

# **SORTWARE INGENIARITZA 2020-21**

## **PROIEKTUA**

### **Aurkibidea**

1. Egileak	1
2. Sarrera	1
3. Eskakizun Bilketa	2
4. Diseinua	14
5. Implementazioa	19
6. Ondorioak	22
7. Bideoaren URL-a	22
8. Kodearen URL-a	22

## **1. Egileak**

Itziar Irastorza, Aitor Fontecha eta Maider Amuchastegui.

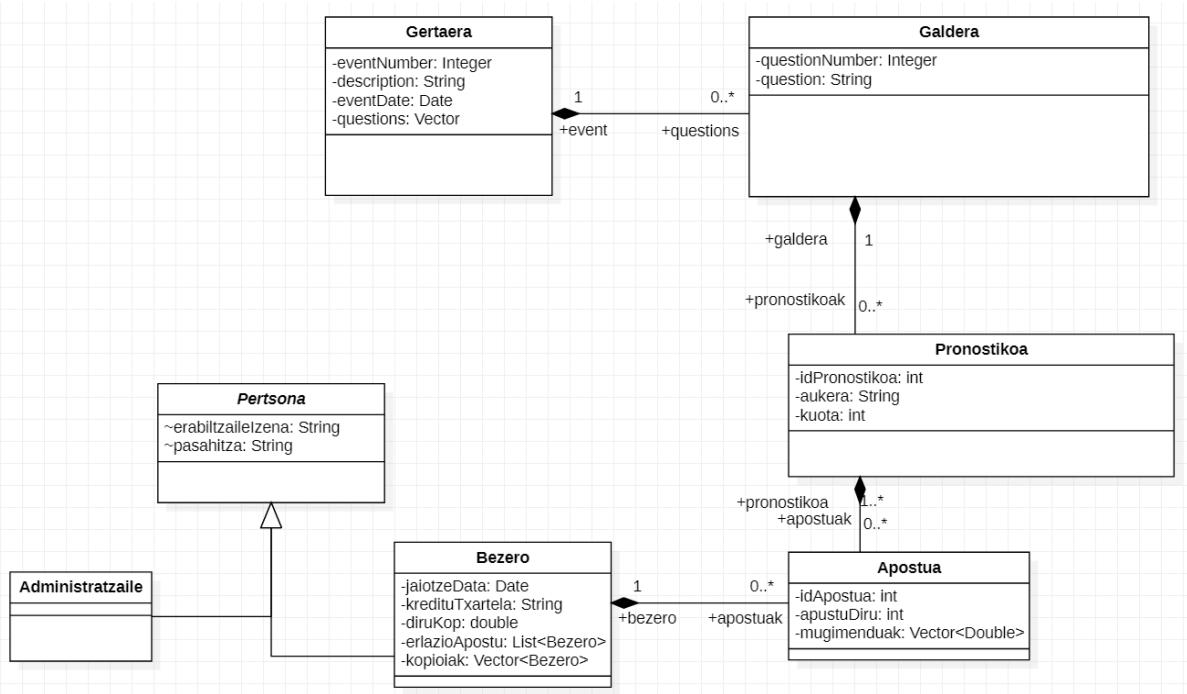
## **2. Sarrera**

Garatu dugun proiektuaren helburua, hiru mailako software arkitektura batean diseinatutako apustuak kudeatzen dituen informazio sistema bat garatzea izan da.

Funtzionalitate nagusiak, besteak beste, apustu egitea, apustu anitza egitea, pronostikoak sortzea, gertaerak sortzea, emaitzak ipintzea, mugimenduak ikustea, erabiltzaileak errepikatzea eta apostu famatuena egitea dira. Apustu egitea, apustu anitza egitea, mugimenduak ikustea, erabiltzailea errepikatzea eta apustu famatuena egitea bezero erregistratu orok egin dezake. Pronostikoak sortzea, gertaerak sortzea eta emaitzak ipintzea administratzaleak soilik egin ahal izango ditu.

### 3. Eskakizun Bilketa

#### 3.1. DOMEINUAREN EREDUA



Proiektu honetako klase nagusietako bat *Gertaera* klasea da. Gertaerak, data desberdinetan egiten diren partida desberdinak izango dira. Gertaera bakoitzeko, galderak izan ahalko ditugu, esaterako, irabazlea zein izango den galdetzeko edota partidan zenbat gol sartuko diren galdetuz. Horregatik, gertaera batek hainbat galdera edo bat ere ez izateko aukera dago, baina galdera bat derrigorrez gertaera konkretu bati lotuta egongo da.

*Galdera* klasea *Pronostikoa* klaseari lotuta dago. Galdera bakoitzak hainbat pronostiko izan ditzake edo bat ere ez. Baina, pronostiko bakoitza galdera zehatz bati lotuta egon beharko da.

*Pronostikoa* klaseko objektuek, hau da, pronostikoek, galdera bati erantzuna ematen die eta erantzun horrek irabazteko duen probabilitatearen araberakoak izaten dira. Pronostiko batek zehazten du apostu batean irabazia zenbatekoa izan ahalko den. Horregatik, *Apostua* klasea eta *Pronostikoa* klasea lotuta daude.

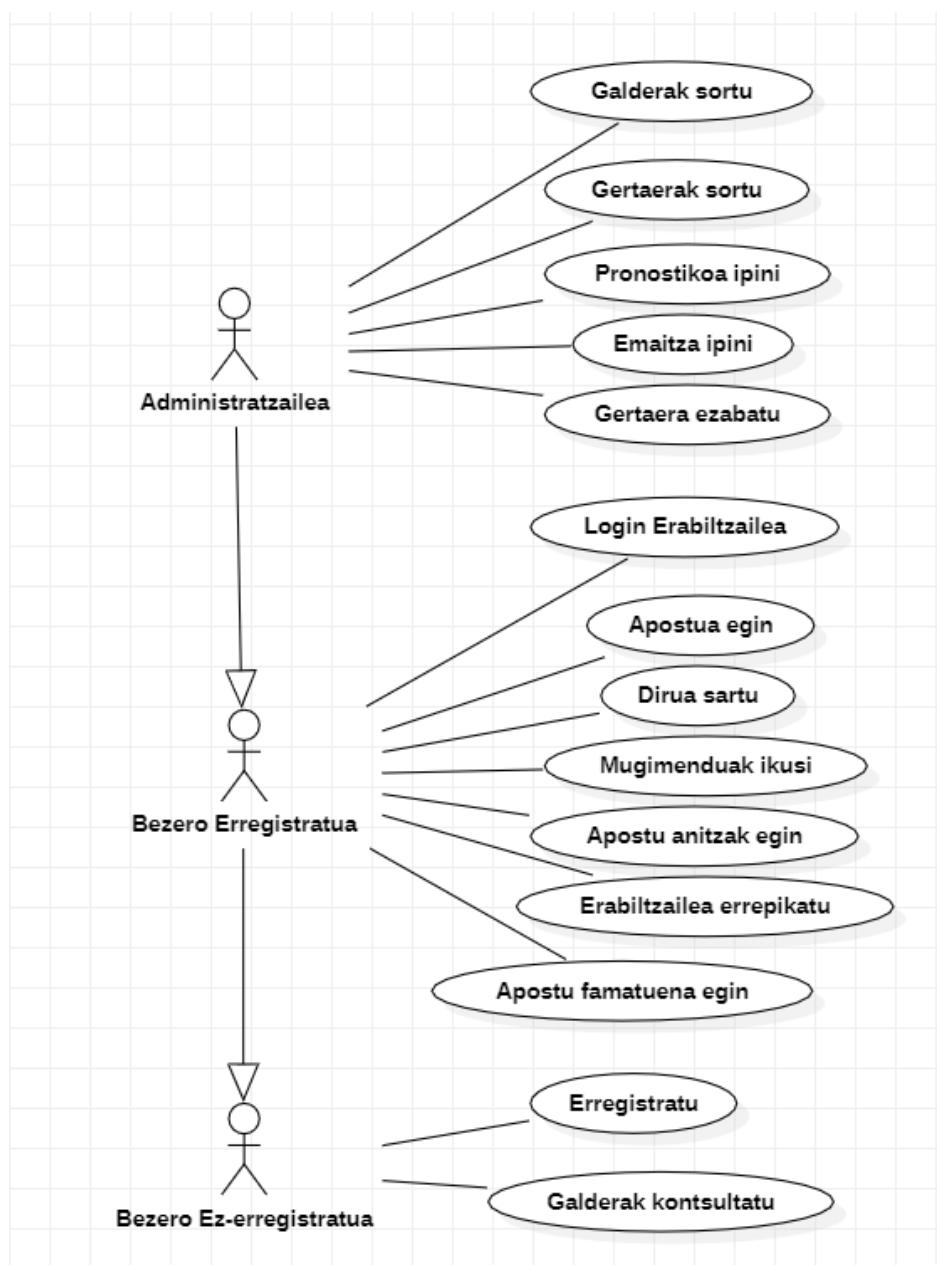
Pronostiko batek hainbat apostu izan ahalko ditu edo bat ere ez, eta apustu batek gutxienez, pronostiko bat izan beharko du. Azken finean, apustu egiten dugunean pronostiko bat aukeratzen dugu, gure iritziz pronostiko irabazlea izango dena. Beraz, ezin dugu apustu egin pronostikorik gabe.

Azkenik, aipatu behar da proiektuak bi erabiltzaile mota dituela, administratzailea eta bezeroa. Klase hauek *Pertsona* klase abstraktutik heredatzen dira. Bi erabiltzaile motek, erabiltzaile izena

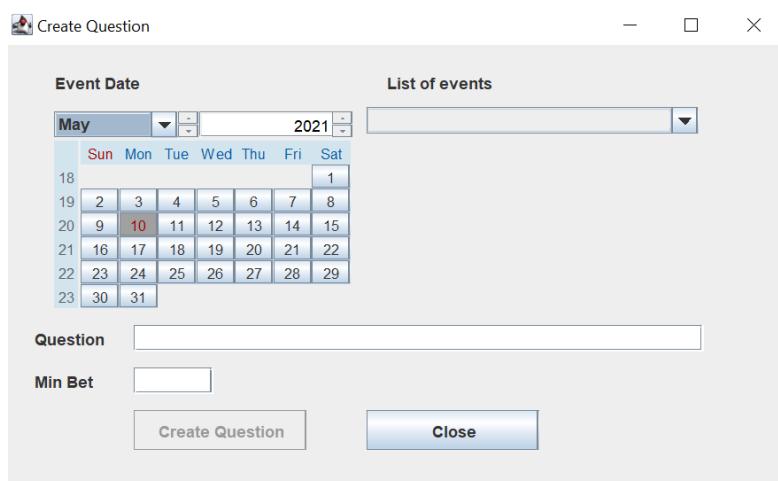
eta pasahitza izango dituzte, login egin ahal izateko, horregatik, bi atributu hauek *Pertsona* klasearen atributuak izango dira, eta beste biek heredatu egingo dituzte.

Bezeroek, apustu egin ahal izango dute. Horregatik, *Bezero* eta *Apostua* klaseak erlazionatuta daude. Bezero batek hainbat apostu izango diu edo bat ere ez, baina apustu batek bezero bat izango du nahitaez. Esaterako, bezero erregistratu berri batek ez du apusturik izango, baina ez da existituko inork egin ez duen apusturik.

### 3.2. ERABILPEN KASUAK



## GALDERAK SORTU:



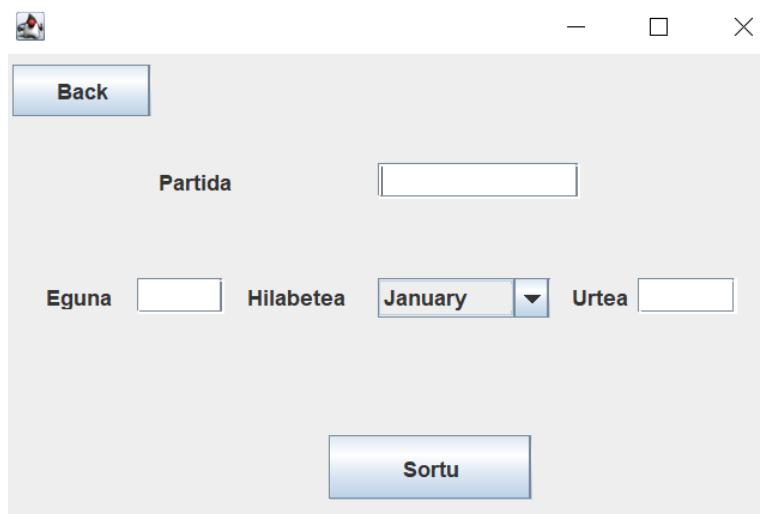
Oinarrizko fluxua:

1. Administratzailak data aukeratzen du.
2. Sistemak data horretako gertaerak erakusten ditu.
3. Administratzailak gertaera aukeratzen du.
4. Administratzailak galdera eta betMinimoa betetzen ditu.
5. Sistemak galdera berria gehitzen du.

Fluxu alternatiboa:

1. Data horretan ez daude gertaerak. Galdera ezin da sortu.
2. Galdera ez da idatzi. Galdera ezin da sortu.
3. BetMinimoa hutsik dago ala ez da balio numerikoa. Galdera ezin da sortu.
4. Gertaeraren data iraungi da. Galdera ezin da sortu.

## GERTAERAK SORTU:



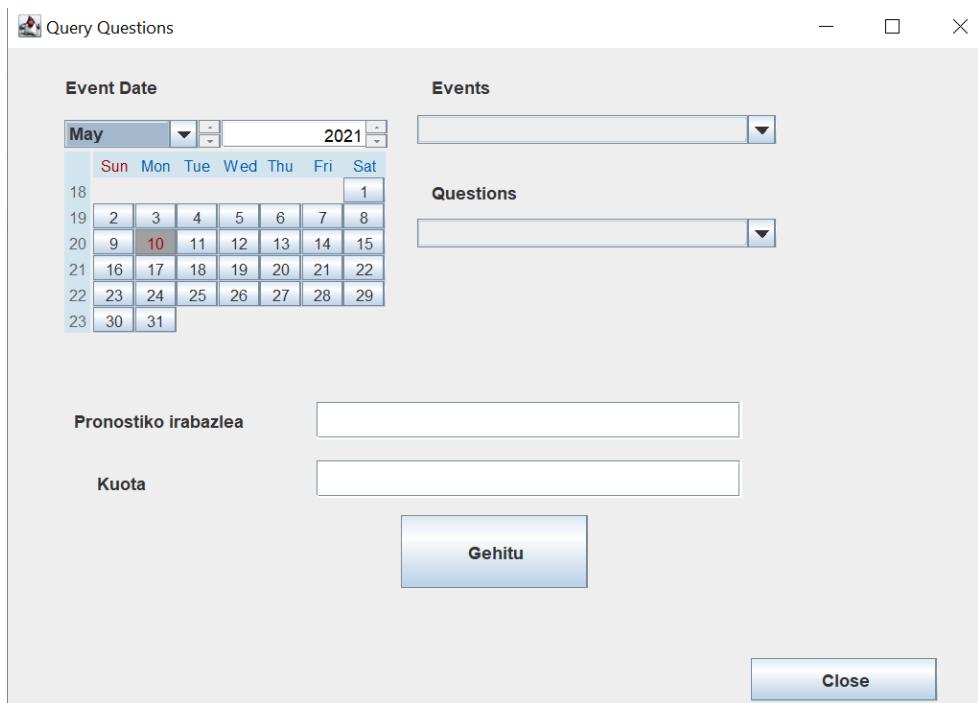
Oinarrizko fluxua:

1. Administratzailak gertaeralzena, eguna, hilabetea eta urtea datuak betetzen ditu.
2. Sistemak datuak zuzenak diren aztertzen ditu.
3. Sistemak gertaera berria gehitzen du.

Fluxu alternatiboa:

1. Data horretan dagoeneko gertaera existitzen da. Gertaera ezin da sortu.
2. Data ez da zuzena. Gertaera ezin da sortu.
3. Gertaeraren data iraungi da. Galdera ezin da sortu.

## PRONOSTIKOA IPINI:



Oinarrizko fluxua:

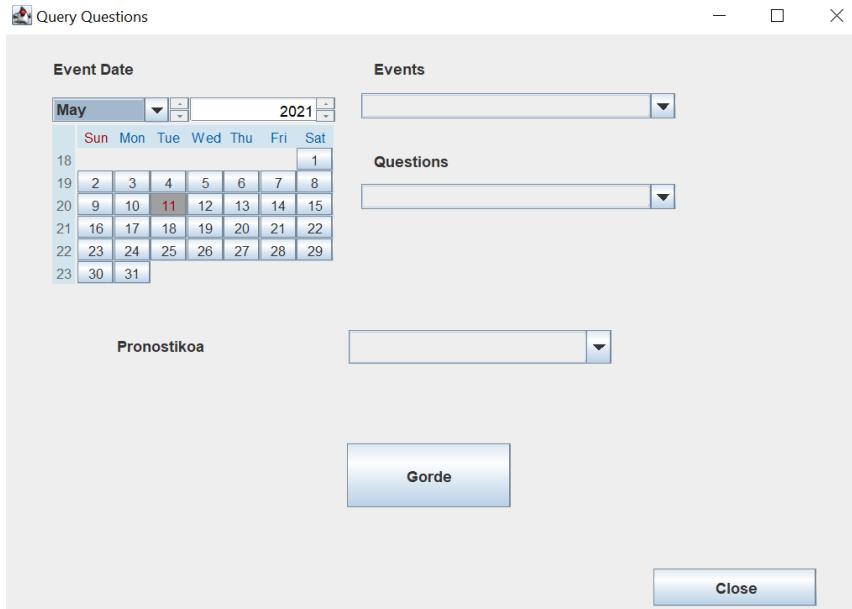
1. Administratzailak data, gertaera eta galdera aukeratzen ditu.
2. Sistemak galdera horretako pronostikoak erakusten dizkio.
3. Administratzailak pronostikoa eta kuota betetzen ditu.
4. Sistemak pronostiko berria gordetzen du.

Fluxu alternatiboa:

1. Data horretan ez dago gertaerarik.
2. Gertaera horretan ez dago galderarik.

3. Pronostikoaren formatua ez da zuzena.

### EMAITZA IPINI:



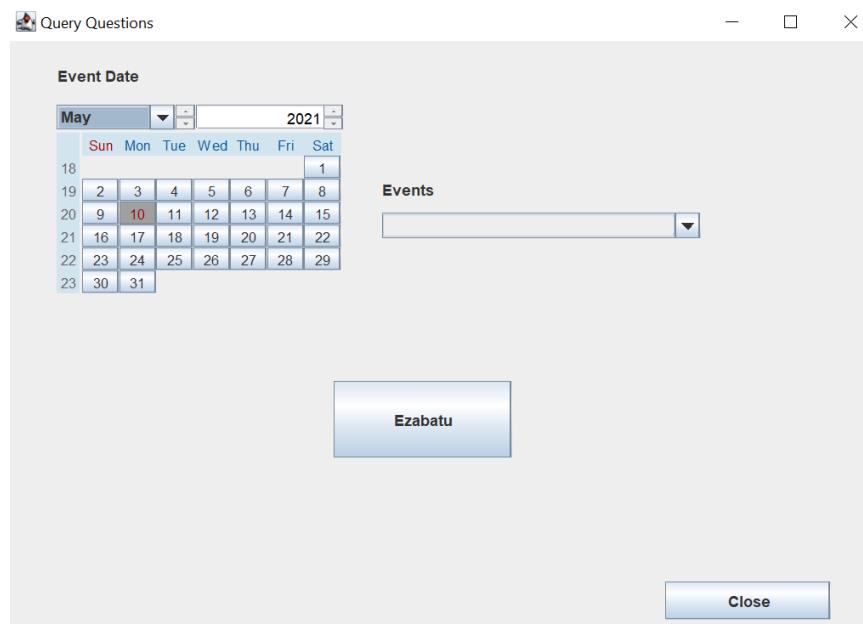
Oinarrizko fluxua:

1. Administratzailak data aukeratzen du.
2. Sistemak data horretako gertaerak erakusten ditu.
3. Administratzailak gertaera aukeratzen du.
4. Administratzailak emaitza betetzen ditu.
5. Sistemak emaitza gordetzen du.
6. Sistemak bezero bakoitzaren irabaziak eguneratzen ditu.

Fluxu alternatiboa:

1. Aukeratutako datan ez daude gertaerak.
2. Aukeratutako dataren formatua ez da zuzena.
3. Emaitza ez da idatzi. Emaitza ezin da gorde.
4. Gertaeraren data ez da igaro. Ezin da emaitza jarri amaitu/jokatu ez den gertaera batean.

## GERTAERA EZABATU:



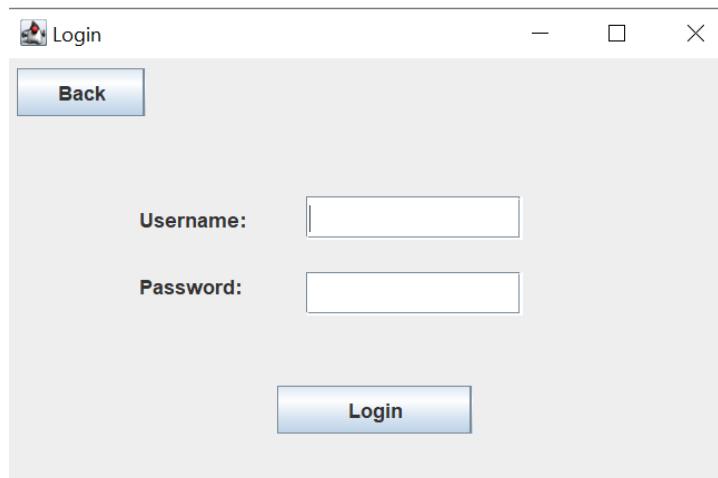
Oinarrizko fluxua:

1. Administratzaileak data aukeratzen du.
2. Sistemak data horretako gertaerak erakusten ditu.
3. Administratzaileak gertaera aukeratzen du.
4. Sistemak gertaera ezabatzen du.

Fluxu alternatiboa:

1. Aukeratutako datan ez daude gertaerak.
2. Aukeratutako dataren formatua ez da zuzena.

## LOGIN:



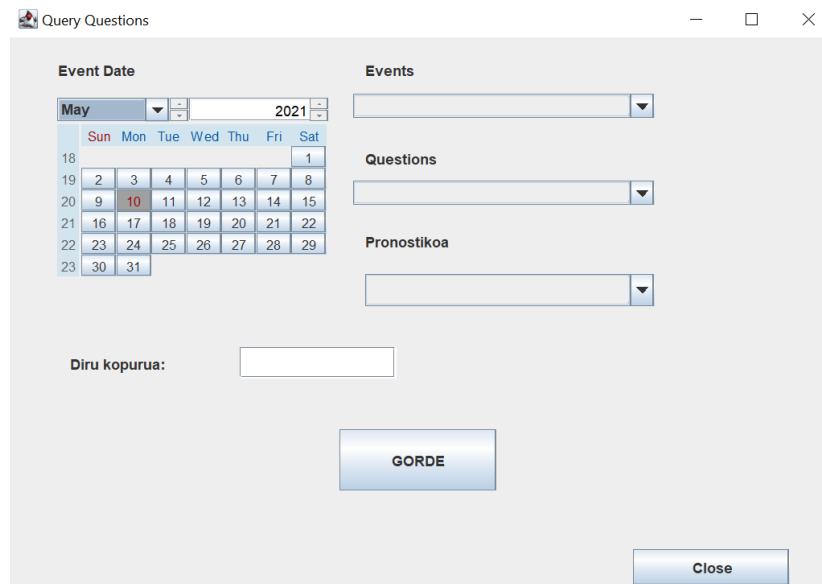
Oinarrizko fluxua:

1. Bezero Erregistratuak erabiltzailea eta pasahitzaz betetzen ditu.
2. Sistemak rola Bezeroa den begiratzen du.
3. Sistemak Bezeroa pantaila erakusten du.

Fluxu alternatiboa:

1. Erabiltzailea edota pasahitzaz hutsik daude. Login Bezeroa ezin da egin.
2. Erabiltzailea edota pasahitzaz ez dira zuzenak. Login Bezeroa ezin da egin.
3. Rola ez da Bezeroa. Login Bezeroa ezin da egin.

### APOSTUA EGIN:



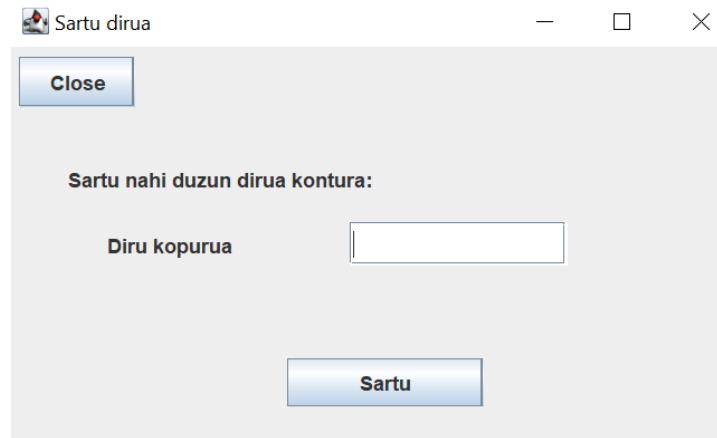
Oinarrizko fluxua:

1. Bezeroak data aukeratzen du.
2. Sistemak data horretako gertaerak erakusten ditu.
3. Bezeroak gertaera aukeratzen du.
4. Sistemak gertaera horrek dituen galderak erakusten ditu.
5. Bezeroak galdera aukeratzen du.
6. Sistemak galdera horrek dituen pronostikoak erakusten ditu.
7. Bezeroak pronostikoa aukeratzen du eta pronostiko horren gainean apostatu nahi duen diru kopurua betetzen du.
8. Sistemak apostu berria gordetzen du.

Fluxu alternatiboa:

1. Data horretan ez dago gertaerarik.
2. Dataren formatua ez da zuzena.
3. Apostatu nahi den dirua ez da betMinimora heltzen.
4. Gertaeraren data iraungi da. Ezin da aposturik egin.

### DIRUA SARTU:



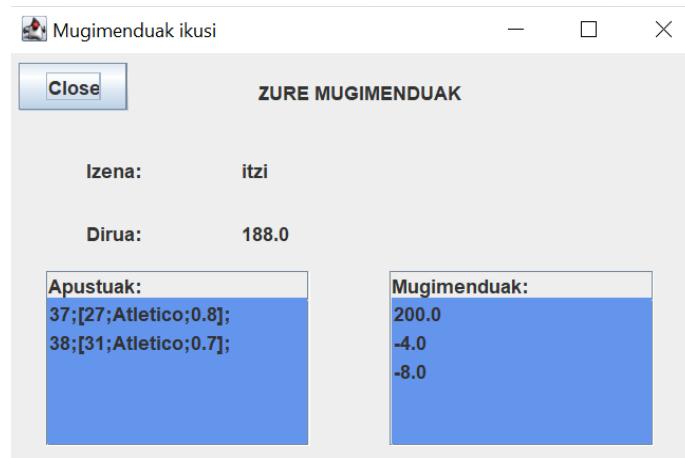
Oinarrizko fluxua:

1. Bezeroak sartu nahi duen diru kantitatea aukeratzen du.
2. Sistemak bezeroaren diru kantitatea eguneratzen du.

Fluxu alternatiboa:

1. Sartu nahi den diru kopurua ez da minimora iristen. Gutxienez 1.00 euro sartu beharko dira.

### MUGIMENDUAK IKUSI:



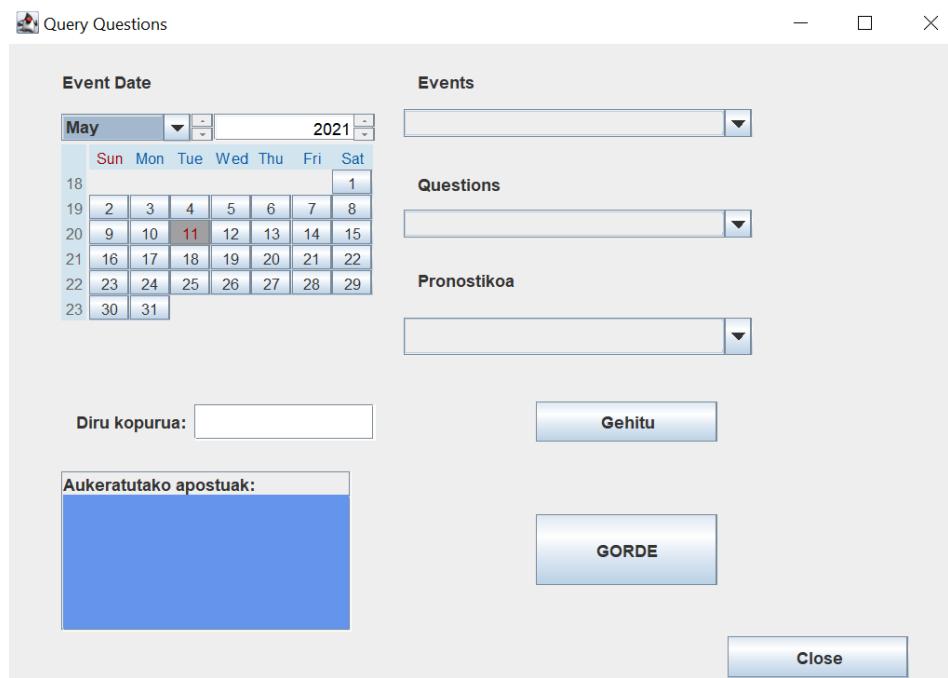
Oinarrizko fluxua:

1. Sistemak bezeroak egin dituen mugimenduak erakusten dizkio.

Fluxu alternatiboa:

1. Bezeroak ez du mugimendurik.

### APOSTU ANITZAK EGIN:



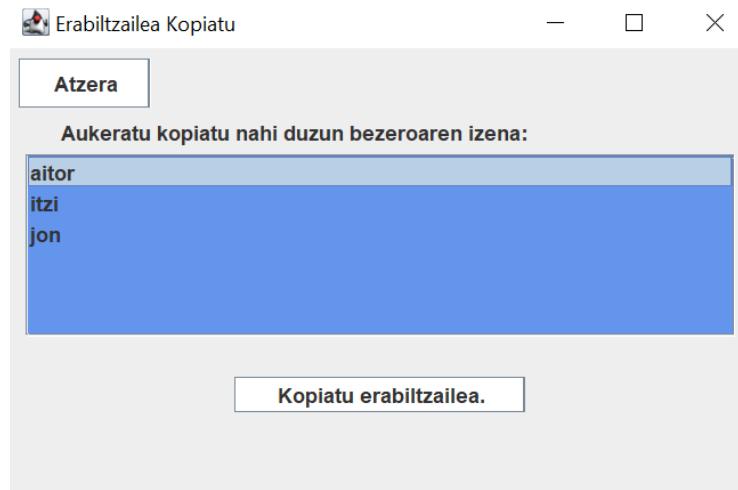
Oinarrizko fluxua:

1. Bezeroak data aukeratzen du.
2. Sistemak data horretako gertaerak erakusten ditu.
3. Bezeroak gertaera aukeratzen du.
4. Sistemak gertaera horrek dituen galderak erakusten ditu.
5. Bezeroak galdera aukeratzen du.
6. Sistemak galdera horrek dituen pronostikoak erakusten ditu.
7. Bezeroak pronostikoa aukeratzen du.
8. Bezeroak apostua zerrendara gehitzen du.
9. Bezeroak apostu gehiago gehitzen ditu zerrendara.
10. Bezeroak apostu anitz horren gainean apostatu nahi duen diru kopurua betetzen du.
11. Sistemak apostu anitza gordeko du.

Fluxu alternatiboa:

1. Data horretan ez dago gertaerarik.
2. Apostatu nahi den dirua ez da betMinimora heltzen.
3. Gertaeraren data iraungi da. Ezin da aposturik egin.
4. Apostatu nahi den dirua ez da betMinimora heltzen.

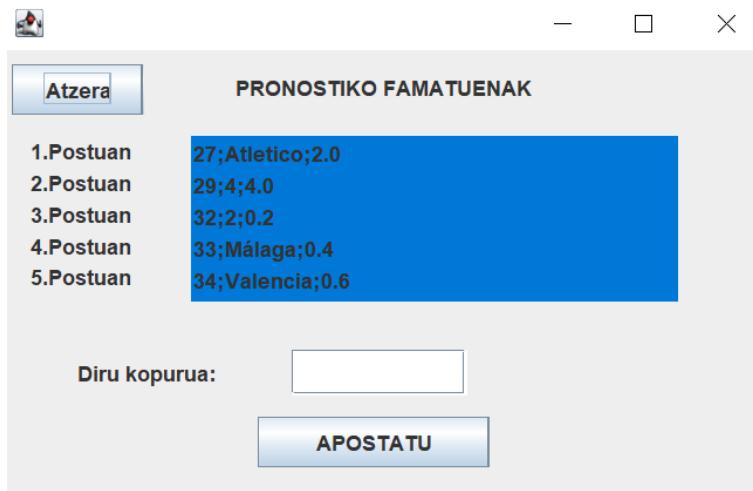
### **ERABILTZAILEA ERREPIKATU:**



Oinarrizko fluxua:

1. Sistemak erabiltzaileen zerrenda erakutsiko ditu.
2. Bezeroak erabiltzailea aukeratuko du.
3. Sistemak bezeroaren aukera gordetzen du.

### **APOSTU FAMATUENA EGIN:**



Oinarrizko fluxua:

1. Sistemak bezeroek gehien apostatu dituzten bost apostuak erakusten ditu.
2. Bezeroak apostuetako bat aukeratuko du.
3. Bezeroak apostatu nahi duen diru kopurua sartzen du.
4. Sistemak apostu berria gordetzen du.

Fluxu alternatiboa:

1. Apostatu nahi den dirua ez da betMinimora heltzen.
2. Gertaeraren data iraungi da. Ezin da aposturik egin.

#### **ERREGISTRATU:**

The screenshot shows a registration form titled "Register". The form includes fields for "Username", "Password", and "Repeat password", each with an empty input box. Below these are fields for "Birth date" (Year, Month, Day) and "Kreditu txartela", both with empty input boxes. A "Back" button is located in the top-left corner, and a "Register" button is in the bottom-right corner.

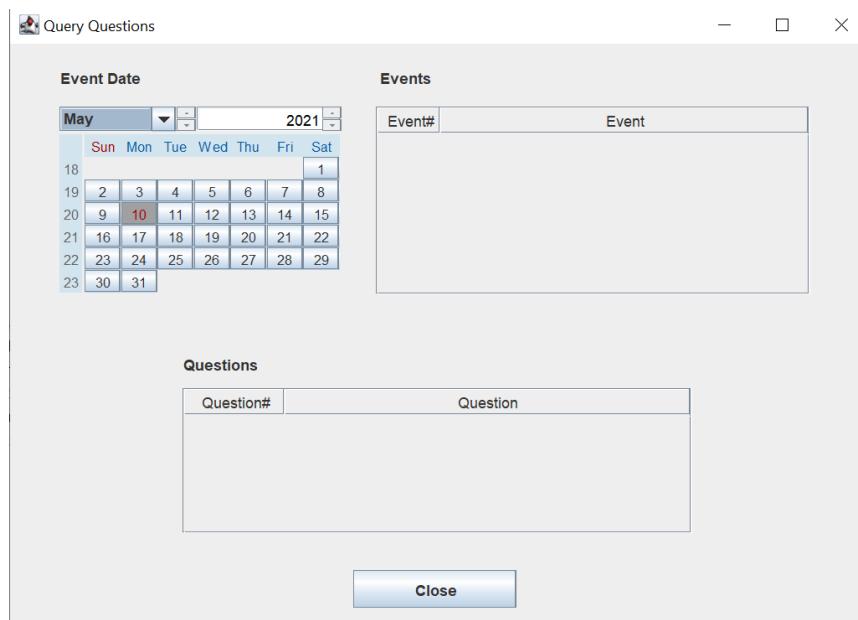
Oinarrizko fluxua:

1. Bezero ez-erregistratuak erabiltzailea, pasahitza, pasahitza errepikatu, jaiotze data eta kreditu txartela betetzen ditu.
2. Sistemak erabiltzailea, pasahitza, jaiotze data eta kreditu txartela gordetzen ditu.
3. Sistemak Login pantaila erakusten du.

Fluxu alternatiboa:

1. Erabiltzailea, pasahitza, pasahitza errepikatu, jaiotze data edota kreditu txartela hutsik daude. Erregistratu ezin da egin.
2. Erabiltzailea dagoeneko datu basean dago. Erregistratu ezin da egin.
3. Jaiotze data ez da legezkoa (+18 urte). Erregistratu ezin da egin.
4. Kreditu txartela ez da zuzena. Erregistratu ezin da egin.

## GALDERAK KONTSULTATU:



Oinarrizko fluxua:

1. Administratzailak data aukeratzen du.
2. Sistemak data horretako gertaerak erakusten ditu.
3. Administratzailak gertaera aukeratzen du.
4. Sistemak gertaera horretako galderak erakusten ditu.

Fluxu alternatiboa:

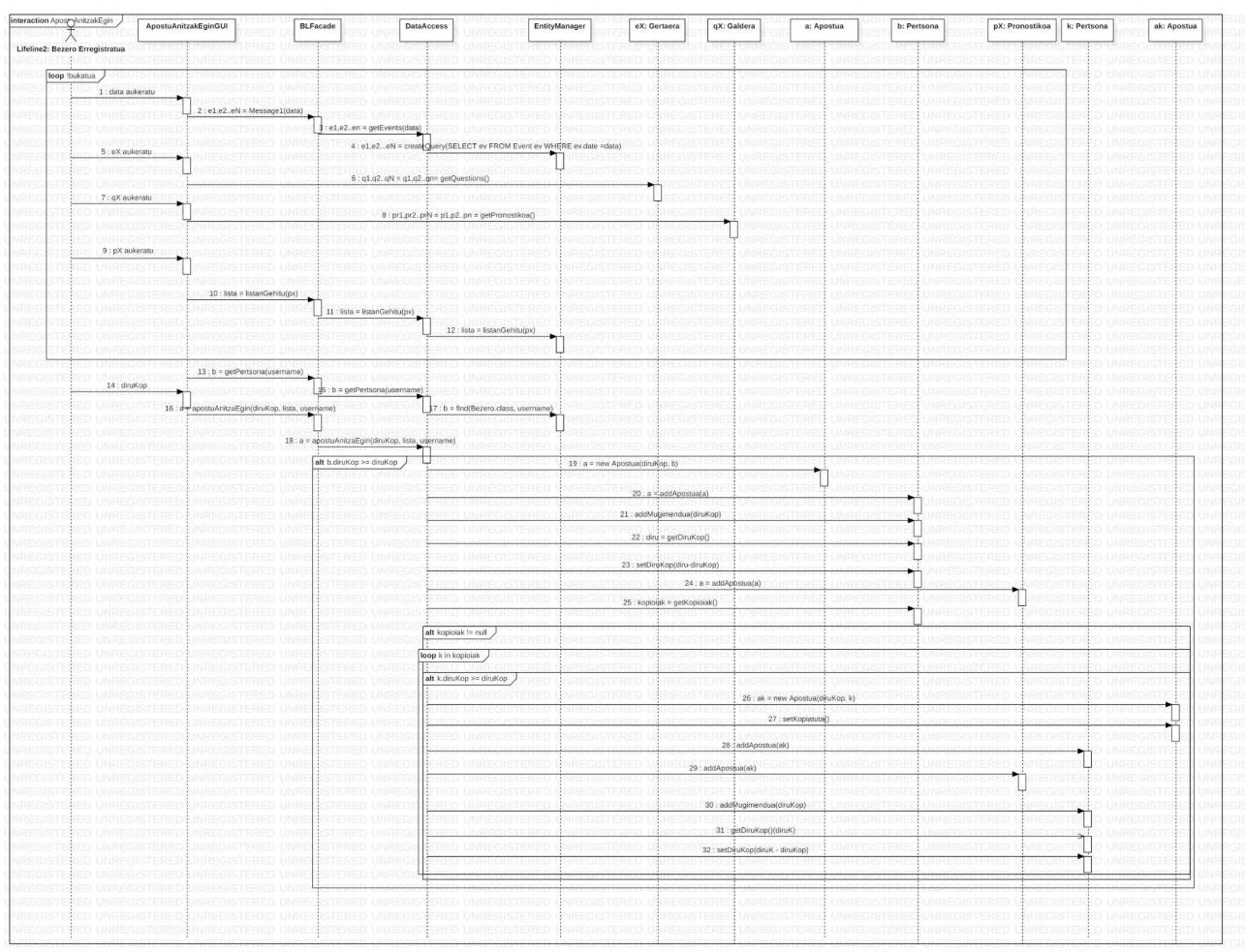
1. Data horretan ez daude gertaerak. Galderak ezin dira erakutsi.
2. Gertaera horretan ez daude galderak. Galderak ezin dira erakutsi.

## 4. Diseinua

### 4.1. SEKUENTZIA DIAGRAMAK

Hirugarren iterazioan garatutako sekuentzia diagramak:

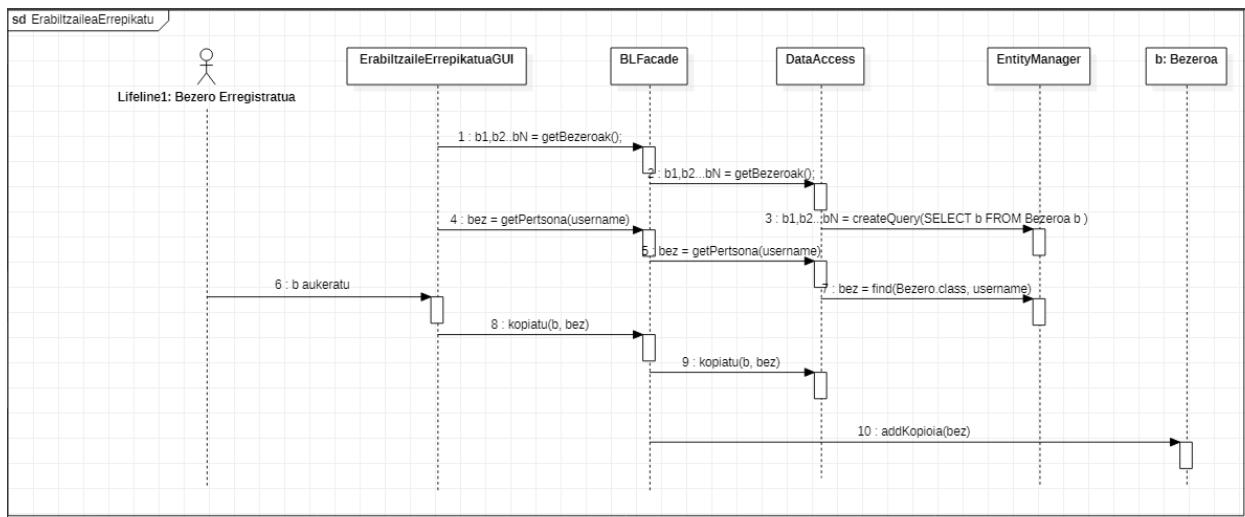
#### APOSTU ANITZAK EGIN:



Bezeroak data aukeratzen du, data horretako gertaera bat aukeratzen du, gertaera horrek dituen galdera bat aukeratzen du, eta azkenik, galdera horrek dituen pronostikoetako bat aukeratzen du. Eta ekintza hau nahi beste aldiz errepikatzen du, apostu bakoitza zerrendara gehituz.

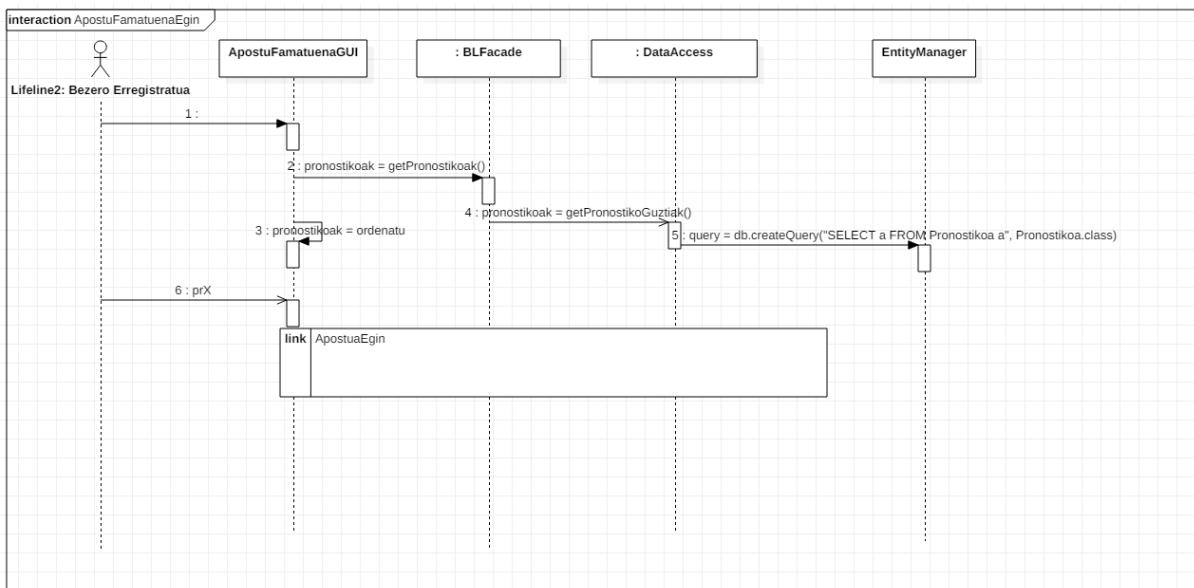
Apostu anitza egiteko, bezeroak apostatu nahi duen diru kopurua betetzen du eta apostu anitza egiten da. Horretarako, apostu berria sortzen eta gehitzen da eta erabiltzaileari apostuan gastatu duen dirua kentzen zaio. Azkenik, pronostikoan apostua gehitzen da.

## ERABILTZAILEA ERREPIKATU:



Sistemak erabiltzaileen zerrenda erakutsiko dio erabiltzailerari, horretarako datu-baseko gainerako erabiltzaile guztiak lortuz, administratzalea izan ezik. Ondoren, erabiltzaileak beste erabiltzaile bat aukeratuko du eta aukeratu duen erabiltzailearen kopioien zerrendan gehituko da.

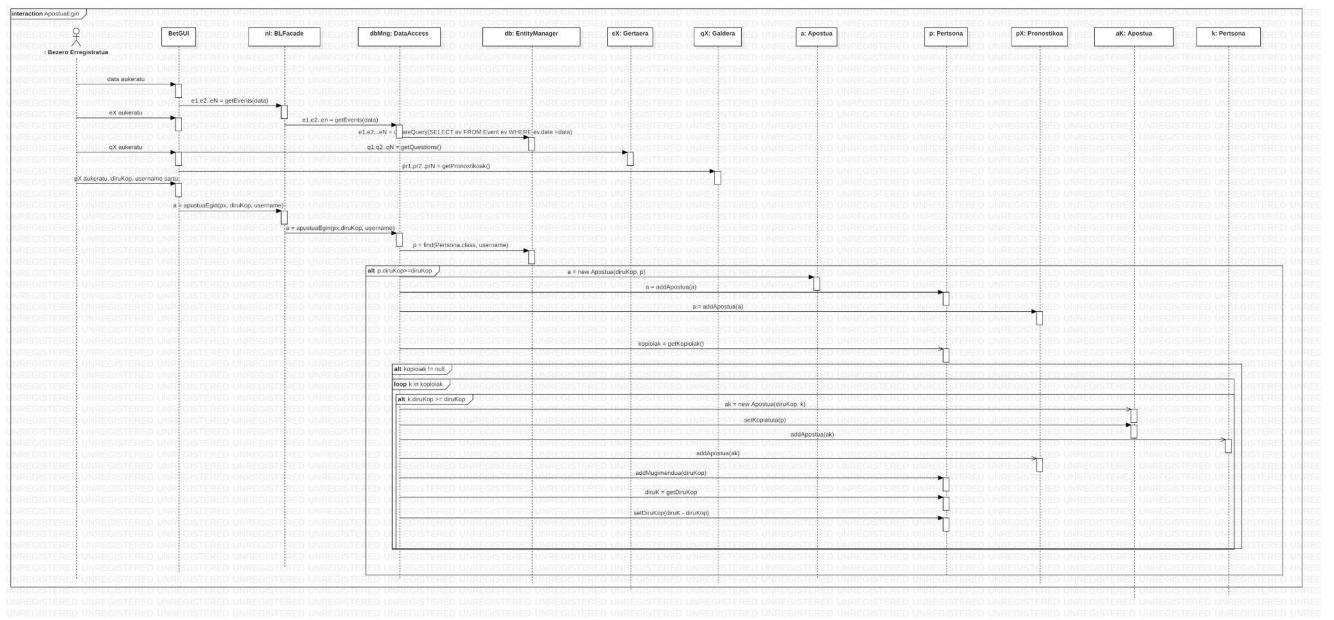
## APOSTU FAMATUENA EGIN:



Sistemak bezeroek gehien apostatu dituzten bost apostuak erakusten ditu eta bezeroak hauetako bat eta apostatu nahi duen diru kopurua sartzen ditu. Ondoren, sistemak apostu berria gordetzeko, ApostuaEgin erabilpen kasuaren sekuentzia diagrama jarraitzen du.

Hirugarren iterazioan eguneratutako sekuentzia diagramak:

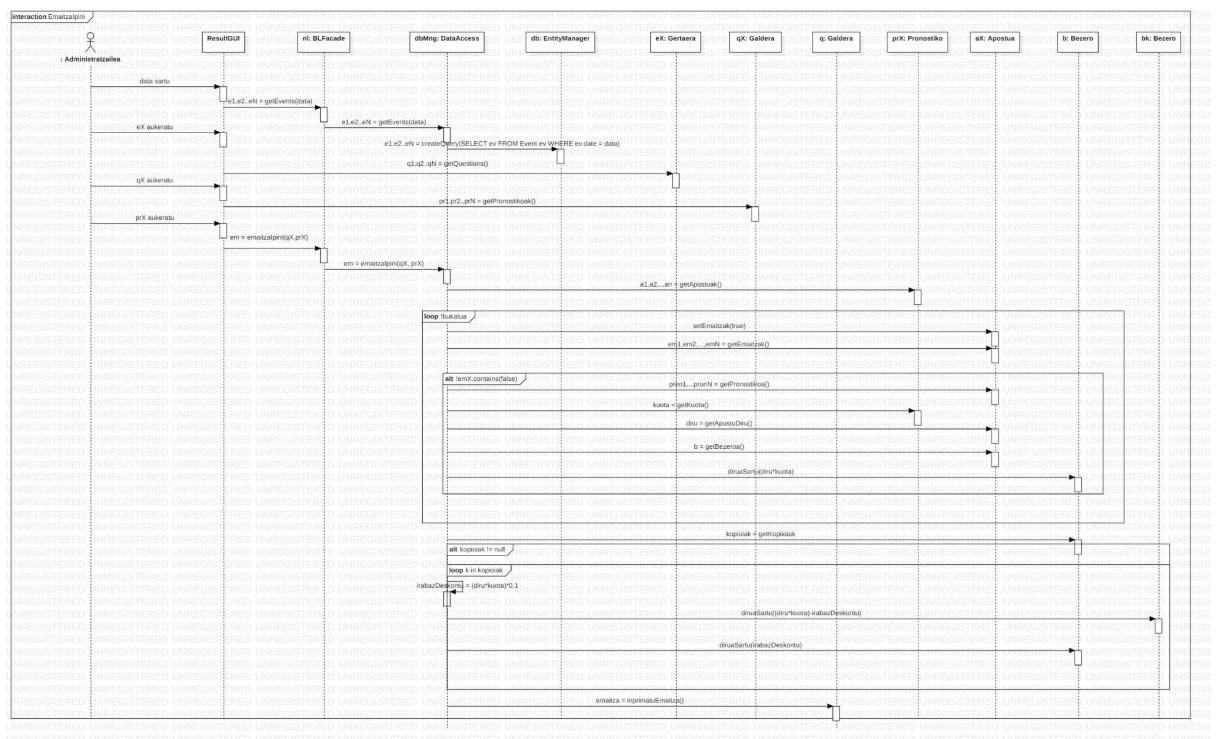
### APOSTUA EGIN:



Iterazio honetan erabiltzailea kopiatu implementatu dugunez, apostua egin erabilpen kasua eguneratu dugu. Orain, *Bezeroek* kopioien lista bat dutenez, lista horren barruan dauden bezeroei apostu bera gehitu behar zaie, baldin eta apostatu nahi den diru kopurua duten.

Kopioiei gehitutako apostuak balio boolean bat izango dute kopiatutako apostua den ala ez jakiteko eta String bat kopiatzen duen erabiltzailearekin. Honekin, emaitza ipintzerakoan jakingo dugu nori eman behar diogun irabaziaren %10-a.

## EMAITZA IPINI:



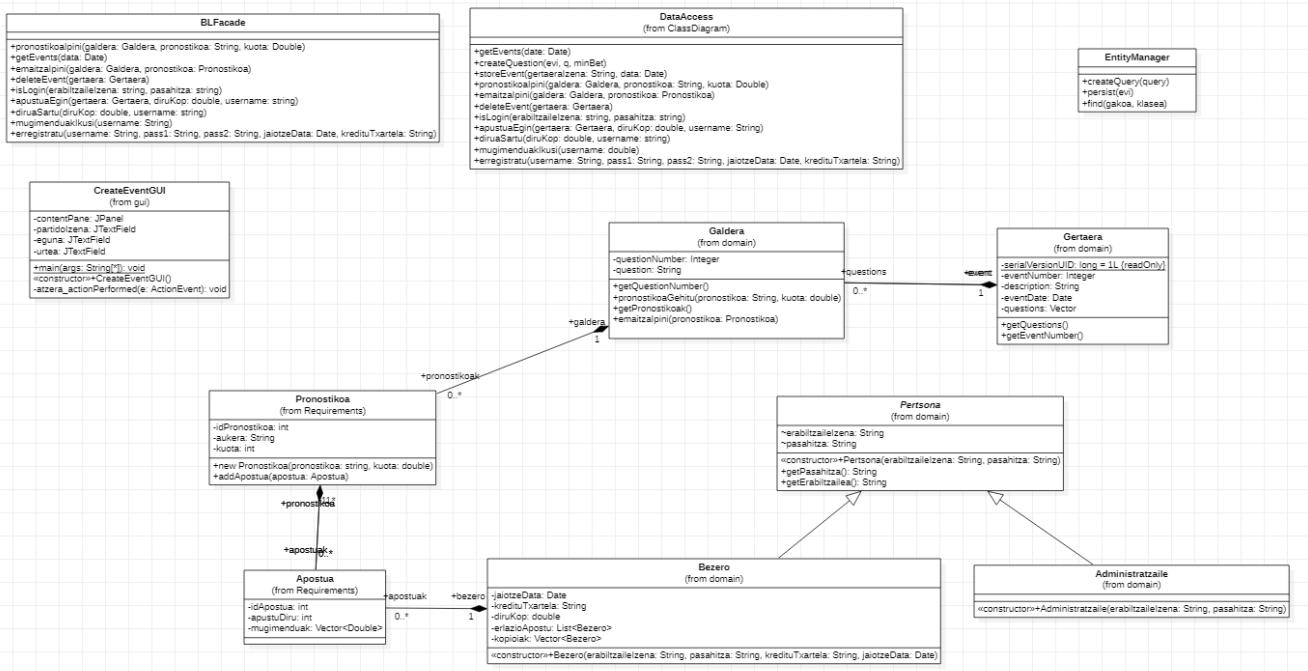
Apustu anitzak erabilpen kasua gehitu dugunez, emaitzak ipini egunerau dugu. Orain, emaitzak ipintzerakoan, begiratu behar dugu ea egindako apustuaren emaitzak adierazten dituen bektorean elementu guztiak *true* diren dirua gehitu baino lehen.

Aldi berean, apostu anitzak irabazterakoan pronostiko guztien kuotak gehitu behar dira eta ondoren apostatutako diruari biderkatu.

Erabiltzailea kopiati erabilpen kasua gehitu dugu ere, beraz, apustua irabazterakoan irabazitako diruaren %10-a kendu beharko zaio apustu hori kopiaturako apustua bada eta %10 apustu hori kopiati dion bezeroari gehitu behar zaio.

## 4.2. KLASE DIAGRAMA

Guk, iterazio honetan, ez dugu klase berririk sortu, soilik lehen zeudenei aldaketa batzuk egin dizkiogu.



## 5. Implementazioa

- **public Pertsona isLogin(String username, String pasahitza);**

*isLogin* funtzioak, *Pertsona* baten logina egiten du, datuak ziurtatuz. Horretarako, datu-basean begiratzen du ea erabiltzailea existitzen den. Horrela bada, erabiltzailea itzultzen du, eta bestela, null itzultzen du.

- **public boolean storeGuest(String username, String pass1, String pass2, Date jaiotzeData, String kredituTxartela);**

*storeGuest* funtzioak, *Bezero* bat erregistratzen du. Horretarako, bezero berri bat sortzen du erabiltzaile izen, pasahitz, kreditu txartel zenbaki eta jaiotze datarekin. Erregistratu nahi den erabiltzailea existitzen ez den ziurtatzen du, eta horrela bada, datu-basean gordetzen du. Gainera, erabiltzaile batek baldintza batzuk betar behar ditu, esaterako, 18 urte baino gehiago izan behar ditu, kreditu txartelaren zenbakiak 16 digitu izan behar ditu eta pasahitza bi aldiz idaztean, bi pasahitzak berdinak izan behar dira. Funtzio honek, errore mezua pantailaratuko du hauetako baldintzaren bat betetzen ez bada. Azkenik, dena ondo joan bada *true* itzuliko du, eta bestela, *false* itzuliko du.

- **public boolean existEvent(Date data, Event gertaera);**

Gertaera bat data berdinean existitzen den ala ez aztertzen du funtzio honek.

- **public boolean storeEvent(Event gertaera);**

Gertaera bat sortzen du, eta datu-basean gordetzen du. Horretarako, gertaera dagoeneko existitzen den begiratzen du, eta datu-basean ez badago gorde egiten du. Dagoeneko existitzen bada, mezu bat pantailaratzen du hau abisatuz.

- **public List<Question> getQuestions(Event gertaera);**

Gertaera jakin bat emanda, honek dituen galderen zerrenda itzultzen du.

- **public Pronostikoa pronostikoalpini(Question galdera, Pronostikoa pronostikoa);**

Galdera bati pronostiko berri bat gehituko zaio.

- **public boolean emaitzalpini(Question galdera, Pronostikoa pronostikoa);**

*emaitzalpini* funtzioak apostu baten emaitza jartzen du. Horretarako, galdera horretan dauden pronostiko jakin bateko egindako apustuei emaitza ipintzen zaio.

Ondoren, irabazitako apustua bada apustu hori egin duen erabiltzaileari dirua gehituko zaio. Kopiatutako apustua bada, irabazitako %10-a kopiatu dion erabiltzaileari gehituko zaio.

- **public Event deleteEvent(Event gertaera);**

Gertaera bat emanda, hau datu-basetik ezabatzen du. Gainera, ezabatutako gertaera itzultzen du.

- **public Boolean apustuaEgin(Pronostikoa pronostikoa, double diruKop, String username);**

Bezeroak aukeratutako pronostikoa eta diru kopuruarekin apustu berria sortuko da. Bezero honek kopioiak baditu apustua kopiatuko da bezero horietan.

Pronostikoak daukan apustu bektorean ere gordeko dira, emaitza ipintzerakoan irabazitako apustu bezala ipintzeko.

- **public boolean diruaSartu(double diruKop, String username);**

Diru kopuru bat eta erabiltzaile izen bat emanda, erabiltzailea bilatzen du datu-basean eta dirua sartzen dio. Gainera, sartu daitekeen diru kopuru minimoa 1 euro denez, hau horrela dela konprobatzeko. Dena ondo joan bada *true* itzuliko du, eta *false* bestela.

- **public String getUser();**

Uneko Bezeroaren erabiltzaile izena itzultzen du.

- **public void setUser(String user);**

Uneko bezeroaren erabiltzaile izena gordetzen da aldagai batean.

- **public List<Pronostikoa> getPronostikoak (Question galdera);**

Galdera bat emanda, hau datu-basean aurkitzen du eta galdera honek dituen pronostiko guztiak zerrenda baten itzultzen ditu.

- **public Bezero getPertsona(String username);**

Erabiltzaile izen bat emanda, datu-basean aurkitzen du eta *Bezeroa* itzultzen du.

- **public Question deleteQuestion(Question galdera);**

Galdera bat emanda, hau datu-basetik ezabatzen du. Gainera, ezabatutako galdera itzultzen du.

- **public Boolean apostuAnitzaEgin(List<Pronostikoa> pronostikoak, double diruKop, String username);**

Bezeroak auketatutako hainbat pronostikoak *pronostikoak* bektorera gehituko dira. Bektore hau eta apostatuko den diru kopuruarekin apostu berria sortzen da. Bezero honek kopioiak baditu apustua kopiaturiko da bezero horietan.

Pronostikoak daukan apustu bektorean ere gordeko dira, emaitza ipintzerakoan irabazitako apustu bezala ipintzeko.

- **public List<Pertsona> getBezeroak();**

Datu-basean dauden *Bezero* guztiak itzultzen ditu, zerrenda batean.

- **public void kopiari(Bezero b1, Bezero b2);**

Sarrerako lehenengo bezeroa kopiari nahi den erabiltzailea izango da eta bigarrena kopiari nahi duena. Beraz, bi erabiltzaile hauek datu-basean bilatzen dira eta kopiari nahi den erabiltzaileari, kopioien zerrendan kopiari nahi duena gehitzen zaio.

- **public List<Pronostikoa> getPronostikoGuztiak();**

Datu-basean dauden *Pronostiko* guztiak itzultzen ditu, zerrenda batean.

- **public List<Double> getMugimenduak(String user);**

Erabiltzaile izen bat emanda, bezeroa datu-basean aurkitzen du eta bezero honen mugimenduak lortzen ditu. Azkenik, mugimenduak itzultzen ditu. Mugimendu hauek, erabiltzaileak duen diru kopuruak izango dituen aldaketak gordetzeko dira.

## 6. Ondorioak

- Gure taldeari dagokionez, lan taldearen dinamika oso ona izan dela uste dugu, ondo moldatu gara eta ez dugu arazorik izan.
- Proiectua ez zaigu oso konplexua iruditu. Mugarik ez genuenez, talde bakoitzak nahi beste zaildu zezakeen erabilpen kasu bakoitza.
- Lortutako emaitzak onak izan direla iruditzen zaigu.
- Proiectuetan Oinarritutako Irakaskuntza bitartez gehiago ikasten dela uste dugu. Dena praktikoa denez, gure kabuz ikasi beharrak autonomia eman digu.

## 7. Bideoaren URL-a

[BIDEOA](#)

## 8. Kodearen URL-a

[Bets21 \(github.com\)](#)