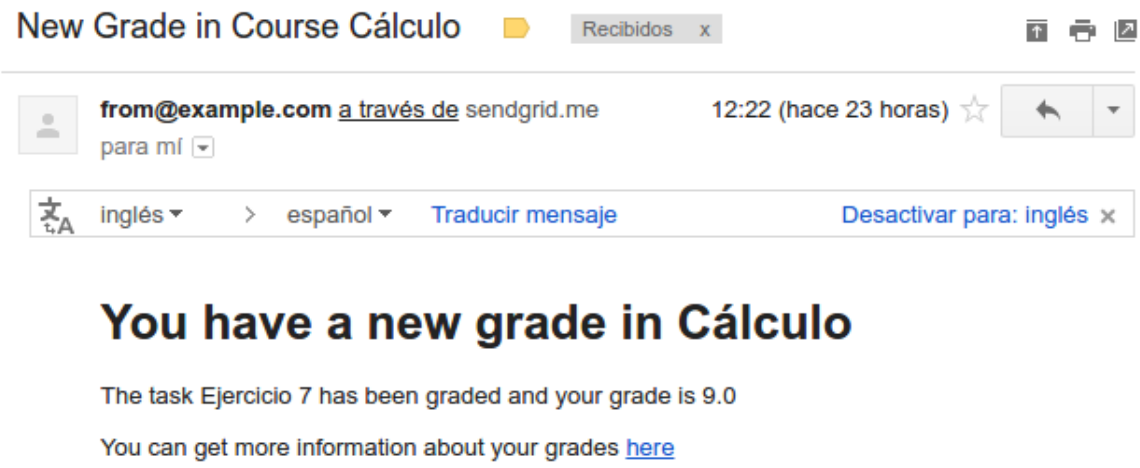
FASEAK	ZEREGINAK		AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
	ak	kasuak				
		Sekuentzia-diagramak	5	0	5	1
GUZTIRA:			93	19,5	80,5	

8.9. Taula: 9. Iterazioa

Bileran erabaki da hurrengo iterazioan administratzailearen funtzionalitatea inplementatuko dela (irakasgaiak sortu/ezabatu eta irakasle/ikasleak gehitu/kendu). Gainera, nota berri bat jasotzean ikasleak emaila jasotzeko funtzionalitatea ere inplementatuko da.

8.10. *10. ITERAZIOA* (2016/06/09 – 2016/06/15)

Iterazio honetan, Administratzailearen funtzionalitate osoa inplementatu dut, eta ikasleei nota berriak jartzean mezuak bidaltzeko funtzionalitatea ere bai.



8.12. Irudia: Nota berria – emaila

FASEAK		ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Betekizunak		Funtzionalak	0	0	0	1
		Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna...)	0	0	0	1
Kudeaketa		Bilerak eta aktak	3,5	0,5	3	1
		Hasierako planifikazioa	0	0	0	1
Formakuntza	Teknikoa	Ruby on Rails	1	2	2	2
		Devise, Authority, Rolify	2	0	0	1
		Google Charts	0	0	0	1

FASEAK			ZEREGINAK		AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
			TDD (Ruby)		1	0	0	1
Garapena	Diseinua		Interfaze Grafikoen Prototipoak		0	0	0	1
	Inplementazioa		Datu basea / Klaseak		2	0	0	1
			Autentifikazioa		2	0	0	1
			Lokalizazioa		1	1	1	2
			Informazioa bistaratzea		2	2	0	2
			Zereginak sortzea		0	0	0	1
			Notak gehitzea		1	0	0	1
			Rspec probak inplementatzea		2	2	6	3
			Beste funtzionalitate batzuk		10	10	6	2
	Probak	Diseinu grafikoa eta analisia	Ikasle eta irakasleekin probak		5	0	5	1
		Funtzionalitateak	Ikasle eta irakasleekin probak		4	0	4	1
		Barne-probak	Sistema eragile eta nabigatzaile eta aparatu ezberdinekin probak egin		6	0	6	1
Dokumentazioa			Instalazio-gida		1	0	1	1
			Memoria		22	1	21	1
			UML Diagramak	Klase-diagrama	5	0	5	1
				Erabilpe	5	0	5	1

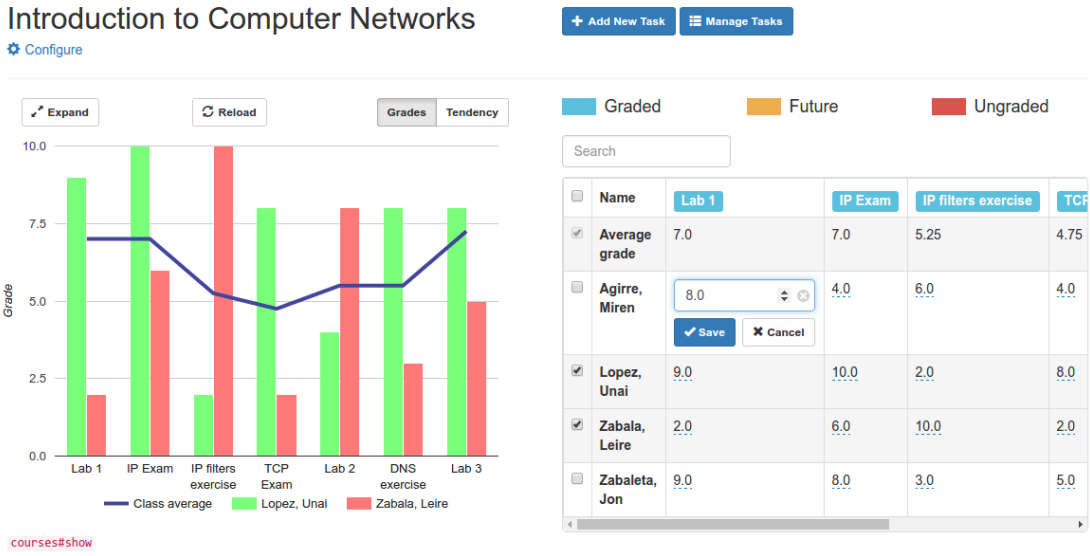
FASEAK	ZEREGINAK		AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
		n kasuak				
		Sekuentzia-diagramak	5	0	5	1
GUZTIRA:			80,5	18,5	70	

8.10. Taula: 10. Iterazioa

Bileran irakasleekin egingo diren probetan erabiliko diren erabiltzaile-historiak onartu dira, eta hurrengo iteraziorako aplikazioaren erabilgarritasuna hobetzeko aldaketak egingo direla erabaki da.

8.11. *11. ITERAZIOA* (2016/06/15 – 2016/06/22)

Iterazio honetan, gunearen erabilgarritasunerako hobekuntzak egin ditut: mugikorreko bista ahalik eta gehien hobetu, botoiei *glyphicon*-ak gehitu funtzioa ahalik eta argiena izateko, eta beste hainbat aldaketa itxuran. Irakaslearen grafikoetan, ikasleenetan bezala, zeregin bakoitzaren batezbesteko nota adierazten duen lerroa gehitu dut, eta taulan batezbesteko hori erakusten duen lerroa ere bai.



8.13. Irudia: Irakasgaiaren orrialdea hobekuntzekin

Irakasgai batean gaudela zereginen zerrenda ikusteko eta zereginak ezabatzeko aukera inplementatu dut, eta irakasgaiaren hasiera/amaiera data aldatzeko aukera ere bai. Ikasle eta irakasleek mezuak zein hizkuntzatan jaso nahi dituzten aukeratzeko aukera inplementatu dut, eta irakasgai batean orrialdera sartzean irakasgai horren hizkuntza aukeratzen da automatikoki.

FASEAK	ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Betekizunak	Funtzionalak	0	0	0	1
	Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna...)	0	0	0	1
Kudeaketa	Bilerak eta	3	0,5	2,5	1

FASEAK			ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
			aktak				
			Hasierako planifikazioa	0	0	0	1
Formakuntza	Teknikoa		Ruby on Rails	2	1	0	2
			Devise, Authority, Rolify	0	0	0	1
			Google Charts	0	0	0	1
			TDD (Ruby)	0	0	0	1
Garapena	Diseinua		Interfaze Grafikoen Prototipoak	0	0	0	1
	Inplementazioa		Datu basea / Klaseak	0	0	0	1
			Autentifikazioa	0	0	0	1
			Lokalizazioa	1	1	0	2
			Informazioa bistaratzea	0	2	0	2
			Zereginak sortzea	0	0	0	1
			Notak gehitzea	0	1	0	1
			Rspec probak inplementatzea	6	4	4	3
			Beste funtzionalitate batzuk	6	10	0	2
	Probak	Diseinu grafikoa eta analisia	Ikasle eta irakasleekin probak	5	0	5	1
		Funtzionalitateak	Ikasle eta irakasleekin probak	4	0	4	1
		Barne-probak	Sistema eragile eta nabigatzaile eta aparatu ezberdinekin probak egin	6	0	6	1
Dokumentazioa			Instalazio-gida	1	0	1	2
			Memoria	21	1	20	1

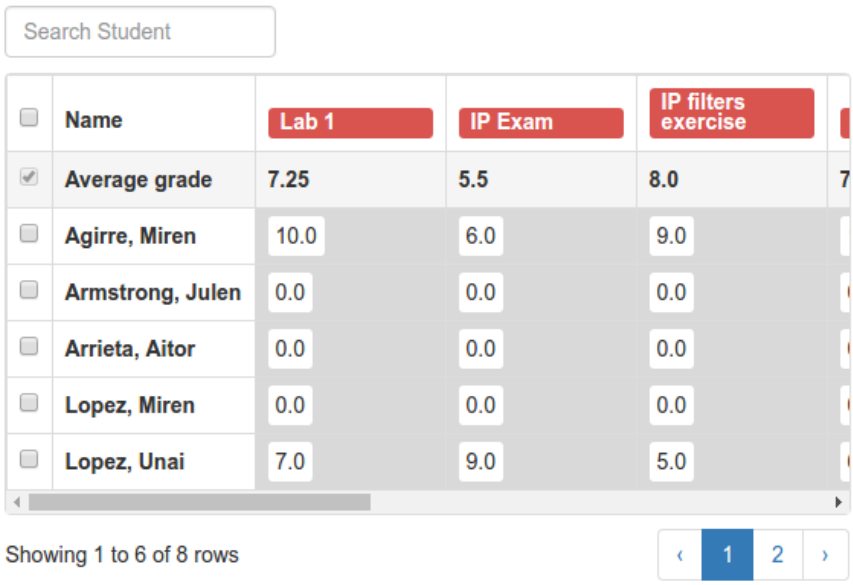
FASEAK	ZEREGINAK		AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
	UML Diagramak	Klase-diagrama	5	0	5	1
		Erabilpenkasuak	5	0	5	1
		Sekuentzia-diagramak	5	0	5	1
GUZTIRA:			70	20,5	57,5	

8.11. Taula: 11. Iterazioa

Bileran joeren kalkuluaren formula zein den argitu da, azkenik ez baitzen argi geratu. Hurrengo iterazioan erabilgarritasun-hobekuntza gehiago egingo dira, eta irakasleak zereginak editatzeko funtzionalitatea inplementatuko da. Gainera, dokumentazioan lana egingo da.

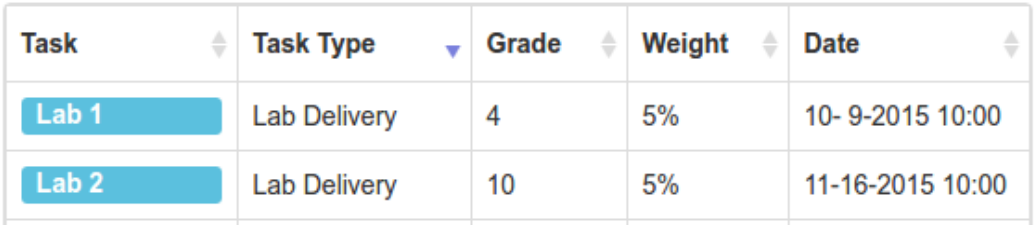
8.12. *12. ITERAZIOA* (2016/06/22 – 2016/06/29)

Iterazio honetan aplikazioaren erabilgarritasunerako hobekuntzak egin ditut. Mugikorrerako bertsioan, tauletako ezabatu/editatu estekak hasieran agertzen dira amaieran agertu ordez. Irakasle eta ikasleen tauletan orrialdetzea (*pagination*) inplementatu dut, orrialdearen layout-a ikasle/zeregin kopuruarekiko independentea izan dadin.



8.14. Irudia: Orrialdetzea

Irakaslearen taulan notak editagarriak direla argitzeko kolore-eskema aldatu dut eta nota berriaren formularioa popup bezala berrezarri dut, taularen barruan zabalduz gero taualren layout-a aldatzen zelako. Ikaslearen taulan zereginak atributuen balioen arabera berrordena daitezke (data, nota etab.).



8.15. Irudia: Taula berrordenatzea

Gainera, irakasleak zereginak editatzeko aukera inplementatu dut. Probak eguneratu ditut aldaketak islatzeko. Azkenik, Erabilpen Kasuak UML formatuan sortu ditut, eta github orrialdean oinarrizko instalazio gida sortu dut.

FASEAK			ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Betekizunak			Funtzionalak	0	0	0	1
			Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna...)	0	0	0	1
Kudeaketa			Bilerak eta aktak	2,5	0,5	2	1
			Hasierako planifikazioa	0	0	0	1
Formakuntza	Teknikoa		Ruby on Rails	0	0	0	1
			Devise, Authority, Rolify	0	0	0	1
			Google Charts	0	0	0	1
			TDD (Ruby)	0	0	0	1
Garapena	Diseinua		Interfaze Grafikoen Prototipoak	0	0	0	1
	Inplementazioa		Datu basea / Klaseak	0	0	0	1
			Autentifikazioa	0	0	0	1
			Lokalizazioa	0	0	0	1
			Informazioa bistaratzea	0	0	0	1
			Zereginak sortzea	0	0	0	1
			Notak gehitzea	0	0	0	1
			Rspec probak inplementatzea	4	3	1	2
			Beste funtzionalitate batzuk	0	6	0	2
	Probak	Diseinu grafikoa eta analisia	Ikasle eta irakasleekin probak	5	0	5	3
		Funtzionalitateak	Ikasle eta irakasleekin probak	4	0	4	3
		Barne-probak	Sistema eragile eta nabigatzaile eta aparatu	6	0	6	3

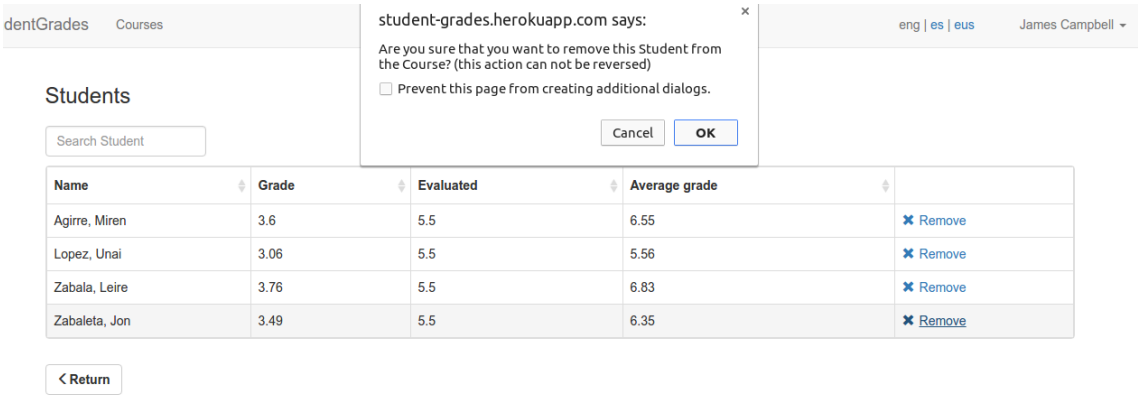
FASEAK	ZEREGINAK		AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
	ezberdinekin probak egin					
Dokumentazioa	Instalazio-gida		1	0,5	0,5	2
	Memoria		20	5	15	3
	UML Diagramak	Klase-diagrama	5	0	5	2
		Erabilpen kasuak	5	4	0	3
		Sekuentzia-diagramak	5	0	5	3
		GUZTIRA:	57,5	19	43,5	

8.12. Taula: 12. Iterazioa

Bileran erabaki da hurrengo bilera bi astetan izango dela, eta bitartean memorian lan egingo dela. Gainera, ikasleak ebaluazio jarraitutik kentzeko funtzionalitatea nola inplementatu pentsatzea proposatu da, nahiz eta ez zaion lehentasun handia emango.

8.13. *13. ITERAZIOA* (2016/06/29 – 2016/07/13)

Iterazio honetan aurreko iterazioan proposatutako ikasleak kentzeko funtzionalitatea inplementatu dut eta memorian lan egin dut.



8.16. Irudia: Irakasgaitik ikasle bat kentzeko aukera

FASEAK		ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Betekizunak		Funtzionalak	0	0	0	1
		Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna...)	0	0	0	1
Kudeaketa		Bilerak eta aktak	2	1	0	1
		Hasierako planifikazioa	0	0	0	1
Formakuntza	Teknikoa	Ruby on Rails	0	0	0	1
		Devise, Authority, Rolify	0	0	0	1
		Google Charts	0	0	0	1
		TDD (Ruby)	0	0	0	1
Garapena	Diseinua	Interfaze Grafikoen Prototipoak	0	0	0	1
	Inplementazioa	Datu basea / Klaseak	0	0	0	1
		Autentifikazioa	0	0	0	1

			Lokalizazioa	0	0	0	1
			Informazioa bistaratzea	0	0	0	1
			Zereginak sortzea	0	0	0	1
			Notak gehitzea	0	0	0	1
			Rspec probak inplementatzea	1	0,5	0,5	2
			Beste funtzionalitate batzuk	0	6	0	2
	Probak	Diseinu grafikoa eta analisia	Ikasle eta irakasleekin probak	5	0	5	3
		Funtzionalitateak	Ikasle eta irakasleekin probak	4	2	2	3
		Barne-probak	Sistema eragile eta nabigatzaile eta aparatu ezberdinekin probak egin	6	0	6	3
Dokumentazioa			Instalazio-gida	0,5	0	0	1
			Memoria	15	5	10	3
			UML Diagramak: Klase-diagrama	5	5	0	1
			UML Diagramak: Erabilpen kasuak	0	0	0	1
			UML Diagramak: Sekuentzia-diagramak	5	0	5	3
			GUZTIRA:	43,5	19,5	28,5	

8.13. Taula: 13. Iterazioa

Bileran erabaki da hurrengo iterazioan irakasleekin egindako probetan proposatutako hobekuntzak inplementatzea, proben emaitzak errepasatu ondoren. Azkenik, memoria amaituko da.

8.14. *14. ITERAZIOA* (2016/07/13 – 2016/07/25)

Iterazio honetan memoria amaitu dut.

FASEAK		ZEREGINAK	AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
Betekizunak		Funtzionalak	0	0	0	1
		Ez-funtzionalak (Erabilgarritasuna, segurtasuna, pribatutasuna...)	0	0	0	1
Kudeaketa		Bilerak eta aktak	0	0	0	1
		Hasierako planifikazioa	0	0	0	1
Formakuntza	Teknikoa	Ruby on Rails	0	0	0	1
		Devise, Authority, Rolify	0	0	0	1
		Google Charts	0	0	0	1
		TDD (Ruby)	0	0	0	1
Garapena	Diseinua	Interfaze Grafikoen Prototipoak	0	0	0	1
	Inplementazioa	Datu basea / Klaseak	0	0	0	1
		Autentifikazioa	0	0	0	1
		Lokalizazioa	0	0	0	1
		Informazioa bistaratzea	0	0	0	1
		Zereginak sortzea	0	0	0	1
		Notak gehitzea	0	0	0	1
		Rspec probak inplementatzea	0,5	0	0	2
		Beste funtzionalitate batzuk	0	0	0	2
	P Diseinu	Ikasle eta	5	0	5	3

FASEAK			ZEREGINAK		AURREKO ESTIMAZIOA	EMANDAKO ORDUAK	ESTIMAZIO BERRIA	LEHENTASUNA
		grafikoa eta analisia	irakasleekin probak					
	probak	Funtzionalitateak	Ikasle eta irakasleekin probak		2	0	2	3
		Barne-probak	Sistema eragile eta nabigatzaile eta aparatu ezberdinekin probak egin		6	1	5	3
Dokumentazioa			Instalazio-gida		0	0	0	1
			Memoria		10	10	0	3
			UML Diagramak	Klase-diagrama	0	0	0	1
				Erabilpen kasuak	0	0	0	1
				Sekuentzia-diagramak	5	0	5	3
				GUZTIRA:	28,5	11	17	

8.14. Taula: 14. Iterazioa

Iterazio hau azkena izango dela erabaki dugu. Izan ere, hasieran planifikatutako 300 orduak gainditu dira dagoeneko (336 ordu eman dira) eta lortutako produktuaren kalitatea helburuko kalitatetik gertu dagoela iruditu zaigu.

Azken taulan ikus daitekeenez, probentzat estimatutako 12 ordu eta sekuentzia-daiagramak sortzeko estimatutako 5 ordu eman gabe geratu dira. Probei dagokienez, estimazioak handiegiak izan direlako izan da, eta sekuentzia-diagramei dagokienez ez zaigu iruditu suposatzen duten lanak pena merezi duenik, aplikazioaren funtzionamendua ulertzen laguntzen duten probak inplementatu ditugula kontuan hartuz.

9

Ondorioak

Atal honetan proiektuan aurrera eraman ondoren ateratako ondorioak azalduko ditugu.

9.1. Azken produktua

Azken iterazioa amaiturik, lortutako produktua ebaluatzeko egoeran gaude. Esango nuke azken emaitza hasieran espero nuena baino hobea izan dela. Framework eta lengoaia berri bat ikasi behar izan dudanez, hasieran emaitzak lortzea zaila izan zen, baina behin erraztasuna hartutakoan gauza interesgarriak egitea posible izan da. Hasierako betekizunak betetzeaz gain, hasieran planifikatu gabeko hainbat funtzionalitate inplementatzeko beharra ikusi ahala produktuaren konplexutasuna haziz joan da. Probei esker (bai software probak eta bai benetako erabiltzaileekin egindakoak) produktuaren kalitatearen inguruko ziurtasun handiagoa lortu dut.

9.2. Hobekuntzak

Aplikazioa hobetzen jarraitzeko asmoa izanez gero, ondorengo funtzionalitateetan lan egingo nuke:

- Notaz gain zereginetan feedback zehatzagoa jasotzeko sistema sortu (oharrak...).
- Irakasleak notak instituzio akademikoak zehazten duen tartea baino lehen jartzen ez baditu mezu bat jasotzeko funtzionalitatea.
- Erabiltzaileen profil konplexuagoak sortu.

9.3. Ikasitako lezioak

9.3.1. Tamaina honetako proiektu bat kudeatzea

Hainbeste orduko lana eskatzen duen proiektu bat aurrera eraman dudan lehen aldia izan da hau. Hala ere, erabilitako metodologiak aplikazioa pixkanaka eraikitzea dakarrenez askotan proiektu handietan izaten den frustrazioa ekiditea ekarri duela esango nuke. Gainera, iterazio amaierako bilerei esker aplikazioa etengabe hobetzeko motibazioa lortzen da, "bezeroa"-rekin harreman estua mantentzen delako.

Hala ere, pertsona bakarreko taldea eta interesatu bakarra izateak benetako egoera profesional baten antz gutxi duela esango nuke, izan ere gaur egun talde-lana ia derrigorrezkoa da proiektu serio bat aurrera eramateko, eta honek kudeaketaren konplexutasuna asko handitzen du.

9.3.2. Erabiltzaileekin egindako proben garrantzia

Aplikazio erabilgarri bat lortzeko modurik onena erabiltzaileekin probak egitea dela ondorioztatu dut. Izan ere, diseinu perfektua dugula pentsatu arren, benetako erabiltzaile batek beti izaten du behar bezain ondo ulertzen ez duen zerbait, edo proposatzeko hobekuntzaren bat. Ikuspuntu

ezberdinak bilduta, ahalik eta erabiltzailei gehienei ulergarri egingo zaien diseinu bat lortzen da.

9.3.3. Software probak

Nahiz eta software probak irakasgairen batean aipatu izan ditugun, proiektu honekin hasi arte ez ditut inoiz benetako probak idatzi. Are gehiago, zertarako balio zuten ere ondo ulertzen ez nuela esango nuke. Rails-ek probak idazteko eskaintzen dituen aukera erraz eta ulergarriei esker (Rspec, Capybara, FactoryGirl) proba hauei nien beldurra galdu diedala esango nuke, eta aplikazioak espero dugun funtzionamendua betetzen duela ziurtatzeko zein modu egokia diren ulertu dut.

9.3.4. Ruby on Rails eta Ruby

Framework eta lengoaia hauen ezagutza ona lortu dudala esango nuke. Behin frameworkaren funtzionamendua barneratzea lortuta, aplikazio berri bate denbora txikian garatzeko gaitasuna dudala esango nuke. Gainera, Rails-en filosofia jarraituz gero kode eraginkor eta ulergarriagoa idazten ikasi dudala esango nuke.

Ruby-ri dagokionez, bere ezaugarri bereziak ulertu ondoren nire lengoaiarik gustukoena bihurtu da, nahiz eta ingurune profesional batean erabiltzeko aukerarik izango dudan ez dakidan, ez baita antzeko beste lengoaia batzuk (bereziki Python) bezain zabaldua.

9.3.5. *Frontend* garapena

Bootstrap erabiltzen ikasita, eta interfaze erabilgarri bat lortzean jarri dugun enfasiari esker, nire *frontend* garapenerako gaitasuna asko hobetu dela esango nuke. Gainera, Google Charts erabiliz grafiko konplexuak egiteko gai naiz orain.

9.4. Merkaturatze-perspektibak

Aplikazio honen antzeko alternatiben konplexutasuna eta zabalkundea ikusirik, esango nuke ez dela oraindik benetako egoera batean erabiliko nukeen aplikazio bat. Hala ere, datorren ikasturtean proiektuko irakasleak bere irakasgairen batean erabiliko duela adierazi didanez, baliteke lortutako emaitzak onak badira etorkizunean aplikazioa garatzen jarraitzea.

Bibliografia

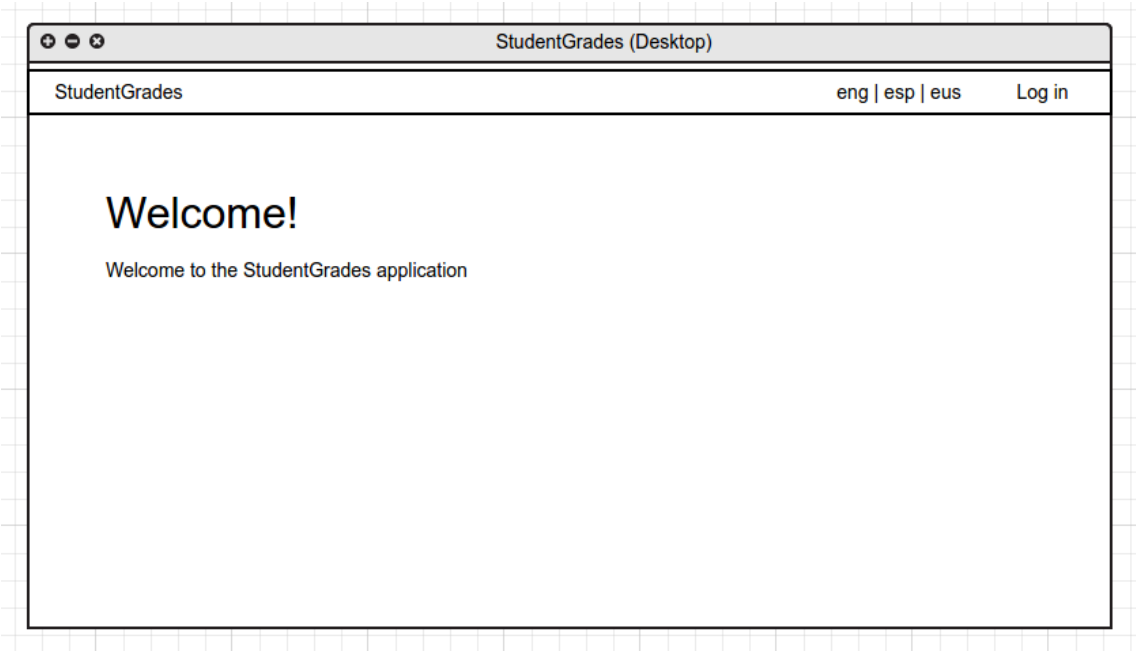
- Sandi Metz. 2013. *Practical Object-Oriented Design in Ruby*. Addison-Wesley.
- Russ Olsen. 2011. *Eloquent Ruby*. Addison-Wesley.
- Douglas Crockford. 2008. *JavaScript: The Good Parts.* O'Reilly.
- Michael Hartl. 2015. *Ruby on Rails Tutorial.* Addison-Wesley.
- Noel Rappin. 2014. *Rails 4 Test Prescriptions*. The Pragmatic Programmers.

A Eranskina

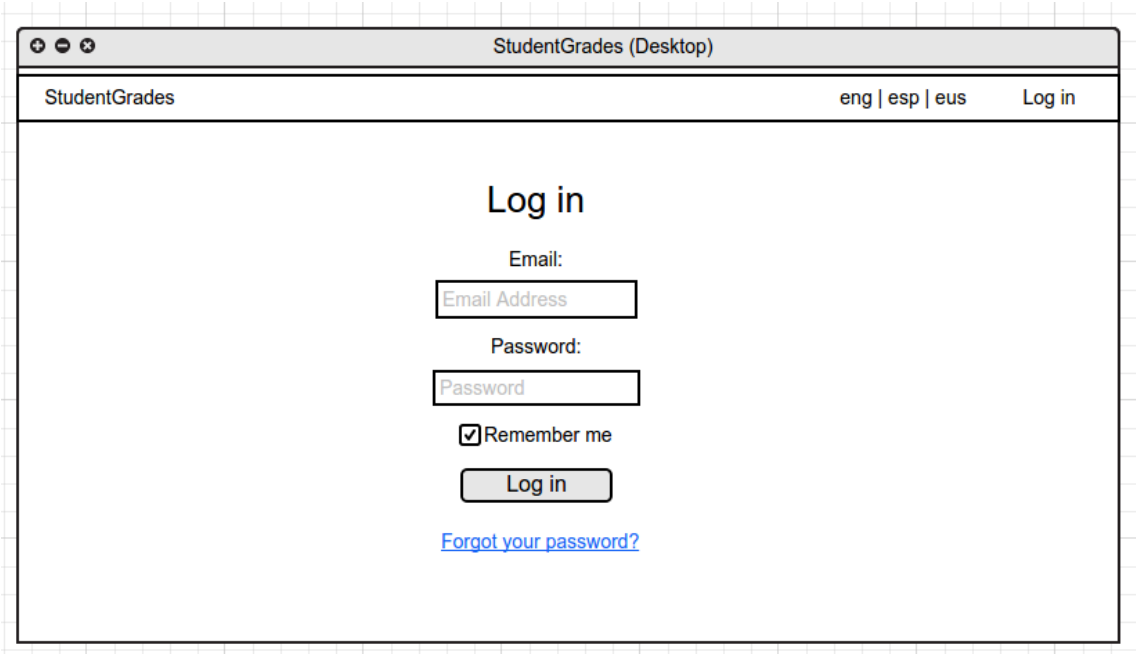
Interfazeen Prototipoak

Eranskin honetan aplikazioa garatzen hasi aurretik hasierako ideia bezala sortutako prootipoak erakusten dira. B Eranskineko probetan ikasleek erabili zituzten prototipo hauek, paperean inprimaturik.

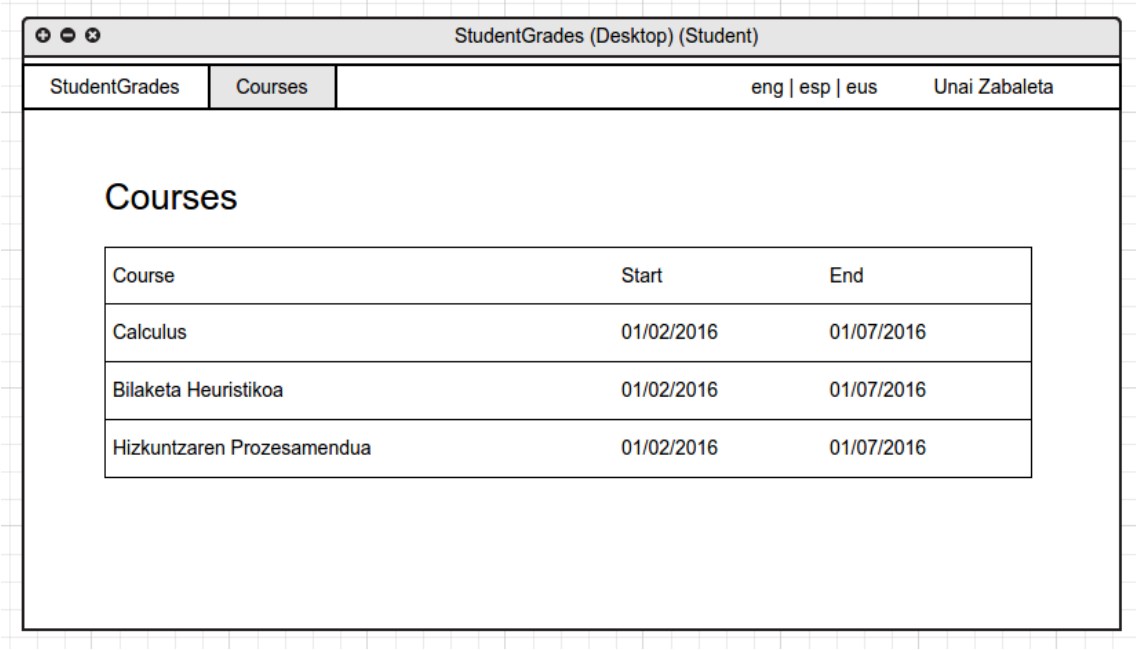
A.1 PROTOTIPOAK



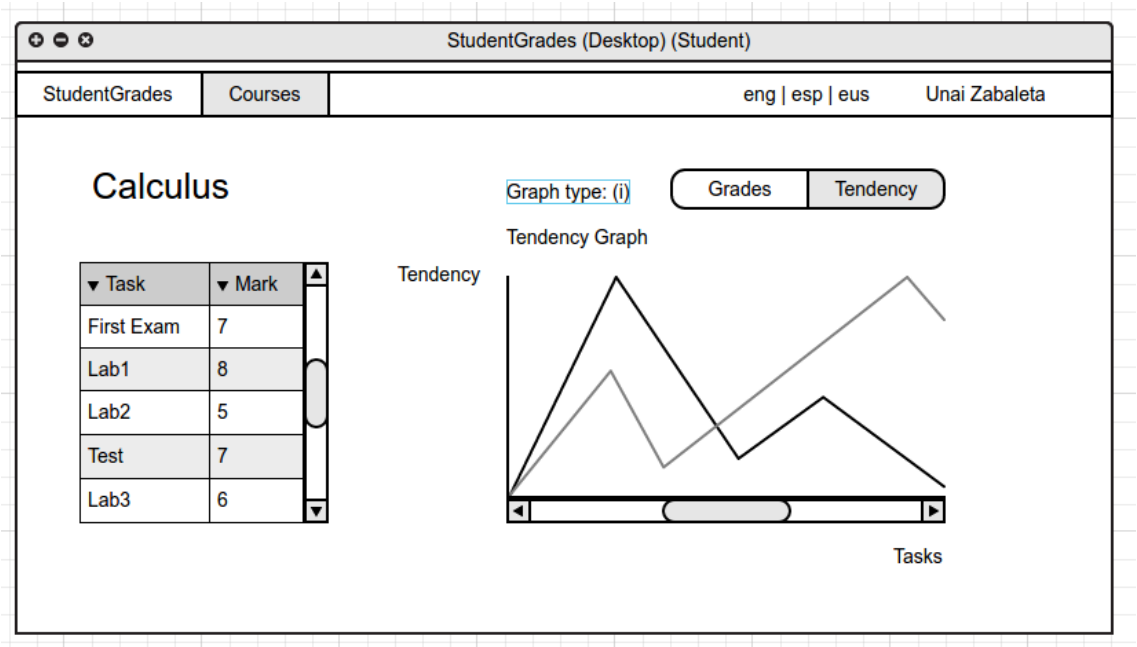
A.1. Irudia: Hasierako orrialdea



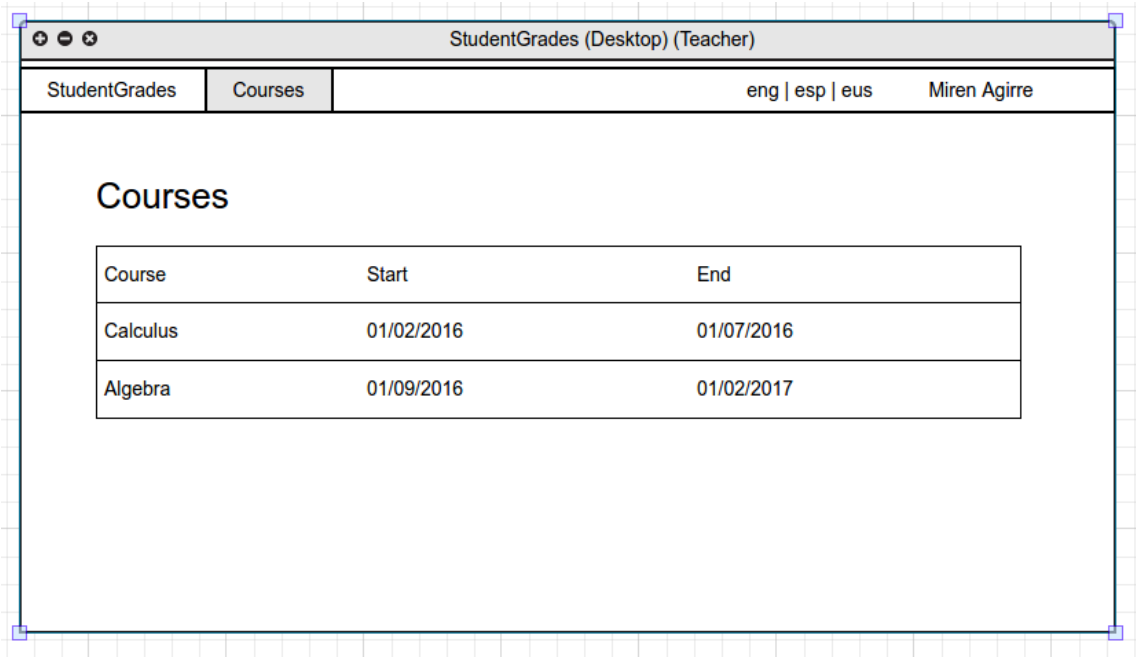
A.2. Irudia: Login orrialdea



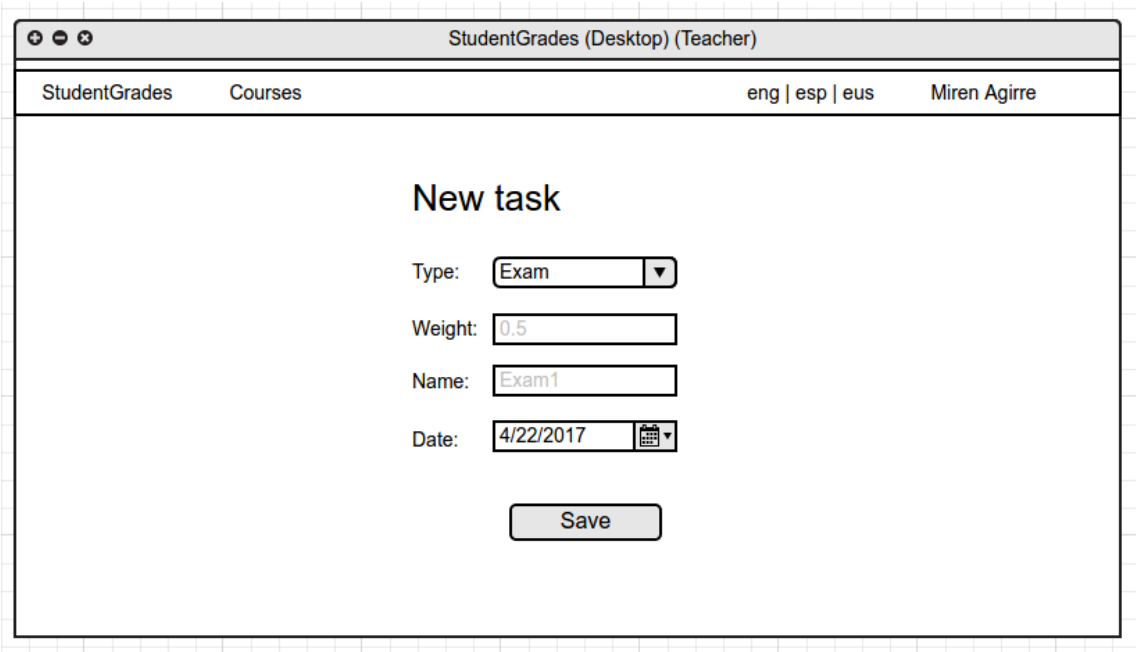
A.3. Irudia: Ikaslearen irakasgaien zerrenda



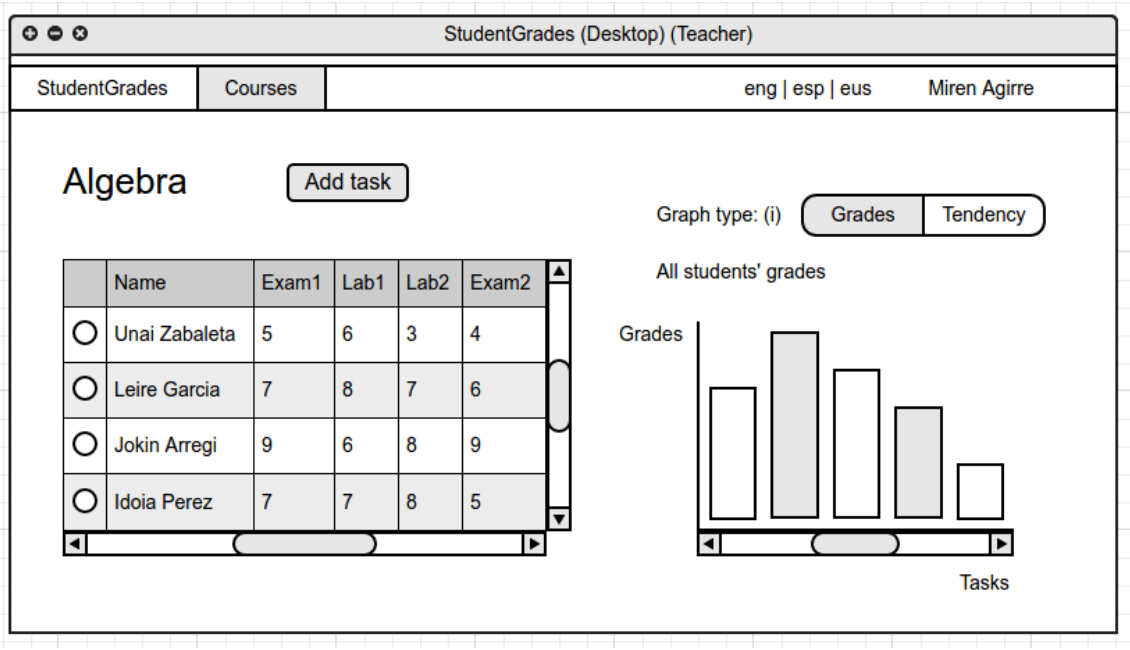
A.4. Irudia: Ikaslearen irakasgaiaren orrialdea



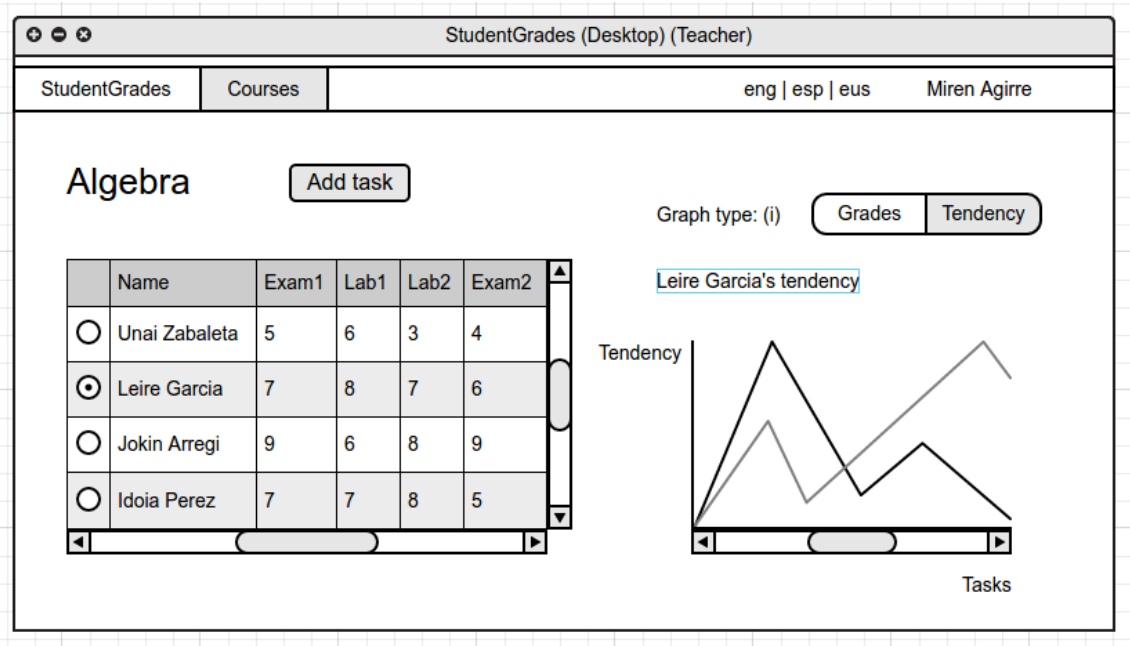
A.5. Irudia: Irakaslearen irakasgaien zerrenda



A.6. Irudia: Irakasleak irakasgaian zeregin berri bat sortzeko formularioa



A.7. Irudia: Irakaslearen irakasgaiaren orrialdea (notak)



A.8. Irudia: Irudia: Irakaslearen irakasgaiaren orrialdea (joerak)

B Eranskina

Erabiltzaileekin egindako proben emaitzak

Eranskin honetan Erabiltzaileekin egin diren ebaluazioak azalduko dira.

Erabiltzaileekin egindako probak beharrezkoak dira aplikazioaren erabilgarritasun egokia ziurtatzeko (betekizun ez-funtzionala). Irismenean (2.1 Atala) aipatu bezala, tresna hau hiru erabiltzaile motak erabiliko dute, eta erabiltzaile mota hauetara zuzendu dira egindako erabilgarritasun-probak:

- Ikasleak: Aplikazioaren erabiltzaile nagusiak, beraien notaren joera ezagutzeko erabiliko dute (bereziki mugikorra erabiliz).
- Irakasleak: Bi unetan erabiliko dute aplikazioa: kurtso hasieran, joera mota konfiguratzeko, eta kurtsoan zehar zereginak sortu eta notak jartzeko (bereziki ordenagailua erabiliz).
- Administratzaileak: Kurtsoa hasi aurretik erabiliko dute aplikazioa, irakasgaiak sortu eta irakasle eta ikasleak ezartzeko.

B.1 EBALUAZIO-TEKNIKA

Proiektu honetako hobekuntza jarraituko metodologia arina jarraituz, zehaztasun maila ezberdineko prototipoak erabiliko dira aplikazioaren garapenaren une ezberdinetan, erabiltzaileak diseinua balidatzeko beharra ikusten denean.

Ibilbide Kognitibo eta Thinking Aloud tekniken konbinazioa erabiliko da, sakontze-teknikez gain [Jakob Nielsen. 1993. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA]. Erabiltzaileek interfaze-proposamen bat ebaluatuko dute zeregin baten (edo gehiagoren) testuinguruan. Amaitzean, ebaluatzaileak aplikazioarekin elkarrekintzan izan duten esperientziaren inguruan egingo dizkien galderak erantzungo dituzte.

Lehen prototipoak bakunak izango dira, proiektuaren hasierako ideia jasoz. Garapenean aurrera egin ahala, prototipoak martxan dagoen aplikazioaren interfazeak izango dira, etengabe hobetuz joango direnak.

B.2 EBALUAZIOAK

B.2.1. IKASLEEKIN

Aplikazioaren garapena hasi aurretik, interfazeen prototipoak sortu genituen, lehen ideia bat osatu eta erabiltzaileen feedback-a jasotzeko (prototipo hauek A Eranskinean daude eskuragarri, eta paperean inprimatu ziren).

Hautatutako zereginak

1. Login egin.

2. (Irakasle bezala) Algebra irakasgaian “Exam3” azterketa sortu eta Jokin Arregi ikasleari nota gehitu azterketa berri horretan.

3. (Irakasle bezala) Leire García ikaslearen notak ikusi Algebra irakasgaian.

4. Bilaketa Heuristikoa irakasgaian zure notak ikusi.

Ikus daitekeenez, ikasleen zereginak oso sinpleak direnez (login egin eta notak ikusi) hasierako bertsio honetan, irakasleek egingo lituzketen bi zereginen prototipoak ere ebaluatzeko erabaki hartu da, irakasle baten lekuan jarriz.

Emaitzak

Lehen erabiltzailea

1. Arazorik ez.

2. Erabiltzaileak uler lezake zeregin berriaren “name” eremua ikasle baten izena dela (ez dago argi zeregin bat ez dela ikasle jakin batentzat, ikasle guztientzat baizik).

3. Taulan radiobutton-en header-a “grafikoa erakutsi” izan behar litzateke radiobutton-ek zer egiten duten argi gera dadin.

4. Taulan zeregin bakoitzaren notarekin batera zeregin horren pisua ikusi behar litzateke.

Bigarren erabiltzailea

1. Hobe litzateke login formularioa nabigatzaile barran erakustea.

2. Zeregin berria sortzeko formularioan “pisua” zer den ez dago argi, hoble litzateke testu-eremuaren aurretik '%' sinboloa erakustea ehuneko bat dela argi gera dadin. Taulan zereginen pisua ere ikusi behar litzateke, eta radiobutton-en ordez checkboxak erabiltzea hobea litzateke, horrela hainabt ikasleren notak konparatzeko. Notak taulan zuzenean sartu ordez, “Notak gehitu” botoia gehitu “Zeregin berria” botoiaren ondoan, eta orrialde berri batean eman taula editatzeko aukera, modu honetan informazioa gehitzea eta informazioa bistaratzea banatuz.

3. Checkbox bati ematean lehenik noten barra-grafikoa erakutsi, ez tendentzia.

4. Taulan zeregin bakoitzaren notarekin batera zeregin horren pisua ikusi behar litzateke. Zereginak motaren arabera eta ondoren kronologikoki edo zuzenean kronologikoki (berrienak lehenik) ordenatu nahi diren hautatzeko aukera izan behar genuke. Motaren arabera elkartuz gero mota horretako zereginen batezbestekonota eta pisua ikusteko aukera izan behar genuke.

Hirugarren erabiltzailea

1. Hasierako orria login orria izan behar litzateke.

2. Taulan notak gehitzeko edo editatzeko botoi bat egon behar litzateke, segurtasuna izateko botoiari ematean bakarrik gordeko direla taulan egin ditugun aldaketak.

3. Ikasle asko daudenean taulan scroll egin beharko denez, taula erakutsi lerro batean eta grafikoa hurrengoan.

4. Arazorik ez

Laugarren erabiltzailea

1. Hasierako orrian aplikazioaren inguruko informazio gehiago erakutsi: taulen eta grafikoen irudiak adibidez. Login egiterakoan, Google-ek egiten duen bezala lehenik posta sartu eta balidatu eta ondoren pasahitza.

2. “pisua” zer den ez da argi ulertzen: azalpen bat gehitu tooltip bezala. Zereginaren izenaren eremua pisuaren eramuaren aurretik jarri. Taulan datuak sartu edo aldatutakoan gordetzeko botoia gehitu.

3. Ikasle guztien informazioa ikustean tendentziak ez du zentzurik, ez eman aukera hori. Radiobutton-en ordez checkbox-ak erabili, hainbat ikasleren datuak konparatzeko. Grafikoko barra bati klik ematean ikasle horren checkbox-a aukeratu eta ondorioz bere zereginen noten informazioa erakutsi. Ikasle bat aukeratzean, grafikoan duen kolore bera ezarri taulako lerroari. Ikasleak Abizena, Izena kontuan hartuta ordenatu taulan, errazago bilatu ahal izateko. Taulan ikasleak bilatzeko testu-eremu bat gehitu.

4. Zereginen data erakutsi. Irakasgaiaren orrialdean irakaslearen informazioa erakutsi (izen-abizenak, emaila), eta irakasgaien zerrendan ere bai. Egutegi bat gehitu irakasgaiaren orrialdean, koloreak zereginaren mota erakusten duelarik, eta tooltip bezala zereginaren izena eta nota (irakaslearen kasuan ere berdina egin, baina nota erakutsi gabe.

Ondorioak eta ebaluazioaren ondoren egindako hobekuntzak

Erabiltzaileek prototipoen inguruan duten iritzia kontuan hartuko da benetako interfaze grafikoa garatzerako orduan, batez ere behin baino gehiagotan aipatu diren arazoak landuko dira (pisua zer den argitu, taulan zereginen nota erakutsi...).

B.2.2. ADMINISTRATZAILEEKIN

Zoritxarrez ez dugu mota honetako lan bat egiten duen erabiltzaile batekin probak egiteko aukerarik izan. Hala ere, funtzionalitate hau irakasleena baino askoz ere sinpleagoa denez hain garrantzitsua ez dela esango nuke.

B.2.3. IRAKASLEEKIN

Hiru irakasleko talde bat hautatu da, produktuaren fase ezberdinetan produktuaren ebaluazioak egiteko. Irakasle hauek ebaluazio jarraituan adituak dira, eta euretako bik gainera erabilgarritasunaren inguruko adituak ere badira. Ebaluazioa 12. iterazioan egin da (proiektuaren amaiera aldera).

Hautatutako zereginak

1. Cálculo irakasgaian taldearen batezbesteko nota ikusi

2. Cálculo irakasgaian taldearen joera ikusi

3. Unai Lópezek Cálculo irakasgaian dituen notak ikusi

4. Unai Lópezek Cálculo irakasgaian duen joera ikusi

5. Cálculo irakasgaian egindako test bat sortu eta notak jarri (pisua 0.75 izango da).

6. *ExamenDerivadas* zeregina ezabatu

Emaitzak

1. Zeregina: Arazoak eman ditu, ez da ondo ulertzen/ikusten taldeko batezbesteko nota. Ikasle bakoitzaren batezbestekoa da, ala talde osoarena?

2. Zeregina: Arazo lexikoa: Joera eta eboluzioa gauza bera al dira? Taldearen joeraren lerroa ez da besteengandik bereizten.

Ebaluazio formatiboko formula ez da ulertzen, ez da argi geratzen.

3. Zeregina: Ez da arazorik izan.

4. Zeregina: Zalantza: zein ehuneko izan da dagoeneko ebaluatua?

5. Zeregina: Zalantza: 6.0 ala 6 jarri behar da? Nota berria idatzi aurretik aurrekoa ezabatu behar al da? Zuzenean idatziz gero gaizki geratzen da (adibidez, 03).

Nota bat hutsik utz al daiteke, ala 0 bat jarri behar al da oraindik nota jarri ez bada?

Zereginaren pisuak dudak sortzen ditu. 0.05 %5 al da?

6. Zeregina: Ez da arazorik izan.

Sakontze paraleloko tekniken emaitzak

1. Hasierako pantailaren ulergarritasunaren inguruko zalantzak ditugu. Beraz, ebaluazioa hasi aurretik erabiltzaileari ondokoa galdetu diogu:

Zein da hasierako pantailan agertzen diren elementuen esanahia? Zertarako balio dute?

Elementu mugikor batek (*slideshow*) aplikazioaren pantailakadak erakusten ditu, informazio modura (ez funtzionala). Esanahiaren ulergarritasunaren emaitzak onak izan dira.

2. Aurreko iterazio batean irakasgai baten ebaluazio mota (formatibo edo akumulatiboa) zein den erraz identifikatzeko zailtasuna detektatu zen. Arazo hori zuzendu ondoren, erabiltzaileei galdetu zaie:

Zer motatako ebaluazioa du irakasgai honek?

Erantzunak zuzenak izan dira.

Ondorioak eta ebaluazioaren ondoren egindako hobekuntzak

Erabiltzaileekin egindako ebaluazioetan, detektatutako zalantzen aurrean hobekuntzak proposatu eta baloratu dira. Azkenik, hobekuntza hauek egin dira:

- Irakasgai bateko taulan, zereginak egindako ordenan agertuko dira (zaharrenak lehendabizi, berrienak azkenik).
- Ebaluazio akumulatiboa hautatzean (irakasgai baten konfiguratzio orrialdean), aurreko noten tasa gris argiz jarri (*disabled*).
- Taularen azpian orrialdetzea egitean agertzen den "Showing 1 to X of X rows" mezua kenduko da.
- Taldearen batezbesteko notaren eta joeraren lerroak beste lerroak baino deigarriagoak izan behar dira, lerro hauen itxura aldatu beharko da.

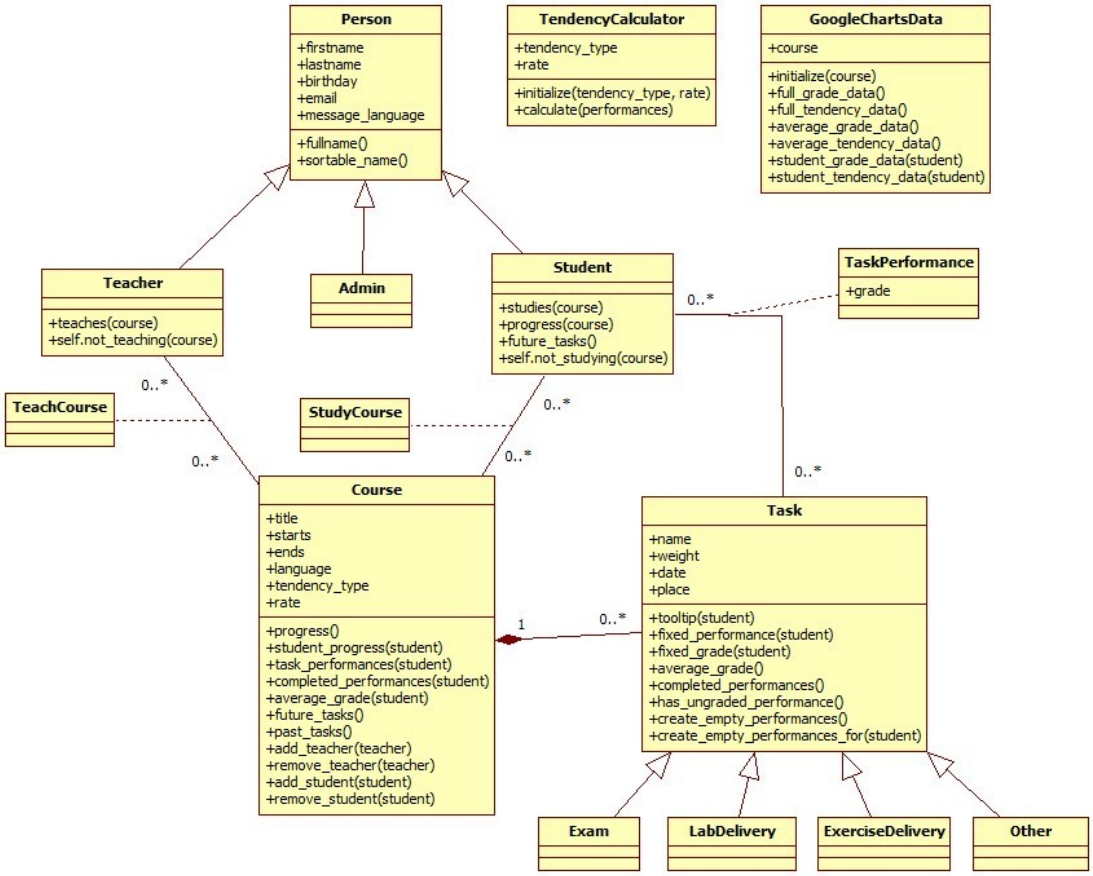
- Zeregina editatzeko orrialdean, Tasa ordez Aurreko noten tasa dela argituko da.
- Unerarte ebaluatu den ehunekoa argi ikusteko lerroa erakutsiko da grafikoan.

C Eranskina

UML Diagramak

Eranskin honetan Aplikazioaren UML diagramak erakutsiko dira.

C.1. KLASE DIAGRAMA



C.1. Irudia: Klase diagrama

Irudiak jarraitzen duen notazioa ondokoa da:

- Gezi txuri hutsek herentzia adierazten dute, objektuei orientatutako programazioan duen esanahiarekin. Beraz, Teacher-ek Person-ekin lotzen duen mota honetako gezi bat badu Person Teacher-en klase gurasoa dela esan nahi du.

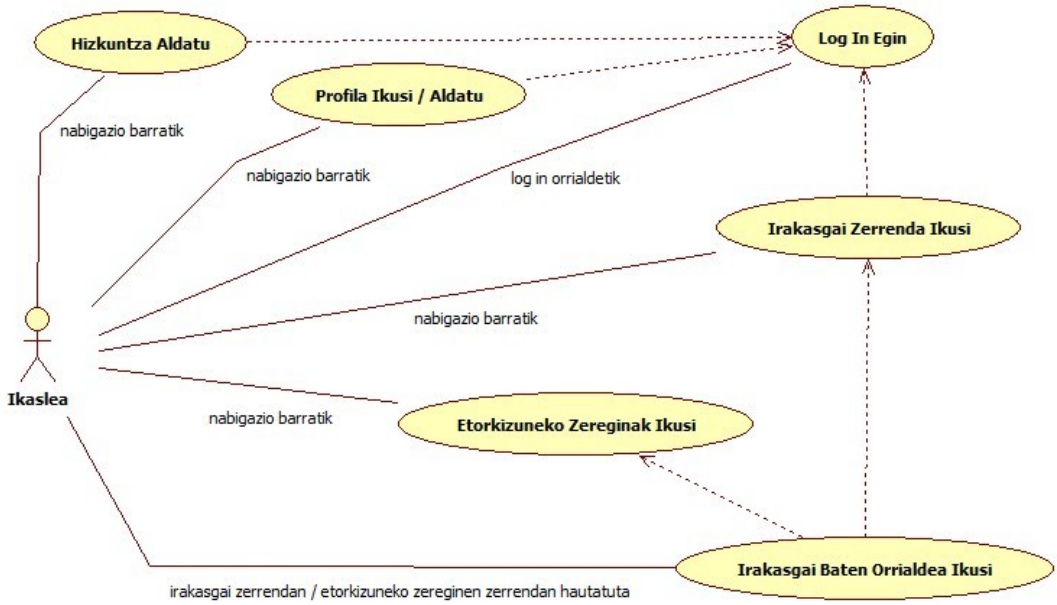
- Bi aldeetan zenbakiak dituzten lerro arruntek asoziazioa adierazten dute. Adibidez, Teacher eta Course lotzen dituen lerroak Teacher-ek hainbat Course dituela adierazten du (kasu honetan, 0..*-k adierazten duenez, 0 edo gehiago), eta Course- era berean hainbat Teacher dituela (aurreko kasuan bezala, 0 edo gehiago).

- Errondo beltz bat duten lerroek konposizioa adierazten dute. Hau da, klase bat beste baten parte izanik baino ezin dela existitu. Gure kasuan, mota honetako lerro batek lotzen ditu Task eta Course. Izan ere, irakasgai baten testuinguruan ez bada zeregin bat ezin da existitu.

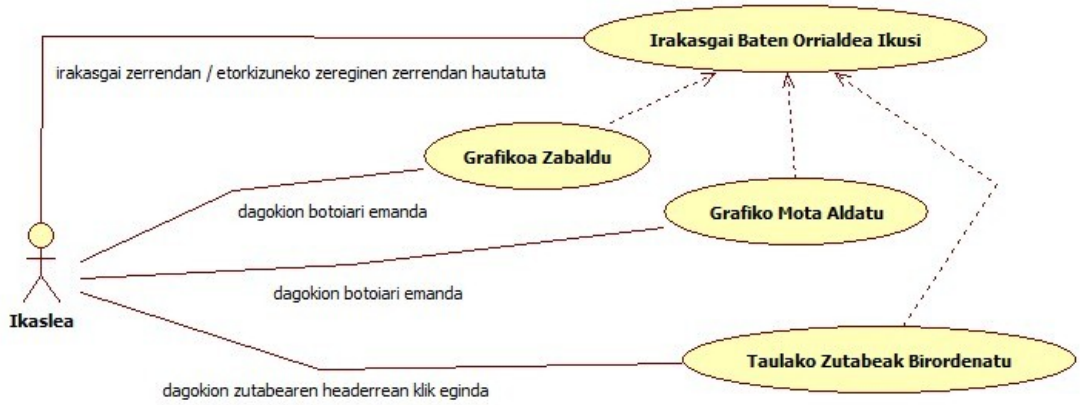
- Azkenik, marra etendunek asoziazio-klase bat adierazten dute. Asoziazio-klase bat beste bi klaseren arteko asoziazio baten inguruko informazioa gordetzen duen klase bat da. Gure adibidean, TaskPerformance klaseak Student baten eta Task baten asoziazioaren inguruko informazioa gordetzen du, hau da, zeregin horretan ikasleak zein nota izan duen.

C.2. ERABILPEN KASUAK

C.2.1. IKASLEEN KASUAK

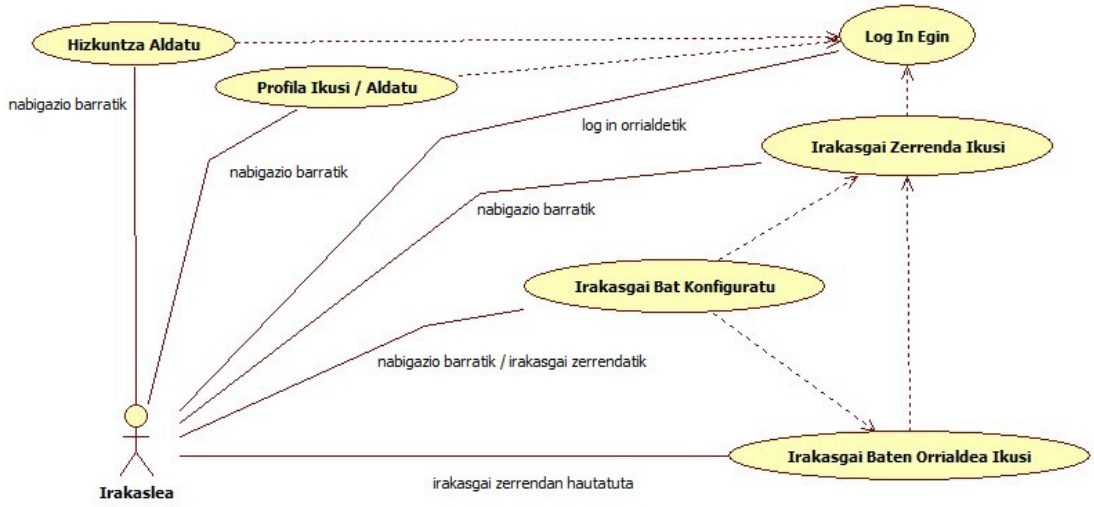


C.2. Irudia: Ikaslearen kasuak 1

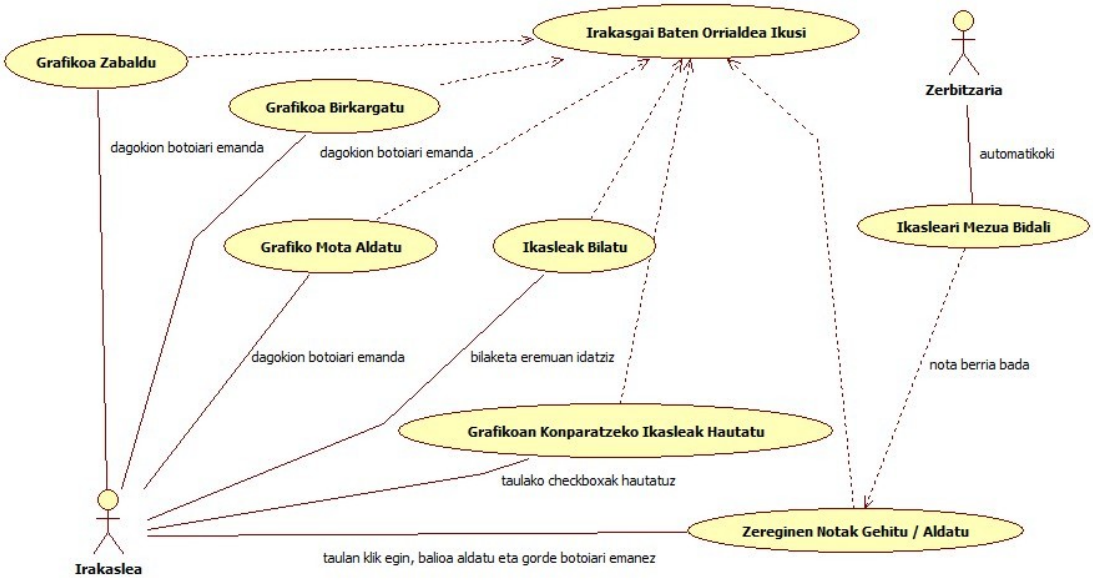


C.3. Irudia: Ikaslearen kasuak 2

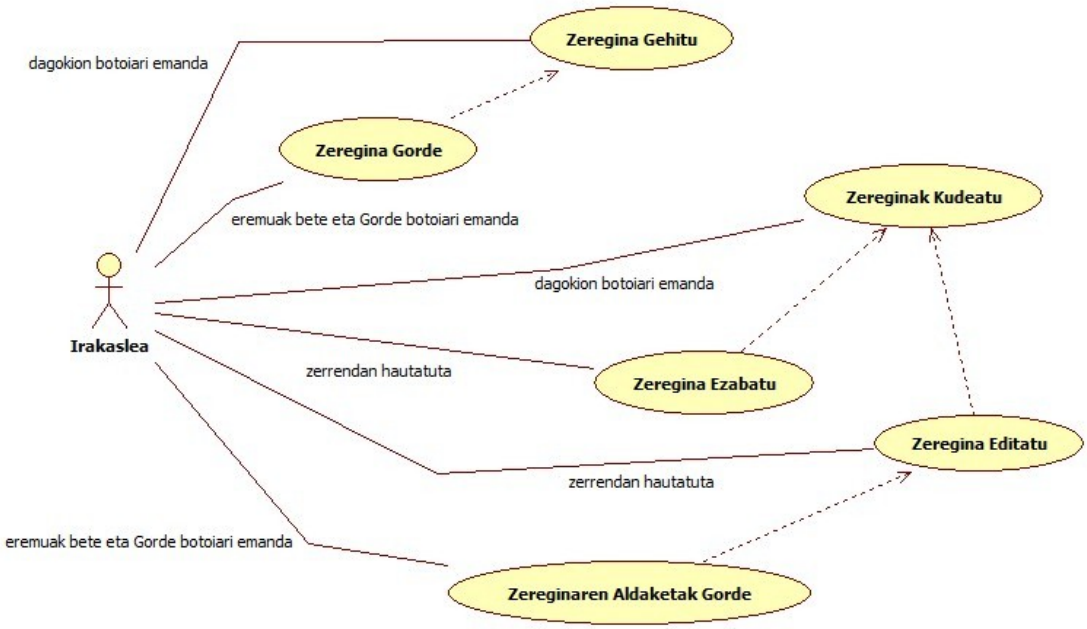
C.2.2. IRAKASLEEN KASUAK



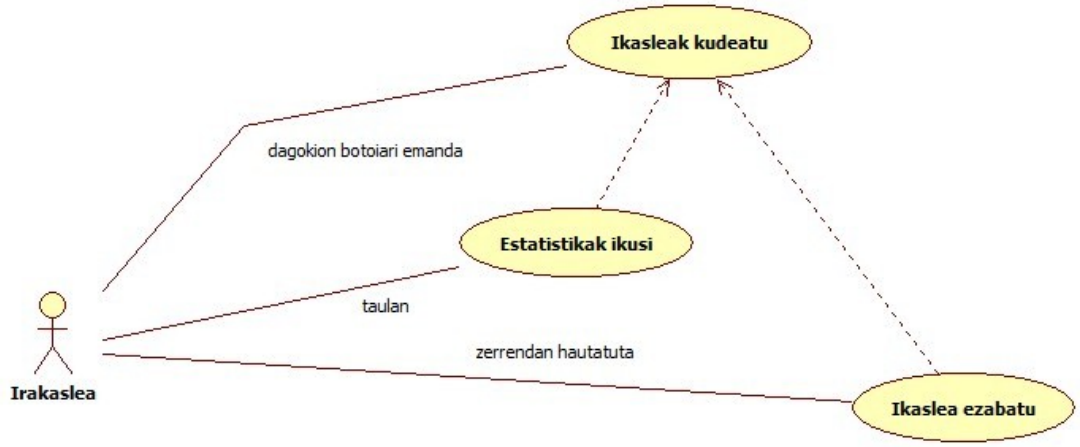
C.4. Irudia: Irakaslearen kasuak 1



C.5. Irudia: Irakaslearen kasuak 2

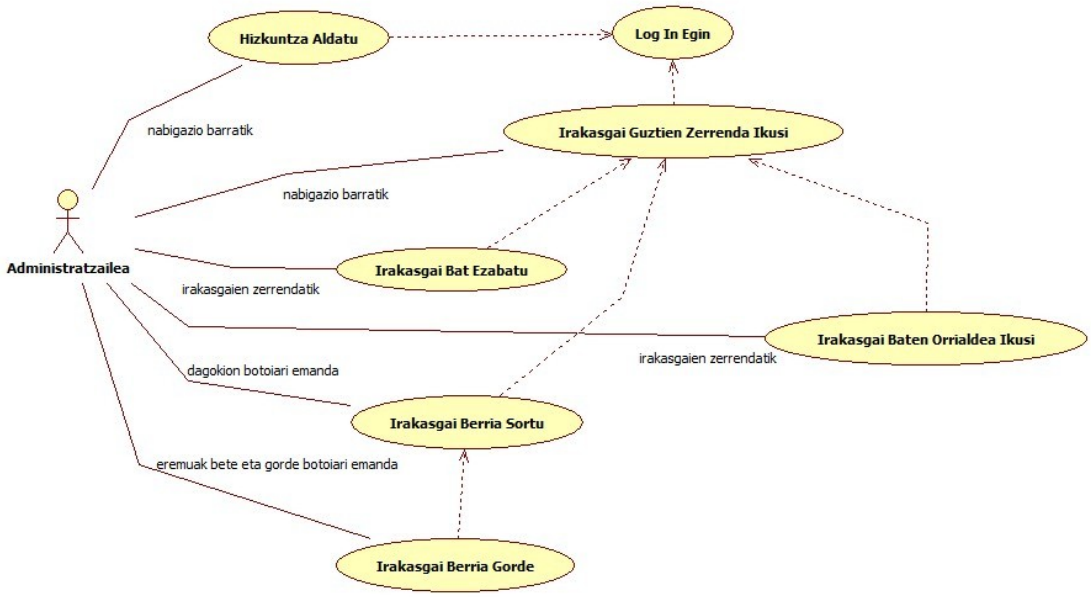


C.6. Irudia: Irakaslearen kasuak 3

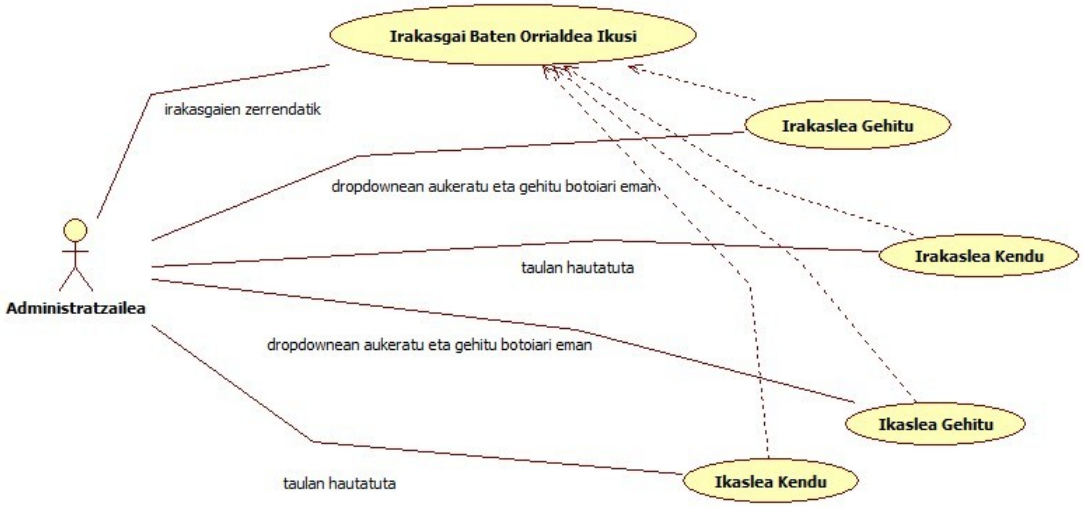


C.7. Irudia: Irakaslearen kasuak 4

C.2.3. ADMINISTRATZAILEEN KASUAK



C.8. Irudia: Administratzailearen kasuak 1



C.9. Irudia: Administratzailearen kasuak 2