

Apustu sistema projektua

Inigo Salazar Xabier Irastorza

2022ko maiatzaren 22a

Gaien Aurkibidea

1	Sarrera	2
2	Web zerbitzuen eta internalizazioaren inguruan	3
3	Eskakizunen bilketa	3
3.1	Domeinuaren eredua	3
3.1.1	Gertaera	5
3.1.2	Galdera	5
3.1.3	Kuota	5
3.1.4	Apustua	5
3.1.5	Erabiltzailea	6
3.1.6	User	6
3.1.7	Admin	6
3.1.8	Kontaktua	6
3.1.9	Mezua	6
3.1.10	Mugimendua	6
3.1.11	Apustu anitza	7
3.2	Erabilpen kasuen eredua	7
3.3	Interfaze grafikoak eta haien gertaera fluxuak	9
3.3.1	Main interfaze grafikoa	9
3.3.2	User interfaze grafikoa	9
3.3.3	Saioa hasi	11
3.3.4	Erregistratu	11
3.3.5	Admin	13
3.3.6	Apustuak egin/begiratu	13
3.3.7	Galderak sortu	15
3.3.8	Gertaerak sortu	16
3.3.9	Kuotak ipini	17
3.3.10	Emaitza jarri	18
3.3.11	Gertaera ezabatu	18
3.3.12	Mezuak bidali	20
3.3.13	Dirua sartu	21
3.3.14	Mugimenduak ikusi	22

3.3.15 Apustuak ezabatu	22
3.3.16 Cash out	23
3.3.17 Apustu anitza egin	24
4 Diseinua	25
4.1 Hirugarren iterazioan garatutako sekuentzia-diagramak	25
4.1.1 Apustu anitza egin	25
4.1.2 Mezuak bidali	28
4.1.3 Cash out egin	30
4.2 Hirugarren iterazioan eguneratutako sekuentzia-diagramak	32
4.2.1 Apustuak ezabatu	32
4.2.2 Emaitza jarri	34
4.2.3 Gertaera ezabatu	36
4.3 Klase diagrama	38
5 Implementazioa	40
5.1 Oinarrizko erabilpen kasuetako klaseak	40
5.1.1 Event klasea	40
5.1.2 Question klasea	41
5.1.3 Kuota klasea	41
5.1.4 Apustua klasea	42
5.1.5 Mugimendua klasea	43
5.2 Apustu anitza erabilpen kasuko klaseak	44
5.2.1 ApustuAnitza klasea	44
5.3 Erabiltzaileak adierazteko klaseak	45
5.3.1 User klasea	46
5.3.2 Admin klasea	46
5.3.3 Erabiltzailea klasea	46
5.4 Mezuak bidali erabilpen kasuko klaseak	47
5.4.1 Kontaktua klasea	47
5.4.2 Mezua klasea	48
6 Ondorioak	49
7 Erreferentziak	49

1 Sarrera

Proiektu honen helburua apustu sistema funtzional bat sortzea izan da, era-biltzaile anitzekoa eta interneten aurki ditzakegun apustu etxeen webguneek eskaintzen dituzten oinarrizko funtzionalitateak eskaintzen dituena. Proiektu honetan garatu den apustu-sistemak futbol partiduetan apostatzeko balio du. Implementatu dugun apustu sistemak aukera ematen du, nola ez, apustuak egiteko, horiek ezabatzeko eta horien jarraipena egiteko historialaren bidez. Hori egingarri egiteko bi erabiltzaile mota bereizten ditu: erabiltzaile arrunta

eta administratzialeak. Lehenak edozein bezerok edukiko lituzkeen funtzionalitateetara sarbidea ematen du, bigarrena aldiz, apustu sistema kudeatuko luketen enpresako langileei begira garatua izan da. Bi erabiltzaile mota hauei esker lortu dugu partidu baten inguruau apustuak sortzeko funtzionalitateak (administratzialearen esku) eta horiei apustuak eginez erantzutekoak (erabiltzaile arruntaren esku) implementatzea. Apustuak kudeatzeko funtzionalitateez gain, txat motako posta elektroniko bat ere implementatua du proiektu honek, erabiltzaileak euren artean komunikatzeko pentsatua.

Txosten honetan proiektuaren zehaztapena, diseinua eta implementazioa egiteko orduan jarraitu ditugun pausoak eta horiek ulertzeko beharrezko azalpenak, proiektuaren dokumentazio gisa balioko dutenak, idatzi ditugu. Proiektuaren garapena Prozesu Bateratua eta SCRUM metodologiak erabiliz bideratu da.

2 Web zerbitzuen eta internalizazioaren inguruau

Proiektu honetan garatu den apustu sistemak web-zerbitzuak eta internalizazioa implementatu ditu bere osotasunean. Hurrengo ataletan azalduko da nola.

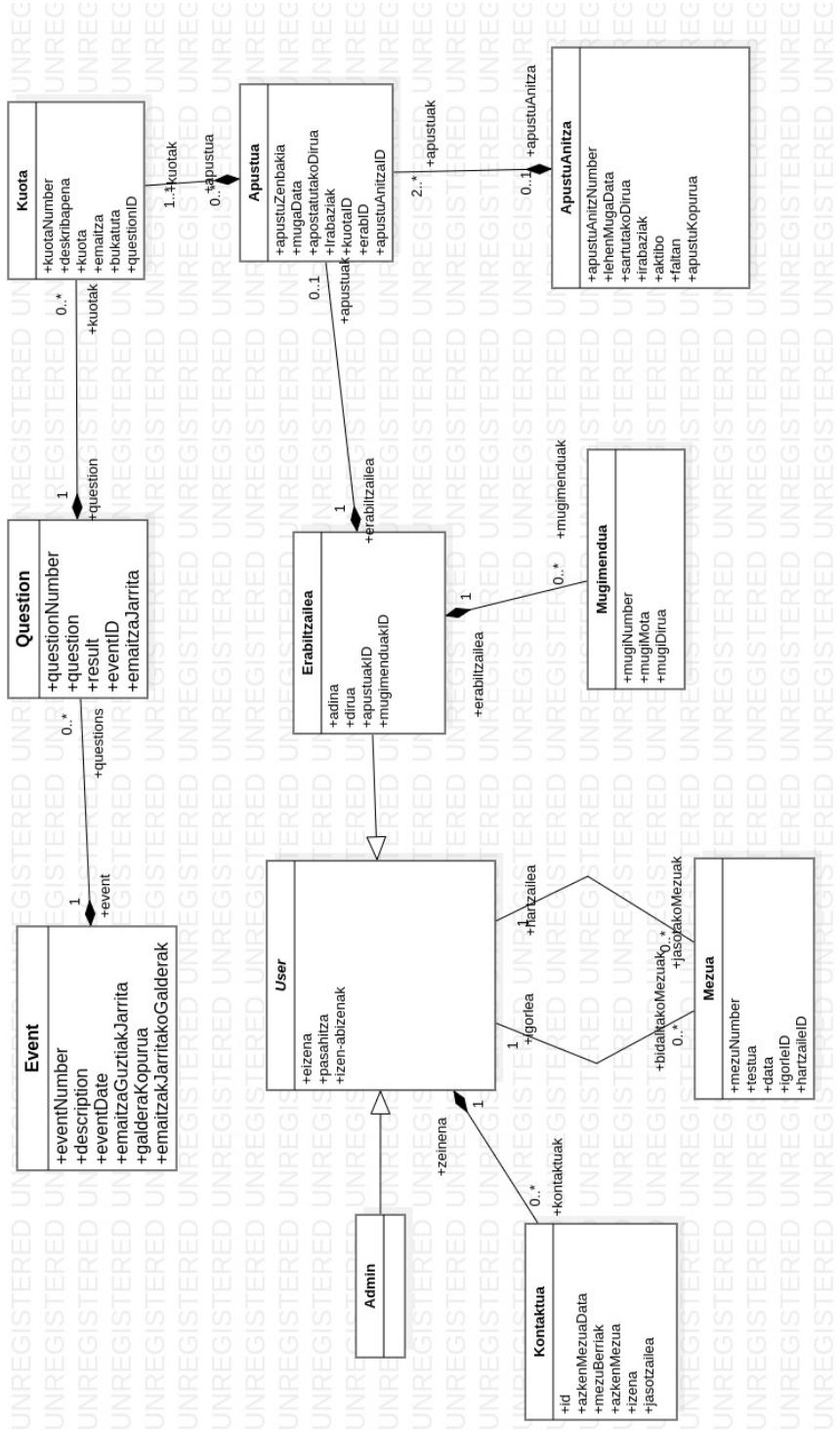
3 Eskakizunen bilketa

Gure proiektuaren eskakizunen bilketa aurki daiteke ondorengo ataletan.

3.1 Domeinuaren eredu

3 ESKAKIZUNEN BILKETA

4



Irudia 1: Apustu sistema proiektuko domeinuaren eredu.

Azpiatal honetan domeinuaren ereduiko entitateen arteko erlazioak eta entitate bakoitzaren zentzua azaltzen saiatuko gara soilik, entitate bakoitzaren atributuen (ezaugarrien) gaineko informazioa implementazioa azaltzen den atalean aurki daiteke.

3.1.1 Gertaera

Gertaerak, domeinuaren ereduauan *Event* klaseak adierazten ditu, gure apustu sisteman futbol partiduak adierazteko erabiltzen dira. Gertaera batek egun eta ordu zehatz batean antolatuta dagoen partidu bat adierazten du eta edozein erabiltzailek horren inguruan apustua egin dezake. Gertaerak galderarekin erlazionatuta daude, izan ere, gertaera bakoitzak galderak eduki ditzake eta horien erantzun posibleen gainean egin daiteke apustu. Galderak bakoitza gertaera bakar batekin erlazionatuta egongo da, galdera horri zentzua ematen dion gertaera izango da hori, hain zuzen.

3.1.2 Galdera

Galderak, domeinuaren ereduauan *Question* klaseak adierazten ditu, apustuen muina dira, izan ere, apustu guztiak galdera baten erantzunen alde edo kontra egin daitezke. Adibidez, "Zeinek irabaziko du partidua?" galdera bat izango litzateke eta apustua haren erantzun baten gainean egingo litzateke, kasu honetan, adibidez, "Errealia" edo "Osasuna". Galdera guztiak, aurrez esan bezala, erreferentzia egiten dieten gertaerarekin erlazionatuta daude. Gainera, kuotekin ere erlazionatuta daude, kuota horiek galdera bakoitzaren erantzun posible bat adierazten baitute.

3.1.3 Kuota

Kuotak, domeinuaren ereduauan *Kuota* klaseak adierazten ditu, aurrez aipatutako galderen erantzun posible bat dira. Erantzuna ematen dioten galderarekin erlazionatuta egoteaz gain, apustuekin erlazionatuta ere badaude. Apustuak erantzun posibleen, hau da, kuoten gainean egiten dira eta beraz, apustua zeren alde egiten den jakiteko kuota klasean gordeta geratzen dira. Hortik datorkigu bi entitate hauen arteko erlazioa.

3.1.4 Apustua

Apustuek, domeinuaren ereduauan *Apustua* klaseak adierazten ditu, izenak esaten duena adierazten dute. Lehen esandakoarengatik kuotekin erlazionatuta daude, baina erabiltzaile batekin ere erlazionatuta egon beharko dira, izan ere, apustu hori pertsona batek egiten du eta horrek erlazio bat sortzen du. Gainera, apustu anitzekin ere erlazionatuta dago, apustua apustu anitza osatzen duen apustu sorta baten parte izan daitekeelako.

3.1.5 Erabiltzailea

Erabiltzaile erregistratuak adierazten ditu, administratzaileak ez direnak. Erabiltzaileek apustuak egin ditzaketenez, apustuekin erlazionatuta dago entitatea. Gainera, mugimenduekin ere erlazionatuta dago, izan ere, entitate horrek erabiltzaile batek egin dituen ekintzak adierazten ditu. Nola ez, bere guraso klasearekin ere erlazionatuta dago, User klasearekin, zeinak erregistratutako erabiltzaile mota guztiak adierazten dituen, Erabiltzailea klaseak adierazten dituen guztiak barne hartuta.

3.1.6 User

Erregistratuta dauden erabiltzaile mota guztiak, erregistratuak eta administratzaileak, adierazten ditu. Hori horrela izanik, Erabiltzailea eta Admin klasearekin erlazionatuta dago. Horretaz gain, Mezua klasearekin ere erlazionatuta dago, izan ere, klase horrek erregistratutako edozein erabiltzailek bidalitako mezuak adierazten ditu. Kontaktua klasearekin duen erlazioa bide beretik doa, edozein erabiltzaile erregistratuk eduki baititzake bere posta zerbitzuan erregistratutako edozein erabiltzailerent kontaktuak.

3.1.7 Admin

Administratzaileak diren erabiltzaileak adierazten ditu eta User klasearen ume klasea da. Hori horrela izanik, klase horrekin hierarkia motako erlazioa du.

3.1.8 Kontaktua

Erabiltzaile erregistratu batek mota bereko beste erabiltzaile batekin elkarritzeta bat izan badu inoiz, edo kontaktuetara gehitu badu, orduan erabiltzaile horiek elkarren kontaktuak dira. Hori adierazten du klase honek, kontzeptu zabaldua da mezularitza aplikazioen artean. Domeinuaren ereduan erlazio bakarra du, User klasearekin, kontaktua norena den adierazgarri egiten baita hala.

3.1.9 Mezua

Erregistratutako erabiltzaile batek beste bati bidalitako edozein mezu adierazten du. Erabiltzaile bakoitzak bi mezu-ontzi eduki behar ditu: batak bidalitako mezuak adieraziko ditu eta besteak jasotakoak. Hori horrela, erlazio bikoitz bat irudikatu dugu domeinuaren ereduan, adar batek erabiltzaile baten bidalitako mezuak adierazten ditu bestea hartzalea izanik eta besteak erabiltzaileak jasotako mezuak beste erabiltzailea igorlea izanik.

3.1.10 Mugimendua

Mugimendu bat erabiltzaile batek egin duen ekintza bat da: apustua egin, apustua ezabatu, apustua irabazi, ... Implementazioko 5.1.5 atalean mugimendu posible guztiak jaso ditugu. Mugimendu oro erabiltzaile batek egiten

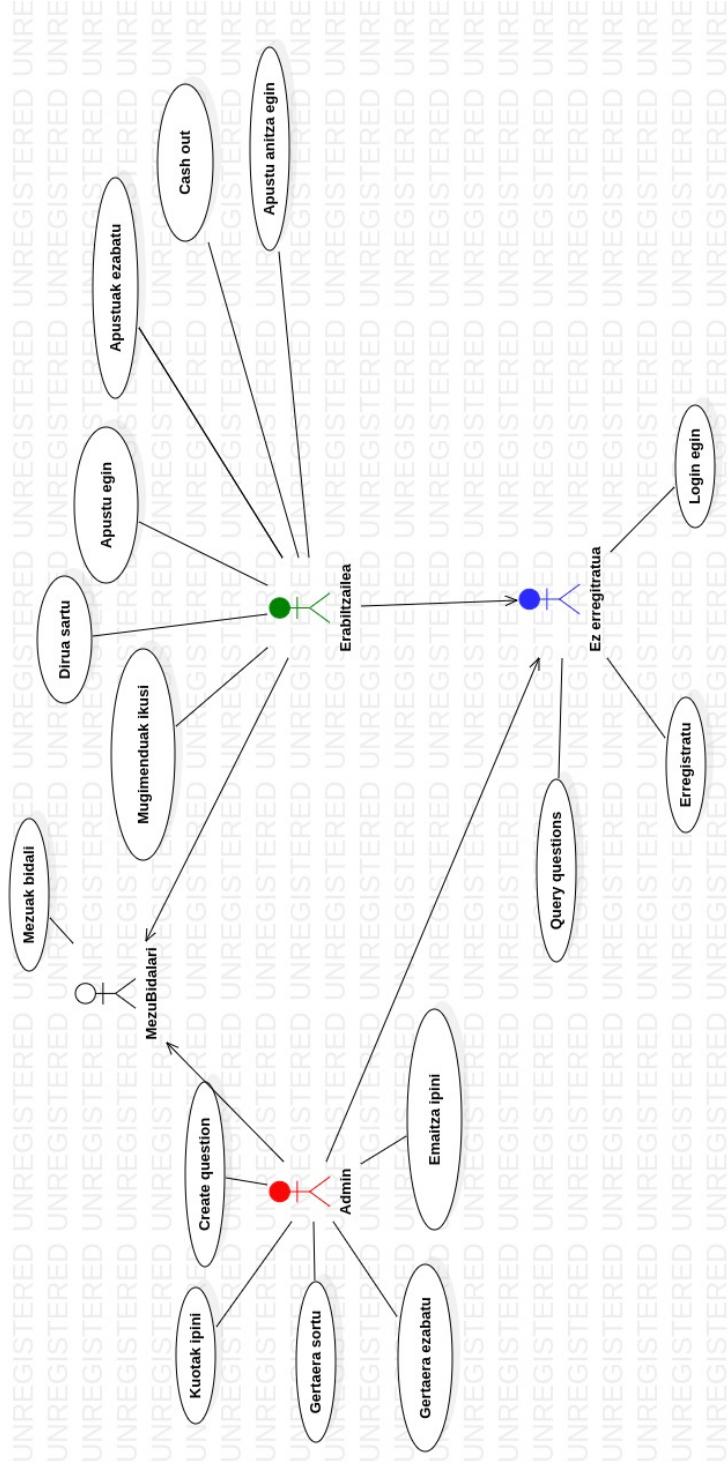
duenez, bistakoa da zein erlazio izango duten bi entitate horiek. Erabiltzaile batetik hainbat mugimendu edo bat ere ez egin ditzake.

3.1.11 Apustu anitza

Apustu anitza, domeinuaren ereduan *ApustuAnitza* bezala adierazia, apustu sorta bat da. Denak batera egin izanagatik erabiltzaileak irabazi gehigarriak lortu ditzake, baina apustu anitzaren barneko apustu bakoitza irabazteagatik erabiltzaileak ez du irabazirik jasoko. Denak irabazi edo bat ere ez, hori da entitate honen eta bera erabiltzen den erabilpen kausaren muina. Apustu anitzak apustuekin erlazionatuta daude hortaz, apustu anitz bat bi apustuk gutxienez osatuko baitute. Gainera, apstuak apustu anitzen parte izan daitzke, edo ez (apustu bakartiak).

3.2 Erabilpen kasuen eredua

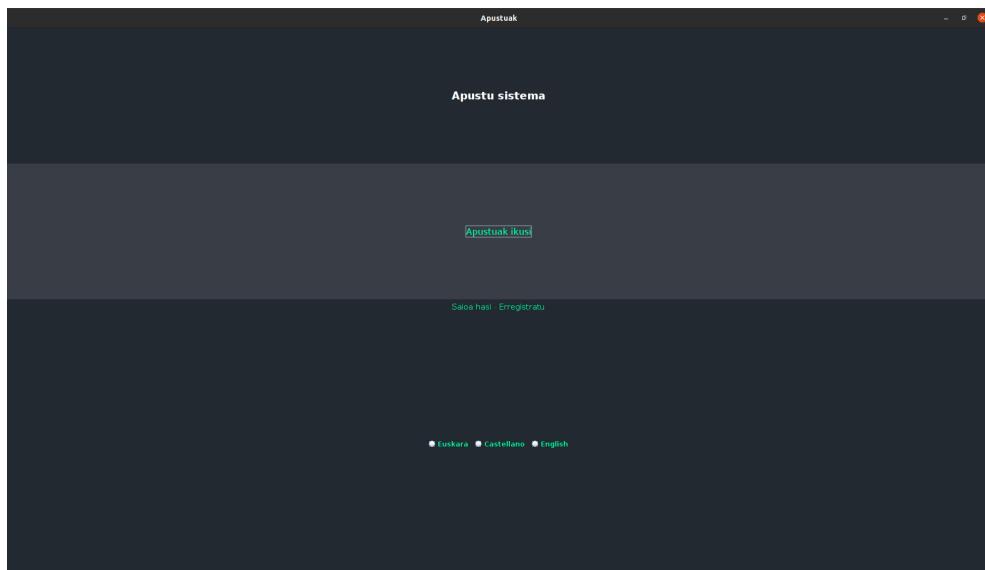
Jarraian aurki daiteke proiektu honen erabilpen kasuen eredua.



Irudia 2: Proiektuaren erabilpen kasuak.

3.3 Interfaze grafikoak eta haien gertaera fluxuak

3.3.1 Main interfaze grafikoa



Irudia 3: Main interfaze grafikoa.

3.3.2 User interfaze grafikoa



Irudia 4: User interfaze grafikoa.

3.3.3 Saioa hasi



Irudia 5: Saioa hasi interfaze grafikoa.

```
# Flow of events

## Basic Flow

1. *System* shows two labels to type username and password
2. *User* types username and password in the labels
3. *User* clicks on login button

## Alternative flow
1. There isn't any user with that username. User cannot login
2. The specified password and username don't match. User cannot login
```

3.3.4 Erregistratu

```
# Flow of events

## Basic Flow

1. *Sistema* erregistratzeko beharrezkoak diren eremuak bistaratzen ditu:
**Erabiltzaile-izena**, **pasahitza**, **izen-abizenak** eta **adina**
2. *Erabiltzailea* eremuak betetzen ditu
```



Irudia 6: Erregistratu interfaze grafikoa.

3. *Erabiltzailea* erregistratu botoia sakatzen du
4. *Sistema* egutegia bistaratzen du

```
## Alternative flow
1. Erabiltzaile-izena jada erregistratuta badago ezin da berriz
erregistratu erabiltzaile berdinarekin
```

3.3.5 Admin



Irudia 7: Admin intrefazeo grafikoa.

3.3.6 Apustuak egin/begiratu



Irudia 8: Apustuak egin edo begiratzeko interfaze grafikoa.

```

# Flow of events
## Basic Flow
1. *Sistemak* aukeratu duzun partidua erakusten du tea txuriune bat bertan
erabiltzaileak zenbat diru sartu nahi duen jartzeko

2. *Erabiltzailea* eremuak betetzen du eta apustua egin botoia sakatzen du
4. *Sistemak* apustua gordetzen du eta UserGUI-ra bueltatzen da.
## Alternative flow
1. Erabiltzaileak diru kantitatea desegokiarekin betetzen du txuriunea,
karaktere desegoki batekin edo ez duelako saldo nahikorik kontuan.
Orduan errorea ateratzen da txuriunea txarto bete duela abisatzu

```

3.3.7 Galderak sortu



Irudia 9: Galdera sortzeko interfaze grafikoa.

```

# Flow of events
## 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**
5. *System* adds the new **question** with a minimun **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.
```

3.3.8 Gertaerak sortu



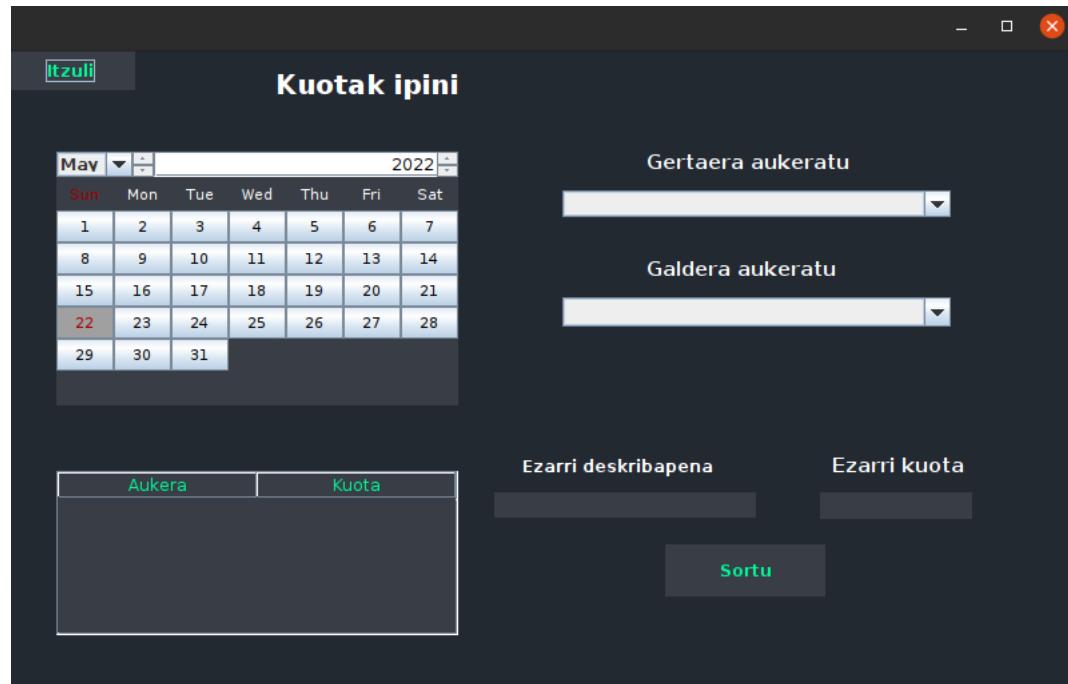
Irudia 10: Gertaerak sortzeko interfaze grafikoa.

```
# Flow of events
## Basic Flow

1. *System* shows two labels, the second bigger than the first,
to type event's name and description. Also shows three lists
to choose a date in a correct format.
2. *Admin* types all the requierments.
3. *User* clicks on create button.
4. *System* creates a new event.

## Alternative flow
1. User does not type the name of the event. Event cannot be created.
2. User clicks on exit button. No event is created.
```

3.3.9 Kuotak ipini



Irudia 11: Kuotak ipintzeko interfaze grafikoa.

```

# Flow of events

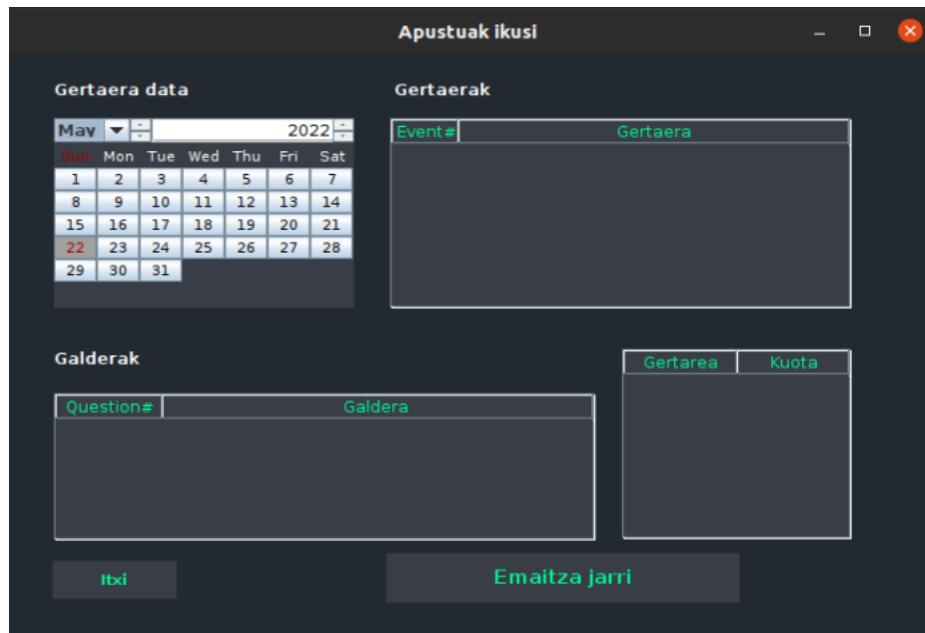
## Basic Flow

1. *System* shows a list of events
2. *Admin* selects a event in list
3. *System* displays the questions of this event
4. *Admin* selects a question
5. *Admin* introduces the cuote or cuotes for this question
6. *System* adds the new quote or quotes to the selected question

## Alternative flow
1. There are no events. Cuote cannot be added.
2. There are no questions for this event. Cuote cannot be added.
3. The cuote field is empty. Cuote cannot be added.

```

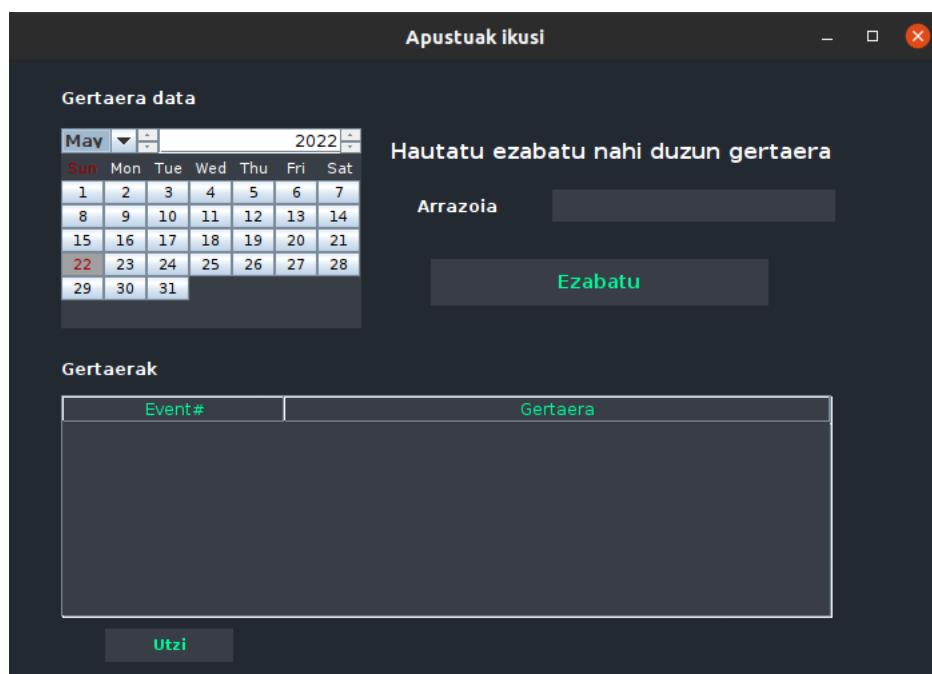
3.3.10 Emaitza jarri



Irudia 12: Emaitza jartzeko interfazea.

- ```
Flow of events
Basic Flow
1. *System*: Erabiltzaileak data jartzeko beharrezko tresnak
(urte, hilabete eta egunaren zerrenda tolesgarriak) pantailaratuko ditu.
2. *User*: Data sartuko du.
3. *System*: Data horretako gertaera guztiak atzituko dira eta
beste zerrenda tolesgarri batean pantailaratuko dira.
4. *User*: Emaitza ipini nahi dion galdera dagoen gertaera hautatuko du.
5. *System*: Gertaera horri dagozkion galderak pantailaratuko ditu beste
zerrenda tolesgarri batean.
6. *User*: Emaitza ezarri nahi dion galdera hautatuko du eta emaitza
sartuko dio.
7. *System*: Emaitza ezarriko du eta datubasea eguneratu.
```

### 3.3.11 Gertaera ezabatu



Irudia 13: Gertaerak ezabatzeko interfazea.

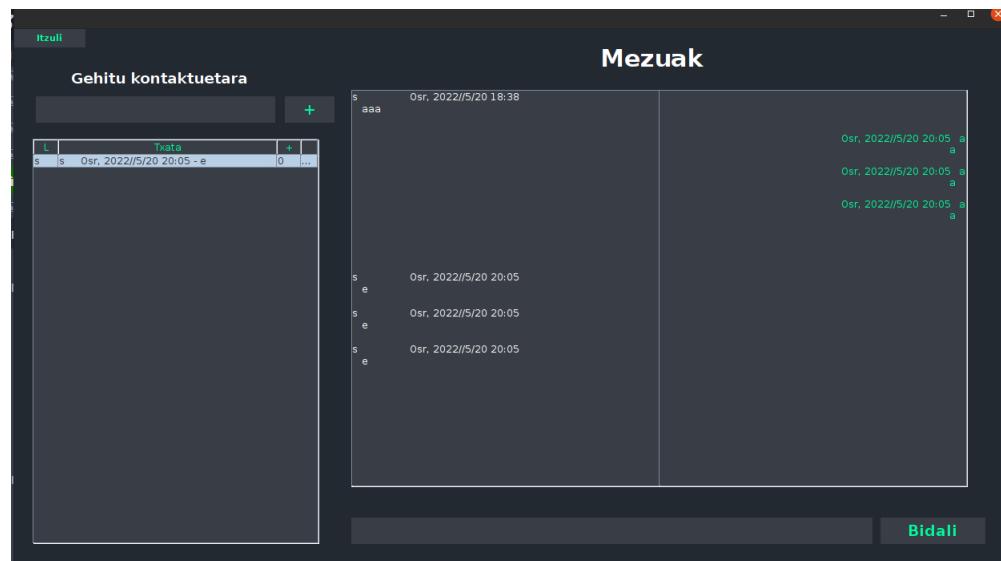
```
Flow of events

Basic Flow

1. *Sistemak* datu datubasean dauden gertaerak erakusten ditu
2. *Admina* ezabatu nahi duen gertaera klikatzen du eta ezabatu botoiari sakatzen du
3. *Sistemak* gertaera ezabatzen du

Alternative flow
1. Gertaerarik ez badaude ez du ezer erakutziko
```

### 3.3.12 Mezuak bidali



Irudia 14: Mezuak bidaltzeko interfazea.

```
Flow of events

Basic Flow

1. *Sistemak* MezuakBidaliGUI interfazea erakusten du.
```

2. \*Erabiltzaileak\* elkarrizketa hasi edo jarraitu nahi duen kontaktua aukeratzen du,edo berri bat gehitzeko eskatu erabiltzaile izena sartuz.
3. \*Sistemak\* kontaktu berria gehitzen du halakorik eskatu bada.
4. \*Sistemak\* erabiltzaileak hautatutako kontaktuarekin izandako elkarrizketa zaharrak erakusten ditu pantailan.
5. \*Erabiltzaileak\* mezu bat idatzi eta bidaltzeko botoia sakatzen du.
6. \*Sistemak\* mezua bidali eta pantaila freskatzen du.

## Alternative flow

1. \*Sistemak\*, erabiltzaileak sartutako erabiltzaile izena ez badagokio inori,errorea pantailaratuko du.

### 3.3.13 Dirua sartu



Irudia 15: Dirua sartzeko interfazea.

```
Flow of events

Basic Flow

1. *Sistemak* txuriune bat erakusten du bertan erabiltzaileak zenbat diru
sartu nahi duen jartzeko
 eta pasahitza konfirmatzeko ere
2. *Erabiltzaileak* eremuak betetzen du eta sartu botoia sakatzen du
4. *Sistema* dirua erabiltzailearen kontuan sartzen da eta UserGUI-ra
```

bueltatzen da.

```
Alternative flow
1. Erabiltzaileak diru kantitatea desegokiarekin betetzen du txuriunea.
Orduan errorea ateratzen da txuriunea txarto bete duela abisatuz
2. Pasahitza txarto sartu badu errore mesua agertuko da eta pasahitza berriro
sartu beharko du
```

### 3.3.14 Mugimenduak ikusi

| Zenbakia | Ekintza | Mugimendua |
| --- | --- | --- |
| 1 | Irabazi | 20.0 |
| 2 | Apostatu | 10.0 |
| 3 | Sartu | 100.0 |
| 4 | Sartu | 100.0 |
| 5 | Apustu anitza egin: | -23.0 |
| \* | --> Apustua Atletico-Athletic, Zeinek irabaziko du ... | --- |
| \* | --> Apustua Eibar-Barcelona, Zeinek irabaziko du?... | --- |
| 6 | Cash out egin | 37.83695652... |
| 7 | Apustua Eibar-Barcelona, Zeinek irabaziko du?, x | -23.0 |

Irudia 16: Mugimenduak ikusteko interfazea.

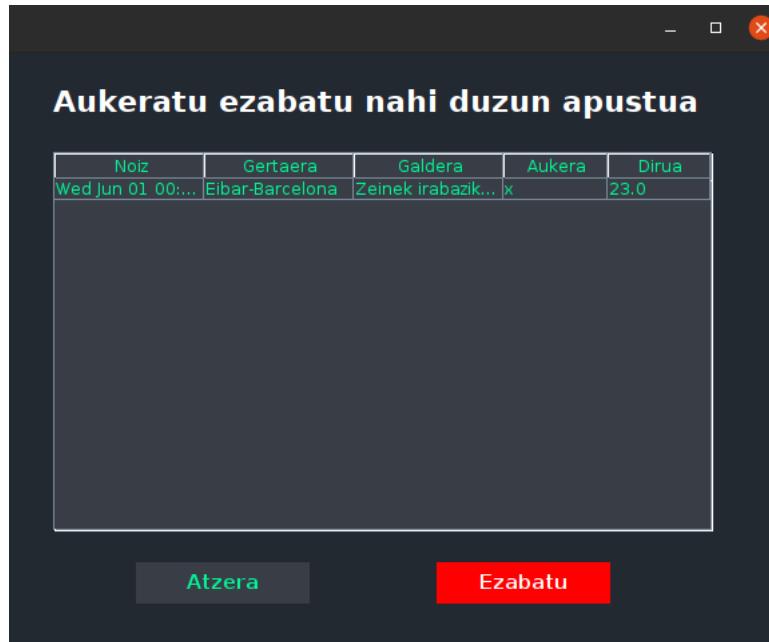
```
Flow of events

Basic Flow

1. *Sistemak* erabiltzailearen mugimenduak erakusten ditu:
Dirua sartu, apustua egin, apustua anulatua, apustua irabazi...

Alternative flow
1. Erabiltzailea oraindik ez du mugimendurik egin lista hutsik dago
```

### 3.3.15 Apustuak ezabatu



Irudia 17: Apustuak ezabatzeko interfazea.

```
Flow of events

Basic Flow

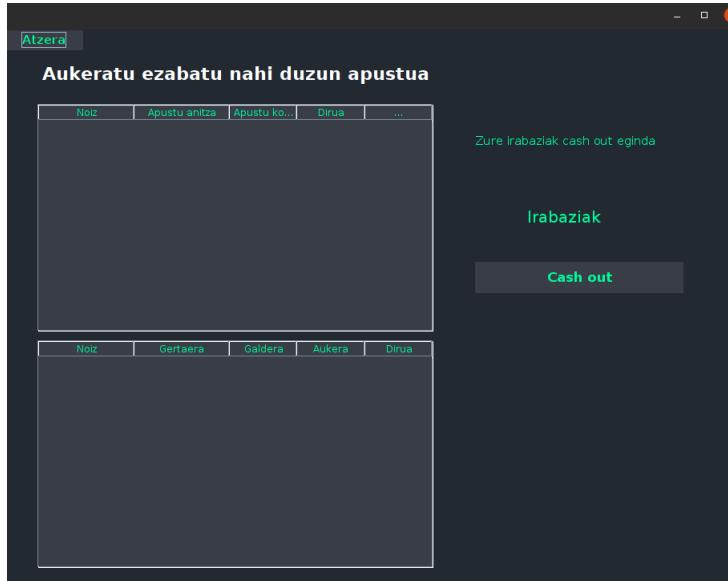
1. *Sistemak* ZureApustuakGUI-interfazea erakusten du. Bertan oraindik aktibo dauden apustuak daude

2. *Erabiltzaileak* borratu nahi duen apustua klikatzen du eta apustua borratu botoia sakatzen du
4. *Sistemak* apustua ezabatzen du eta dirua bueltatzen dio erabiltzaileari

Alternative flow
1. Aukeratutako apustua pasa bada errorea erakutziko du eta ez da dirua bueltatuko
```

### 3.3.16 Cash out

```
Flow of events
```



Irudia 18: Cash out egiteko interfazea.

#### ## Basic Flow

1. \*Sistemak\* CashOutGUI-interfazea erakusten du. Bertan oraindik aktibo dauden apustu anitzak daude
2. \*Erabiltzaileak\* cash out aplikatu nahi dion apustu anitza klikatzen du.
3. \*Sistemak\* apustu anitz hori osatzen duten apustuen zerrenda eta haien informazioa erakusten ditu.
4. \*Erabiltzaileak\* ezabatu nahi duen apustu anitza sakatzen du.
5. \*Sistemak\* apustu anitza ezabatzen du eta beraz, hura osatzen duten apustu guztiak ere bai.

#### ## Alternative flow

1. Apustu anitza borratzeko epemuga igaro bada, errore mezua erakutsiko du \*Sistemak\*

#### 3.3.17 Apustu anitza egin

Erabilpen kasu honek ApustuakEgin erabilpen kasuarekin partekatzen du interfaze grafikoa.

```

Flow of events

Basic Flow

1. *Sistemak* FindQuestionsGUI interfazea erakusten du. Bertan,
egutegi bat ikusgarri dago, gertaerak dituzten egunak beste kolore
batez azpimarratuta dituena.
2. *Erabiltzaileak* egun bat aukeratzen du.
3. *Sistemak* egun horretan jazoko diren gertaerak erakusten ditu.
4. *Erabiltzaileak* geratera horietako bat aukeratzen du.
5. *Sistemak* gertaera horri dagozkion galderen zerrenda erakusten du.
6. *Erabiltzaileak* galdera horietako bat hautatzen du.
7. *Sistemak* galdera horri dagozkion kuotak eta haien kuota balioak
erakusten ditu.
8. *Etrabiltzaileak* horietako bat aukeratzen du.
9. *Sistemak* aukera hori gorde egiten du, apustu bat bezala eta erakutsi
egiten du hautatutako apustuen taulan.
10. *Etrabiltzaileak* nahi adina apustu sor ditzake nahi duen gertaera,
galdera eta kuotetatik.
11. *Etrabiltzaileak* diru kopuru bat sartzen du.
12. *Sistemak* diru kopuru hori sartuz gero erabiltzaileak lortuko lituzkeen
irabaziak erakusten ditu.
13. *Etrabiltzaileak* onarpen botoiari ematen dio.

Alternative flow
1. Hautaketa okerra egiten baitu, errorea erakutsiko du.

```

## 4 Diseinua

### 4.1 Hirugarren iterazioan garatutako sekuentzia-diagramak

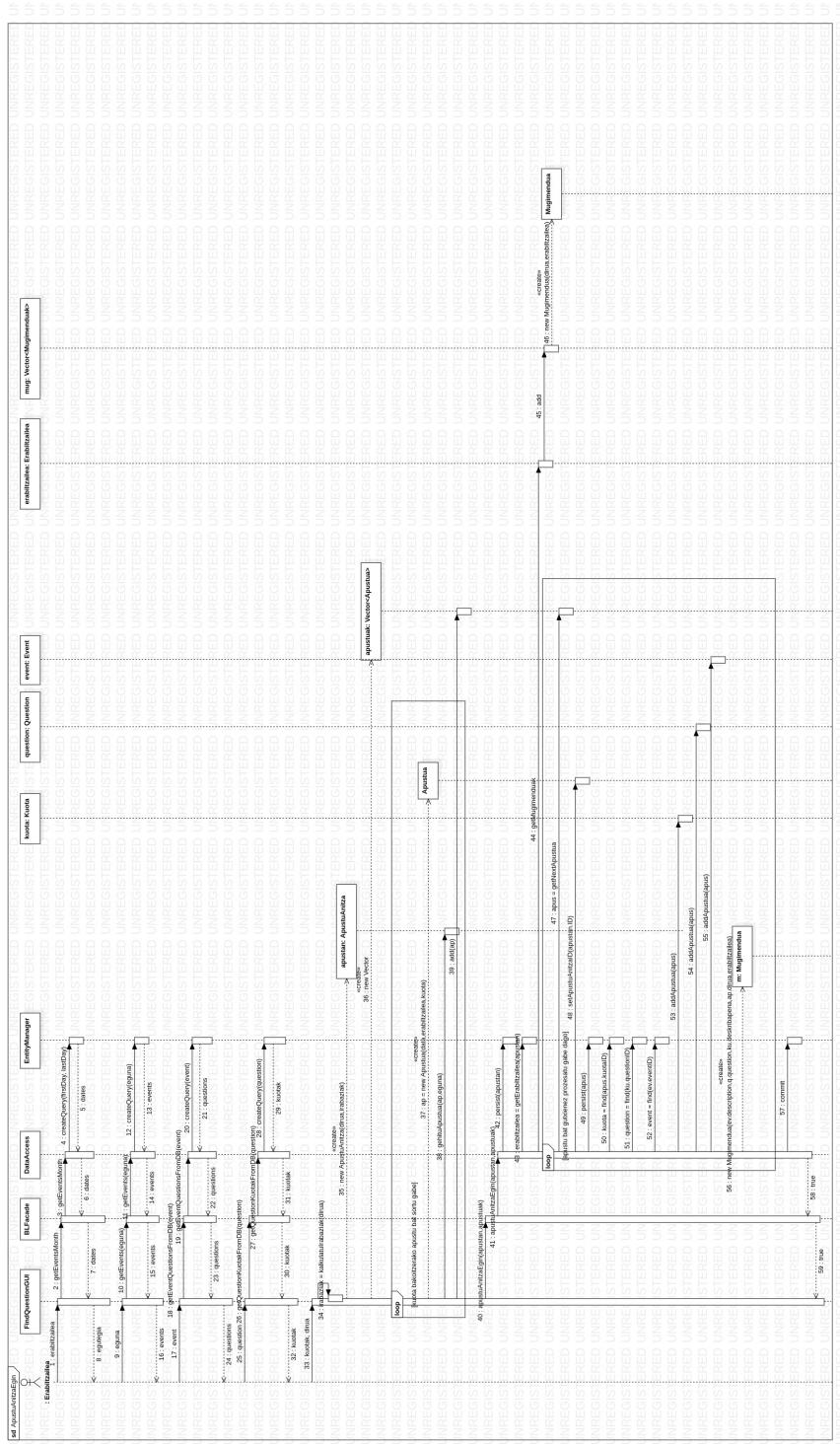
Hirugarren iterazio honetan implementatu behar ziren erabilpen kasu bakoi-tzek sekuentzia diagrama bat egin dugu eta jarraian horien inguruko azalpenak topa daitezke.

#### 4.1.1 Apustu anitza egin

Bi zatitan bana dezakegu sekuentzia diagrama hau eta beraz, berak adierazten duen erabilpen kasua. Lehen zatian, GUIak eta erabiltzaileak elkarrekin kintza handia dute, izan ere, erabiltzaileak hautatzen dituen data, gertaera edo galderen arabera GUIak haien gertaera-zerrendak, galdera-zerrendak edo kuota-zerrendak erakutsi behar ditu. Hori nola egin? Datubaseari galdera bat eginez, erabiltzaileak sartu duen datuari dagokion zerrenda itzul dezan.

Bigarren zatian erabiltzaileak jada erabaki du bera hautua eta dirua ere sartu du. Hori horrela, makinak apustu anitza gorde eta aktibo jarri behar du.

Gorde aurretik ordea, GUIak memorian bertan apustu anitza eta hura osatuko duten apustuak sortu beharko ditu, horrek justifikatzen du diagramako lehen begizta-kutxa. Apustu anitza sortuta dagoenean, GUIak kontrola DataAccess-i igorriko dio, apustu anitz hori datubasean gorde dezan. Hartarako, mugimendu bat sortuko du lehenik, apustu anitza egin duela jakinarazteko etorkizunean erabiltzaileari. Gero apustu anitza datubasean gordeko du eta baita ere bere parte den apustu bakoitza, horietako bakoitzari ere mugimendu bat sortuz.



Irudia 19: Apustu anitza egin sekuentzia-diagrama.

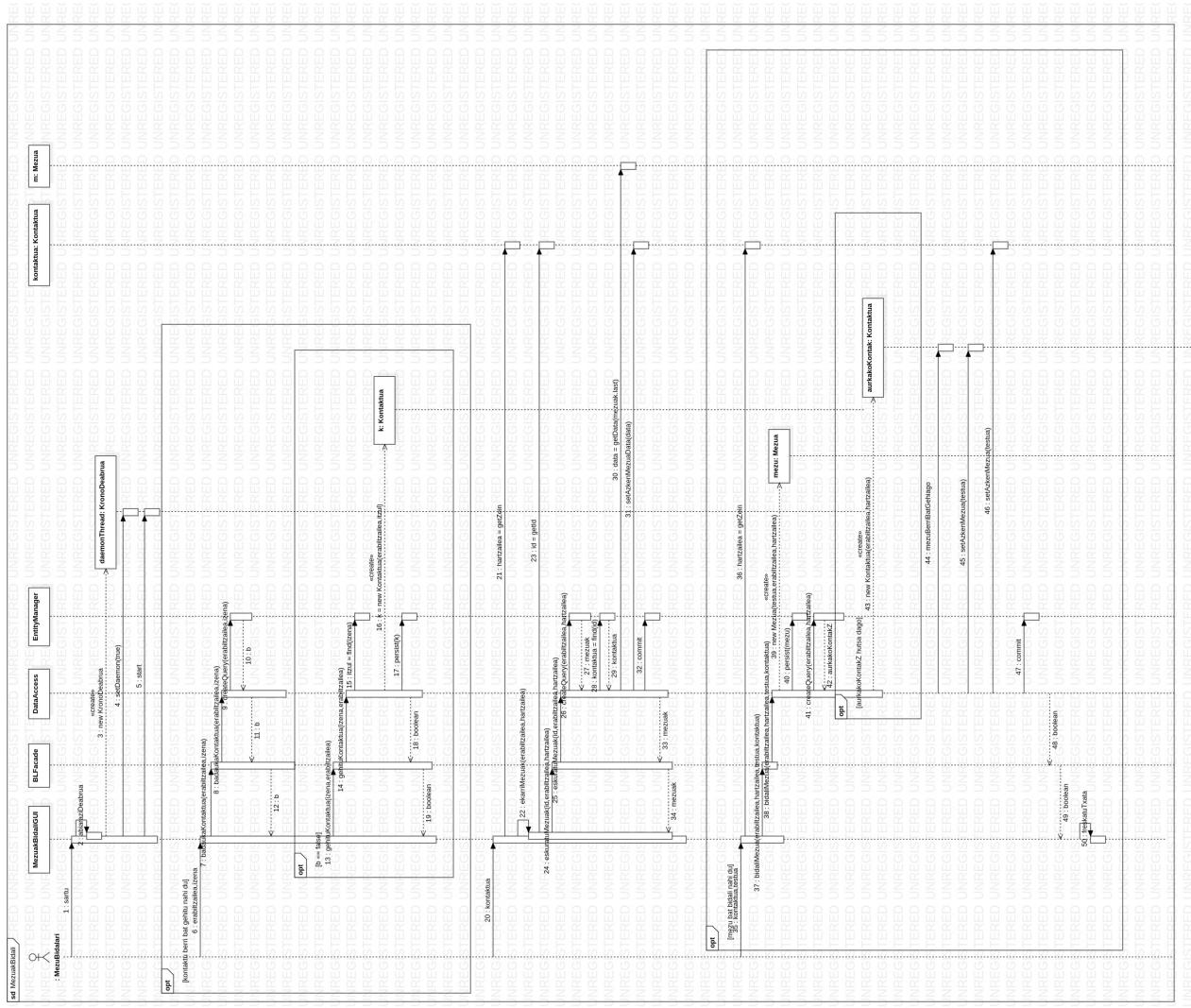
#### 4.1.2 Mezuak bidali

Erabiltzaileari mezuak bidaltzeko aukera eman aurretik funtsezko da honek jasota eduki ditzakeenak edo idazten dagoen bitartean jaso ditzakeenak behar bezala pantilaratzea. Hori horrela izanik, deabru bat abiaraziko da lehenik, segunduro segunduro eta GUIa itxi arte txata freskatuko duena, mezu berrien bila. Kontaktuen zerrenda ere freskatuko du, baina erabiltzaileak bertan hau-taketak egiten ezerosotasunik ez sortzen, sei segundoan behin egingo du fres-katze hori.

Ostera, erabiltzaileak kontaktu berri bat gehitu nahi badu, hora datubasean existitzen den edo ez begiratuko du makinak eta baiezkoan, errorea itzuli. Ezezkoan, kontaktu entitate berri bat sortu eta datubasean gordeko du.

Erabiltzaileak kontaktu bat hautatuz gero, berak adierazten duen erabiltzaileak igorritako mezuak pantilaratu beharko ditu eta hori egiteko, datubaseari dei egin beharko dio. Hori egin aurretik, GUIan bertan dagoen metodo bat- ti dei egiten dio, honek bere gain har dezan pantailan bistarapena egitearen ardura. Datubasean galdera bidez topatzen dira mezu guztiak, erabiltzaileak eskatutakoigorle eta hartzaleak dituzten mezu guztiak eskuratzendirelarik eta gero GUIan ordenatu.

Erabiltzaileak mezu bat bidali nahi badu, azken opt leihoa islatzen den pro- zedura jarraituko da. Erabiltzaileak hautatutako kontaktuak adierazten duen erabiltzaileari bidaliko zaio mezia, beraz, Mezia entitate berri bat sortu eta hartziale gisa kontaktuko erabiltzailea ezarriko zaio (igorle gisa unekoa), ondo-ren datubasean gordetzeko. Behin mezia gordeta, igorlearren eta hartzalearen Kontaktua entitateak eguneratu behar dira, etorkizunean kontaktuen tauleta-ko informazioa (azken mezia eta data) behar bezala egunera daitezzen. Hori egiteko, hartzaleak uneko erabiltzailearekin elkartutako kontakturen bat duen jakin behar da eta ezezkoan, berri bat sortu. Behin kontaktu hori topatuta, mezu berri kopurua handitu egin behar da eta azken mezuak aldatu. Azken pauso hori uneko erabiltzailearen kontaktuan ere egin behar da. Eguneraketa garden bidez egiten dira pauso hauek datubasean.

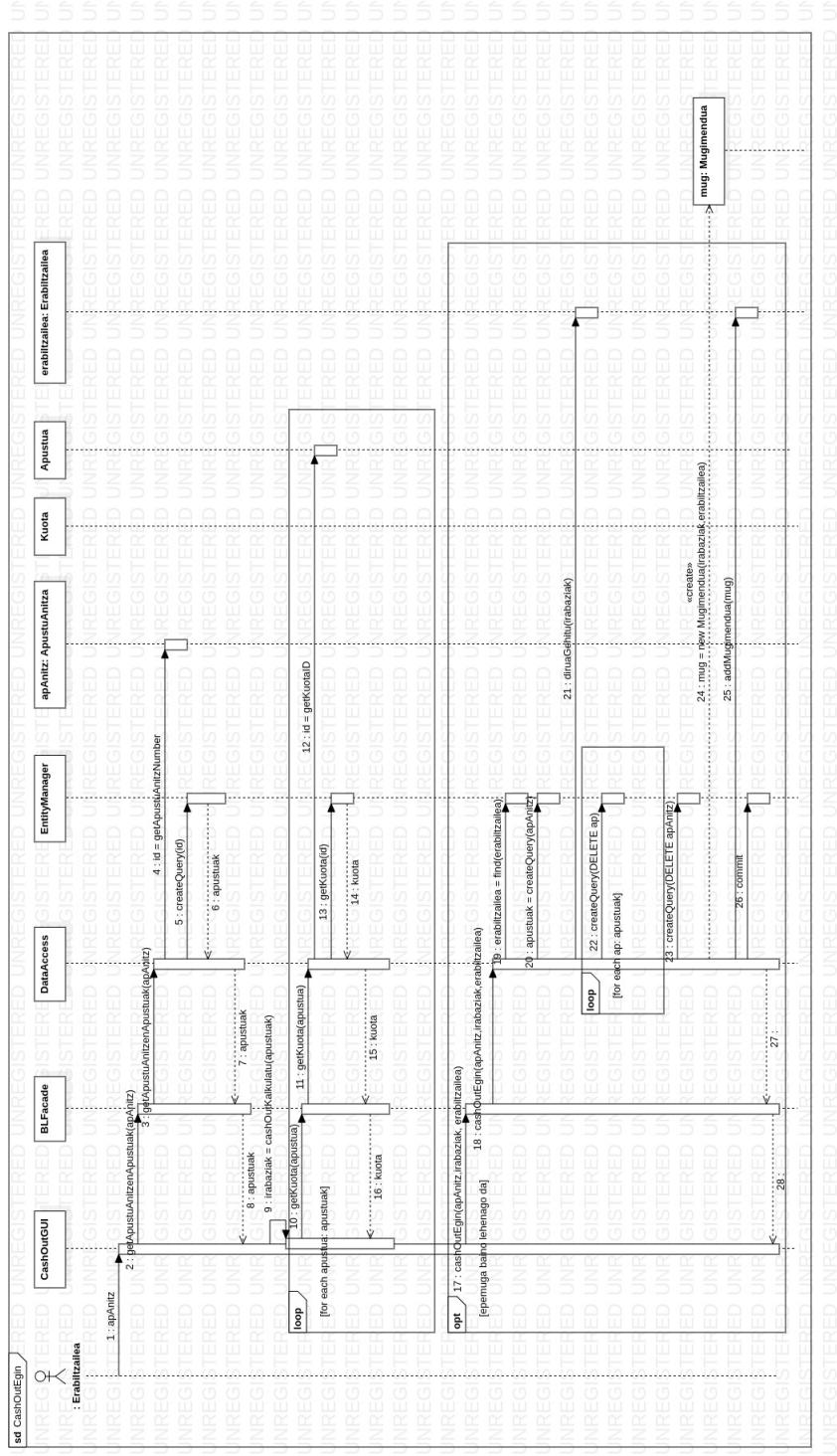


Irudia 20: Mezuak bidali sekuentzia diagrama.

#### 4.1.3 Cash out egin

Lehenik, apustu anitzen zerrendatik apustu anitz bat aukeratzen du erabil-tzaileak eta hura osatzen duten apustuak eta heien informazioa bistartatzen ditu pantailan, lortuko lituzkeen irabaziekin batera. Lehen begizta-kutxa cash out kalkulatzeko metodoak datubasera egiten duen deia islatzen du, apustu bakoitzaren kuotak eskuratu behar baititu irabaziak kalkulatzeko. Gogora de-zagun web zerbitzuak implementazerakoan ezinezko egiten dela entitate bate-tik bestera jauzi egitea datubaseko deiak egin gabe, horra hor datubaserako deien arrazoia.

Behin erabiltzaileak cash out egitea eta non egin erabakita, ekintza hori egi-teko garaiz dabilela ziurtatu behar da. Hala bada, dirua gehitzen zaio erabil-tzaileari eta apustu anitza zein apustuak banan-banan ezabatzen dira datuba-setik, mugimendu bat sortzeaz gain ekintzaren berri emateko.



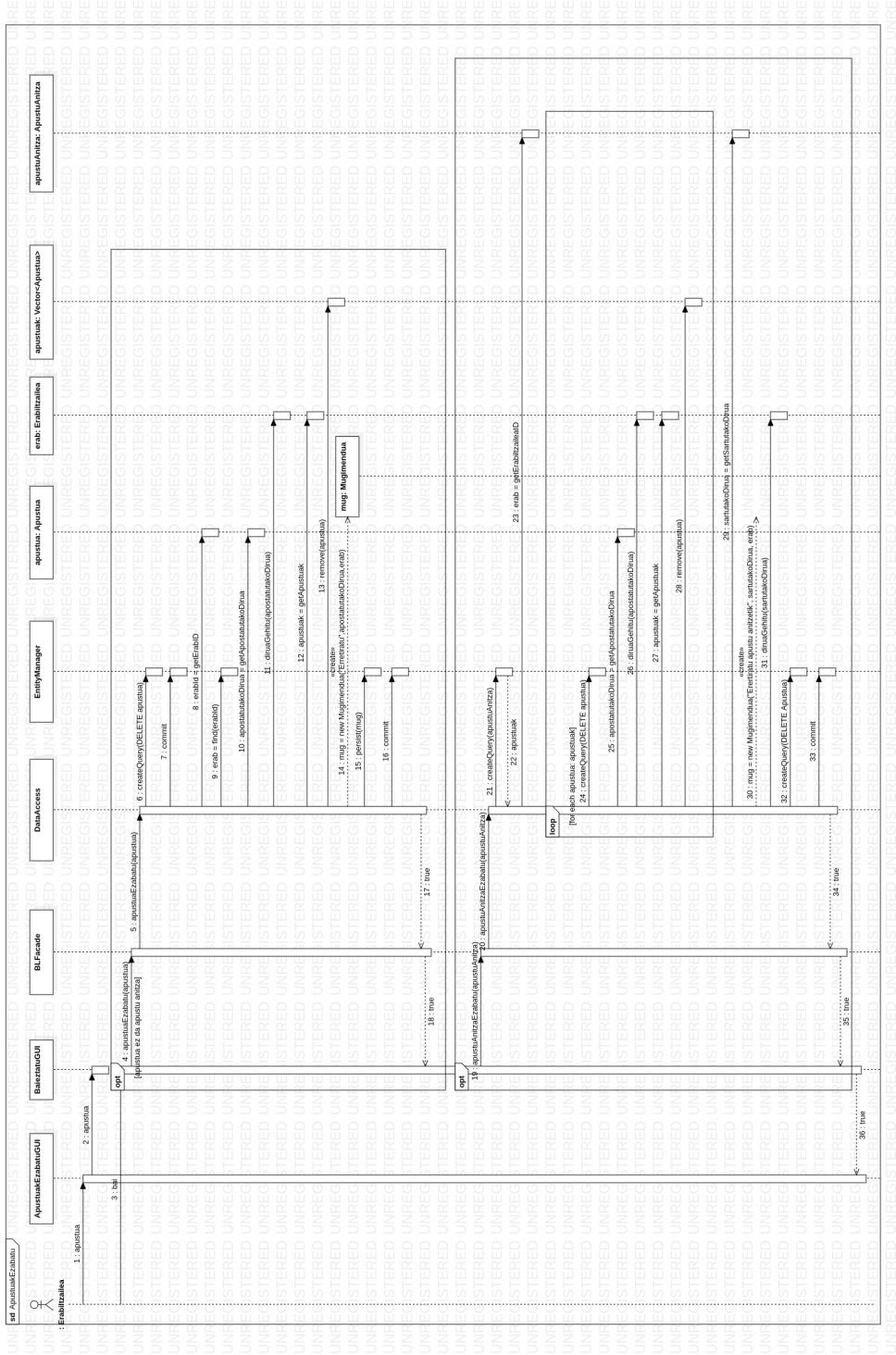
Irudia 21: Cash out egin sekuentzia diagrama.

## 4.2 Hirugarren iterazioan eguneratutako sekuentzia-diagramak

Apustu anitzen implementazioak hiru erabilpen kasu eguneratzera behartu gaitu. Azpian horien sekuentzia diagramen inguruko azalpenak topa daitezke.

### 4.2.1 Apustuak ezabatu

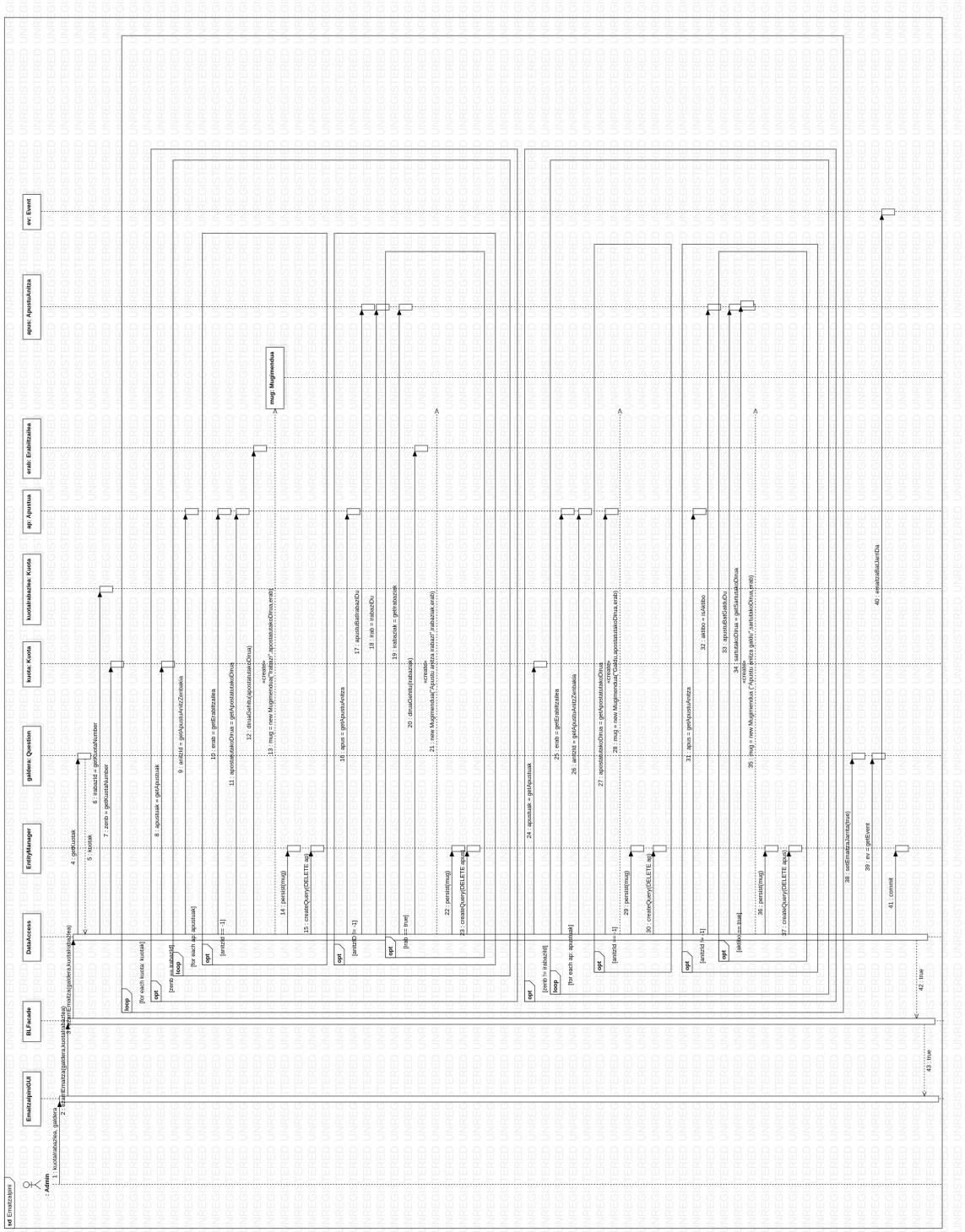
Apustuak ezabatzerako orduan orain apustu hori anitza den edo ez begiratu behar da. Ez bada apustu anitza, ohiko prozedura jarraitu behar da, hau da, apustua ezabatu datubasetik, dirua itzuli erabiltzaileari eta ekintzaren berri emango duen mugimendu bat sortu, ondoren datubasean gordetzea. Apustua anitza bada aldiz, aurreko ekintza guztiak apustu anitza osatzen duten apustu bakoitzeko egin behar dira eta amaieran, apustu anitza bera adierazten duen objektua ere datubasetik ezabatu.



Irudia 22: Apustuak ezabatu sekuentzia diagrama.

#### 4.2.2 Emaitza jarri

Emaitza jarri erabilpen kasuan ere antzeko aldaketa egin behar da. Emaitza jartzerako orduan, emaitza jarri zaion kuotako apustu guztiak banan-banan aztertu behar dira. Apustu horiek anitzak ez badira haien apustu anitzen identifikazio atributuak (anitzId) -1 balioa izango du. Beraz, balio hori due-nerako, kuota irabazoleko apustuen erabiltzaile guztiei dagozkien irabaziak gehitu behar zaizkie saldora, eta irabazi dutela ohartarazteko mugimendu bana sortu. Ondoren, apustuak ezabatzea erabaki dugu, datubasean apus-tu zaharren pilak ez sortzeko. Kuota galtzaileetan prozedura bera jarraitzen da, baina erabiltzaileari ez zaio dirurik gehitzen eta galdu duela adierazten zaio mugimenduan. Apustua anitza denean, hura osatzen duten apustu guz-tiak ezabatu behar dira (apustu anitzaz gain) eta erabiltzaileari gehitu behar zaion dirua apustu anitzetik eskuratu beharko da eta ez apustuetatik, haien "irabaziak" atributu guztiekin zero balioa izango dutelako. Azkenik, galderaren objektuan emaitza jarri zaiola ohartaraziko da eta gertaean ere bere galdera bati emaitza jarri zaiola ere jasoko da. Hala, administratzaleek etorkizunean galderetan emaitzak bi aldiz jartzea eragotziko da.

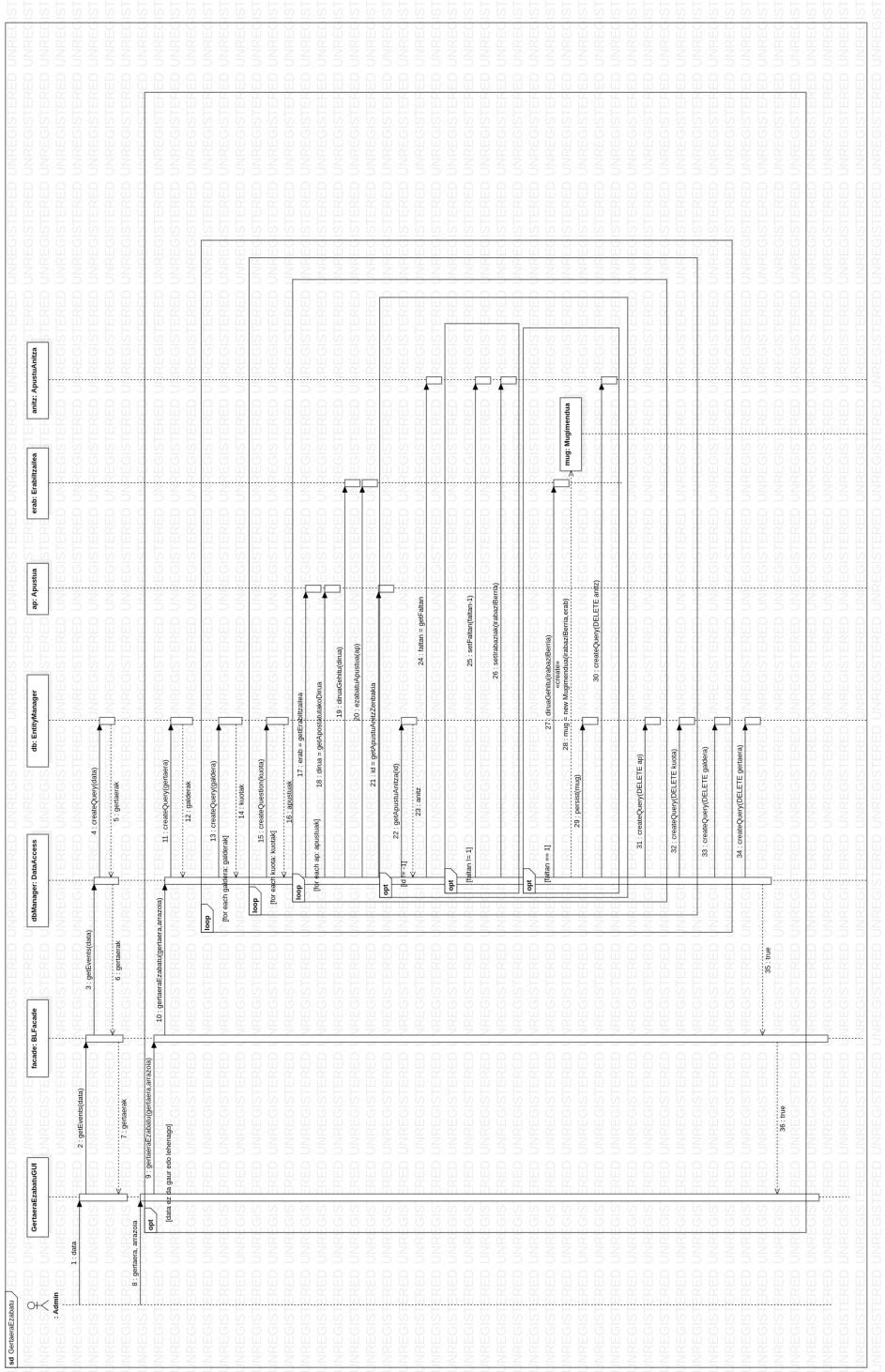


Irudia 23: Emaitza jari selventzia diagrama.

#### 4.2.3 Gertaera ezabatu

Gure aplikazioaren politika geraterak ezabatzeko orduan hurrena da: gertaera horrek apustu aktiboren bat badu, dirua itzuliko zaio apustu hori egin duen erabiltzaileari. Apustu anitzak implementatzean politika zabaldu egin behar izan dugu: apustu anitzen bat aktibo badago, erabiltzaileari gertaera horri esker apustu anitzean lor zezakeen irabazi gehigarria kenduko zaio, baina apustu anitzeko beste apustuak mantendu egingo dira. Geratera ezabatzean beste arazo gehigarri bat apustu anitz bati irabazteko soilik apustu bat irabaztea falta izatea da, kasu horretan, erabilpen kasu honen gain geratuko litzateke halako apustuak irabazi edo galdu diren jakinaraztearen ardura.

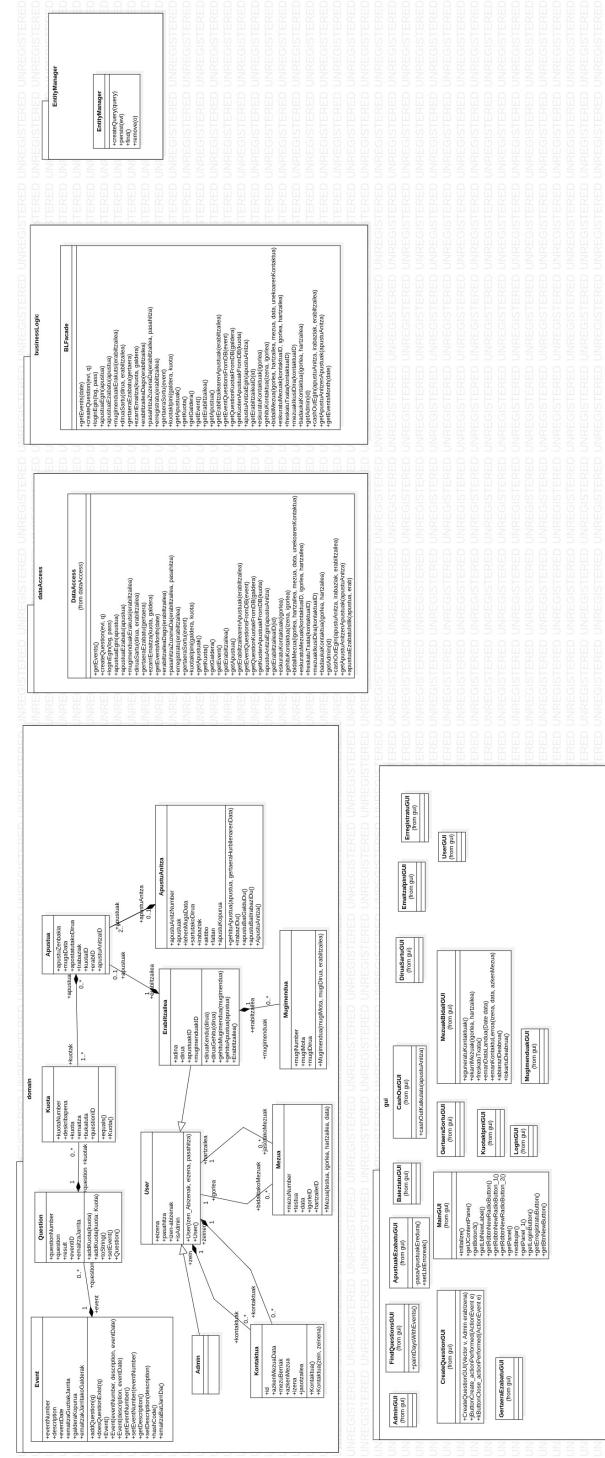
Sekuentzia diagraman ikus dezakegu, admnistratzialeak ohi bezala ezabatu nahi duen gertaera eta hori egitearen arrazoia sartu behar dituela. Beti ere, lehenik gertaera hautatzeko nahiko baliabideak eman behar dizkio makinak erabiltzaileari (gertaeren karga egin behar da). Behin zein gertaera ezabatu behar den jakinda, haren galderetara, azken hauen kuotetara eta azken hauen apustuetara sarbidea lortu behar du makinak. Gero, apustuak ezabatu beharko ditu banan-banan. Uneko apustua ezabatuta, apustua apustu anitza den edo ez begiratu behar du (identifikatzalea -1 al da?). Eta hala bada, hura irabazi edo galdu duen zehazteko zenbat apustu falta zaizkion jakin behar da, faltan atributua atzitzuz. Ezabatu behar den gertaeraren apustua erabakigarrria bada apustu anitz horrentzat, apustu anitz bat irabaztean jarraitzen den prozedura jarraitu beharko da edo galtzen denekoa, kasuaren arabera. Azkenik, kuotak eta galderak ezabatu beharko ditu makinak, haiei lotutako apustu guztiak ezabatu baditu jada. Azken pauso gisa, gertaera ezabatuko da.



Irudia 24: Gertaera ezabatu sekuentzia diagrama.

### 4.3 Klase diagrama

Jarraian aurki daiteke proiektuaren klase diagrama.



Iridia 25: Projektuaren klase diagrama:

## 5 Implementazioa

Atal honetan domeinuko klaseak nola implementatu diren azalduko da. Domeinuko klase bakoitzak apustu sistemako entitate bat adierazten du, besteak beste, gertaerak, galderak, apustuak, kuotak, etab. Atalaren helburua klase bakoitzaren metodoak dokumentatzea bada ere, errepikapenak saihesteko klase bakoitzaren geter eta seter metodoak ez dira ipini eta horiek atzitzen duten atributuaren azalpena atributu bakoitzaren azalpenean irakur daiteke.

Bestalde, ezin aipatu gabe utzi kalse guztiak haien instantzien identifikatzaila gisa jokatzen duen atributu bat dutela eta hau automatikoki sortzen dela exekuzio garaian, jva.xml paketea erabiliz.

### 5.1 Oinarritzko erabilpen kasuetako klaseak

#### 5.1.1 Event klasea

Event klaseak zazpi atributu ditu eta bakoitzaren betebeharra jarraian dagoen zerrendan azaltzen da:

- *Integer eventNumber*: gertaeraren identifikazio-zenbakia, datubasean identifikatzaila gisa jardungo duena.
- *String description*: gertaeraren izena gordetzeko atributua.
- *boolean emaitzaGuztiakJarrita*: gertaeraren galdera guztiak emaitza jarrita badute true balioa edukiko du, bestela false.
- *int galderaKopurua*: gertaerari gehitu zaion galdera kopurua gordetzen duen atributua.
- *int emaitzakJarritakoGalderak*: gertaeraren zenbat galderak duten emaitza jarrita adierazten du.
- *Date eventData*: gertaeraren data gordetzeko atributua.
- *Vector<Question> questions*: gertaeraren galdera guztiak gordetzeko bektorea.

Hiru eraikitzaile sortu dira: atributu gabekoa (web zerbitzuek eskatua), deskribapena eta data parametro gisa eskatzen dituena eta horiez gain gertaeraren identifikatzaila eskatzen duena. Eraikitzaileek parametro bakoitza dagon kion atributuari esleitzen diote eta *galderaKopurua* eta *emaitzakJarritakoGalderak* atributuak 0 zenbakarekin hasieratzen direla ziurtatzen dute.

Ohiko metodoez gain, ondorengo metodoak implementatu dira klase honetan:

- *Question addQuestion (String question)*: parametro gisa galdera adierazten duen karaktere katea sartuta, galdera bat gehitzen dio gertaeraren galdera bektoreari eta *galderaKopurua* atributua eguneratzen du. Azkenik, galdera itzultzen du, Question klasearen instantzia bezala.

- *boolean DoesQuestionExists (String question)*: parametro gisa galdera bat adierazten duen karaktere-katea sartuta, hura gertaeraren galdera bektorea dagoen edo ez aditzera ematen du. Bektorean baldin badago, true itzuliko du, bestela false.
- *void emaitzaBatJarriDa ()*: gertaeraren galdera bati emaitza bat jarri zaionean baliatzeko pentsatuta dago. *emaitzakJarritakoGalderak* eguneratzen du aipatutako kasuan eguneratu beharko lukeen moduan eta gertaeraren galdera guztiek emaitza jarrita badute *emaitzaGuztiakJarrita* aldagaien horren berri ematen du.

### 5.1.2 Question klasea

Question klaseak zazpi atributu ditu:

- *Integer questionNumber*: klasearen instantzia bakoitzaren identifikatzalea.
- *String question*: galderaren testua gordetzeko atributua.
- *int eventID*: galderari dagokion gertaeraren identifikatzailea gordetzeko atributua. Web zerbitzuen implementazioan erabilia.
- *boolean emaitzaJarrita*: emaitza jarrita baldin badu galderak, egia balia egongo da gordeta atributu honetan, bestela gezurra.
- *String result*: galderaren emaitza gordetzeko balio duen atributua.
- *Event event*: galderari dagokion gertaera gordetzeko balio duen atributua.
- *Vector<Kuota>*: galderari dagozkion kuotak gordetzeko erabiltzen den bektorea.

Eraikitzailee dagokienean, ohiko parametrorik gabeko eraikitzaileaz gain, hiru eraikitzaile ditu. Lehenengoan, galderaren identifikadoreoa, galdera eta galdera parte den gertaera sartu behar dira parametro bezala; bigarrenean, galdera eta galdera parte den gertaera; hirugarrenean aldiz, galdera soilik. Identifikatzailea automatikoki sortzen da parametro gisa sartzen ez bazaio, klase guztietan gertatzen den bezala.

Metodoei dagokienean, ohikoetaz gain erabiltzen direnek ez dute azalpen bezirkik merezi, izan ere, bakoitzak bere izenak esaten duena zehatz-mehatz edo ia zehatz-mehatz egiten du.

### 5.1.3 Kuota klasea

Kuota klaseak bederatzi atributu ditu eta aurreko biak ez bezala, bere osotu sunean guk implementatua izan da.

- *Integer kuotaNumber*: kuota bakoitzaren identifikatzailea, datubasean gordetzea ahalbidetuko duena.
- *String deskripapena*: kuotaren izena, hau da, kuotari dagokion galderaren erantzun posible bat.
- *double kuota*: erantzun posible horri ezarriko zaion kuota.
- *boolean emaitza*: kuotak aukera posible bat adierazten duenez, aukera hori emaitza baldin bada true balioa edukiko du atributu honek, bestela false.
- *boolean bukatuta*: kuotari dagokion galdera parte den gertaera bukatuta dagoen edo ez adierazten duen atributua.
- *int questionID*: kuotari dagokion galderaren identifikatzailea.
- *Question galdera*: kuotari dagokion galdera.
- *Vector<Apustua> apustuak*: kuota honen gainean egin diren apustuak gordetzen dituen bektorea. Hau da, bektore horretako apustu anitzak osatzen dituzten apustuen artetik bat kuota honen gainean eginda dago.
- *Vector<ApustuAnitza> apustuAnitzak*: kuota honen gainean egin diren apustu anitzak gordetzen dituen bektorea. Hau da, bektore horretako apustu anitzak osatzen dituzten apustuen artetik bat kuota honen gainean eginda dago.

Klaseak parametrorik gabeko eraikitzaleaz gain, hiru parametroko eraikitzai le bat du. Sartu behar zaizkion parametroak kuotaren deskripapena, kuotaznenbakia eta kuotari dagokion galdera dira. Eraikitzale hau erabiltzen da nagusiki.

Ohiko geter eta seterrez gain ez ditu aparteko metodorik klase honek.

#### 5.1.4 Apustua klasea

Apustua klasea hamar atributuz osatuta dago, bakoitzaren betebeharra jarreraian agertzen da:

- *Integer apustuZenbakia*: apustua identifikatzeko balio du, datubaseari begira definitu da,
- *Date mugaData*: erabiltzaileak bere apustua bertan behera uzteko aukera izango duen azken eguna gordetzeko balio du. Egun honen ondoren, apustua behin betikoa bihurtuko da.
- *double aposatutakoDirua*: erabiltzaileak apostatu duen diru kopurua gordetzeko atributua.
- *double irabaziak*: erabiltzaileak apustu hau irabaziz gero lortuko lukeen dirua gordetzeko balio duen atributua.

- *int kuotaID*: apustuari dagokion kuotaren identifikatzailea.
- *String erabID*: apustua egin duen erabiltzailearen identifikatzailea, hau da, erabiltzaile-izena.
- *ApustuAnitza apustuAnitza*: apustua parte den apustu anitza. Ez bada apustu anitz baten parte, null balioa edukiko du atributu honek.
- *Integer apustuAnitzaID*: apustua parte den apustu anitzaren identifikazio-zenbakia. Ez bada apustu anitz baten parte, -1 balioa edukiko du atributu honek.
- *Erabiltzailea erabiltzailea*: apustua egin duen erabiltzailea gordetzeko balio duen atributua.
- *Kuota kuota*: apustuari dagokion kuota gordetzeko balio duen atributua.

Klase honek ohiko parametrorik gabeko eraikitzaileaz gain, beste bi ditu. Lehenak muga data, apostatuko dirua, irabaziak, apustua egin duen identifikatzailea eta apustuari dagokion kuota ditu parametro gisa eta hauek dagozkien atributuei esleitzearruraturatzen da, identifikatzaileak dituzten atributuei jarri behar zaizkien balioak eskuratu eta jartzeaz gain. Bigarrenak lau parametro ditu eta ekintza berdinak egiten ditu: muga data, apostatutako dirua, erabiltzailea eta kuota.

Klasearen metodoak geler eta seterrak dira nagusiki eta ez direnak, klaseek eduki ohi dituzten ohiko metodoak dira, berezitasunik gabekoak denak.

### 5.1.5 Mugimendua klasea

- *Integer mugiNumber*: mugimendua identifikatzen duen atributua.
- *String mugiMota*: mugimenduaren mota adierazten duen atributua, aurrrera adieraziko dira mota posibletak.
- *double mugiDirua*: mugimendua egitearen ondorioz, erabiltzaileak galdu edo irabazi duen dirua adierazten duen atributua.
- *Erabiltzailea erabiltzailea*: mugimenduari dagokion erabiltzailea, hau da, mugimendua zein erabiltzailek egin duen.

Klaseak ohiko parametrorik gabeko eraikitzaileaz gain, hiru parametroko eraikitzaile bat du. Eraikitzaile honi mugimendu mota, mugitu den dirua eta mugimendua egin duen erabiltzailea sartu behar zaizkio. Zer da mugimendu mota? Mugimenduak adierazten duenaren arabera zortzi mugimendu mota bereiz daitezke:

- Apustua egin: erabiltzaileak apustu bat egin du, diru galera dakar horrek.

- Apustua ezabatu: erabiltzaileak aurrez egindako apustu bat ezabatu du, dirua berreskuratzea dakar horrek.
- Apustua irabazi: erabiltzaileak apustu bat irabazi du eta beraz, dirua ere bai.
- Apustua galdu: erabiltzaileak apustu bat galdu du eta beraz, dirua ere bai.
- Apustu anitza irabazi: erabiltzaileak apustu anitza irabazi du eta beraz, dirua ere bai.
- Apustu anitza galdu: erabiltzaileak apustu anitza galdu du eta beraz, dirua ere bai.
- Dirua sartu: erabiltzaileak dirua sartu du kontura eta horrek diru kopuru handitzea dakar.
- Gertaera bertan behera geratzea: erabiltzaileak bertan behera geratu den gertaera batean apustu egin badu, dirua itzuliko zaio. Berreskurapena dakar horrek.

Eraikitzaileak mugimendu horietako bat sartuta, mugimenduko dirua gehitu edo kendu egin den ondorioztatuko du eta hala adierazita utzi, ondoren pantailaratzeak behar bezala egiteko. Gainerako metodo guztiak ohikoak dira.

## 5.2 Apustu anitza erabilpen kasuko klaseak

Apustu anitzen funtzionalitatea ahalbidetzeko balio duten klaseak ageri dira jarraian.

### 5.2.1 ApustuAnitza klasea

Apustu anitzen klaseak zortzi atributu ditu eta bakoitzak honakoa adierazten du:

- *Integer apustuAnitzNumber*: apustu anitzaren identifikatzaila gordetze-ko balio duen atributua.
- *Vector<Apustua> apustuak*: apustu anitza osatzen duten apustuak, hau da, erabiltzaileak egindako apustuak eta ondoren apustu anitzaren baitan bildu dituenak.
- *Date lehenMugaData*: erabiltzaileak apustu anitza bertan behera uzteko aukera izango duen azken eguna. Egun hori apustu anitza osatzen duten apustuen artetik emaitza jakingo den lehen apustuaren gertaeraren eguna izango da. Izan ere, apustu bat jada irabazi edo galdu duen jakinda, apustu anitza ezingo du ezabatu.
- *double sartutakoDirua* erabiltzaileak apostatu duen dirua.

- *double irabaziak*: erabiltzaileak apustu anitza irabaziko balu lortuko lu-keen dirua.
- *boolean aktibo*: apustu anitza aktibo dagoen edo ez adierazten duen atri-butua. Apustu anitza martxan baldin badago (ez da ezabatu) true ba-liaoa edukiko du, bestela false.
- *int faltan*: apustu anitza irabazteko oraindik ere irabazi behar dituen apustuak. Gogora dezagun apustu anitza hainbat gertaeraren gainean egindako apustuez osatuta dagoela eta hauek egun desberdinatan ger-ta daitezkeela, beraz, apustu anitzaren emaitza azken gertaera bukatu ondoren jakingo da. Kontagailu honek uneoro apustu anitzeko zenbat apustuk oraindik emaitza jarrita ez duten zenbatuko du.
- *apustuKopurua*: apustu anitza osatzen duten apustu kopurua.

Klaseak ohiko parametrorik gabeko eraikitzailea eta bi parametroko eraikitzaile bat ditu. Bigarren honek sartutako dirua eta irabaziak eskatzen ditu para-metro gisa. Hasieraketak egiterako orduan, irabaztea geratzen zaizkion apustu kopurua zero zenbakiaz hasieratzen du, apustuKopurua bezalaxe eta aktibo gisa ezartzen da apustua. Modu honetan, azpian azalduko den *gehituApustua* metodoa erabiltzera bultzatzen da programatzalea apustu anitzera apustu bat gehitu nahi duenean, atributuetan eguneraketak behar bezala egiten direla ziurtatzeko.

Klase honetan ondorengo metodoak dira azpimarragarriak:

- *void gehituApustua (Apustua ap, Date gertaeraHurbilenarenData*: apus-tu anitza apustu bat gehitzerako orduan erabili behar den metodoa. Apustu kopurua eta irabazteko falta den apustu kopuruaren atributuak eguneratzen ditu, apustua gehitu eta apustu anitzetik erretiratzeko az-ken egunaren data eguneratzeaz gain.
- *boolean irabaziDu ()*: erabiltzaileak apustu anitza irabazi duen edo ez itzuliko du, hartarako beharrezko atributuak begiratuz.
- *void apustuBatGalduDu ()*: erabiltzaileak apustu bat galdu duela jakiten denean deitu behar den metodoa, apustu anitzari aktibotasuna kenduko diona.
- *void apustuBatIrabaziDu ()*: erabiltzaileak apustu bat irabazi duela ja-kiten denean deitu behar den metodoa, apustua aktibo badago apustu anitza irabazteko apustu bat gutxiago falta zaiola adierazita utziko due-na.

### 5.3 Erabiltzaileak adierazteko klaseak

Jarraian dauden klaseak aplikazioan aurki daitezkeen erabiltzaile motak adie-razteko erabili dira.

### 5.3.1 User klasea

Aplikazioko edozein erabiltzaile User motako erabiltzailea da. Hau da, klase hau administratzale eta erabiltzaile arrunten guraso klasea da. Jarraian aurki daitezke, hortaz, aplikazioko erabiltzaile erregistratu guztiak dituzten atributuak:

- *String izena\_abizenak*: erabiltzailearen izena eta bizenak gordetzenko balio duen atributua.
- *String eizena*: klasearen instantziaren identifikatzaile gisa jokatzen duen atributua, erabiltzailearen ezizena gordetzenko ere balio duena. Identifikatzaile gisa erabiltzen denez, aplikazioak ez duitu onartzen erabiltzaile izen bereko bi erabiltzaile desberdin.
- *String pasahitza*: erabiltzailearen kontura sartzeko asmatu behar den pasahitza gordetzen duen atributua.
- *Vector<User> kontaktuak*: txataren erabilpen kasura bideratuta dagoen bektorea da, instantziak adierazten duen erabiltzaileak zein erabiltzaile-rekin elkarritzeta bat zabalduta duen adierazten du. Elkarrizketak mota desberdinak erabiltzaileen artean gerta daitezkeenez, bektoreak User motako objektuak edukiko ditu.

Bi eraikitzaile ditu: ohiko parametrorik gabekoa eta hiru parametroduna. Azken honek izen abizenak, erabiltzaile izena eta pasahitza eskatzen ditu eta horiek dagozkien atributiei esleitzenten dizkie. Klaseak ez du aparteko metodorik, geter eta seterrak soilik.

### 5.3.2 Admin klasea

Admin klasea administratzale maila duten erabiltzaileak adierazteko erabiltzen da aplikazio honetan. User klasearen ume klasea da. Erabilpen kasuen atalean deskribatu da administratzale batek zein aprteko eskubide dituen. Klaseak bere guraso klaseak dituen atributuetaz gain ez du atributu gehiagorik.

Ohiko parametrorik gabeko eraikitzaileaz gain, beste eraikitzaile bat du. Eraikitzaile horrek hiru parametro ditu: izen\_abizenak, erabiltzaile izena eta pasahitza eta parametroak guraso klasearen eraikitzaileera igortzeaz arduratzen da. Metodoei dagokienean, guraso klasearen atributuetatik informazioa eskratzeaz edo ezartzeaz arduratzen diren geter eta seter metodoak soilik ditu.

### 5.3.3 Erabiltzailea klasea

Erabiltzailea klasea administratzale maila ez duten baina erregistratuta dauden erabiltzaileak adierazteko erabiltzen da apliakzio honetan. User klasearen ume klasea da. Erabilpen kasuen atalean deskribatu da erabiltzaile arrunt batek zein eskubide dituen. Klaseak bere guraso klaseak dituen atributuez gain honako atributu hauek ditu:

- *int adina*: erabiltzailearen adina gordetzen da hemen, gogora dezagun erregistroa gauzatzeko baldinztetako bat adin nagusikoa izatea dela.
- *double dirua*: erabiltzaileak duen saldoa gordetzeko erabiltzen den atributua.
- *Vector<Integer> mugimenduakID*: erabiltzaileak egindako mugimendu guztien identifikatzailak gordetzen dituen bektorea.
- *Vector<Integer> apustuaKID*: erabiltzaileak egindako apustu guztien identifikatzailak gordetzeko balio duen bektorea.
- *Vector<Mugimendua> mugimenduak*: erabiltzaileak egindako mugimendu guztiak gordetzen dituen bektorea.
- *Vector<Apustua> apustuak*: erabiltzaileak egindako apustu guztiak gordetzen dituen bektorea.

Klaseak ohiko parametrorik gabeko eraikitzailaz gain, lau parametroko bat du. Eraikitzale honek izen abizenak, erabiltzaile izena, adina eta pasahitza eskatzen ditu. Metodoei dagokienean, ohiko geter eta seterrak arduratzen dira identifikatzailerei bektoreak eguneratzeaz, mugimendu eta apustuak dituzten bektoreen geter eta seterrak baliatuz.

## 5.4 Mezuak bidali erabilpen kasuko klaseak

Mezuak bidaltzearen erabilpen kasua ahalbidetzeko baliatzen dira azpiko bi klaseak.

### 5.4.1 Kontaktua klasea

Erabiltzaile batek dituen kontaktuak, elkarritzketen definizioaren antzekoa da kontaktuaren definizioa, adierazteaz arduratzen da klase hau. Funtsezkoa da mezuak bidaltzerako orduan nori bidali jakiteko eta batez ere, erabiltzaileak bidalitako mezuak modu antolatuan gordetzeko datubasean. Jarraian ageri diren zortzi atributuak ditu:

- *Integer id*: kontaktuaren identifikatzaila, datubasera begira.
- *String izena*: kontaktua dagokion erabiltzailearen erabiltzaile izena gordetzen duen atributua.
- *String jasotzailea*: kontaktua dagokion erabiltzailea igorletzat hartuta, atributu honetan hartzailaren erabiltzaile izena gordeko da.
- *Date azkenMezuaData*: kontaktuak adierazten duen elkarrizketan igorri den azken mezuaren data gordetzeko atributua.
- *User zein*: kontaktuak zein erabiltzaile duen hartzailertzat, bere erabiltzaile instantzia gordeko da atributu honetan.

- *User zeinena*: kontaktua zeinena den (igorlea), bere erabiltzaile instantzia gordeko da atributu honetan.
- *int mezuBerriak*: igorleak kontaktuko mezuak ikusi zituenetik hartzaleak igorri duen mezu kopurua gordetzeko atributua.
- *String azkenMezua*: kontaktuak adierazten duen elkarrizketan igorri den azken mezuaren testua gordetzen duen atributua.

Beste klase guztietan ez bezala, erlazioak adierazten dituzten atributuak ez daude `@XmlAttribute` anotazioaz azpimarratuta, zergatik? Ez dagoelako arriskurik begiztak sortzeko, izan ere, beste klaseek ez dute erlaziorik klase honekin, erlazioak aldebakarrekoak direlako. Honek erraztu egiten ditu web zerbitzuak implementatzerako orduan datu basera egin beharreko deiak, atributuak ez baitira blokeatzen begiztak saihesteko.

Klaseak bi eraikitzaile ditu: parametrorik gabekoa eta bi parametroduna. Bigarren hau da erabiliena eta kontaktua zeinena den eta zein duen hartzailetzat sartea eskatzen du. Hori eginda, identifikatzaileak gordetzen dituzten atributuen balioak ondorioztatzeko gai da, erlazioak adierazten ez dituzten atributuak eguneratzeaz gain. Metodo guztiak geler eta seter motakoak dira.

#### 5.4.2 Mezua klasea

Mezuak adierazteaz arduratzen da klase hau eta zazpi atributuz osatuta dago:

- *Integer mezuNumber*: klasearen instantzia bakoitzaren identifikatzailea, datubaseari begira.
- *Date data*: mezua igorri den data osoa gordetzen duen atributua.
- *String testua*: mezua bera, hau da, testua.
- *User igorlea*: mezua bidali duen erabiltzailea gordetzen duen atributua.
- *User hartzailea*: mezua iritsi behar zaion erabiltzailea gordetzen duen atributua.
- *String igorleID*: mezua bidali duen erabiltzailearen identifikatzailea gordetzen duen atributua.
- *String hartzaileID*: mezua jaso behar duen erabiltzailearen identifikatzailea gordetzen duen atributua.

Klaseak bi eraikitzaile ditu: parametrorik gabekoa eta lau parametroduna. Lau parametroko honek testua, igorlea, hartzailea eta data sartzeko eskatzen ditu eta gainerako atributu guztiak dagozkien balioekin hasieratzeaz arduratzen da. Metodo guztiak geler edo seter motakoak dira.

## 6 Ondorioak

Proiektua apustu sistema baten prototipo txiki bat da, gabeziak dituena. Harren diseinua, funtzionalitateak eta erakargarritasuna egokiak eta ondo planteatuak izan direla uste dugu, baina jakin badakigu implementazioan, bereziki eraginkortasun aldetik, zer hobetu baduela. Etorkizunean hautsi beharko liratzekeen mugen artean dago hainbat GUIetan, baina bereziki apustuak ezabatzeko GUIlean, apustu kopuru ertain bat egin ondoren kargatzeko behar duten denbora. Izan ere, karga horiek motelak dira datubasera egiten diren dei ugarien ondorioz.

Etorkizunean garatu daitekeen beste erabilpen kasu bat bote bidezko apustuena da, herri, elkarte edo laguntalde mailako apustuetara begira. Erabilpen kasu honetan erabiltzaileek kudeatuko lukete botea, haiiek sortu eta haiiek jarrita parte hartzeko sartu beharko litzatekeen dirua. Galdera bat jarrita, erantzun zuzena asmatu dutenek irabaziko lukete botea (gertaera honetan parte hartu duten erabiltzaileek sartutako diru totala) eta irabazle horien artean banatuko litzateke. Enpresak ehuneko bateko komisio bat ordainduko luke bote horretatik apustua antolatzeko webgunea uzteagatik erabiltzaile antolatzaila eta parte hartzaleei. Era honetan, herri mailako esku pilota edo antzeko txapelketetan apustuak egiteko aukera emango genuke, administratzialeek zuzenean sortu behar izan gabe eta erabiltzaileei sisteman parte hartzeko aukera gehiago emanez.

Hala ere, informatikan hasiberriak garen bi ikaslek garatu dugu proiektu hau eta pertsonalki, hilabete gutxi batzuetan asko ikasi dugula nabaritu dugu.

Duela hilabete batzuk ez genuen inolaz ere lortuko halako proiektu bat garatzea eta orain, atzera begira, egin dugun aurrerapena pozteko modukoa dela ikusten dugu. Proiektua garatzeko SCRUM metodologia baliatu da eta iterazioka banatu dugu proiektua. Iterazio bakoitzaren ondoren emaitza exekutarri bat lortu dugu eta eskailerak igotzen balego bezala, proiektuak aurrera nola egin duen ikusteko aukera eman zaigu. Alde horretatik gustora egon gara metodologiarekin. Hala ere, proiektua egiteko denbora asko behar izan dugu, hasieran uste genuena halako bi edo hiru eta horrek proiektuaren kalitatean nolabait eragin duela ere badakigu. Hurrengo proiektuei begira lana ahalik eta modu jarraituenean eta denbora aldetik lasai egitearen garrantzia azpimarratu nahiko genuke, halako proiektuen garapenean beti daudelako ezustekoak.

## 7 Erreferentziak

Azalpen laburtuak dituen bideoa hemen aurki daiteke: [klik egin](#).