

# **SOFTWARE INGENIARITZA AZKEN EMANGARRIA (3. ITERAZIOA)**

eman ta zabal zazu



Universidad  
del País Vasco

Euskal Herriko  
Unibertsitatea

# Aurkibidea

<b>Egileak</b>	<b>3</b>
<b>Web Zerbitzua eta Internalizazioa garatu al dugu?</b>	<b>4</b>
<b>Sarrera</b>	<b>5</b>
<b>Eskakizun bilketa</b>	<b>7</b>
Domeinu eredua	7
Erabilpen kasuen diagrama	9
Gertaera fluxu eta interfazeak	11
Create question	11
Hilabeteko diruak hasieratu	11
Langileen puntuazioak ikusi	12
Gertaera sortu	12
Pronostikoa ipini	13
Emitza ipini	13
Gertaera ezabatu	14
Bezerao lortu	15
Bezeraoari mezua bidali	15
Elkarrizketa gelditu	16
Elkarrizketa amaitu	16
Apustua ezabatu	17
Etekinak ikusi	17
Mugimenduak ikusi	18
Dirua sartu	18
Apustua egin	19
Arreta zerbitzua eskatu	20
Langileari mezua bidali	20
Langilea puntuatu	21
Errepikatzeko eskaera egin	21
Errepikatzeko eskaerari erantzun	22
Errepikapena baieztatu	23
Ezarpenak aldatu	24
Errepikapena ezabatu	24
Query questions	24
Erregistratu	25
Login	25
<b>Diseinua</b>	<b>27</b>
Sekuentzia diagramak	27
Apustua egin	27
Emitza ipini	28
Gertaera ezabatu	29
Apustua ezabatu	30

Arreta zerbitzua eskatu	31
Langileari mezua bidali	31
Elkarrizketa gelditu	32
Bezeroa lortu	32
Bezeroari mezua bidali	33
Elkarrizketa amaitu	33
Langilea puntuatu	34
Errepikatzeko eskaera egin	34
Errepikatzeko eskaerari erantzun	36
Errepikapena baieztatu	37
Ezarpenak aldatu	37
Errepikapena ezabatu	38
Hilabeteko diruak hasieratu	38
Langileen puntuazioa ikusi	38
Klase diagrama	39
<b>Inplementazioa</b>	<b>41</b>
<b>Ondorioak</b>	<b>45</b>
<b>URLak</b>	<b>47</b>
Bideoaren URLa	47
Kodearen URLa	48
Aurkezpenaren URLa	48

Egileak

Oier Elola Urquizu

Tarek Chamkhi Ermina

Unax Lazkanotegi Forcen

Josu Loidi Gorostidi

## Web Zerbitzua eta Internalizazioa garatu al dugu?

Bai, amaieran dagoen bideoan ikus daitekeen moduan, hainbat ApplicationLauncher aldi berean exekuta ditzakegu, eta honetaz gain interfaze berdinak hizkuntza ezberdinetan erabiltzeko aukera dago.

# Sarrera

Proiektu honen helburua, hiru mailako software arkitektura batean diseinatutako apustuak kudeatzen dituen informazio sistema baten garapena egitea zen. Hiru iteraziotan zehar, lanean jardun ostean proiektu txukun bat egitea lortu dugu. Lau erabiltzaile mota (bezero ez erregistratuak, bezero erregistratuak, langileak, administratzaileak) ditu gure sistemak, eta bakoitzari funtzionalitate ezberdin batzuk esleitu dizkiogu.

Bezero ez erregistratu batek ezer gutxi egin dezake. Zehazki, gertaeren galderak kontsultatu eta noski erregistratu. Login egiteko erabilpen-kasua ere hor du, baina, eztabaidagarria da, hau bezero ez erregistratuen erabilpen-kasu bat edota denena ote den.

Bezero erregistratuari dagokionez, apustu anizkoitzak egiteko aukera du (nola ez!), baina baita hauek ezabatzekoa ere. Apustu bat ezabatuz gero, jokaturiko dirua itzuliko zaio. Hori bai, gertaera amaitu bada, ezingo dio apusturik egin ezta hari egindako apustu bat ezabatu ere. Diru kontuak aipaturik, kontura dirua sartzeko aukera ere badu bezeroak. Baita bere kontuan izan diren diru mugimenduak eskuratu duen etekinarekin batera ikustekoa ere. Orain arte aipaturiko funtzionalitateak inplementazio aldetik nahiko "sinpleak" izan dira. Baina badira beste bi, dezente konplikatu zaizkigunak, edo hobeto esanda, dezente konplikatu ditugunak.

Hauetako bat beste bezero bat errepikatzea da. Erabilpen kasu hau, nahiko sinple egin zitekeen. Baina, guk ahalik eta osatuen egon zedin nahi genuen. Bezero batek beste bat errepikatu nahi badu, eskaera bat bidali beharko dio. Horretarako, bere izena bilatzaile batean jarri eta mezu bat bidali besterik ez du egin behar. Mezu honetan, bezeroak apostatzen duen euroko berak apostatuko duena eta hilabeteen apustu errepikatuekin gehienez gastatu nahi duena adierazi behar du. Baina kontuz, errepikatu nahi diogun bezeroak kontu pribatu bat badu, ezingo diogu jarraitu. Zeren bai, erabiltzaile batek bere kontua pribatu jar dezake. Dena ondo badoa, bezeroari eskaera bat iristen zaio. Berak erabakitzen du onartu ala ez. Eskaera ukatzen badu, amaitu da negoziazioa. Aldiz onartzen badu, nahi duen komisia adierazi behar du. Bidaltzean jatorrizko bezeroari mezu bat agertzen zaio, komisia adieraziz. Onar dezake ala ez.

Beraz, gure sistemari esker errepikapen eskaera guztiak uka ditzakegu (pribatu jarrita) edo nahi ditugunak onar ditzakegu nahi dugun komisiorekin. Baina, demagun bezero batek denak onartu nahi dituela, denei komisia bera ezarri nahi diela eta ez duela nahi aritu eskaera guztiak erantzuten. Egin dezake? Bai. Eraitza automatikoa aktiba dezake komisia batekin. Norbaitek errepikapen eskaera bat bidaltzen badio, automatikoki onarpen mezu bat jasotzen du honek.

Beraz, ikusi da errepikatzen hasteko sistema nahiko ongi osaturik dagoela. Baina, zer inplikitzen du errepikatzen egoteak? Bezero errepikatu batek apustua egiten badu apustu bera egiten zaie bere errepikatzaile guztiei, sarturiko dirua euroko apostatzen duten zifrarekin bidertuz. Hori bai, kontuan dirua izan behar dute eta hilabeteko diruak ezin du amaiturik egon. Honez gain, bezero errepikatu batek, apustu bat ezabatzen badu, eta bere errepikatzaileek apustu hori errepikatu bazien, hauei ere ezabatu egiten zaie. Honetaz gain,

apustua irabaziz gero errepikatzerako garaian onartu zen komisioa kobratzen du bezero errepikatuak.

Errepikatzearekin amaitzeko, errepikatzailleak eta errepikatuak ezabatzeko erabilpen kasu bat inplementatu da.

Bezero erregistratuak duen azken erabilpen kasua, arreta zerbitzua da. Bezeroak, arreta zerbitzu bat eska dezake eta hortik aurrera langile batekin (edo gehiagorekin, geroxeago ikusiko dugun moduan) hitz egiten jardungo du elkarrikketa amaitu arte. Amaitzen denean, jasoriko zerbitzua puntuatzeko eskatzen zaio.

Langileei erreparatzen badiegu, hauek gertaera eta pronostikoak sortzeko aukera dute. Baita gertaera bat ezabatzeko eta gertaera baten galdera bati emaitza ipintzeko ere. Honetaz gain, bezeroei arreta zerbitzua emateko aukera du. Erabilpen kasu bati esker, erantzun zain dauden bezeroak lortu ditzake. Langile bati esleitu zaion bezero bat ez zaio inor gehiagori esleituko. Bezeroak eskuratzeaz gain, lorturiko bezero baten mezuak irakurri eta erantzun ditzake. Honetaz gain, bezeroekin elkarrikketa gelditu dezake. Bi modutara egin dezake hau, etenda edo amaituta. Elkarrikketa eteten da, amaitu ez den elkarrikketa bat denean eta langileak elkarrikketarekin jarraitu ezin duenean. Etendako elkarrikketa bat edozein langilek mezu eta guzti berreskura dezake bezeroa lortzeko erabilpen kasuarekin. Amaituriko elkarrikketa bat ezin da berreskuratu, gainera elkarrikketa bat amaitzen denean, lehen esan bezala, arreta zerbitzua jaso duen bezeroari balorazio mezu bat bidaltzen zaio.

Azkenik, administratzaileari gehitu zaion funtzionalitate interesgarri bat sistemako langile guztiek izan dituzten balorazioak ikustearena da.





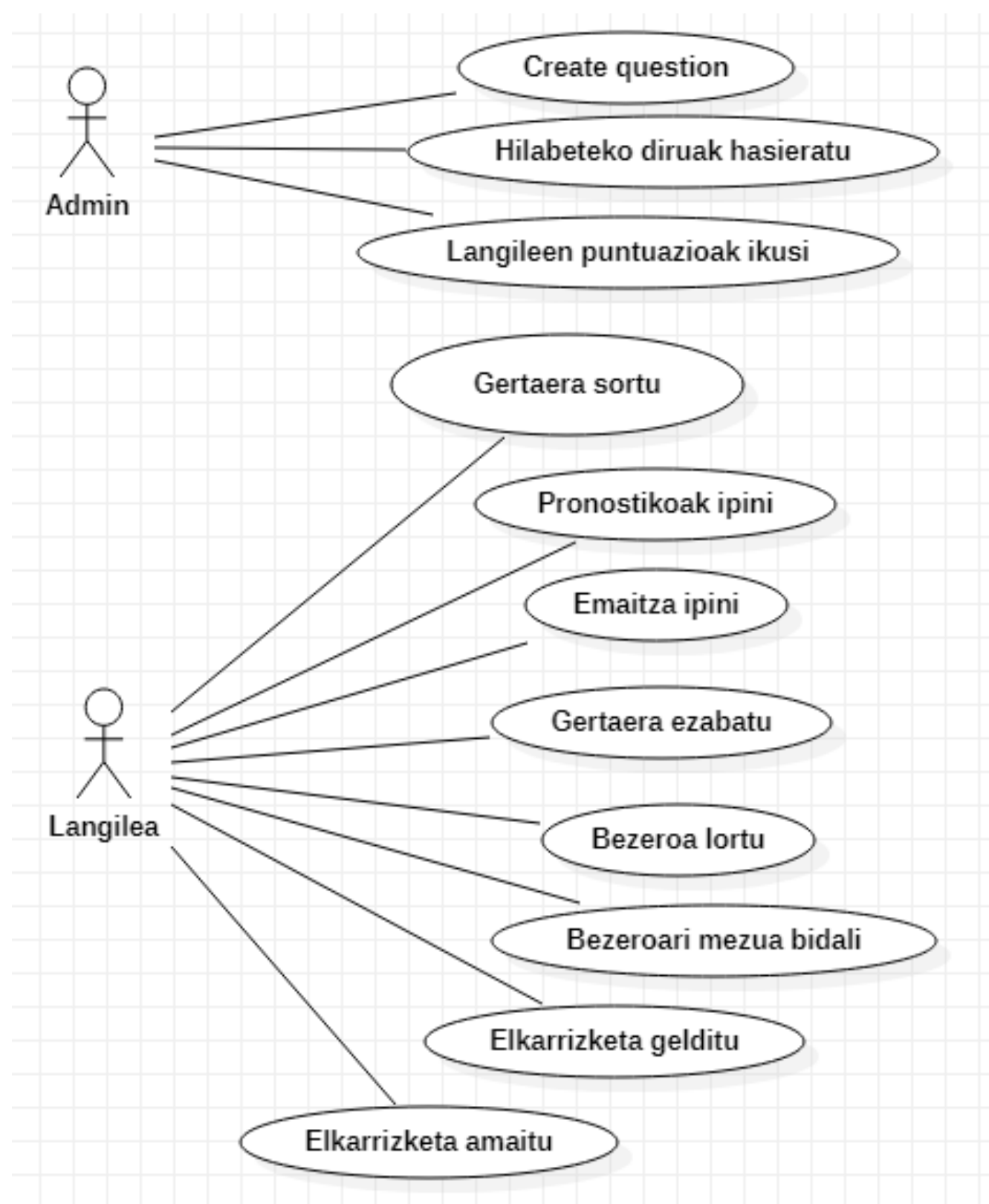
minimoa jasotzen dute. Aldi berean, gertaera hauek hainbat pronostiko izan ditzakete. Pronostiko hauek zenbaki, deskripzio eta kuota batez daude osaturik. Hau guztia izanda, nola lotzen dira apustuak? Apustuak azken finean galderen emaitza posibleen sorta baten alde diru kopuru bat botatzean sortzen dira. Emaitza posible hauek pronostikoak dira. Ondorioz, pronostikoak apustuekin erlazionaturik daude. Hain zuzen, apustu bakoitzeko pronostiko bat edo gehiago izango ditugu. Apustu hauek, identifikadoreaz gain, jokaturiko dirua, errepikatuak ote diren, pronostiko kopurua eta asmatutako dirua (apustua irabazi ote den kontrolatzeko) eta kuota jasotzen dituzte. Eta nola ez, apustu hauek bezeroek egiten dituztenez, bezeroekiko erlazioa adierazi behar da.

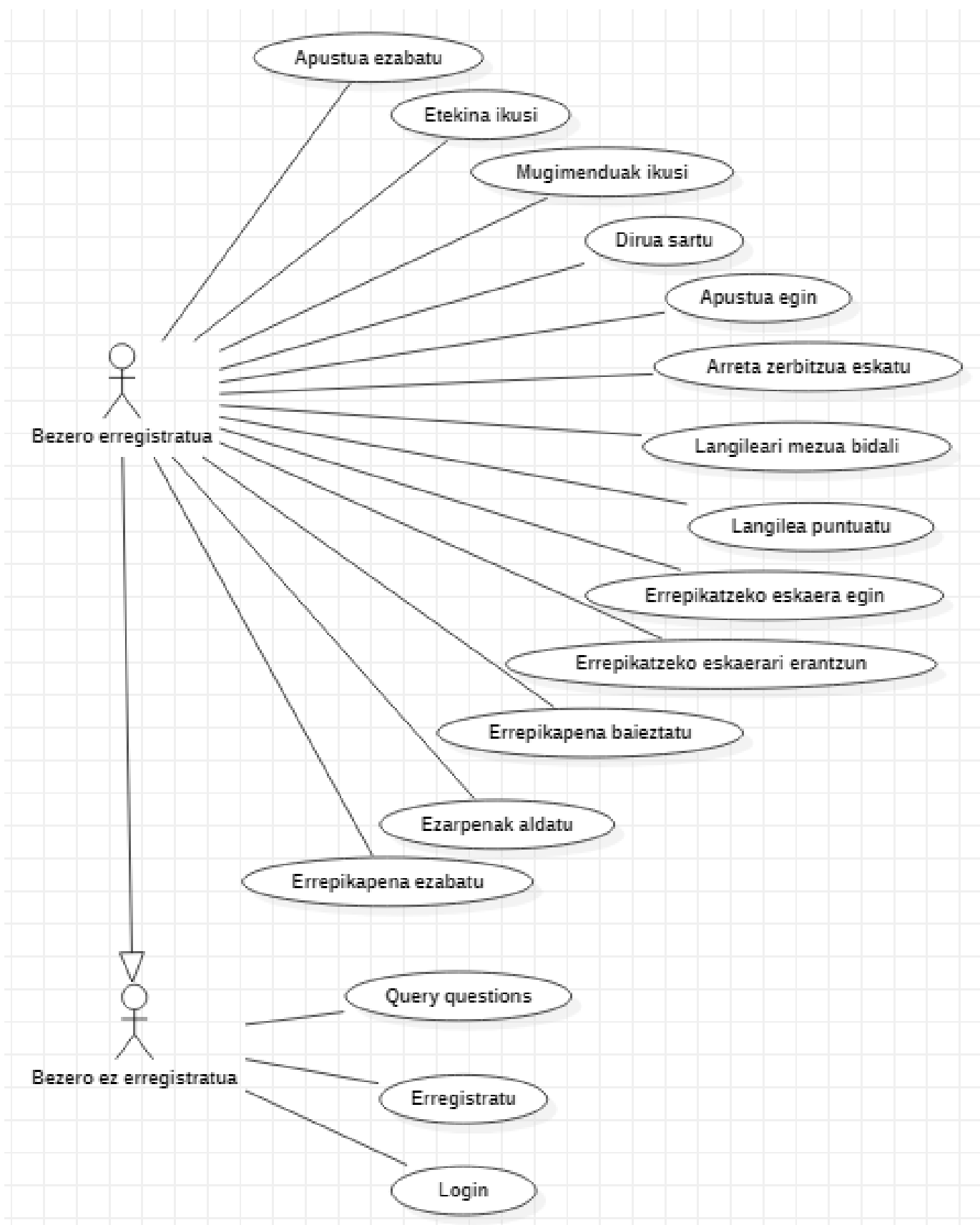
Bezeroek izaten dituzten diru mugimenduak kontrolatzeko ez da misterio handiegirik behar. Nahikoa da Mugimendua klase bat ezartzea identifikadore, deskripzio, diru aldaketa, data eta mota adieraziz eta hau Bezero erregistratuekin lotzea.

Hau dena aipaturik mezuen inguruko komentarioak falta dira. Ikus daitekeen moduan bi mezu mota topa ditzakegu: bezeroartekoak eta arreta mezuak. Biek konpartitzen dituzte, zenbaki bat, data bat, ea irakurrita ote dauden eta mezua bera. Baina, badituzte beraien aldeak. Bezeroarteko mezuak, errepikapen eskaerak egiteko erabiltzen dira. Ondorioz, negoziatzen diren balioak (komisioa, hilabetean jokatu dena...) jaso behar dituzte. Gainera mezu hauek bezeroartean gelditzen dira eta ez dute funtzionalitate gehiegirik eskatzen. Bezero batek, heltzen zaizkion mezuak irakurri, erantzun eta ezabatu egin ditzake. Beste ezertarako ahalmenik ez dute eman behar. Ondorioz, nahikoa da zuzenean mezu hauek bezeroekin lotzea. Bezero batek bidalitako mezuak eta jasotako mezuak izango ditu. Ondorioz, mezu batek hartzaile eta igorle atalak jasotzen ditu. Mezu bat erantzun behar denean nori bidali behar zaion jakiteko, igorlea zein den begiratu besterik ez da egin behar.

Baina, arreta mezuei dagokienez, dezente aldatzen da egoera. Izatez arreta mezuak langile eta bezeroak lotzen ditu. Beraz, bezeroarteko mezuak bezala adieraztea egokia al litzateke? Ez guztiz. Bezeroartekoen kasuan, hartzailea beti bezero bat da eta igorlea ere bai. Baina, arreta mezuetan ez da horrela. Batzuetan hartzailea bezero bat da eta beste batzuetan langile bat. Ezberdintasun horretatik abiatuz, ikus daiteke zerbait ezberdina behintzat planteatu beharra geneukala. Honetaz gain, interesgarria da langileak, bezeroarekin elkarriketa bat edukitzen hari direnean ordurarteko mezu guztiak jasorik edukitzea. Ez dirudi oso zaila hau lortzea, bezero batek gehienez elkarriketa bakar bat izango du langile batekin, ezta? ArretaMezua kasetik uneko langilea eta bezero bera duten mezuak jasota, elkarriketa lortzen dugu. Bai hori egia da, baina gehi diezaiogun hitz bat lehengo esaldiari. Bezero batek gehienez elkarriketa bakar bat izango du **aktibo** langile batekin. Unean bertan bai, baina zer pasatzen da lehengo elkarriketekin? Bezeroak arreta mezuak euren postontzitik ezabatzeraz behartu behar al ditugu? Beharbada jasorik eduki nahi ditu mezuak. Beraz, langile batek uneko elkarriketa bakarrik lortu nahi badu, mezuak nolabait multzokatuta izan behar ditugu. Honek klase berri baten beharraren pista eman zigun. Hori dela eta ArretaElkarriketa klasea sortu genuen. Hau langile baten eta bezero baten artekoa da, eta bi ArretaMezu talde ditu, langileak bidalitakoak eta bezeroak bidalitakoak. Elkarriketa hau amaituta egon daiteke edo ez. Amaiturik ez dagoen bitartean langilearentzat ikusgai egongo da. Amaitzean langileari kendu egingo zaio baina bezeroak oraindik mezuak ikusteko aukera izango du. Hau honela eginik, igorle hartzaile arazoa ere konpondu genuen eta elkarriketa gelditu edota amaitzeko erabilpen kasuak asko erraztu genituen.

## Erabilpen kasuen diagrama





## Gertaera fluxu eta interfazeak

### Create question

#### Basic Flow

1. *System* shows a Calendar where days with events are highlighted
2. *Admin* selects a **Date** in a Calendar
3. *System* displays the **events** of this date
4. *Admin* selects an **event**
5. *Admin* introduces a **question** and a minimum **betting price**
6. *System* adds the new **question** with a minimum **betting price** to the selected event

#### Alternative flow

1. There are no events on this date. Question cannot be added.
2. The question field is empty. Question cannot be added.
3. The minimum betting price is empty or it is not a number. Question cannot be added.
4. Event date has already finished (event day is before current day). Question cannot be added.

The screenshot shows a 'Create Question' window. It contains a calendar for May 2021. The calendar grid shows days from 1 to 31. The 14th of May is highlighted in red. To the right of the calendar is a 'List of events' dropdown menu. Below the calendar, there is a 'Question' text field and a 'Min Bet' input field. At the bottom, there are two buttons: 'Create Question' and 'Close'.

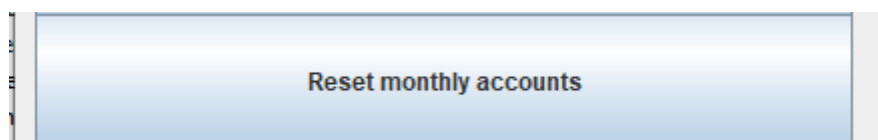
### Hilabeteko diruak hasieratu

#### Basic Flow

1. *Sistemak* datu-baseko Errepikapena objektu guztiei hilabeteen geratzzen den diru kopurua hasieratzen die.

#### Alternative flow

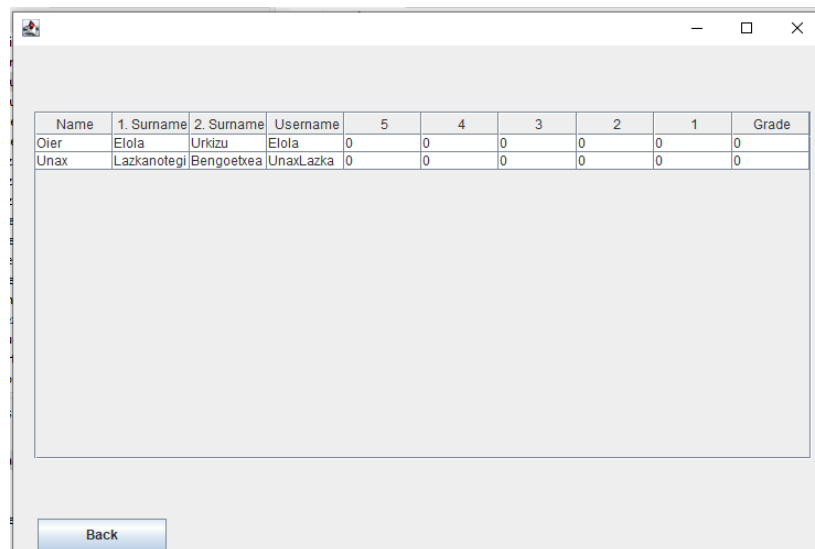
1. Hilabeteen behin exekutatu da.



Langileen puntuazioak ikusi

## Basic Flow

1. *Sistemak* datu-basetik langile guztien puntuazioak lortzen ditu eta pantailan erakusten ditu.



The screenshot shows a software window with a table of employee scores. The table has columns for Name, 1. Surname, 2. Surname, Username, and five score columns (5, 4, 3, 2, 1), plus a Grade column. Two employees are listed: Oier Elola Urkizu and Unax Lazkanotegi Bengoetxea. Both have scores of 0 in all five columns and a Grade of 0. A 'Back' button is located at the bottom left of the window.

Name	1. Surname	2. Surname	Username	5	4	3	2	1	Grade
Oier	Elola	Urkizu	Elola	0	0	0	0	0	0
Unax	Lazkanotegi	Bengoetxea	UnaxLazka	0	0	0	0	0	0

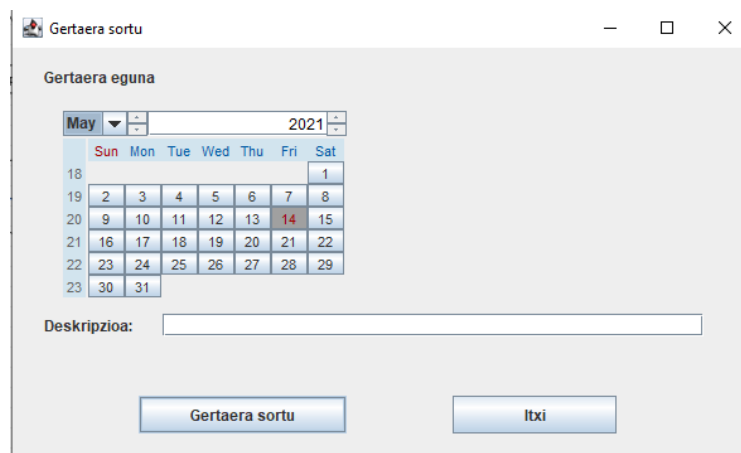
Gertaera sortu

## Basic Flow

1. *Sistemak* informazioa txertatzeko baliabideak erakusten ditu.
2. *Langileak* gertaeraren data eta deskribapena sartu eta botoia sakatzen du.
3. *Sistemak* gertaera zenbakia sortu eta gertaera gehitzen du (onarpen mezua).

## Alternative flow

1. Txertatutako data jada pasata badago, gertaera ez da sortuko (errore mezua).
2. Deskribapen hutsa txertatzen bada, gertaera ez da onartuko (errore mezua).
3. Sarrerako data eta deskribapena dituen gertaera bat existitzen bada datu-basean, gertaera ez da sortuko (errore mezua).



The screenshot shows a 'Gertaera sortu' window. It features a date picker set to May 2021, a calendar view showing the month of May, and a text input field for 'Deskripzioa:'. At the bottom, there are two buttons: 'Gertaera sortu' and 'Itxi'.

## Pronostikoa ipini

### Basic Flow

1. Sistemak egutegi bat erakusten du gertaerak dituzten egunak azpimarraturik dituena.
2. Langileak data bat aukeratzen du egutegian
3. Sistemak data horretan dauden gertaerak erakusten ditu.
4. Langileak gertaera bat aukeratzen du.
5. Sistemak gertaera horren inguruan dauden galderak pantailaratzen ditu.
6. Langileak galdera bat aukeratzen du eta deskripzio eta kuota bat idazten ditu eta botoia sakatzen du.
7. Sistemak pronostiko berri bat sortzen du eta datu-basean gehitzen du.

### Alternative flow

1. Deskripzio edota kuota hutsa ematen bada, ez da pronostikoa sortuko (errore mezua).
2. Kuota ez badago zenbakiz osatua edo 1 baino txikiagoa bada ez da pronostikoa sortuko (errore mezua).
3. Aukeratu den galderak jada, baldin badauka deskripzio berdineko pronostiko bat, ez da pronostikoa sortuko (errore mezua).

**Create pronostic**

**Event Date**

May 2021

Sun	Mon	Tue	Wed	Thu	Fri	Sat
18						1
19	2	3	4	5	6	7
20	9	10	11	12	13	14
21	16	17	18	19	20	21
22	23	24	25	26	27	28
23	30	31				

**List of events**

**List of questions**

**Description:**

**Fee:**

**Create** **Close**

## Emaizta ipini

### Basic Flow

1. Sistemak egutegi bat erakusten du gertaerak dituzten egunak azpimarraturik dituena.
2. Langileak data bat aukeratzen du egutegian
3. Sistemak data horretan dauden gertaerak erakusten ditu.
4. Langileak gertaera bat aukeratzen du.
5. Sistemak gertaera horren inguruan dauden galderak pantailaratzen ditu.
6. Langileak galdera bat aukeratzen du.
7. Sistemak galderaren pronostiko guztiak erakusten ditu.
8. Langileak pronostiko bat aukeratzen du eta botoia sakatzen du.
9. Sistemak galderaren emaitza atributua eguneratzen du eta pronostikoaren alde apustu egindako bezeroen dirua eguneratzen du eta trantsazkio berri bat gehitzen die.

### Alternative flow

1. Gertaera oraindik ez bada bukatu, ez da emaitza jartzen utziko (errore mezua).
2. Apusturen bat errepikatua bada, bezero errepikatuari komisioa ordainduko zaio.

**Set result**

Event Date: May 2021

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

List of events:

List of questions:

List of pronostics:

Create Close

## Gertaera ezabatu

### Basic Flow

1. *Sistemak* egutegi bat erakusten du gertaerak dituzten egunak azpimarraturik dituen.
2. *Langileak* data bat aukeratzen du egutegian
3. *Sistemak* data horretan dauden gertaerak erakusten ditu.
4. *Langileak* gertaera bat aukeratzen du eta botoia sakatzen du.
5. *Sistemak* gertaera horren, galderen baten pronostikoren bat duten apustu guztiei pronostikoa kentzen die. Jarraian gertaera ezabatzen du.

### Alternative Flow

1. Gertaera jada amaitu bada ez da ezabatzen utziko.
2. Apustuak pronostiko bakar bat bazuen, dirua itzuliko zaio bezeroari. Gainera, apustua errepikatua bazen, hilabeteko zifra eguneratzen da.
3. Gelditu diren pronostiko guztiak jada asmatu badira, apustuaren dirua ematen zaio bezeroari eta errepikatua bada komisia errepikatuari.

**Remove event**

Event Date: May 2021

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

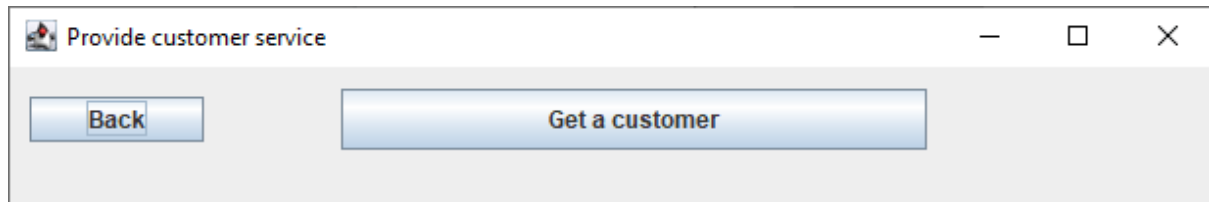
List of events:

Remove Close

## Bezeroa lortu

### Basic Flow

1. *Sistemak* datu-basetik esleitu gabeko eta amaitu gabeko elkarrizketa bat hartzen du eta uneko langileari esleitzen dio.



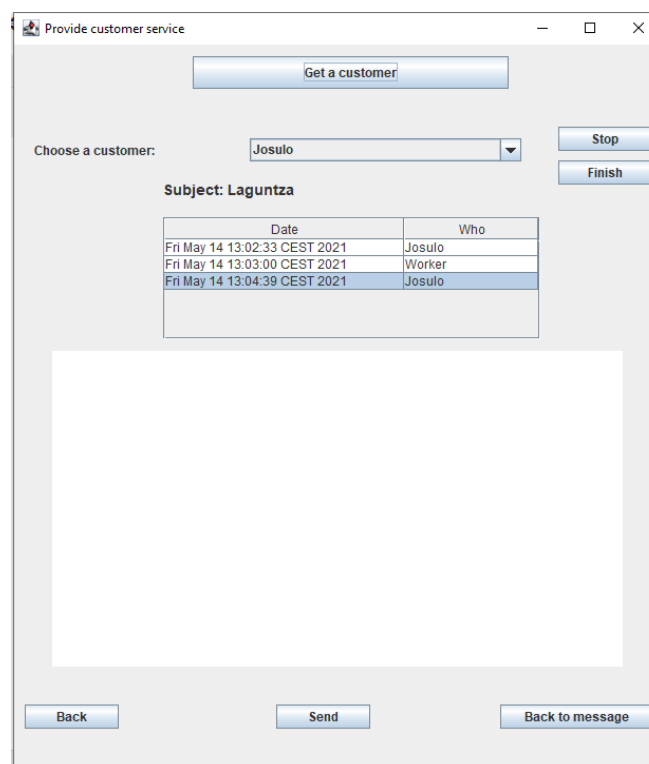
## Bezzeroari mezua bidali

### Basic Flow

1. *Sistemak* langileak dituen elkarrizketak erakusten ditu.
2. *Langileak* elkarrizketa bat aukeratzen du.
3. *Sistemak* elkarrizketako mezu guztiak erakusten ditu.
4. *Langileak* azken mezua hartzen du eta testu bat idazten du.
5. *Sistemak* elkarrizketari mezu berri bat gehitzen dio.

### Alternative flow

1. Azken mezua langileak bidalia bada, ezingo zaio mezurik bidali.

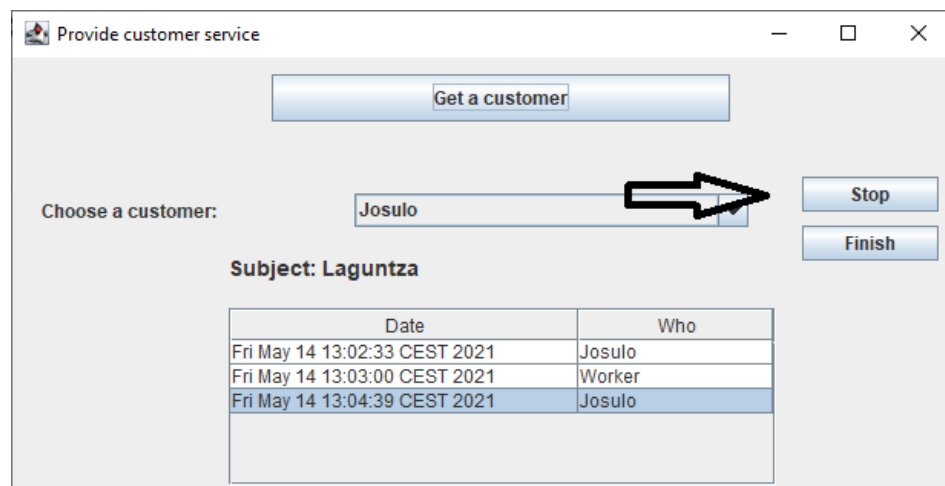




## Elkarrizketa gelditu

### Basic Flow

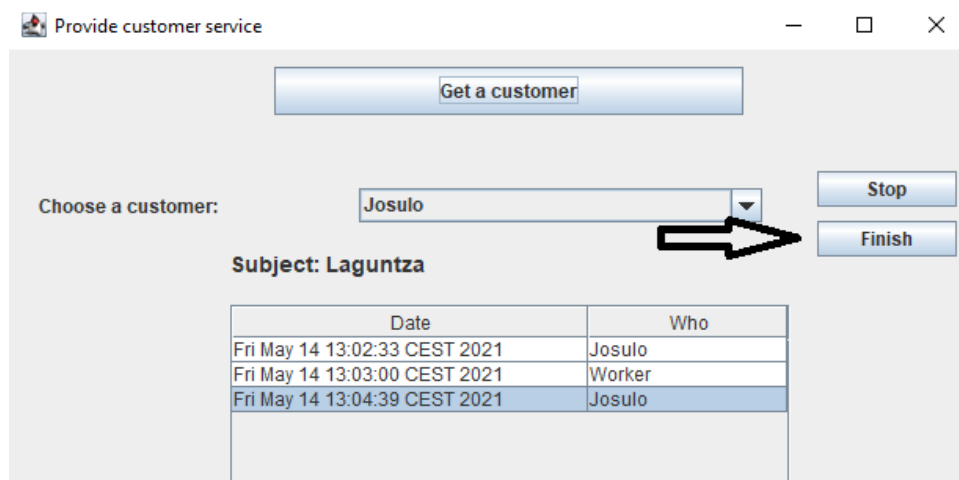
1. *Sistemak* langilearen elkarrizketak erakusten ditu.
2. *Langileak* elkarrizketa bat aukeratzen du.
3. *Sistemak* elkarrizketa esleitu gabe jartzen du, beste langile batek hartzeko aukera izan dezan.



## Elkarrizketa amaitu

### Basic Flow

1. *Sistemak* langilearen elkarrizketak erakusten ditu.
2. *Langileak* elkarrizketa bat aukeratzen du.
3. *Sistemak* elkarrizketatik bezeroak bidalitako mezu guztiak borratzen ditu eta baita bezeroak ezabaturik dituen langileak bidalitako mezuak ere.
4. *Sistemak* elkarrizketa amaitutzat emateko mezu berri bat gehitzen du elkarrizketara.



## Apustua ezabatu

### Basic Flow

1. *Sistemak* bezeroak eginik dituen apustu guztiak erakusten ditu.
2. *Bezero erregistratuak* apustu bat aukeratzen du.
3. *Sistemak* apustua datu-basetik ezabatzen du eta bezeroari dirua itzultzen dio.

### Alternative flow

1. Apustuaren gertaeraren bat jada hasi bada edo amaitu bada, apustua ez da ezabatuko (errore mezua).

Delete bet

Choose a bet

1

Event Date	Event	Question	Result	Fee	Final result
2021/Mar/17	Eibar-Celta	Zeinek irabaziko du partidua?	X	1.5	-
2021/Mar/17	Eibar-Celta	Zeinek sartuko du lehenengo gola?	1	1.2	-

Played: 2.0      Total fee: 1.799999999...      Potential returns: 3.599999999...

Remove

Back

## Etekina ikusi

### Basic Flow

1. *Bezero erregistratuak* botoia sakatzen du.
2. *Sistemak* bezeroaren mugimenduak hartu eta etekina kalkulatu ostean pantailaratu egingo du.

See the benefit

Played: 2.0      Won: 0.0

Yield: -2.0

Current balance: 80.0

Close

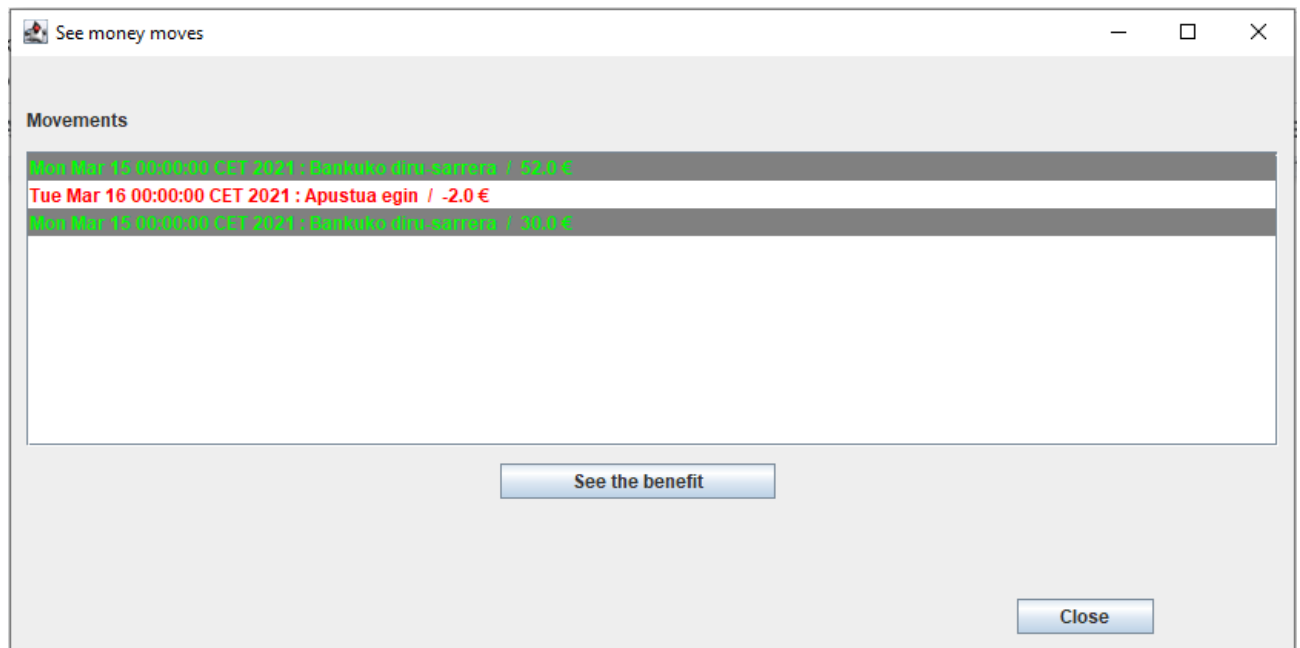
Mugimenduak ikusi

## Basic Flow

1. *Sistemak* bezeroaren mugimendu guztiak erakusten ditu.

## Alternative flow

1. Bezeroak ez badu mugimendurik, mezu baten bidez adieraziko zaio.



Dirua sartu

## Basic Flow

1. *Sistemak* diru kantitatea testuz sartzeko baliabideak eskaintzen ditu.
2. *Bezero erregistratuak* diru kantitatea idazten du eta botoia sakatzen du.
3. *Sistemak* bezeroaren diru kantitatea eguneratzen du eta transakzio berri bat gehitzen dio.

## Alternative flow

1. Testu hutsa sartzen bada edo ez badago zenbaki positiboz osaturik, dirua ez da sartuko (errore mezua).

## Apustua egin

### Basic Flow

1. Sistemak egutegi bat erakusten du gertaerak dituzten egunak azpimarraturik dituen.
2. Bezero erregistratuak data bat aukeratzen du egutegian
3. Sistemak data horretan dauden gertaerak erakusten ditu.
4. Bezero erregistratuak gertaera bat aukeratzen du.
5. Sistemak gertaera horren inguruan dauden galderak pantailaratzen ditu.
6. Bezero erregistratuak galdera bat aukeratzen du.
7. Sistemak galderaren pronostiko guztiak erakusten ditu.
8. Bezero erregistratuak pronostiko bat aukeratzen du.
9. Lehen 8 pausuak nahi adina aldiz errepikatzen dira pronostiko sorta bat lortu arte.
10. Bezero erregistratuak diru kantitate bat ematen du.
11. Sistemak apustu berri bat sortzen du, bezeroaren dirua eguneratzen du eta transakzio berri bat gehitzen dio.

### Alternative flow

1. Gertaeraren bat hasi edo amaitu bada, ez da apustua egiten utziko (errore mezua).
2. Diru kantitatea, galderak eskatzen duen minimoa baino txikiagoa bada, ez da apustua egiten utziko (errore mezua).
3. Bezeroak ez badauka apostatu nahi duen adina diru, ez da apustua egiten utziko (errore mezua).
4. Bezeroak errepikatzaileak baditu, hauei ere apustua gehituko die dagokion diru kantitatearekin.

The screenshot shows a 'Make bet' application window. It features a calendar for June 2021, a list of events (Atlético-Athletic), a list of questions (Zeinek irabaziko du partidua?), and a list of pronostics (1 : 1.2). The window also displays the minimum bet (1.0) and the total fee (1.2). At the bottom, there is a 'Bet' field, a 'Create' button, and a 'Close' button.

**Make bet**

**List of events**

June 2021

Atlético-Athletic

**List of questions**

Zeinek irabaziko du partidua?

**List of pronostics**

1 : 1.2

Min bet: 1.0

Quit Add

Date	Event	Question	Result	Fee
2021/Jun/17	Atlético-Athletic	Zeinek irabaziko du partidua?	1	1.2

Bet: € Min bet: 1.0 Total fee: 1.2

Create Close

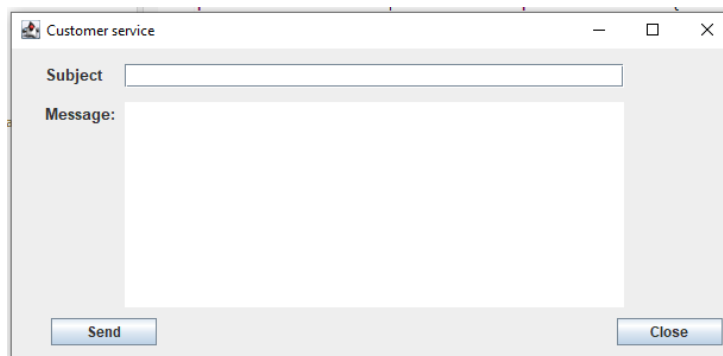
## Arreta zerbitzua eskatu

### Basic Flow

1. *Bezera erregistratuak* gai bat eta mezu bat idazten ditu.
2. *Sistemak* esleitu gabeko elkarrizketa berri bat sortzen du eta bertan mezua gehitzen du.

### Alternative flow

1. Mezua hutsa bada, ez da bidaliko (errore mezua).
2. Bezeroa arreta zerbitzua jasotzen ari bada mezua ez da bidaliko.



Customer service

Subject

Message:

Send Close

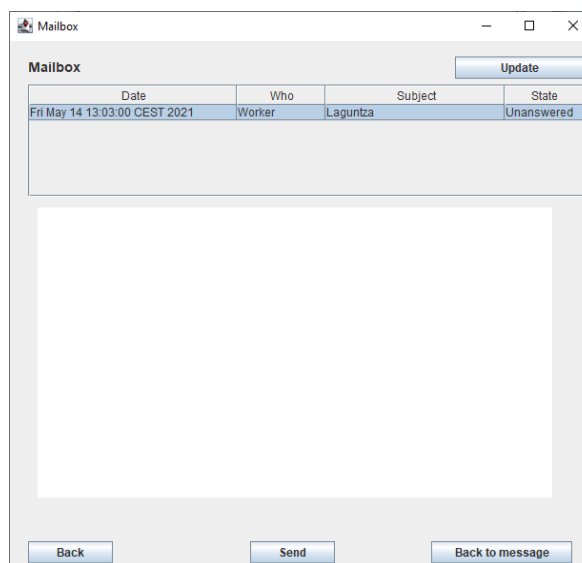
## Langileari mezua bidali

### Basic Flow

1. *Bezeraok* erantzun gabeko langile baten mezu bat hartzen du, eta erantzuteko testua idazten du.
2. *Sistemak* mezuaren elkarrizketari mezu berri bat gehitzen dio.

### Alternative flow

1. Mezua hutsa bada, ez da bidaliko (errore mezua).



Mailbox

Update

Date	Who	Subject	State
Fri May 14 13:03:00 CEST 2021	Worker	Laguntza	Unanswered

Back Send Back to message

## Langilea puntuatu

### Basic Flow

1. *Bezero erregistratuak* erantzun gabeko elkarrizketa amaierako mezu bat hartzen du eta 1etik 5erako zenbaki bat ematen du.
2. *Sistemak* elkarrizketa amaitu duen langileari puntuazio bat gehitzen dio.

The screenshot shows a web application window titled 'Mailbox'. It contains a table with two rows of messages. The first row is selected. Below the table is a feedback form for the selected message. The form includes a 'Hello:' label, a message body, a 'Thank you very much,' line, a 'Company Administrator' signature, and a rating section with five radio buttons. The fourth radio button is selected. At the bottom are 'Back' and 'Send' buttons.

Date	Who	Subject	State
Fri May 14 13:03:00 CEST 2021	Worker	Laguntza	Answered
Fri May 14 13:15:33 CEST 2021	Administrator	Laguntza	Unanswered

Hello:

This message is to inform you that your conversation with a worker has just ended. We hope you found our service helpful, how would you rate the service you received? Please fill out the form below and send it to us. It will help us to improve.

Thank you very much,

Company Administrator

Rate from 1 to 5

☐ ☐ ☐ ☒ ☐

## Errepikatzeko eskaera egin

### Basic Flow

1. *Bezero erregistratuak* string bat ematen du.
2. *Sistemak* haiekin errepikapen erlaziorik ez duen, publikoak diren eta euren erabiltzaile izenean sartutako string-a duten bezeroak itzultzen ditu.
3. *Bezero erregistratuak* bezero bat aukeratzen du, mezu bat idazten du, hilabeteen apostatu nahi duena zehazten du eta bezeroak apostatuko duen euroko apostatuko duena zehazten du.
4. *Sistemak* bezeroari mezua bidaltzen dio sarturiko datuekin.

### Alternative flow

1. Mezua hutsa bada, ez da bidaliko (errore mezua).
2. Bigarren bezeroak errepikapen automatikoa ezarririk badu, lehen bezeroari onarpen mezua helduko zaio.

Request to repeat

Search

Josulo

**User name: Josulo**

Bets made: 1      Bets won: 0      Percentage of success: % 0.0

Played: 2.0      Won 0.0      Yield -2.0

Request to repeat

Close

Request to repeat

Search

Josulo

For: Josulo

For every euro he bets you will bet:      Maximum monthly bet:

Send      Close

Errepikatzeko eskaerari erantzun

## Basic Flow

1. *Bezero erregistratuak* eskaera mezu bat hartzen du eta komisio balio bat ematen du.
2. *Sistemak* bezero eskatzaileari onarpen mezu bat bidaltzen dio.

Mailbox

Mailbox

Update

Date	Who	Subject	State
Fri May 14 13:03:00 CEST 2021	Worker	Laguntza	Answered
Fri May 14 13:15:33 CEST 2021	Administrator	Laguntza	Unanswered
Fri May 14 13:20:22 CEST 2021	Tarek12301	Repeat request	Unanswered

Who: Tarek12301

Message: I need your help

Conditions established:

-For each euro he will bet: 2.0€

-Maximum monthly bet: 50.0€

Do you accept?

☒ Yes
☐ No

Your benefit:

/1

Send

Back

Errepikapena baieztatu

## Basic Flow

1. *Bezero erregistratuak* onarpen mezu bat hartzen du eta onartzen duela adierazten du.
2. *Sistemak* bezero errepikatuari jakinarazpen mezu bat bidaltzen dio.

Mailbox

Mailbox

Update

Date	Who	Subject	State
Fri May 14 13:21:31 CEST 2021	Josulo	Acceptance	Unanswered

Who: Josulo

Message: I accept it, but %20.0 of the profits for me.

Conditions established:

-For each euro he will bet: 2.0€

-Maximum monthly bet: 50.0€

-Commission: %20.0

Do you accept?

☒ Yes
☐ No

Send

Back



## Ezarpenak aldatu

### Basic Flow

1. *Bezero erregistratuak* kontu publikoa edo pribatua nahi duen eta emaitza automatikoa ezarri nahi al duen zehazten du (baiezko kasuan, komisioa ere bai).
2. *Sistemak* bezeroaren datuak eguneratzen ditu.



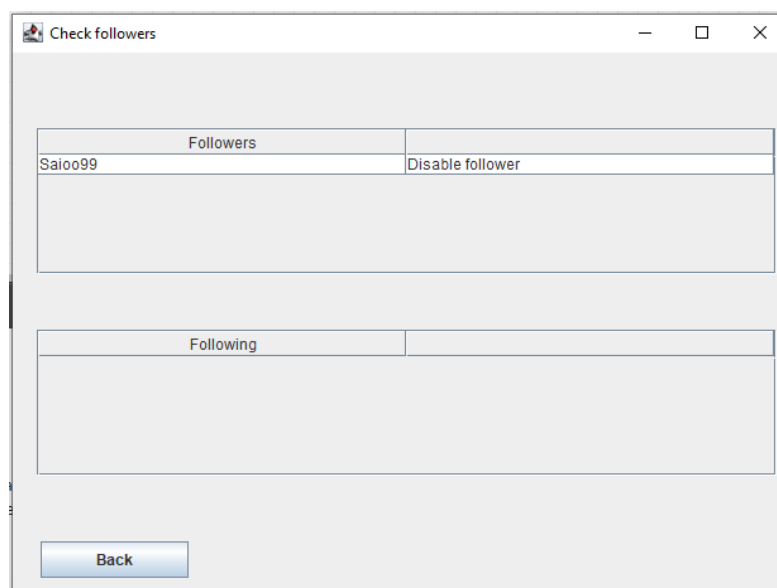
The screenshot shows a 'Settings' window with the following options:

- Type of account: ☐ Private, ☒ Public
- Automatic response activated: ☒ Yes, ☐ No
- Commission:  /1
- Buttons: Save, Close

## Errepikapena ezabatu

### Basic Flow

1. *Bezero erregistratuak* errepikapen bat aukeratzen du (bera errepikatzaile edo bera errepikatu dena).
2. *Sistemak* errepikapeneko bi bezeroei errepikapena ezabatu ostean, datu-basetik ezabatzen du..



The screenshot shows a 'Check followers' window with two tables:

Followers	
Saioo99	Disable follower

Following	
-----------	--

At the bottom of the window is a 'Back' button.

## Query questions

### Basic Flow

1. *Bezero erregistratuak* errepikapen bat aukeratzen du (bera errepikatzaile edo bera errepikatu dena).
2. *Sistemak* errepikapeneko bi bezeroei errepikapena ezabatu ostean, datu-basetik ezabatzen du..

**Query Questions**

**Event Date**

May 2021

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

**Events**

Event#	Event

**Questions**

Question#	Question

**Close**

## Erregistratu

### Basic Flow

1. Sistemak datuak txertatzeko baliabideak erakusten ditu.
2. Bezero ez erregistratuak bere datuak ematen ditu eta botoia sakatzen du.
3. Sistemak datuak hartu, bezero berri bat sortu eta datu-basera gehitzen du.

### Alternative flow

1. Datuen bat ez bada eman, erregistroa ez da onartuko (datuaren ondoan errore mezua)
2. Erabiltzaile izena jada datu-basean aurkitzen bada, erregistroa ez da onartuko (erabiltzaile izenaren ondoan errore mezua).
3. Pasahitzak zortzi karaktere baino gutxiago baditu erregistroa ez da onartuko (pasahitzaren ondoan errore mezua).
4. Bi pasahitzak ezberdinak badira, erregistroa ez da onartuko (pasahitzaren ondoan errore mezua).
5. E-mailak ez badu e-mail egitura betetzen, erregistroa ez da onartuko (errore mezua e-mailaren ondoan)
6. Telefono zenbakiaren luzera 9ren ezberdina bada eta ez badago zenbakiz osatua, erregistroa ez da onartzen (errore mezua telefono).
7. Erregistratu nahian dabilen pertsonak 18 urte baino gutxiago baditu (gaurko data - jaiotze data < 18 urte) erregistroa ez da onartuko (jaiotze-dataren gainean errore mezua).

**Register**

Name:

First surname:

Second surname:

User name:

Password:

Repeat password:

E-mail:

Phone Number:

Birth date:

Day:  Month:  Year:

**Register**

**Close**

# Login

## Basic Flow

1. *Sistemak* datuak txertatzeko baliabideak erakusten ditu.
2. *Bezero ez erregistratuak* erabiltzaile izena eta pasahitza eman eta botoia sakatzen du.
3. *Sistemak* datuak egiaztatzen ditu eta zuzenak badira, pertsonaren interfazea zabaltzen du.

## Alternative flow

1. Erabiltzaile izena eta pasahitza duen Pertsonarik ez bada aurkitzen datu basean, logina baztertuko da (errore mezua).
2. Daturen bat ez bada sartu, logina baztertuko da (errore mezua).

The image shows a standard Windows-style dialog box titled "Log in". It contains two text input fields: "User name:" and "Password:". Below these fields is a blue "Login" button. At the bottom of the window, there is a question "Don't you have an account yet?" followed by two buttons: "Back" and "Register". The window has standard minimize, maximize, and close buttons in the title bar.

# Diseinua

## Sekuentzia diagramak

*Adi! Sekuentzia diagramak dokumentuan txertatzeko eskatzen zen arren, hauen gehiengoa handiegia denez, ezinezkoa zitzaigun era ulergarri eta txukun batean hauek dokumentuan txertatzea, ondorioz azalpenak bakarrik gehitu dira.*

## Apustua egin

Bezeroa ApustuaEginGUI interfazean aurkitzen da, non interfazeak eskura emango dizkion baliabideak apustu bat egiteko `getEvents()` metodoa erabilita, data berri bat aukeratzen duen unean. Honek, data horretako gertaera guztiak erakusten ditu, aukeratutako gertaeraren galdera guztiak eta aukeratutako galderaren pronostiko guztiak ere.

Behin datu guzti hauek ikusgai dituenean, bezeroak aukeratzen duen pronostikoa lista batera gehitzeko aukera dauka, "Add" botoiarekin, lista batean gordeta eta aukera bezala, pronostikoren bat listatik ezabatu nahi badu, aukera izango du "Quit" botoia sakatuz. Kontuan izan behar da programak ez duela ahalbidetzen listara gehitzea galdera bati erreferentzia egiten dioten pronostiko bat baino gehiago.

Bezeroak erabaki duenean zein pronostikori apustu egin nahi dioen, apostatzeko diru kopurua adierazi eta "Create" botoia sakatu behar du, honela apustua sortzeko prozesua hasita: honek negozio logikari deitu egingo dion `apustuaEgin()` metodoa erabiliko du eta pronostikoen lista, adierazitako diru kopurua eta bezeroa datuak emango ditu parametro bezala. Gainera, bezero bera itzuliko du, baina une horretan sortuko den apustu berria gehituta. Ondoren, negozio logikak dei berdina egingo dio datu baseari.

Behin datu baseari dei egin zaionean, bertan egingo dira aldaketa berrien kalkuluak eta horretarako, lehenik eta behin bezeroa erazagutu behar da eta horregatik parametro gisa pasa dugun bezeroaren erabiltzaile izena jasoko dugu `getErabiltzaileIzena()` metodoarekin eta `entityManager`-ean bilatuko dugu `find()` metodoa erabiliz. Metodo honekin jasoko dugun objektua, hau da, bezeroa, erabili beharko dugu. Horretaz gain, pronostikoen listarekin berdina egin beharko dugu, for bat erabiliz, pronostiko guztiak erazagutu eta lista berri batean gordeta.

Behin objektua lortuta, transakzioa hasten da eta uneko bezeroari `addApustua()` metodoa erabilita eta pronostiko list, diru kopurua eta null parametroak bidalita, datu hauek dituen apustu bat sortzen da eta bezeroaren apustuen listan gehituko da. Gainera, metodo honek sortutako apustua itzuliko dio datu baseari.

Honekin batera, daukagun sortako pronostiko guztiei sortutako apustu berria gehitu egingo die eta entityManager-ari persist() metodoarekin apustu objektu berria gehituko du.

Hau guztia egin ondoren, programak uneko bezeroak dituen errepikatzaileak aztertzen ditu, getErrepikatzaileak() metodoa erabilita eta lista berri bat jasoko du. Lista hori hutsa badago, ez du ezer egingo. Bestela, errepikatzaile guztiei apustua gehitzen saiatuko da.

Horretarako baldintza batzuk bete behar dituzte errepikatzaileek: getHilabeteHonetanGeratzenDena() metodoa erabilita hilabetean errepikatzerakoan duen diru limitera iritsi den eta ez bada iritsi, errepikatzerako orduan automatikoki apostatuko duen diru kopurua baino gehiago duen apostatzea begiratuko du. Ez bada horrela, apostatzeko diru kopurua getHilabeteHonetanGeratzenDena() metodoak itzultako balioa izango du.

Behin ikusita zein izango den apostatu nahi duen diru kopurua, errepikatzaileak bere kontuan diru kopuru hori badu, addApustua() metodoa erabilita errepikatzen duen bezeroari apustu berria gehituko zaio (pronostiko lista, diru kopurua eta errepikatzen dion bezeroaren objektua parametroak emanik). Horretaz gain, apustuan dauden pronostiko guztiei apustua gehituko zaie apustuen listetara.

Jarraian, errepikatzen duen bezeroari addMugimendua() metodoarekin eta parametro batzuk pasata, mugimendu berri bat gehituko dio, apustua adieraziz eta gainera eguneratuHilHonetanGeratzenDena() metodoa erabilita eta apostatutako diru kopurua adierazita, hilabetean errepikatu dezakeen diru kopurua eguneratuko du. Ondoren, persist() metodoa erabilita, apustua entityManager-era gehituko du.

Azkenik, hasierako bezeroari addMugimendua() metodoa erabilita mugimendu berria gehituko zaio eta transakzioa amaituko da.

## Emailtza ipini

Langilea EmailtzaIpiniGUI interfazeaz aurkitzen da eta interfazeak esakintzen dizkion baliabideekin, data bat aukeratzeko aukera izango du, non aukeratutako data horretan dauden gertaera guztiak jasoko dituen getEvents() metodoaren bidez.

Gertaera guztiak ikusterakoan, bat aukeratu eta horren galderak adieraziko dizkigu, langileak baten bat aukeratu dezan. Aukeratzen duen galderaren pronostikoak adieraziko dizkigu eta momentu horretan aukera emango digu pronostikoaren emailtza jartzeko.

Emailtza zein den adierazterako orduan, emailtzaIpini() metodoa erabiliko da, negozio logikari dei eginez. Horretarako beharrezkoa da gertaera emailtza ipintzerako orduan baino lehenago gertatu izan dadila, bestela ezin izango zelako emailtzarik ipini. Gainera, uneko galdera eta pronostikoa parametro bezala emango zaizkio negozio logikari.

Datu basearekin, berdina egingo da, parametro berdinak emanez.

Behin datu basean dagoenean programa, identifikadoreen bidez, pronostikoa eta galdera entityManager-etik jasoko dira eta jarraian transakzioa hasieratuko da.

Transakzioa hasterakoan, lehenengo galderari emaitza adieraziko dio, setResult() metodoa erabilita eta pronostikotik getDeskripzioa() objektua parametro bezala emanik.

Behin pronostiko irabazlea adierazita, pronostikoaren apustu guztiak jasoko dira, getApustuak() metodoa erabilita eta apustu bakoitzarekin hau egingo du:

eguneratuAsmatuKop() metodoa erabiliko du eta asmatutakoKop == pronostikoKop bada, true balioa itzuliko du. True itzultzen badu, agindu gehiago egingo ditu, bestela bertan behera geratuko da eta transakzioa amaituko da.

True-ren kasuan, getErrepikatua() metodoa erabiliko du eta null balioa ez badu itzultzen, esan nahiko du beste bezero bati errepikatu zaion apostua dela eta orduan beste bezeroak jasoko duen komisioa kalkulatu du.

Kalkulu hori egiteko, getKopurua() (apostatu duen diru kopurua), getKuota() (apostuaren kuota) eta getKomisioa() (errepikatuaren komisioa) beharko ditugu. Datu horiek jasota komisioa kalkulatu du eta gero irabazitako diru guztia jakingo dugu, irabazi totala - komisioa eginda.

Azkenean, addMugimendua() erabilita, apustuaren bezeroari irabaziaren mugimendua gehituko zaio.

## Gertaera ezabatu

Langilea GertaeraEzabatuGUI interfazean aurkitzen da eta dituen baliabideak, data bat aukeratzeko dira eta behin data bat aukeratuta, automatikoki getEvents() metodoa martxan jarriko da eta programak egun horretako gertaera guztiak erakutsiko ditu.

“Remove” botoia sakatuko du eta orduan negozio logikak eskaintzen duen ezabatuGertaera() metodoari dei egingo dio. Hori bai, hau egin ahal izateko, data etorkizunekoa izan beharko da. Parametro gisa, gertaera pasatuko zaio.

Datu basearekin, berdina egingo da, parametro berdina emanaz.

Datu basean, getEventNumber() metodoa erabiliko da gertaeraren identifikadorea jasotzeko eta entityManager-aren find() metodoa erabilita, gertaera jasoko dugu.

Une honetan, programak, gertaera horren galdera guztiak aztertuko ditu, galdera bakoitzaren pronostiko guztiak ere eta pronostiko bakoitzaren apustuak ere.

Hasteko, apustuari bere pronostiko listatik une horretan aztertzen hari den pronostikoa ezabatuko zaio, removePronostikoa() metodoa erabilita eta apustuari geratzen zaizkion pronostiko kopurua itzuliko du.

Gainera, `getBezerao()` (uneko apustuaren bezeroa jasotzeko), `getKopurua()` (apustuan apostatu egin den diru kopurua itzultzeko), `getErrepikatua()` (apustua errepikatu zaion bezeroari), `getAsmatutakoKop()` (apustuan asmatu diren pronostiko kopuruak) eta `getPronostikoaKop()` (apustua dauden pronostiko kopurua) metodoak erabiliko ditu.

Orduan, `removePronostikoa()` metodoak itzuli duen kopurua 0 bada, kalkulu gehiago egin beharko dira. Gainera `getErrepikatua()` metodoak bezero bat itzuli badu, errepikapen horri apustuaren diru kopurua eguneratu beharko zaio, `eguneratuHilHonetanGeratzenDena()` metodoa erabilita eta `getKopurua()` metodoak emandako balioa parametro gisa emanda.

Honen ondoren, bezeroari mugimendu berri bat adierazten zaio, `addMugimendua()` metodoa erabilita, gertaera ezabatu dela adieraziz eta dirua itzulita eta `removeApustua()` metodoa erabili eta apustua parametro gisa ematen zaio.

Beste kasuan, hau da, `removePronostikoa()` metodoak 0 balioa ez badu itzultzen, `getAsmatutakoKop()` eta `getPronostikoKop()` metodoek ematen dituzten balioak berdinak badira, beste aukera posible bat egongo da.

Hasteko, `getKuotaTotala()` metodoa erabiliko du eta `getErrepikatua()` metodoak bezero bat itzuli badu, errepikatuak kobratuko lukeen komisioa kalkulatu dio eta `addMugimendua()` metodoarekin mugimendu bat adieraziko dio komisioaren dirua itzultzeko.

Ondoren, irabaziare diru guztia kalkulatu du eta komisioa kenduko dio. Behin hori egin, apustua irabaztearen mugimendua adieraziko dio `addMugimendua()` metodoarekin.

Hau guztia apustu guztiekin egin ondoren, datu baseak `entityManager`-aren `remove()` metodoa erabiliko du eta gertaera ezabatuko du. Behin ezabatuta, transakzioa amaituko du.

## Apustua ezabatu

Bezerao `ApustuaEzabatuGUI` interfazean egonik, lehendabizi berak egindako apustuak lortzen dira `getApustuak()` metodoaren bidez, non jasoko dena izango den Apustua motako objektuen bektorea.

Behin bezeroak ikusgai dituela, bat hautatuko du eta Apustua objektu horren Pronostikoa motako objektuak lortuko ditu `getPronostikoaKop()` metodoaren bidez eta interfazeak erakutsiko ditu. Hori bada ezabatu nahi duen Apustua, behean azalduko zaio "Delete" izeneko botoia eta hor sakatuta prozesua martxan jarriko da: negozio logikako `deleteApustua()` metodoari dietzen zaio parametro gisa hautatutako Apustua objektua eman. Aldi berean, metodo honek Bezerao motako objektua itzuliko du, zehazki une horretan interfazean aurkitzen den Bezerao objektua izango da baina eguneratua. Negozio logikak datu-basearekin jarriko da harremanetan eta `deleteApustua()` metodoari deituko dio parametro berdina eman.

Datu-basean jarraitzen den prozedura honakoa da, guztia transakzio baten barruan burutuz: lehendabizi Apustua objektua lortzen da datu-basetik, eta `getPronostikoak()` metodotik lortutako Pronostikoa objektu bakoitza begiratzeko da, ziurtatuz dagoeneko ez dela oraindik amaitu Pronostiko horren gertaera. Hala izanez gero, Pronostikoa objektuak bere Apustua objektu zerrendatik ezabatuko du `removeApustua()` eginez, eta aurkako kasuan `EventFinished()` salbuespena jaurtiko du.

Aurrera jarraituz, `getBezeroa()` eginez apustua egin duen Bezeroa objektua lortzen da, eta bere Apustua objektu zerrendatik ezabatuko da `removeApustua()` eginez eta jokaturako apustu kopurua jaitsi. Ezabatu duen apustua errepikapen baten bidez sortua izan bada (`getErrepikatua()!=null`), Errepikapena objektua lortu eta hil horretan geratzen den diru kantitatea eguneratzen da apostatu duen diru kantitatea gehituz (lehendabizi `getErrepikapena()` egiten da eta Errepikapena objektua lortzean `eguneratuHilHonetanGeratzenDena()` egiten da). Hori eginik, Mugimendua objektu berri bat sortuko da eta Bezeroa objektuaren Mugimendua objektuen bektorerara gehituko da `addMugimendua()` eginez, kasu honetan testua "Apustua ezabatu" izanik.

Norbanakoaren apustua ezabaturik dago, baina datu-basetik oraindik ez (hori izango da azken agindua); baina, gure aplikazioaren kudeaketa politikak dio bezero bat errepikatua bada, eta egindako apustu bat ezabatzen badu, errepikatzaleei ere ezabatu egingo zaie apustu hori. Beraz, orain errepikatzaleak lortu behar dira (`getErrepikatzaleak()`) eta banan-banan joan egiten ondorengo metodologia: oraindik Apustua objektu hori aurkitzen bada bere Apustua objektuen zerrendan, bertatik ezabatu (`removeApustua()`) eta Mugimendua berria sortu (`addMugimendua()`) behar da, eta Pronostikoa objektu bakoitzeko zerrendatik ere ezabatu (`removeApustua()`). Errepikapena objektua ere eguneratu beharrean dago, zehazki `hilabeteHonetanGeratzenDena` atributua apostatutako kantitatea gehitu behar baita (eguneratuHilHonetanGeratzenDena()). Azkenik, datubasetik ezabatzen da errepikapen apustu hori.

Behin errepikapen guztiekin prozesu hori egitean, datu-basetik ezabatzen da apustua eta transakzioa amaitutzat emanik, Bezeroa motako objektua eguneratua itzuliko da.

## Arreta zerbitzua eskatu

Bezeroa ArretaZerbitzuaEskatuGUI interfazean aurkituko da, non gaia eta testua idatziko dituen bere arazoa adieraziz. Behin hau buruturik, botoia sakatuko du, eta honek datuak prozesatuko ditu ea ongi dauden ala ez. Ongi egonez gero, aurrera jarraituko du, eta negozio logikako `arretaElkarrizketaSortu()` metodoari emango dizkio alde batetik Bezeroa objektua eta bestetik jasotako edukia: gaia eta testua. Metodo honek ArretaElkarrizketa motako objektu bat itzuliko du eta interfazeak erabiliko du. Negozio logikan jarraituz, datu-baseko `arretaElkarrizketaSortu()` metodoari deituko dio eta parametro berdinak emango dizkio. Azkenik, datu-basean honako pausoak jarraitzen dira `arretaElkarrizketaSortu()` metodoan: Bezeroa motako objektua lortzen da datu-basetik lehendabizi, eta behin lortuta, transakzio bat hasten du: lehendabizi ArretaElkarrizketa objektu berri bat sortzen du `addElkarrizketa()` metodoaren bidez, parametroa gaia izango da, eta objektu hori itzultzen du metodoak. Objektu hori izanik, `addMezua()` metodoa erabiltzen da, ArretaMezua motako



instantzia berri bat sortu eta ArretaElkarriketa objektuari gehitzeko. Azkenik, datu-basean gordetzen da ArretaElkarriketa objektua persist() eginez eta transakzioa itxi egingo da.

## Langileari mezua bidali

Bezeroa PostontziaGUI interfazera sartzean martxan jarriko den lehendabiziko gauza izango da jasotako mezu guztiak lortzea; horretarako, negozio logikako getMezuak() metodoari deitze zaio parametro gisa Bezeroa objektua emanik. Metodo honen helburua izango da emandako Bezeroa motako objektuak datu-basean dituen Mezua motako objektu guztiak lortzea. Horretarako, datu-basera joko du getMezuak() metodoak getMezuak() izeneko metodoaren bidez eta parametro gisa jasotako Bezeroa objektua emango dio. Datu-basean, Bezeroa objektua lortuko du eta getMezuak() metodoari deituko dio jasotako Mezua motako objektu guztiak lortzeko (detaile txikibat, hauek dataren arabera ordenaturik aurkitzen dira).

Beraz, bezeroak jasotako Mezua motako objektu guztiak ikusten ditu eta bat hautatzen du, zehazki ArretaMezua motako langile batek bidalia, eta testu bat idazten dio bidaltzeko. Beraz, botoia sakatzean emandako parametroak ongi daudela ziurtatu ostean prozesua martxan jartzen da: ArretaMezua objektuaren ArretaElkarriketa objektua lortzen du getElkarriketa() metodoaren bidez (interfazean inplementatua dagoena), eta ondoren negozio logikako arretaMezuaBidali() metodoari deitzen dio parametro gisa testua, true boolearra eta lortutako ArretaElkarriketa objektua.

Negozio logikan, datu-baseko arretaMezuaBidali() metodoari deitzen zaio parametro berdinak emanez, eta datu-basean jarraituko den sekuentzia honakoa da: ArretaElkarriketa objektua lortuko du datu-basetik eta objektu horretara ArretaMezua motako objektu berri bat gehituko du transakzio berri batean. Azkenik, negozio logikako mezualrakurri() metodoari deitzen zaio hautatutako ArretaMezua emanik parametro gisa, non metodo honen helburua izango den datu-basean objektua lortzea eta ondoren irakurrita atributua true ezartzea setIrakurrita(true) metodoaren bidez.

## Elkarriketa gelditu

ArretaZerbitzuaEmanGUI interfazera sartzean langilea, lehendabiziko gauza egiten dena honakoa da: langile horrek dituen ArretaElkarriketa motako instantzia guztiak lortzen dira. Horretarako, aldez aurretik negozio logikatik getLangilea() metodoari dei egiten zaio parametro gisa Langilea motako objektu zaharra emanik, eguneratutako Langile motako objektua lortzeko. Hori izanik, getArretaElkarriketa() metodoari deitzen zaio ArretaElkarriketa objektuz osatutako bektorea lortzeko. Hauek interfazeak pantailaratu egingo ditu eta hauetako bat hautatuko du langileak. "Gelditu" botoia sakatzean, negozio logikako geldituElkarriketa() metodoa jaurtiko da, non parametro gisa hautatutako ArretaElkarriketa motako instantzia hartuko duen. Honek, aldi berean, datu-baseko geldituElkarriketa() metodoari deituko dio, eta azken honi berak jasotako parametroa emango dio. Datu-basean geldituElkarriketa() metodoak honela funtzionatzen du: ArretaElkarriketa objektu hori lortzen du datu-basetik, eta transakzio baten bidez, ArretaElkarriketako gelditu() metodoari deituko dio, non honek egiten duena izango da

langileari bere elkarriketa zerrendatik ezabatu `removeElkarriketa(this)` eginez, eta ondoren atributu hori null-era bihurtuz.

## Bezeroa lortu

Langilea `ArretaZerbitzuaEmanGUI` interfazeaz egonik, goian botoi bat ikusi dezake “Get A Costumer” izeneko, eta hau sakatzean prozesu bat hasten da, non helburua izango den `ArretaElkarriketa` motako instantzia bat lortzea non Langilea motako objekturik ez dagoen lanean berarekin (hau da, langilea atributuaren balioa null izatea) eta amaituta ez egotea. Horretarako, `bezeroaEsleitu()` izeneko metodoari deituko zaio negozio logikatik, parametro gisa Langilea objektua emanik, eta `ArretaElkarriketa` motako instantzia bat itzuliko du ondoren interfazeak erabiltzeko. Negozio logikan, datu-baseko `bezeroaEsleitu()` metodoari deitzen dio eta parametro berdina emango dio. Datu-basean, metodo honek honela funtzionatzen du: `createQuery` bidez lortuko ditu `ArretaElkarriketa` motako objektuan non bere langilea atributua null izango den eta amaituta atributuak false izango duen gordeta. Zerrenda hori izanik, emandako Langile objektua lortuko du datu-basetik, eta transakzio berri bat hasiko du bi gauza egiteko: alde batetik, `ArretaElkarriketa` objektu horri Langilea motako objektua esleitzeko (`setLangilea()` eginez<sup>9</sup> eta bestetik Langilea objektuari `ArretaElkarriketa` motako objektua gehitu bere elkarriketa zerrendara (`addElkarriketa()` metodoa).

## Bezeroari mezua bidali

Langilea `ArretaZerbitzuaEmanGUI` interfazera sartzean, lehendabiziko gauza egingo dena izango da langile horrek amaitu gabe dituen `ArretaElkarriketa` guztiak lortzea izango da; horretarako, Langilea motako objektu eguneratua lortuko da negozio logikako `getLangilea()` metodoarekin, parametro gisa langilearen erabiltzaile izena emanik. Behin Langile objektu hori izanik, `ArretaElkarriketa` objektuak dituen bektorea lortuko da Langilea objektutik `getArretaElkarriketak()` metodoaren bidez.

Horietatik elkarriketa bat hautaturik, mezu bat idatziko du langileak eta behean aurkitzen den botoia sakatzean, bezeroari mezua bidaltzeko prozesua martxan jarriko da baldin eta emandako datuak egokiak badira. Negozio logikako `arretaMezuaBidali()` metodoari deituko zaio prozesu hau aurrera eramateko, honek emandako `ArretaElkarriketa` objektu eguneratua itzuliz. Beraz, parametro gisa joango den lehenengo gauza izango da hautatutako `ArretaElkarriketa` objektua, eta beste biak izango dira false boolearra eta idatzitako mezua.

Negozio logikako metodoak datu-baseko `arretaMezuaBidali()` metodoari dei egingo dio eta parametro berdina emango dizkio, eta behin datu-basean egonik honako ordena jarraituko da: `ArretaElkarriketa` objektua lortuko da datu-basetik, eta transakzio berri bat abiatuko da `ArretaMezua` motako objektua gehitzeko bertara `addMezua` eginez.

## Elkarriketa amaitu

Elkarriketa geldituen oso antzeko sekuentzia diagrama du, baina aldaketa dator botoia sakatzerakoan eta hortik aurrera jarraitzen den prozesuan: “Amaitu” botoia sakatzean, negozio logikako amaituElkarriketa() metodoari deitzen zaio parametro gisa hautatutako ArretaElkarriketa motako instantzia. Metodo honek, aldi berean, datu-baseko amaituElkarriketa() metodoari deitzen dio, eta elkarriketa gelditu erabilpen kasuan bezala, negozio logikako metodoak jasotako parametroa datu-baseko metodoari ematen dio. Behin datu-baseko metodoak honela funtzionatzen du: datu-basetik ArretaElkarriketa objektu hori lortzen du eta transakzio bat abiatzen du ondorengo egiteko: bezeroak bidalitako ArretaMezua motako objektuen bektorea lortzen ditu lehendabizi getBezeroakBidalitakoak() metodoari esker eta hauek datu-basetik ezabatzen ditu; jarraian, getLangilea() eginez elkarriketan parte hartu duen Langilearen objektua lortuko du eta bere elkarriketa zerrendatik ezabatuko du removeElkarriketa(elkar) eginez. Lehenengo zatia amaituta, bigarren zatia hasten da orain: langileak bidalitako ArretaMezua motako objektuak lortuko ditu getLangileakBidalitakoak() metodoaren bidez, eta hauek aztertzen joango da banan-banan: ikusgaBezeroarentzat atributua false bada (atributuaren balioa iskusgarriBezeroarentzat() metodoa erabiliz lortuko du) datu-basetik ezabatuko du, kontrako kasuan irakurrita atributuaren balioa true ezarriko du setIrakurrita(true) eginez. Jarraian, mezu berri bat gehituko du ArretaElkarriketa objektuari, non bere edukia hutsa izango den eta azkeneko mezua delako adieraziko du setAzkena(true) eginez. Transakzioa amaitzeko, setAmaituta(true) egingo du ArretaElkarriketa objektuan.

## Langilea puntuatu

Bezeroak bere PostontziaGUI interfazeaz sartzean, ikusi ditzake jaso dituen mezu guztiak. Horien artean, mezu bat egongo da administratzaileak bidalitakoa, non honen amaieran azalduko den puntuazioa ezartzeko toki bat. Mezuak erakusteko prozesua kasu honetan ez du zentzurik erakustek; izan ere, mezu konkretu bat baita administratzailearena, eta langile batekin elkarriketa amaitzen denean soilik bidaltzen dena, berez badakigu zein izango den hautatutako mezua. Orduan, puntuazio bat jarriko dugu 1etik 5era, zenbaki osoa izango dena, eta botoia sakatzean martxan jarriko da prozesua: negozio logikako gehituPuntuazioa() metodoari deituko dio parametro gisa Bezeroa objektua, puntuazioa eta mezuaren ArretaElkarriketa objektua emanik (interfazeaz badago metodo bat getElkarriketa() izenekoa ArretaMezua motako objektua emanik bere ArretaElkarriketa itzultzen duena). Metodo honek, datu-baseko gehituPuntuazioa() metodoari deitu eta parametro berdinak emango dizkio honek lan guztia egiteko. Azkenik, datu-baseko metodoak jarraian datorren prozedura burutuko du: datu-basetik ArretaElkarriketa motako objektua lortuko du, eta objektu hori izanik getLangilea() eginez elkarriketa horretan parte hartu duen langilearen Langilea objektua lortuko du. Transakzio bat abiatuz, Langilea klaseko addBalorazioa() metodoari deituko dio non parametro gisa puntuazioa balioa emango dion; honela, Langileak bere barnean duen balorazioak izeneko atributuan gordeko da bere balioa.

## Errepikatzeko eskaera egin

Bezeroak ErrepikatzekoEskaeraEginGUI interfazean dagoenean, ikusi dezake bilatzaile moduko bat, non berak karaktere batzuk idatziko dituen eta “Search” botoia sakatzean, bilaketa bat egingo da datu-basean non Bezeroa objektuaren erabiltzailelzena atributuak karaktere sekuentzia hori izango duen. Ez hori bakarrik, beste hainbat irizpide ere bete behar ditu: unean interfaze hori erabiltzen ari den Bezeroa objektuaren erabiltzailelzen atributu ezberdina izan behar du, ikusgarritasuna baimenduta izan behar du aplikazioko beste bezeroekiko (publikoa atributua true balioa izatea, zehazki), uneko bezeroaren BezeroartekoMezua klaseko objekturik ez izatea eta dagoeneko bi bezeroen artean erlaziorik ez izatea (Errepikapena objektu baten bitartez esan daiteke erlazioa dutela bi Bezero objektu ezberdinek). Baldintza horiek betetzen dituzten Bezeroa motako objektuak lortzen diren negozio logikako getBezeroak() metodoaren bidez, non parametro gisa bezeroak sartutako karaktere sekuentzia eta Bezeroa objektua izango dituen. Negozio-logikako metodo honek, datu-baseko izen bereko metodoari deitzen dio parametro berdinak emanez, eta zati garrantzitsuena hemen aurkitzen da.

Lehendabizi Bezeroa objektua lortzen da datu-basetik, eta ondoren createQuery bidez Bezeroa objektu guztiak. Behin horiek izanik, iterazio batean sartuta honako prozedura ematen da aurrera (teorian if baten barruan sartzen dira baina hemen zatika azalduko dugu): isPublikoa() erabiliz publikoa atributuaren balioa lortzen da, getErabiltzailelzena() erabiliz erabiltzailelzena, baduMezua() funtzioaren bidez parametro gisa emandako bezeroak ea aztertzen ari den bezeroaren mezurik baduen ala ez lortzen da boolear batez adierazita; horretarako, bai bidalitako eta jasotako mezuetan ea badagoen Bezeroa objektu hori, eta azkenik, errepikapenErlazioaDu() funtzioa erabiliz jakingo da ea bi Bezero objektuek baduten biak lotzen dituen Errepikapena klaseko objekturik, balio boolear baten bidez adierazita. Ba, publikoa atributuak true balioa badu, erabiltzaile izen biak ezberdinak badira eta karaktere sekuentzia hori badu, baduMezua() funtzioak false itzultzen badu eta errepikapenErlazioaDu() funtzioak false itzultzen badu, irteerako Bezeroa objektuen bektorean sartuko da.

Behin lortuta Bezeroa objektu horiek, bat hautatzen da, eta zenbait datu ematen zaizkio interfazeari: mezua, euro kopuru maximoa gastatuko duena hilabeteko eta bezeroaren euro bakoitzerako berak zenbat apostatuko duen. Datu hauek interfazeak hartzen ditu eta hautatutako Bezeroa objektutik komisiAutomatikoa atributuaren balioa hartuko du getKomisiAutomatikoa() getter-ari esker. komisiAutomatikoa atributuaren arabera, mezu ezberdinak bidaliko dira eta hemen bi kasuan aztertuko ditugu.

- Baldin eta komisiAutomatikoaren balioa -1 bada, negozio logikako bidaliMezua() metodoari dei egingo zaio, parametro gisa honakoak bidaliz: mezua unean aplikazioa erabiltzen ari den Bezeroa objektuak bidaliko du eta helburua hautatutako Bezeroa objektua izango da, mezuaren testua, gaia “Repeat Request”, mota “eskaera”, euroko apostatuko duen kantitatea, hilabeteko maximoa eta -1 balioa. Negozio logikak datu-baseko izen berdineko funtzioari jasotako parametro guztiak emango dizkio, eta prozesu hau emango da aurrera: datu-basean bi Bezero objektuak bilatu eta hartzen dira, eta transakzio batean sortzen da bidaliko den BezeroartekoMezua (objektu hau bidaltzen duenak sortzen du) addBidalitakoBezeroMezua() funtzioaren

bidez, non BezeroartekoMezua jasoko duen itzuleran. Ondoren, jasotzaileak jasoko du bere BezeroartekoMezua objektuen bektorean addJasotakoBezeroMezua() funtzioa erabiliz, eta azkenik persist eginez gordeko da datu-basean objektu berria.

- Aurkako kasuan ere negozio logikako bidaliMezua() metodoari dei egiten zaio, baina parametroak aldatzen dira: mezua hautatutako Bezeroa objektuak bidaliko du eta helburua aplikazioa erabiltzen ari den Bezeroa klasea izango da, testua "IAccept" izango da, gaia "Acceptance", mota "errepikatuak eskaera onartu", euroko apostatuko duen kantitatea, hilabeteko maximoa eta komisiAutomatikoa atributuaren balioa. Hemendik aurrerako prozedura, bestearen berdina da: datu-baseko bidaliMezua() metodoari dei egiten dio parametro berdinak emanaz eta prozedura jarraituz bi Bezero objektuak bilatzen ditu eta BezeroartekoMezua klaseko objektu berri bat sortzen du datu horiekin eta bi Bezeroek dagokien bektorean gordetzen dute.

Kontua da komisiAutomatikoa atributuaren balioak duela garrantzia: 0 eta 1 arteko balioa denean adierazi nahi da automatikoki erantzuten duela sistemak eta mezua bidaltzen dio errepikatuaren izenean esanez "Ados, onartzen dut eta komisihoa honakoa da: X" non X hori konstantea den bezero guztientzat; baina, -1 balioa badu esan nahi du errepikatu izango den bezero horrek adostu egin nahi duela errepikatzaile bakoitzarengandik jaso nahi duen komisihoa; beraz, bezero bakoitzarekin izango duen tratua ere ezberdina izango da.

## Errepikatzeko eskaerari erantzun

Bezeroa PostontziaGUI interfazean bere mezuak izango ditu; izan ere, interfazea martxan jarri orduko getMezuak() metodoak unean interfazea erabiltzen ari den Bezeroa objektuaren Mezua klaseko objektu guztiak lortzen ditu datu-basetik. Behin mezuak ikusgarri dituela, bezero baten mezu bat hautatzen du: baldin eta mezu honen mota "eskaera" bada (getMota() erabiliz lor daiteke zer motatako mezua izango den BezeroartekoMezu hori) baliabide batzuk erakutsiko ditu pantailan eta erabiltzen ari den bezeroak bete egin beharko ditu aurrera jarraitzeko: onartu izeneko hautaketa egin beharko du "Yes" ala "No" izenekoa. Onartu atributuaren balioa "Yes" bada, eta onura atributuaren balioa 0-1 artean aurkitzen bada (balio hau mezuari getZenbatErrepikatuarentzat() eginez lor daiteke), mezu bat bidaliko dio negozio logikako bidaliMezua() metodoa erabiliz, non igorlea unean aplikazioa erabiltzen ari den Bezeroa izango den, hartzailea eskaera bidali duen Bezeroa objektua izango da, mezua "IAccept", gaia "Acceptance", mota "errepikatuak eskaera onartu" eta beste datuak hautatutako mezutik lortutakoak dira getter bidez (hartzailea ere hautatutako mezuan geter bat erabiliz lortua izan da).

Beraz, negozio logikak datu basean izen berdina duen metodoari dei egiten dio eta parametro berdinak ematen dizkio, hemendik aurrerako prozesua lehenagotik ere aipatu dugu eta modu laburtuan aipatuko dugu: Bezeroa objektuak lortzen ditu datu-basetik, eta transakzio batean BezeroartekoMezua objektua sortu eta Bezero bakoitzari dagokion BezeroartekoMezua bektorean gordeko dio, azkenik datu-basean gordez persist eginik. Ez hori bakarrik, hautatutako mezua irakurrita dagoela adierazteko mezualrakurri() metodoa erabiliko da negozio logikakoa, non datu-basean bilatuko duen BezeroartekoMezua objektua

eta setIrakurrita(true) egingo duen. Interfazeko instantzian zuzenean setIrakurrita(true) eginik nahikoa izango da.

Azkenik, aurkako kasua aztertu behar da; izan ere, onartu “No” bada, mezua bidaliko dio eskaera egiten ari denari bai, baina mezu honek edukia ezberdina izango da; izan ere, testu gisa “NoAcceptRequest”, gaia “Denial” eta mota “errepikapen eskaera ukatu” izango baita, baina igorlea, hartzailea eta errepikapenaren inguruko datuak berdinak izango dira; beraz, aurreko prozedura berdina jarraituko luke: negozio logikako mezuaBidali() metodoari parametro guzti horiek pasa, honek datu-basearekin konektatu eta mezuaBidali() metodoari pasa parametro guztiak, eta azken honek Bezeroa objektuak lortu, BezeroartekoMezua objektu berria sortu, bi Bezeroa objektuei gehitu eta datu-basean gorde persist eginez. Bukatzeko, “Yes” kasuan bezala, irakurrita dagoela adierazteko interfazeko instantziari setIrakurrita(true) egin eta negozio logikako mezualrakurrita() metodoari dei egin.

## Errepikapena baieztatu

Bezeroa PostontziaGUI interfazean egongo da, eta BezeroartekoMezu bat izango du, non erantzun bat eman behar duen; hau da, errepikatuak erantzun dio eta proposamen bat egiten dio eta erantzuna izango da baiezkoa ala ezezkoa, aukeratzeko botoi baten bidez emango du: bai ala ez izango dira botoiak. Beraz, bai sakatzen bada, esan nahi du onartzen duela eskaera eta errepikatzen hasiko dela. Eta hori da erabilpen kasu hau azken finean, errepikapen proposamena onartua izatea.

Beraz, baiezkoa izan da eta botoia sakatzen du onartzeko; beraz, prozesua martxan jartzen da: lehendabizi negozio logikako bidaliMezua() metodoari dei egiten zaio, kasu honetan aplikazioa erabiltzen ari denak errepikatuari bidaliko dio mezua esanez “StartRepeat”, gaia “StartRepeating”, mota “errepikatzen hasi” eta errepikapeneko balioak lehenagotik definituak izango direnekin. Beraz, datu-basera joko du izen berdineko metodoari dei egiteko parametro berdinak emanez, eta errepikapen eskaera egin eta errepikapen eskaera erantzun diagrametan egiten den berdina egiten du bidaliMezua() metodoak: Bezero objektuak lortu, BezeroartekoMezua klaseko objektua sortu eta bi Bezero objektuetan gehitu dagokien atributuan eta datu-basean gorde, guztia transakzio berdinean eginez.

Hori egina dago, baina orain mezua irakurrita dagoela adierazi behar da; horretarako, mezualrakurri() metodoarekin egingo da parametro gisa hautatutako mezua izanik. Metodo honetan, datu-basera joko da eta bertan BezeroartekoMezua objektu hori lortuko da eta irakurrita atributuaren balioa true definituko setIrakurrita(true) eginez, guztia transakzio baten barruan.

Azkenik, Errepikapena objektua sortu beharra dago, eta horretarako errepikatu() metodoa dugu parametro gisa errepikatzaila izango den Bezeroa objektua, errepikatua izango den Bezeroa objektua, euroko apostatuko duen kantitatea, hilabeteko maximoa eta komisioa emanez, azken hiru parametro hauek BezeroartekoMezutik lortu ahal izango ditu. Beraz, datu-baseko izen bereko metodoari dei egingo dio parametro berdinak emanez eta datu-basean honako sekuentzia eramango da aurrera: Bezeroa klaseko bi objektuak lortuko dira datu-basetik, eta transakzio batean errepikatuari Errepikapena objektua sorraraziko

diogu `addErrepikatzalea()` metodoaren bidez; izan ere, metodo honek Errepikapena objektua sortzeaz gain itzuli egiten baitu eta. Behin objektu hori izanik, errepikatzaleari gehituko diogu `addErrepikatua()` metodoaren bidez, eta azkenik datu-basean gordeko dugu.

## Ezarpenak aldatu

Bezeroa EzarpenakGUI interfazean aurkituko da, non hemen zenbait datu emango dituen bere ezarpenak aldatzeko: ikusgarritasuna eta komisio automatikoa.. Behin datuak emanik, gordetzeko botoiari emango dio, eta interfazeak datuak ongi sartu direla ziurtatuko du, eta hauek egokiak badira, ezarpenak eguneratzera igaroko da; horretarako, negozio logikako eguneratuEzarpenak() metodoari deituko dio parametro gisa Bezeroa motako objektua eta lortutako datuak emango dizkio. Negozio logikak datu-baseko eguneratuEzarpenak() metodoari deituko dio aldi berean, parametro berdinak emanez, non azken hau arduratuko den lehendabizi Bezeroa objektua lortzen datu-basetik eta ondoren, transakzio baten bidez, Bezeroa objektuak duen eguneratuEzarpenak() metodoa erabiliko du datuak aldatzeko, azken honi parametro gisa hasieran interfazeak lortutako ikusgarritasuna eta komisio automatikoa.

## Errepikapena ezabatu

Bezeroak ErrepikatzaleakGUI interfazea irekitzean, interfazea bere jarraitzaile eta jarraituen errepikapenak lortuko ditu: horretarako eguneratu() izeneko metodo bat dago, non honek negozio logikako getErrepikatzaleak() eta getErrepikapenak() metodoei dei egiten dien bi kasuetan Bezeroa motako objektua emanez. Horrela, datu-basera joango da eta datu-basetik Bezeroa motako objektuaren errepikatzaleak eta errepikatuak atributuak osatzen dituen Errepikapena motako bektorea lortuko du. Bi bektoreak lortuta, datuak interfazearen bidez pantailaratuko ditu.

Behin interfazean datuak izanik, Bezeroak bi zerrendetatik bat hautatuko du, berdin du zein zerrendatik izan prozedura berdina da: Errepikapena hautatua izanik, negozio logikako jarraitzeariUtzi() metodoaren bidez Errepikapena motako objektu hori datu-basetik ezabatuko du, baina aurretik bezero bakoitzari ezabatuko zaio Errepikapena objektu hori.

## Hilabeteko diruak hasieratu

Admin erabiltzaileak bere AdminGUI interfazean botoi bat izango du “Reset montly account” izenekoa (defektuzko izena), non hau sakatzean erabilpen kasua aurrera eramango den: negozio logikako eguneratuErrepikapenak() metodoari dei egiten zaio, non honek aldi berean datu-baseko eguneratuErrepikapenak() metodoari dei egiten dion. Helburua honakoa da: datu-basean aurkitzen diren Errepikapena motako objektu guztietan hilabeteHonetanGeratzenDena izeneko atributua hasieratzea. Beraz, datu-baseko eguneratuErrepikapenak() metodoa hastean, honek lehendabizi createQuery baten bidez datu-basean aurki daitezkeen Errepikapena motako objektu guztiak lortuko ditu; behin

lorturik, bakoitzarekin jarraituko beharreko prozesua honakoa da: `getHilabetekoMax()` getterra erabiliz, hilabeteen gehienez gastatuko duen balioa lortuko dugu eta ondoren lortutako balio hori ezarri behar zaio `hilabeteHonetanGeratzenDena` atributuari, gure kasuan `setHilabeteHonetanGeratzenDena()` setterraren bidez lor dezakegu hori parametro gisa lehendik lortutako balioa emanez.

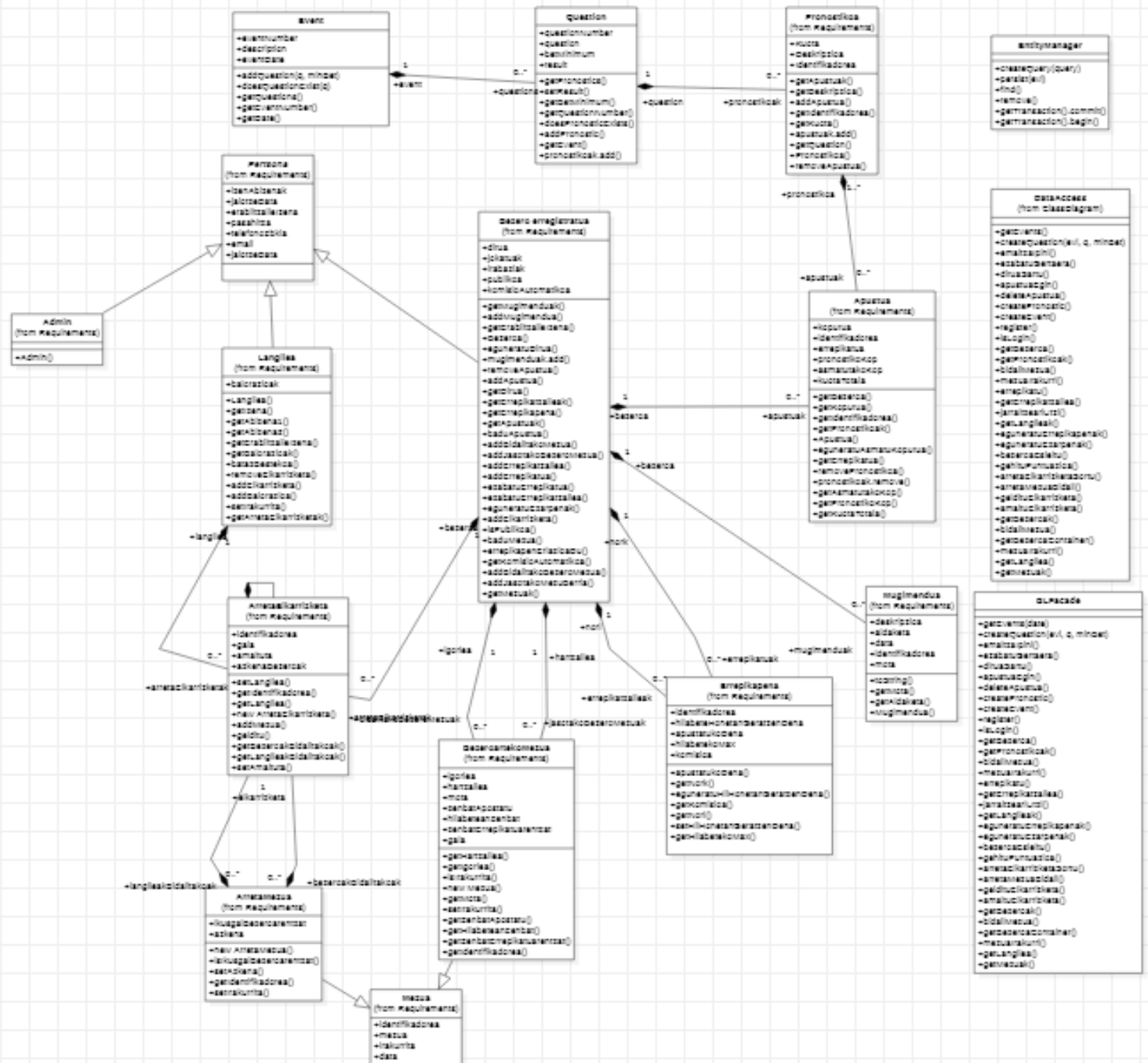
## Langileen puntuazioa ikusi

Erabilpen kasu honen sekuentzia diagramaren azalpena honakoa da: Admin erabiltzaileak `PuntuazioakIkusiGUI` izeneko interfazea irekitzean negozio logikako, `BLFacade` zehazki, `getLangileak()` metodoari deitzen dio, non honek datu-basean aurkitzen diren Langile motako objektuen bektorea itzuliko dion. Horretarako, datu-baseko `getLangileak()` metodoari deituko dio negozio logikatik, non lehenengoak `createQuery` baten bidez datu-basean aurkitzen diren Langile motako objektu guztiak itzuliko dizkion bektore batean.

Behin bektorea izanik, iterazio batean sartuko da, non bektoreko elementu bakoitza hartuta honako prozesua jarraituko duen azkenik pantailan Admin-ak ikusteko langile bakoitzaren inguruko informazioa: `getIzena()`, `getAbizena1()` eta `getAbizena2()` metodoen bitartez langilearen izen-abizenak lortuko ditu, eta `getErabiltzaileIzena()` erabiliz langileak aplikazioa erabiltzeko duen erabiltzaile izena lortuko du; jarraian balorazioak lortuko du `getBalorazioak()` eginez eta `batazBestekoa()` metodoari esker langilearen batez besteko puntuazioa lortuko dira. Azkenik, lortutako elementu bakoitza `PuntuazioakIkusiGUI` interfazeak hartu eta taula batean adierazita langile bakoitzaren informazioa erakusten du.



## Klase diagrama



# Inplementazioa

Gure erabilpen kasu guztiak posible egin ahal izateko, metodo ugari gehitu behar izan ditugu, BLFacade interfazeaz. Jarraian hauek zerrendatuko ditugu eta bakoitzak zertarako balio duen adieraziko dugu:

**public Pertsona isLogin(String erabiltzaileIzena, String pasahitza)**

Metodo honi, erabiltzaile izen bat eta pasahitz bat emanda, datu basean bi datu hauek dituen Pertsona klaseko objekturik baldin badago, pertsona bera itzultzen du. Bestela null balioa. Kontu batean saioa hasteko erabiltzen da.

**public Pertsona register(String izena, String abizena1, String abizena2, String erabiltzaileIzena, String pasahitza, String telefonoa, String emaila, Date jaiotzeData, String mota) throws UserAlreadyExist**

Bezero berri baten datuak emanda, erabiltzaile izen bereko Pertsona klaseko objektu berdin bat ez badago, erabiltzaile berri bat sortzen du. Hiru motatako erabiltzaileak sortzeko balio du, baina bezeroak sortzeko bakarrik aprobetxatu da. Erabiltzaile izen bereko erabiltzailearen bat existitzen bada salbuespen bat jaurtitzen du.

**public void createEvent(String description, Date eventDate) throws EventAlreadyExist**

Deskripzio bat eta gertaera data bat emanda gertaera berri bat sortzen du datu basean. Deskripzio bereko gertaeraren bat existitzen bada, salbuespen bat jaurtitzen du.

**Pronostikoa createPronostic(Question question, String description, double kuota) throws PronosticAlreadyExist**

Galdera bat, deskripzio bat eta kuota bat emanda, galderari sarturiko deskripzio eta kuota dituen pronostiko bat gehitzen dio. Galderak deskripzio bereko pronostiko bat baldin badauka, salbuespen bat jaurtitzen du.

**public void emaitzalpini(Question question, Pronostikoa pronostikoa)**

Galdera bat eta pronostiko bat emanda, galderari emaitza gisa eman den pronostikoa ezartzen zaio. Honetaz gain, pronostikoari eginiko apustuen bezeroei dagozkien diru mugimenduak eta dirua ematen zaizkie. Apustua errepikatua izan bada, dagokion bezero errepikatuari bere komisioa ematen zaio, eta komisio hori, apustuaren bezeroari deskontatzen zaio.

**public Bezeroa apustuaEgin(ArrayList<Pronostikoa> pronostikoak, double a, Bezeroa bezero)**

Pronostiko bat edo batzuk, diru kantitate bat eta bezero bat emanda. Bezeroari, pronostiko horren aldeko apustu bat egiten zaio, adieraziriko diru kantitatearekin. Honetaz

gain, bezeroak errepikatzailleak baditu, hauei ere apustu bera gehitzen zaie. Baina, bezero originalak sartu duen diru kantitatea errepikapenean adierazirik duten zifrarekin (euroko apostatuko dutena) biderkatuz. Bezero errepikatauek apustua egiteko dirurik ez badute ez zaie egingo. Eta hilabeteen gehienez apostatuko duten diru kopurua ez bada iristen baina zerbait gelditzen bada, diru horrekin guztiarekin egingo zaio apustua. Azkenik, bezero errepikatuei hilabeean apostatzeko gelditzen zaien zifra eguneratzen dute.

#### **public Bezeroa deleteApustua(Apustua a) throws EventFinished**

Apustu bat emanik, datu basetik hau ezabatzen du eta apustuaren bezeroari dirua itzultzen dio. Apustu hau errepikatzaileraren batek errepikatua bazuen, hauei ere apustua ezabatzen zaie, dirua itzultzen die eta hilabeteen apostatuko dutena eguneratzen die goruntz. Apustuaren pronostiko baten baten, galderaren gertaera amaiturik badago, salbuespena jaurtitzen du.

#### **public Bezeroa diruaSartu(double u, Bezeroa bezero)**

Metodo honi bezero bat eta diru kantitate bat pasata, bezeroari mugimendu bat eta beraz dirua gehitzen dio.

#### **public void ezabatuGertaera(Event event)**

Gertaera bat pasata, gertaera hau datu-basetik ezabatzen da. Gertaera honen galderaren bati baten batek apustua egin bazion, apustuari pronostiko hori ezabatzen zaio. Pronostiko gehiago baditu, bertan uzten da, bestela datu basetik ezabatzen da eta bezeroari dirua bueltatzen zaio. Apustu errepikatua bazen, hilabeteen gelditzen den diru kopurua eguneratzen zaio errepikapenari. Pronostikoa ezabatu ostean gelditu diren pronostiko guztiak asmatuak bazeuden, apustua irabazita kontsideratzen da eta bezeroari dirua ematen zaio. Apustua errepikatua bazen, errepikatuari dagokion komisioa ematen zaio.

#### **public Bezeroa getBezeroa(String ErabiltzaileIzena)**

Erabiltzaile izen bat pasata, erabiltzaile izen horri dagokion bezeroa pasatzen du. Datu-basean dagoen bezero eguneratua eskuratzeko erabiltzen da metodo hau. Adibidez, apustu eta mugimendu eguneratuak eskuratzeko.

#### **public Langilea getLangilea(String ErabiltzaileIzena)**

Aurrekoaren berdina baina langileak eskuratzeko.

#### **public Vector<Bezeroa> getBezeroak(String username, Bezeroa bezeroa)**

String bat eta bezero bat pasata, beren erabiltzaile izenean sarturiko String-a daukaten, kontu publikoa duten, eta sarturiko bezeroarekin errepikapen erlazorik ez duten bezeroak eskuratzen ditu metodo honek. Errepikapen eskaeraren bilatzailean erabilia.

**public Bezeroa bidaliMezua(Bezeroa nork, Bezeroa nori, String mezua, String gaia, String mota, double zenbatApostatu, double hilabeteanZenbat, double zenbatErrepikatuarentzat)**

Bi bezero adierazita, bata besteari aderaziko datu guztiak biltzen dituen mezu bat bidaltzeko balio du. Hau da, nork bezeroaren bidalitako mezuetan mezu bat gehitzen da, baita nori bezeroaren jasotakoetan.

**public Vector<Mezua> getMezuak(Bezeroa bezeroa)**

Bezero bat emanda, honek jasota dituen bezeroarteko mezu guztiak bueltatzen ditu.

**public void mezualrakurri(Mezua mezua)**

Mezu bat pasata, mezua irakurri dela adierazten da datu basean.

**public void removeMezua(Mezua mezua)**

Mezu bat pasata, mezua bezeroartekoa bada zuzenean datu-basetik ezabatzen da. Arreta mezu bat bada, elkarriketa ez bada amaitu bezeroarentzat ikusgaitz gisa jartzen da (langileek oraindik ikusteko aukera izan dezaten). Elkarriketa amaiturik badago, ezabatu egiten da mezua, eta elkarriketak jada ez badu langileak bidaliriko mezurik elkarriketa ezanbatzen da datu basetik.

**public Bezeroa eguneratuEzarpenak(Bezeroa b, double komisioa, boolean publikoa)**

Bezero bat, komisio automatiko bat eta kontua publikoa edo pribatua jarri nahi den adierazirik, bezeroaren komisioAutomatikoa eta publikoa atributuak eguneratzen dira.

**public void errepikatu(Bezeroa nork, Bezeroa nori, double apustatukoDena, double hilabetekoMax, double komisioa)**

Bi bezero eta errepikapen ezaugarri batzuk emanik, Errepikapena klaseko objektu berri bat sortzen da adierazitako datu eta bezeroekin.

**public Vector<PronostikoaContainer> getPronostikoak(Apustua a)**

Apustu bat emanik, bere pronostikoak, pronostikoen galdera eta gertaerekin batera itzultzen ditu.

**public ArretaElkarriketa arretaMezuaBidali(ArretaElkarriketa elkarriketa, String mezua, boolean langileari)**

ArretaElkarriketa bat, mezu bat eta ea mezua langilearentzat doan adierazirik, elkarriketari arreta mezu berri bat gehitzen dio.

**public ArretaElkarriketa bezeroaEsleitu(Langilea langilea)**

Langile bat pasata, langilerik esleitu gabe eta amaitu gabe dagoen elkarriketarik existitzen bada datu basean, elkarriketari sarrerako langilea esleitzen zaio eta elkarriketa bueltatzen da.

**public ArretaElkarriketa arretaElkarriketaSortu(Bezeroa bezeroa, String gaia, String mezua)**

Bezero bat, gai bat eta mezu bat emanik, inongo langileri esleitu gabe dagoen elkarriketa bat sortzen da adieraziko bezero eta mezuarekin. Jarraian, mezu berri bat sortzen da eta elkarriketari gehitzen zaio.

**public BezeroaContainer getBezeroaContainer(Bezeroa b)**

Bezero bat pasata, bezeroaren kontainer bat itzultzen du, elkarriketak ere jasorik dituen.

**public void geldituElkarriketa(ArretaElkarriketa ae)**

Elkarriketa inungo langileri esleitu gabe ezartzen da.

**public void amaituElkarriketa(ArretaElkarriketa ae)**

Elkarriketa bat pasarik, bezeroak bidalitako mezu guztiak eta bezeroak ezabaturik dituen langileak bidalitako mezu guztiak ezabatzen zaizkio, eta irakurri gabeak irakurrita jartzen zaizkio. Honetaz gain azken mezu bat gehitzen da elkarriketara, balorazio mezua errepresentatzen duena.

**public void gehituPuntuazioa(ArretaElkarriketa l, Integer x)**

Arreta elkarriketa bat eta balorazio bat emanda, elkarriketaren langileari, puntuazio berri bat gehitzen zaio.

**public void eguneratuErrepikapenak()**

Metodo honen bidez, datu-baseko Errepikapena objektu guztiei hilabeteen gelditzen dena hilabeteen gastatuko dena baliora hasieratzen zaie.

**public Vector<Langilea> getLangileak()**

Metodo honi esker, datu basean dauden langile guztiak eskuratzen dira.

**public ArrayList<ErrepikatuakContainer> getErrepikatzailleak(Bezeroa bezeroa)**

Metodo honi bezero bat pasarik, bere errepikatzeileak lortzen ditugu kontainer batean sarturik.

**public ArrayList<ErrepikatuakContainer> getErrepikatzailleak(Bezeroa bezeroa)**

Metodo honi bezero bat pasarik, bere errepikatzailleak lortzen ditugu kontainer batean sarturik.

**public void jarraitzeariUtzi(Errepikapena errepikapena)**

Metodo honi errepikapen bat emanik errepikapen hau datu basetik ezabatzen du.

**public ArrayList<ErrepikatuakContainer> getErrepikapenak(Bezeroa bezeroa)**

Metodo honi bezero bat pasarik, berak errepikatzen dituenak lortzen ditugu kontainer batean sarturik.

**public ArretaElkarrizketa getArretaElkarrizketa(ArretaElkarrizketa ae)**

Metodo honi ArretaElkarrizketa bat pasata honen bertsio eguneratua lortzen dugu.

# Ondorioak

Lan honetatik hainbat ondorio atera daitezke. Hasteko aipatu behar genuke oso interesgarria izan dela proiektuetan garatutako irakaskuntza lantzea. Askotan entzun izan dugu ikasleok, teoria asko jakin arren praktika gabe ez zoazela inora. Irakasgaiari aplikatu den metodologiari esker zuzenean lanean hasi ginen egiten ari ginenaren inguruan ezer handirik jakin gabe. Pentsaezina zirudien klase teoriko sakon bat bera gabe gai izango ginatekeela sortu dugun proiektua sortzeko. Baina, apurka-apurka iterazioz iterazio gaitasunak lantzen joan gara eta asko ikasi dugu. Datu basea ukitu bakarrik egiten duen login erabilpen kasu xume batekin hasi ginen, eta azkenean arreta zerbitzu dotore bat inplementatzen amaitu dugu. Zenbait ikasgai teorikotan baino askoz gehiago ikasi dugularen sentsazioa daukagu. Gainera, askoz entretenigarriagoa da era praktikoa batean ematea ikasgaia, etorkizunean izan dezakegun lanpostuaren antza hartzen duen egoera bat bizi baitugu.

Honetaz gain, talde batean Scrum Masterrak duen garrantziaz ohartu gara. Behar beharrezkoa da talde lan batean denek lana nola doan dakien pertsona bat egotea. Pertsona hau egongo ez balitz, lana ongi antolatzea zaila litzateke, taldekide bakoitzak ez luke garbi jakingo zer egin behar duen. Gehitzeko, gure taldean orokorrean giro ona egon da. Pare bat “haserrealdi” txiki albo batera utzita ez dugu arazorik izan taldean lan egiteko. Esango genuke nahiko ongi bildu garela eta lan polit bat denon artean aurrera egoki eramateko gai izan garela.

Ikusi dugu kodetzen hasi aurretik ideiak garbi edukitzea behar-beharrezkoa dela. Lehen begiratuan badirudi domeinuek, sekuentzia diagramak... eta bestelakoek ez dutela gehiegi aportatzen eta kodetzen zoazen elean erraz alda daitezken gauzak direla. Baina, horrela ez dela konturatu gara. Azken iterazioan domeinu eredua eta sekuentzia diagramak egin ostean, kodetzen hasi ginen. Baina, hobe izango zatekeen domeinuari bigarren buelta bat eman izan bagenio. Kodeak erdian ohartu ginen egin genuen domeinua eta sekuentzia diagramak ez zirela bideragarriak nahi genituen funtzionalitate guztiak lortzeko. Ondorioz, inplementazio prozesua gelditu, eta domeinua berriro planteatzeari ekin genion. Denbora ugari galdu genuen honetan. Beraz, datorren urteko ikasleei ematen diegun gomendio bat domeinuak eta sekuentzia diagramak guztiz ondo daudela ziurtatu arte kodetzen ez hastea da. Denbora asko aurrez daiteke.

Irakasgaiaren inguruko gure iritzia dagokionez, asko gustatu zaigun ikasgai bat izan da, bizitzan zehar baliagarria suertatuko zaiguna. Gainera zailtasun maila egokia du. Lanak egunean emanez gero ez da konplikatu. Hori bai, ez pasa burutik asinatura honetako lana azkeneko unerako uztea, ez duzu denborarik izango (datorren urtekoentzat beste gomendio bat). Gainera interesgarria izan da asinaturari zailtasun maila guk jarri ahal izatea. Hau da, lehen bi iterazioetan nahiko bideratuta joan da kontua, baina azkeneko bi erabilpen kasuek (errepikapena eta librea) joku handia eman zezaketen. Talde bakoitzak zer egiteko gai zen erakutsi behar zuen. Helmuga potente baina sinesgarri bat jarritz gero, erroka polit batean bihurtu daiteke lana.

Amaitzeko, esan dezakegu, oso gustura gelditu garela eginiko lanarekin. Beharbada izango ditu detektatu ez ditugun akatsak eta baita hobetzeko gauza ugari ere. Halere oso harro gaude eginiko lanaz.



# URLak

## Bideoaren URLa

<https://youtu.be/R2JSZYysca8>

## Kodearen URLa

<https://gitlab.com/josuloidigorostidi/bets21azkeniterazioa>

## Aurkezpenaren URLa

<https://docs.google.com/presentation/d/1QFbXDWDFcBTQuE9Tzn9pZRbwtrwMcEfqEhXY-ONnyJA/edit?usp=sharing>