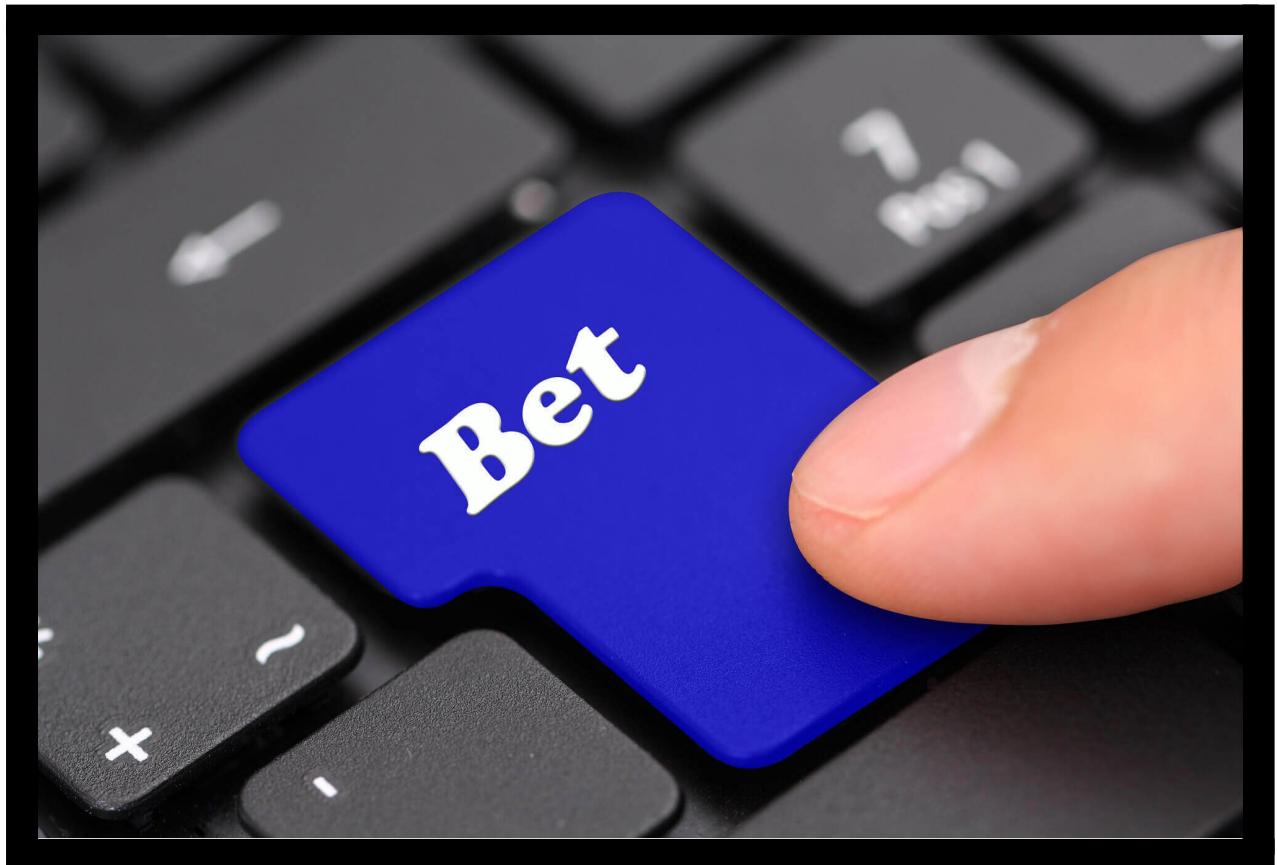


# BETS21



Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

## **Aurkibidea**

<b>Aurkibidea</b>	<b>2</b>
<b>Web Zerbitzuak</b>	<b>2</b>
<b>Internalizazioa</b>	<b>2</b>
<b>Eskakizun bilketa</b>	<b>6</b>
Domeinuaren eredua	6
Erabilpen kasuen eredua	8
GUILak	10
<b>Jarraitu GUI</b>	<b>17</b>
<b>Diseinua</b>	<b>29</b>
Sekuentzia diagramak	29
Klase diagrama	36
<b>Implementazioa</b>	<b>39</b>
<b>Ondorioak</b>	<b>46</b>
<b>Bideoa</b>	<b>47</b>

## **Web Zerbitzuak**

Ez dugu implementatu.

## **Internalizazioa**

Bai, implementatu dugu.

## **Sarrera**

Proiektu honetan helburua apustu sistema bat garatzea da, iteraziotan, hauetako bakoitzean proiektua osatuz.

Sisteman hiru erabiltzaile mota egongo dira, erregistratu gabea, erregistratua, eta azkenik, administratzailea. Hauetako bakoitzak gauza desberdinak egin ahal izango ditu sistemaren barruan.

Sistema garatzeko unean jadanik garatutako 0.iterazio bat izan da abiapuntua, funtzionalidate batzuk jadanik implementatuta zituena:

- Galderak ikusi: edozein erabiltzaile motari galderak ikusteko aukera ematen dio.
- Galderak sortu: administratzaileak galderak sortu ahal ditu hainbat parametroren bidez (galderaren testua, eta apostatzeko diru kopuru minimoa).

Abiapuntu horretatik, lehenengo iterazioan ondoren erabilpen kasuak garatu ditugu:

- Login: Erabiltzaile bati bere kontura sartzeko aukera ematen dio. Honetarako bere erabiltzaile izena eta pasahitza sartu behar ditu. Login egin ostean sistemak erabakiko du erabiltzaile hori administratzaile bat edo erabiltzaile arrunt bat det.
- Erregistratu: erregistratu gabe dagoen erabiltzaile bati kontu bat sortzeko aukera ematen dio, horretarako beharrezkoak diren parametro guztiak sartuz sistemara (username, izena, abizena, adina...).
- Gertaerak sortu: administratzaile batek gertaerak sortu ahal izango ditu gertaera zein den eta data adieraziz.
- Kuotak sortu: sistemako administratzaileak galdera desberdinatan kuotak sortu ahal izango ditu.

Bigarren iterazioan, berriz, ondorengo funtzionalitateak gehitu zaizkio sistemari:

- Dirua sartu: erabiltzaile erregistratuak (soilik) dirua sartu ahal du bere kontura.
- Diru mugimenduak ikusi: gertaera desberdinaren ondorioz gertatutako diru mugimendu desberdinak sistemak erregistratuko ditu erabiltzaile erregistratuaren kontuan. Apostu bat egitean apostatutako dirua, apostua irabaztean irabazitako...
- Apostatu: erabiltzaile erregistratuak apostatzeko aukera izango du. Honetarako, apostatu nahi duen diru kopurua eta ze kuotatan apostatu nahi duen adierazi beharko du.

- 
- **Apostua ezabatu**: Erabiltzaile erregistratuak bere apostuak ezabatzeko aukera izango du.
  - **Gertaera ezabatu**: Sistemako administratzaileak gertaerak ezabatu ahal izango ditu. Gertaera ezabatzean, azken honek barneratzen dituen galdera, kuota eta apustuak ere ezabatuko dira, honek eragingo dituen diru mugimendu guztiak burutuz. apostu anitz bat badago, adibidez 3 kuotakoa, non 2 asmatu diren, ezabatzen den gertaeran badago 3. kuota, apostua irabazitzat emango da lehenengo bi kuotekin eta erabiltzaileari dagokion dirua ordainduko zaio.
  - **Emaitzak ipini**: sistemako administratzaileak jadanik pasa diren gertaeretako galderetako emaitzak ezarri ahal izango ditu. probak errazago egiteko asmoarekin oraindik pasa ez diren gertaeretako galderetan ere erantzunak ipini daitezke.

Amaitzeko, hirugarren iterazioan ondorengoak garatu ditugu:

- **Apostu anitza sortu**: erabiltzaile erregistratuak apostu anitzak ere egin ahal izango ditu. apostu arruntak kuota bat bakarrik dutenak dira, apostu anitzak kuota bat baino gehiago dituztenak. Apostu anitz bat irabazteko beharrezkoa da guztiak asmatzea.
- **Mezuak bidali**: Sistemako administratzaileak eta erabiltzaile erregistratuak mezuak bidali ahal izango dituzte. Mezuak txat pribatuetan edo taldeetan bidali ahal dira.
- **Gertaera bikoiztu**: Sisteako administratzaileak gertaera bat kopiatu ahal izango du beste egun batean. Gertaerarekin batera bere galdera eta kuota guztiak kopiatuko dira.
- **Jarraitu erabiltzailea**: Erabiltzaile (erregistratuak soilik) batek beste bat (edo gehiago) jarraitzeko aukera izango du. Honekin, jarraitutako erabiltzaileak egindako apostu guztiak modu automatikoan egingo dira, beti ere sartutako parametroek ahalbidetzen badute. Erabiltzaile bat jarraitzeko unean, zenbat diru bideratu nahi den adierazi behar da honela diru hori erreserbatzeko. Bestalde, jarraitutako erabiltzaileak apostatutakoaren ze ehuneko apostatu nahi den ere adierazi ahal izango du erabiltzaileak.  
Edozein momentutan izango du jarraitzeari usteko aukera.
- **Sortu taldea**: Sistemako administratzaileek eta erabiltzaile erregistratuek chat taldeak sortzeko aukera izango dute. Taldea sortzean, taldearen izena eta partaideak adierazi behar dira. Taldea sortzen duen pertsona taldeko adminisitratzailea izango da.

## Software ingeniaritza

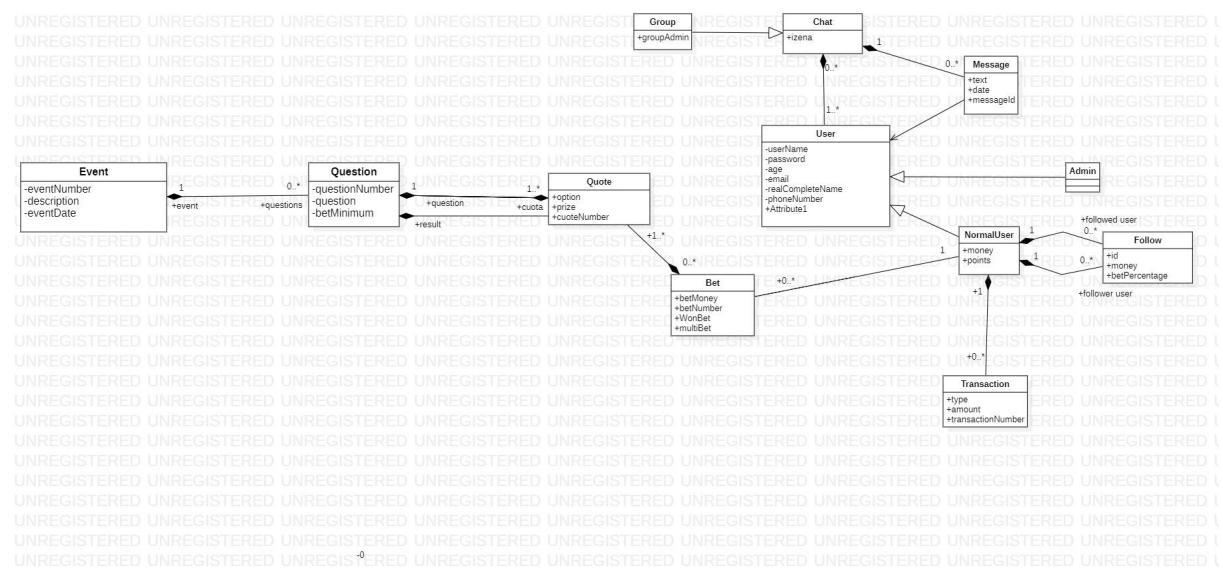
Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

- 
- Sartu erabiltzailea taldera: Taldeko administratzaileak beste edozein erabiltzaile (edo administratzaile) taldean sartu ahal izango du.
  - Bota erabiltzailea taldetik: Taldeko administratzaileak edozein taldekide taldetik bota dezake.
  - Utzi taldea: Talde bateko partaideek taldea utzi dezakete. Taldea usten duena taldeko administratzailea bada, taldekide listan lehenengo posizioan dagoena izango da administratzaile berria.
  - Ezabatu taldea: Taldeko administratzaileak taldea ezabatu dezake.

## Eskakizun bilketa

### Domeinuaren eredu

Gure proiektuaren garapenean zehar, 0.iteraziotik jasotako domeinu ereduaren diseinuak zenbait txertaketa zein aldaketa nabarmen jasan behar izan ditu. Hori horrela, hasiera batean gure apostu sistemaren garapenerako abiapuntu modura jaso genuen diseinuak honela amaitu du:



Lehenik eta behin, domeinu eredu honetan gertaeren(**Event**) klasea dugu, gure apostu sistemaren oinarri dena. Klase honek gertaeraren deskripzioa eta haren data jasotzen ditu, galderen(**Question**) klasearekin duen erlazioaz gain, zeinetan, gertaera batek galdera bat baino gehiago eduki ditzazkeela eta galdera bat gertaera jakin bat dagokiola definitzen den.

“**Question**” klasearekin jarraituz, ikus genezake gertaerari buruz eginiko galdera eta honetan apostatu ahal izateko onartuko den diru kopuru minimoa direla bere atributuak. Honetaz gain, kuoten(**Quote**) klasearekin bi erlazio zuzen mantentzen ditu. Batean, galdera batek kuota bat baino gehiago edukiko dituela eta kuota bat galdera jakin bat dagokiola zehazten da. Bestean, aldiz, galderen klasean kuoten klaseko atributu bat ezartzen da, gertaeraren inguruko galdera jakin baten emaitza ezartzeko balioko duena. Galdera erantzutakoan eginiko aukeraketa eta aukeraketa horren arabera irabazi daitekeen diru kopurua izango dira aipatutako “**Quote**” klase horren ezaugarriak.

Behin gertaerak, galderak eta hauen emaitzen kuotak azaldu ditugula, ezinbestekoa da apostu(**Bet**) klase bat sortzea erabiltzaileek apostuak egin ahal ditzaten. Klase hau, batetik kuoten klasearekin, eta bestetik erabiltzaile arrunten(**NormalUser**) klasearekin erlazionatzen da, izan ere, apostu jakin bat kuota batia ala gehiagorri eta erabiltzaile jakin bat egokitu beharko zaio. Hori horrela, kuota jakin batean egin diren apustuen zerrenda jaso beharko du “**Quote**” klaseak, apustu ezberdinatan parte hartu duen erabiltzaile arrunt batek egingo duen

## Software ingeniaritza

Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

---

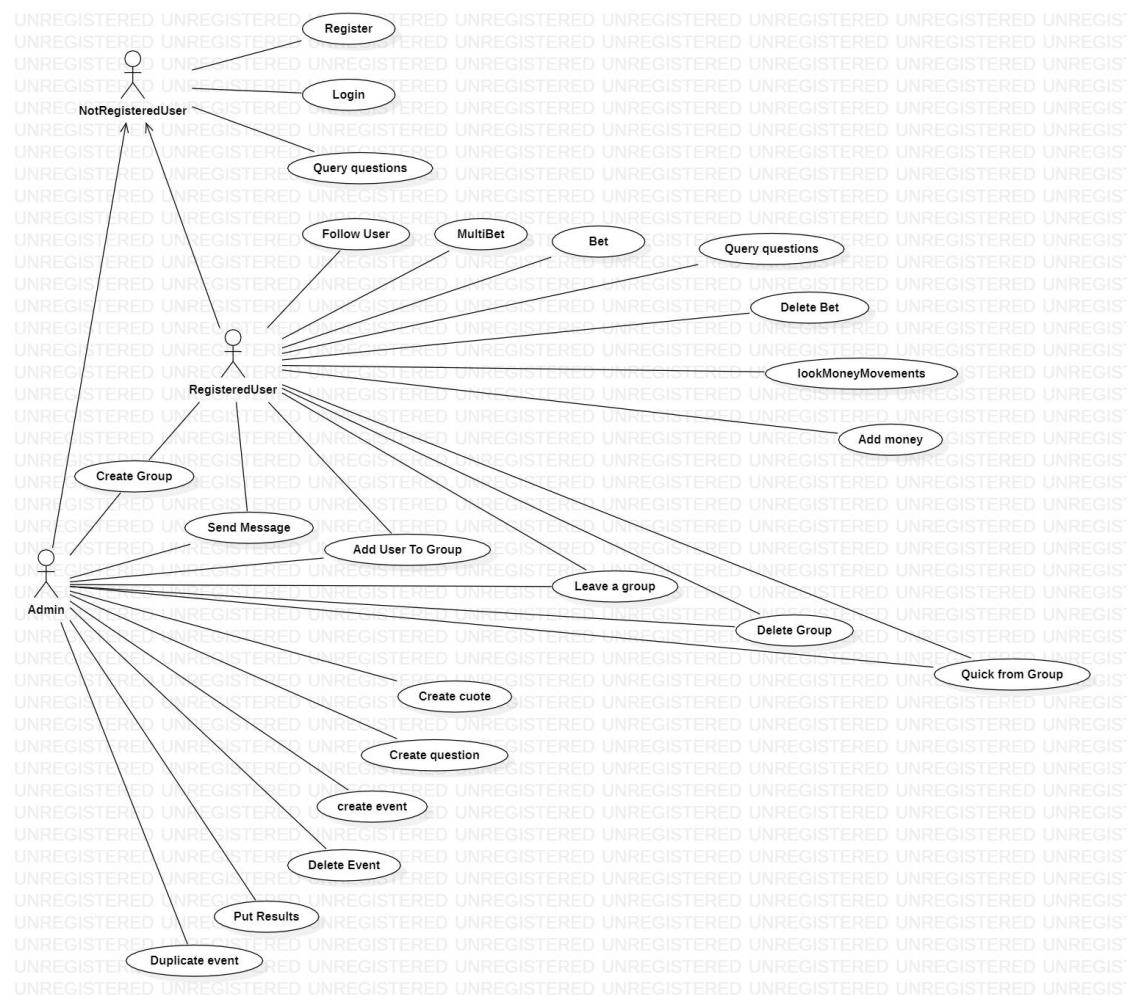
antzera. Erlazio hauetaz gain, “**Bet**” klaseak erabiltzaileak apostatutako dirua, egindako apostua kuota anitzekoa ala bakarreko den eta ea egindako iragarpena asmatu duen izango ditu atributu.

Esan bezala, erabiltzaile arruntak(**NormalUser**) izango dira apustu sistema honetan apostatu ahal izango dutenak. Horretarako, ezinbesteko dugu transakzioak(**Transaction**) bideratzeko klase bat txertatzea, diru mugimenduak egiteko baita hauen jarraipenean aritzeko ere. Honen arabera, erabiltzaile arruntek egindako transakzioen lista bat edukiko dute, zeinetan transakzio mota eta mugitutako dirua azalduko den. Bestalde, erabiltzaileak elkarren artean jarraitzeko funtzionalitatea ere implementatu behar izan dugunez, beharrezkoa ikusi dugu jarraiketak(**Follow**) egin ahal izateko ere klase bat atxikitzea, zeinak bi erlazio edukiko dituen “**NormalUser**” klasearekin. Bi erlazioei esker, erabiltzaile arrunt batek, egindako jarraiketen zerrenda bat eta haien jarraitzaleak jasotzen dituen beste zerrenda bat edukiko ditu. Jarraiketa(**Follow**) horietan, nork zein jarraitu duen ageriko da, jarraitzerakoan sartutako diru kopuru eta portzentai jakin bat ere aukeratuz, jarraitzaleek parametro horien arabera jarraitzen duten erabiltzailearen apostuak automatikoki egin ditzaten.

Erabiltzaile motei dagokienez, erregistratu diren erabiltzaile arruntak eta apostu sistemako administratzaileak bereiztearren, “**User**” superklasea sortu dugu, bertan definituriko ezaugarri orokorrak(datu pertsonalak, erabiltzaile izena, pasahitza...) erabiltzaile arruntek(**NormalUser**) eta administratzaileek(**Admin**) herentzia bidez jasoz. Ikusitako guztiaz gain, erabiltzaile arrunten egindako iragarpenekin lortutako emaitzen araberako puntuazio bat egokituko zaile eta gainera, une horretan sisteman sartua duten dirua ikusteko aukera ere izango dute.

Amaitzeko, eratu dugun domeinu ereduaren azken erlazioak, erabiltzaileak, implementatutako txateatzeko sistemarekin mantentzen dituenak dira. Erlazio honen arabera, erabiltzaile batek dagozkion txatak edukiko ditu zerrenda batean, eta txatek(**Chat**), aldiz, taldeko partaideen zerrenda bat gordeko dute (taldea bada 2 baino gehiago), taldera bidalitako mezuen zerrenda bat eta taldeko administratzailea den erabiltzailea izatearekin batera.

## Erabilpen kasuen eredu



Atxikitutako diagrama erreparatuz, ikus genezake garatu dugun apustu sistemaren erabilpen kasuak hiru erabiltzaile mota ezberdinan banatzen direla: erregistratu gabeko erabiltzailea(**NotRegisteredUser**), erregistratutako erabiltzailea(**RegisteredUser**) eta administratzailea(**Admin**). Azken biek, erregistratu gabeko erabiltzailearen funtzionalitateak jasotzen dituzte herentzia bidez, funtzionalitate hauek, apustu sistemaren orrian erregistratzea(**Register**), bertan saioa hastea(**Login**) eta gertaeeren kontsultak egitea(**Query questions**) izanik.

Erregistratu gabeko erabiltzailearen ekintzezagin, erregistratutako erabiltzaileek eta administratzaileek zenbait erabilpen kasu partekatzen dituzte, hain zuzen ere, erabiltzaile ezberdinen arteko elkarkomunikaziora zuzendutako erabilpen kasuak. Horien artean ondorengoak ditugu: txateatzeko taldeak sortzeko aukera(**Create Group**) eta hauek ezabatzekoa(**Delete Group**), talde jakin batera mezuak bidali ahal izatea(**Send Message**), talde hori uztea(**Leave Group**), baita erabiltzaileak taldera gehitzeko(**Add User To Group**) edota taldetik botatzeko ahalmena(**Quick from Group**) ere, azken biak taldeko administratzaileak bakarrik gauzatu ditzazkelarik.

## Software ingeniaritza

Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

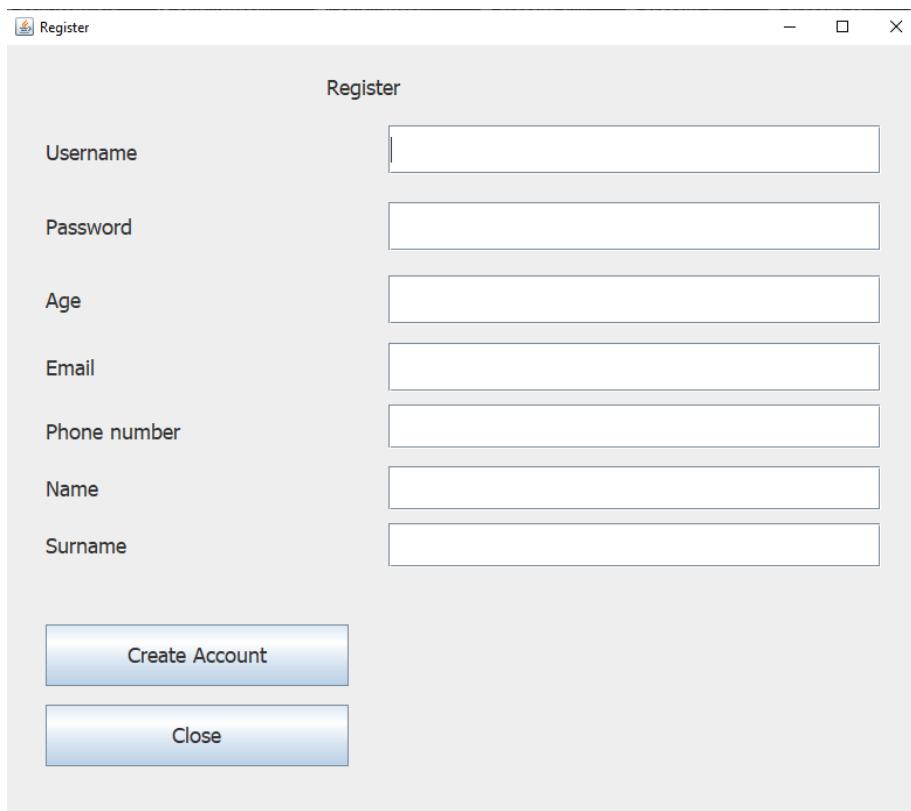
---

Azkenik, gainontzeko erabilpen kasu guztiak genituzke, zehatzago esanda, apustu sistemako erabiltzaile erregistratuei dagozkienak eta sistema honetako administratzaleek dituzten pribilegioak. Esandakoak esanda, erregistratutako erabiltzaileei dagokienez, hauen erabilpen posibleen artean ditugu, erabiltzaileak jarraitzea, jarraitutako erabiltzailearen apostuak automatikoki eginez(**Follow User**); kuota bakarreko(**Bet**) edo kuota anitzeko apostuak egitea(**MultiBet**); egindako apostuak ezabatzeko aukera, sistemak apostutako dirua itzuliz(**Delete Bet**). Hori guztiaz gain, diru mugimenduekin lotutako funtzionalitateak ere badituzte, zeinetan dirua gehitzeko(**AddMoney**) edota egindako transakzioen jarraipen zuzena(**lookMoneyMovements**) egiteko aukera duten erabiltzaile erregistratuek.

Bestalde, aurretik esan bezala, apustu sistema honetako administratzaleak ditugu, sistemaren betekizunen zuzentasuna bermatzearen ardura dutenak. Hori horrela, erabiltzaile mota honek, gertaerak sortzeko(**Create Event**), ezabatzeko(**Delete Event**) eta bikoitzeko(**Duplicate Event**) aukera izango du, kuotak ezartzeko(**Create Cuote**), galderak sortzeko(**Create question**) eta hauen emaitzak esleitzeko(**Put Results**) boterearekin batera.

## GULak

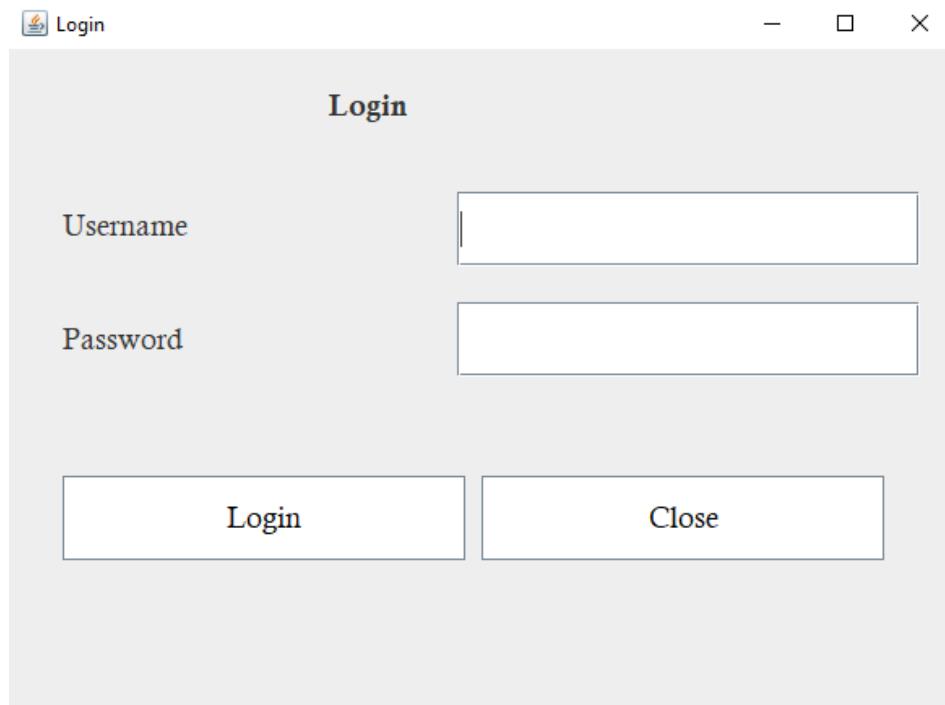
- Register GUI



```
# Flow of events
## Basic Flow
1. *System* shows a Interface where the user can login or select to register
2. *User* selects to register and *system* redirects *User* to another interface
3. *User* introduces the necessary data
4. *System* checks if the username is already used
5. *System* checks if the password has more than 4 characters.
6. *System* checks if the email is already used
7. *System* checks if the phone number is already used
8. *System* checks if the user has 18 or more years.
9. *System* creates a new NormalUser account and saves it.
10. *System* redirects *User* to the main page

## Alternative flow
1. *User* enters wrong the data
2. *System* sends a message showing what is wrong (used username, used email, used phone number,
   user has less than 18 years or the password isn't valid)
3. *User* enters data again
```

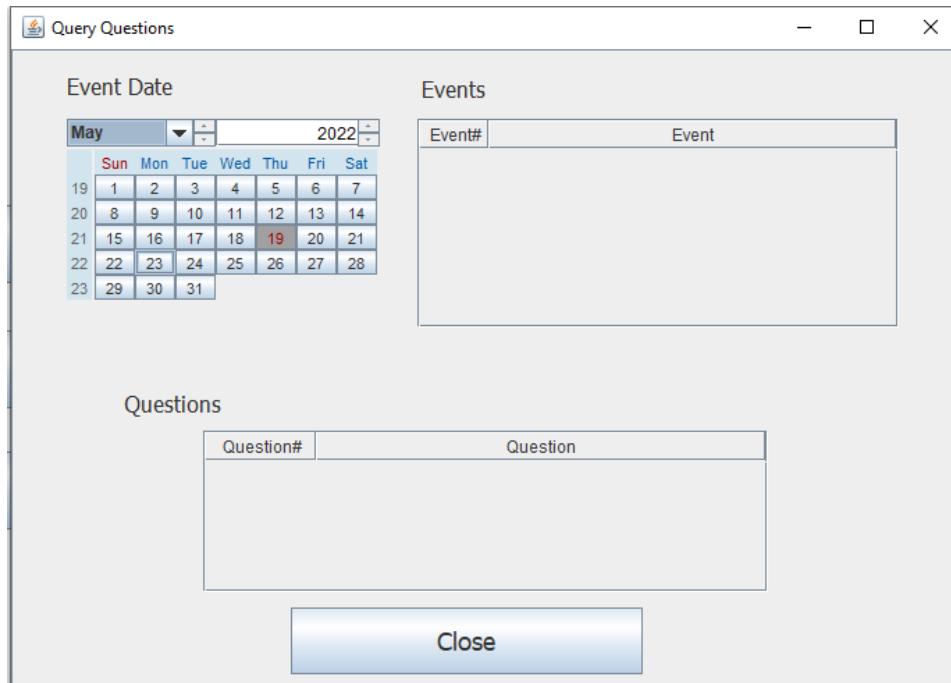
- Login GUI



```
# Flow of events
## Basic Flow
1. *System* shows a Interface where the user can login
2. *User* enters the neccesary data
3. *System* check that the account exists in the database
4. *System* checks that the password is correct
5. *System* redirects *User* to the main page

## Alternative flow
1. *User* enters wrong the data
2. *System* sends a message showing what is wrong
3. *User* enters data again
```

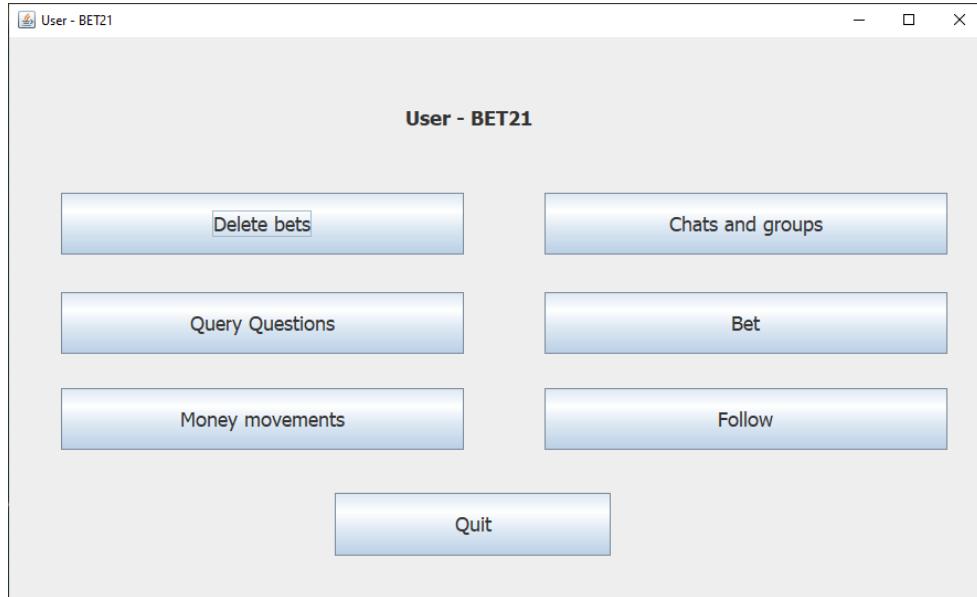
- Galderak ikusi GUI



```
# 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. *System* displays the **questions** associated to the selected **event**

## Alternative flow
1. There are no events on this date. Events cannot be shown.
2. There are no questions associated to the event. Questions cannot be shown.
```

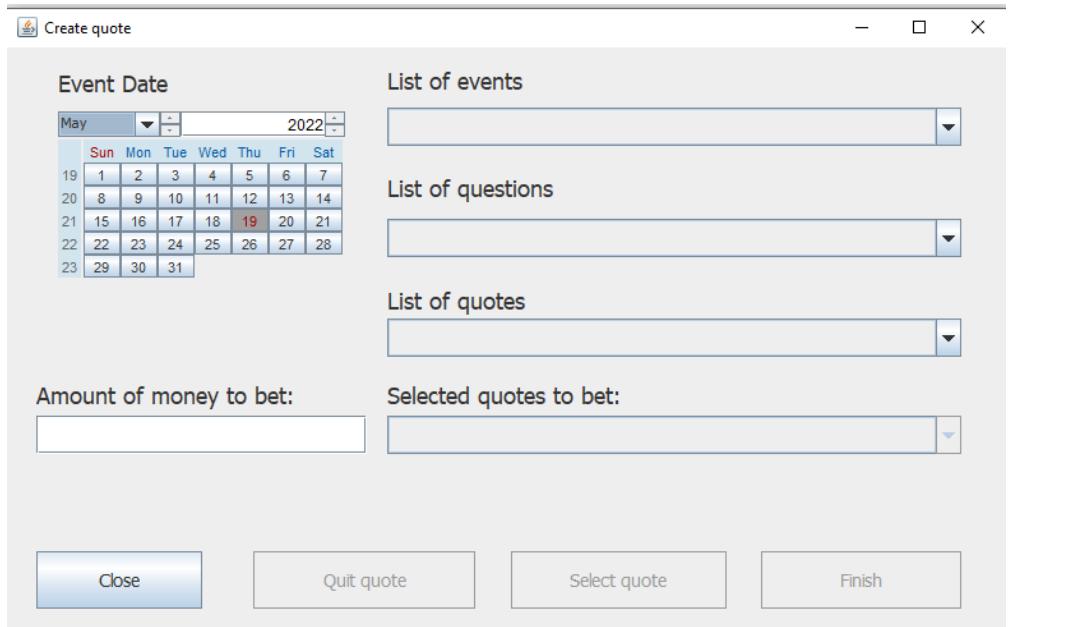
- Erabiltzaile arrunta GUI



# Software ingeniaritza

Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

- Apostatu (apostu arruntak/apostu anitzak) GUI



The screenshot shows a Windows-style application window titled "Create quote". On the left, there is a "Event Date" section with a calendar for May 2022. The date 19 is highlighted in red. To the right of the calendar are three dropdown menus labeled "List of events", "List of questions", and "List of quotes", each with a downward arrow icon. Below these dropdowns are two input fields: "Amount of money to bet:" and "Selected quotes to bet:", both with downward arrow icons. At the bottom of the window are four buttons: "Close", "Quit quote", "Select quote", and "Finish".

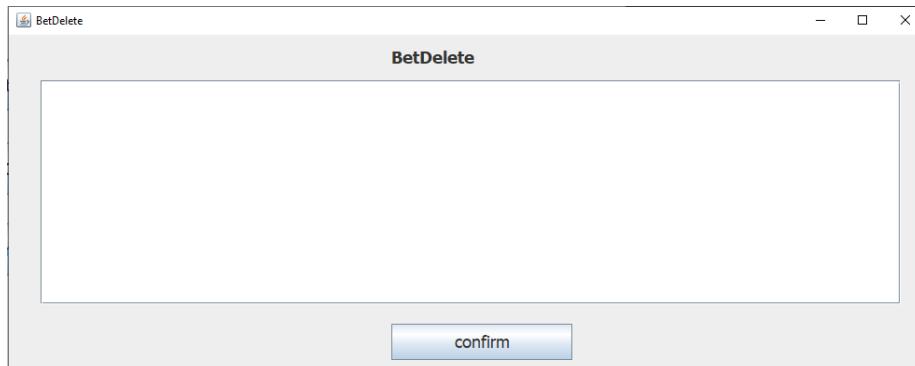
```
# Flow of events
## Basic Flow
1. *User* selects a day
2. *System* show the **Events** of that day.
3. *User* selects an event
4. *System* show the **Questions** of the selected **Event**
5. *System* shows the **Cuotes** of the selected **Question**
6. *User* selects one of that **Cuotes** and insert money amount to bet
7. *System* checks if the date has passed and that the money amount or cuote are !=null
8. *System* checks that the **User** didn't bet n that **cuote** before
9. *System* redirects *User* to another GUI
10. *System* creates the **Bet** and add it to the **User** betlist and the **Cuote** betList.
11. *System* saves the Bet

## Alternative flow
1. Date is expired
2. *User* enters more money than the amount he has.
3. *User* tries to bet in an bet that he has already bet.
```

```
# Flow of events
## Basic Flow
1. *User* selects a day
2. *System* show the **Events** of that day.
3. *User* selects an event
4. *System* show the **Questions** of the selected **Event**
5. *System* shows the **Cuotes** of the selected **Question**
6. *User* selects the **Cuotes** and insert money amount to bet
7. *System* checks that the *User* has that money amount and that the date is no passed.
8. *System* checks that the **User** hasn't bet in any of that cuotes before.
9. *System* rest money to **User**
10. *System* creates the bet
11. *System* adds the bet to the User bet list.
12. *System* adds the bet to the quotes bet list.

## Alternative flow
1. Date is expired, so finish
2. *User* enters more money than the amount he has. Finish
3. *User* tries to bet in a quoote that he has already bet. Finish
```

- Apostua ezabatu GUI



```
# Flow of events

## Basic Flow

1. **User** clicks the button to see his **Bets**
5. *System* shows the **Cuotes**
2. *System* shows the bet of the user
3. *User* confirm that he wants to delete the selected **Bet**
4. *System* return the money to the **User**
5. *System* deletes the bet

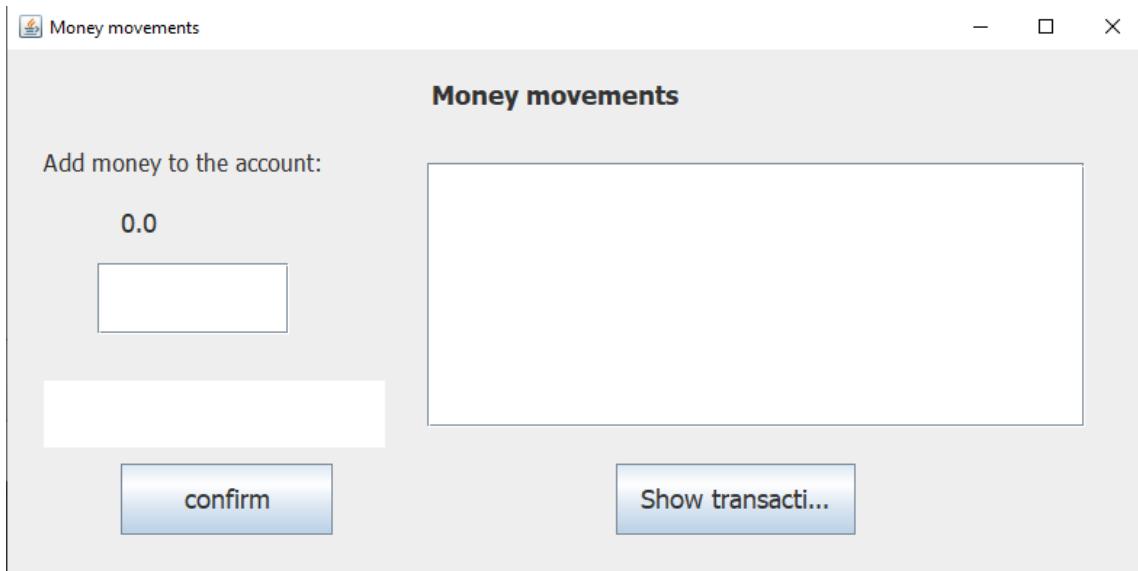
## Alternative flow

1. *User* didn't bet in that cuote
2. *User** tries to delete a **Bet** without selecting any.
```

## Software ingeniaritza

Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

- Dirua sartu GUI / Diru mugimenduak ikusi



```
# Flow of events
## Basic Flow
1. *System* shows a Interface where the user can addMoney
2. *User* enters the neccesary money
3. *System* check that the amount of money is bigger than 0
4. *System* adds the transaction (add money) to the user transaction list using type and amount
5. *System* updates *User* money
```

```
## Alternative flow
1. *User* enters an amount of money = 0 or lower
2. *System* sends a message showing what is wrong
3. *User* enters amount again
```

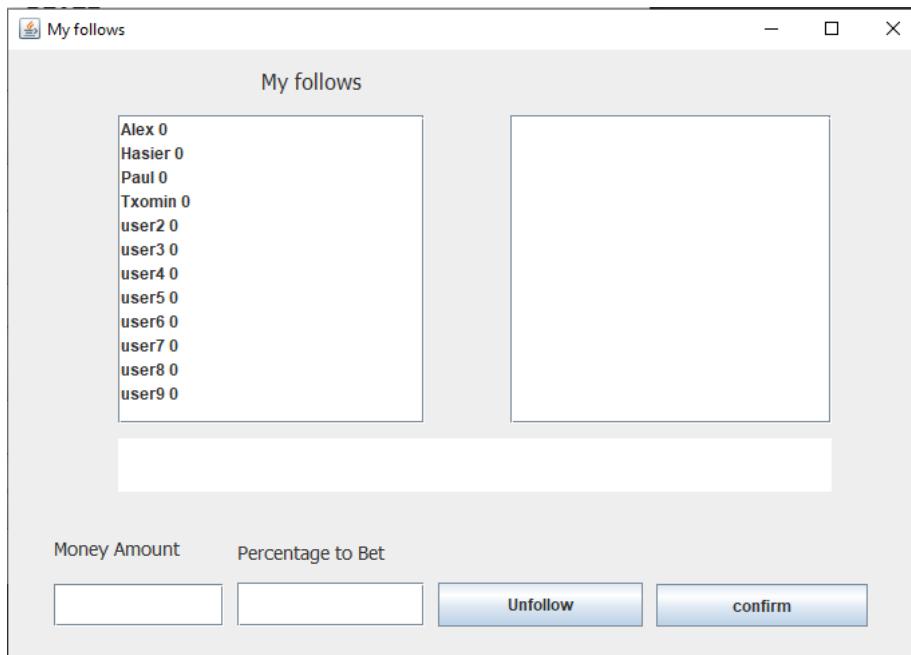
```
# Flow of events
## Basic Flow
1. *System* shows a Interface where the user can see his money transactions
2. *User* clicks the button
3. *System* shows to the user his transactions
```

```
## Alternative Flow
1. There are no transactions for that *User*
```

## Software ingeniaritza

Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

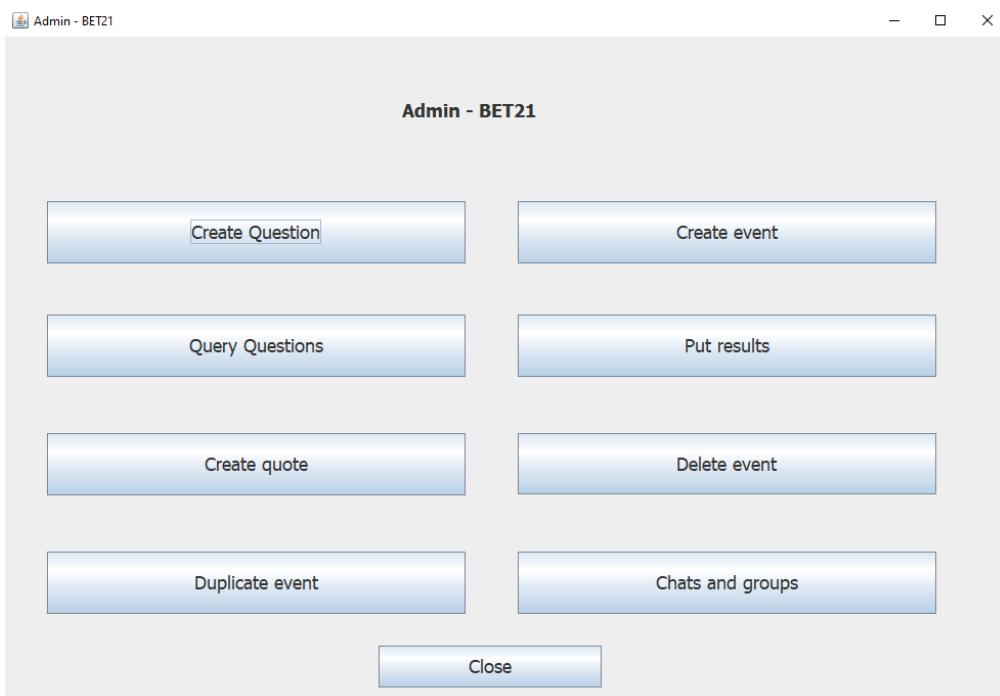
- Jarraitu GUI



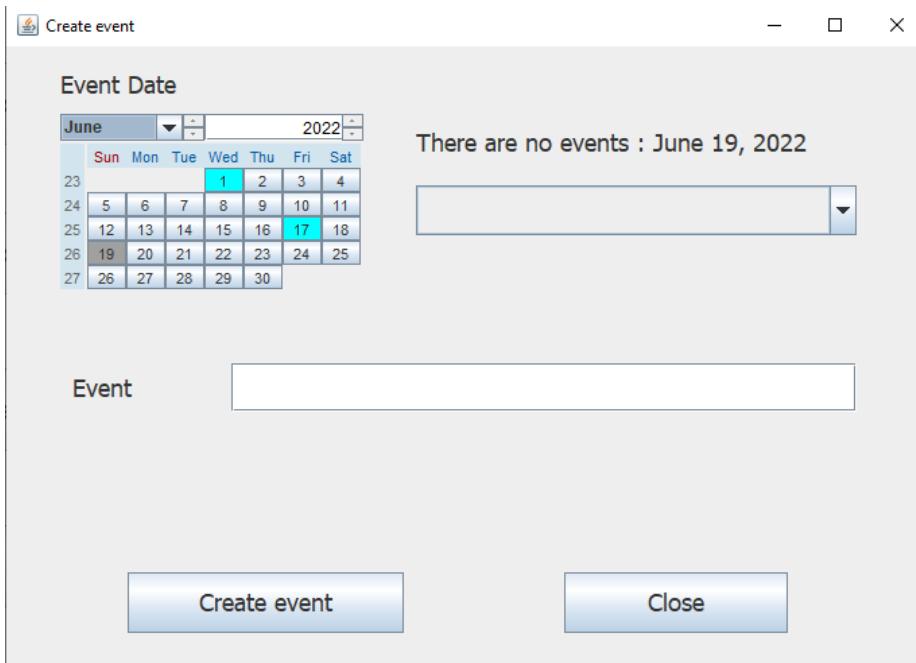
```
# Flow of events
## Basic Flow
1. *System* shows an interface where appears a list of users with some stats
2. **User** chooses from the list a **User** which wants to follow
3. **User** enters the amount of money that he wants to use on the bets of the user is going to follow and what percentage of the **User** he wants to bet.
4. System adds the **Follow** to Users

## Alternative flow
1. *User* entered values that are wrong
2. *User* tries to follow another user with more money than he has
```

- Admin GUI



- Gertaera sortu GUI



```
# Flow of events

## Basic Flow

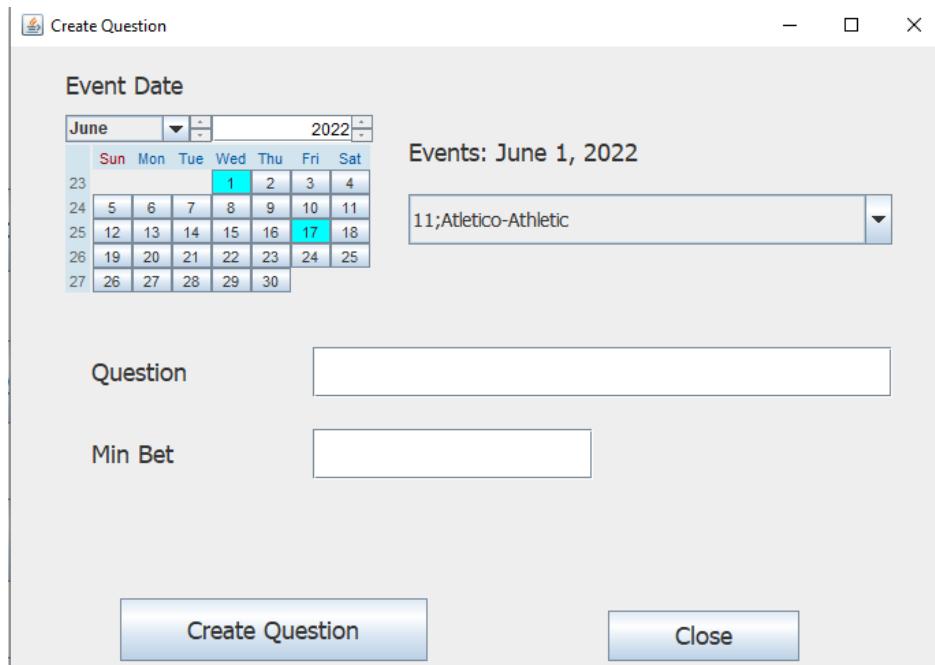
1. *System* shows a Calendar
2. *Admin* selects a **Date** in a Calendar
3. *System* displays the **Events** of this **date**
4. *Admin* introduces a **Event** to that date
5. *System* adds the new **Event** to the selected **date**

## Alternative flow
1. *System* sends a message that the date is wrong
2. *Admin* selects another date
3. That **Event** for that date is already created.
```

## Software ingeniaritza

Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

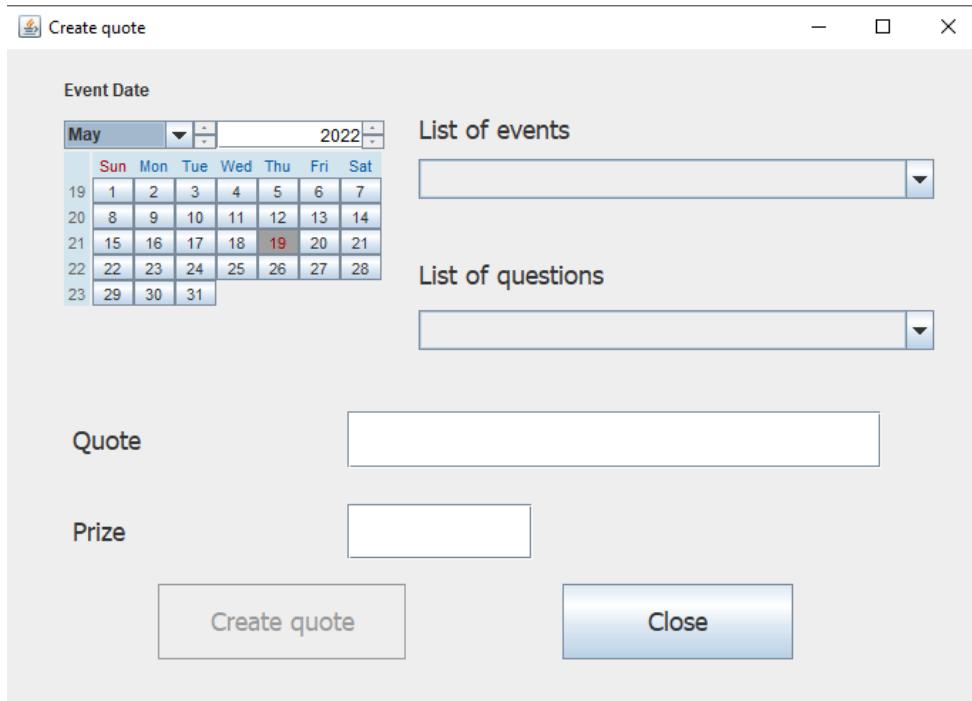
- Galdera sortu GUI



```
# 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 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.
```

- Kuota sortu GUI



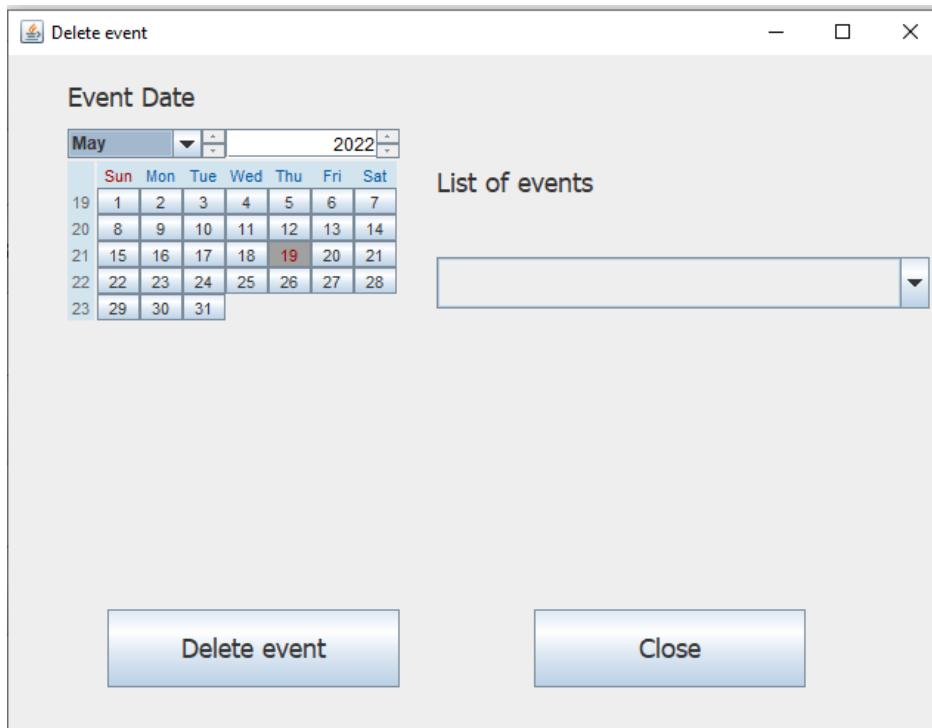
```
# Flow of events
## Basic Flow
1. *System* shows a Calendar where days with events are highlighted
2. *Admin* selects a **Date** in a Calendar
3. *System* displays the **events** of this **date**
4. *Admin* selects an **event**
5. *System* gets the questions of the selected **Event**
6. *Admin* selects a **question**
7. *Admin* introduces a **cuote**
7. *System* adds the new **cuote** to the selected **question**

## Alternative flow
1. There are no event for the selected day.
2. There are no question on this event. Cuote cannot be added.
3. Question date has already finished (event day is before current day). Cuote cannot be added.
4. Cuote already exists for that question
5. Date already passed
```

## Software ingeniaritza

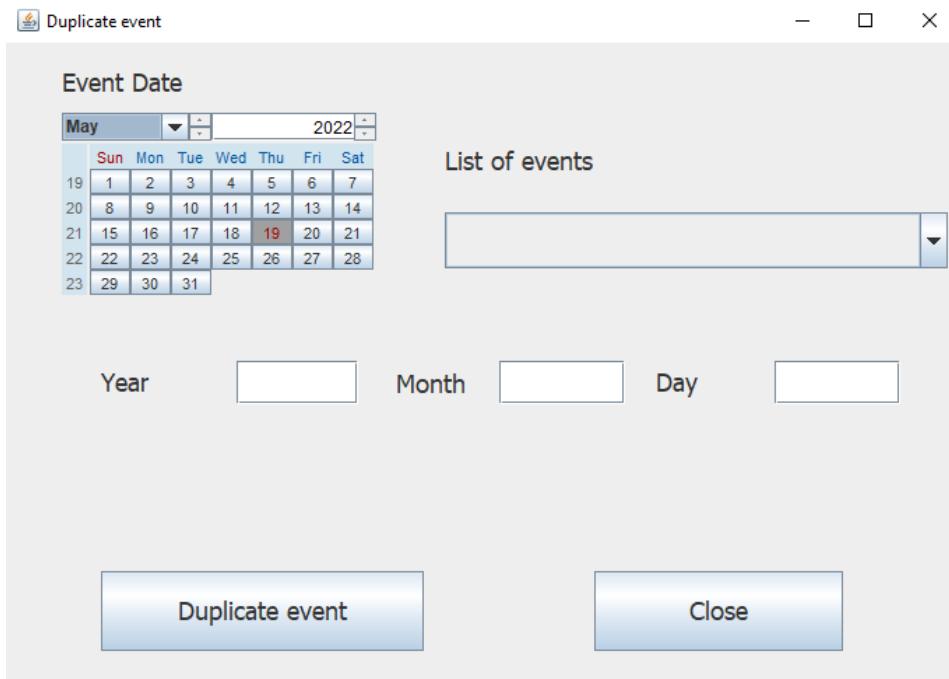
Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

- Gertaera ezabatu



```
# Flow of events
## Basic Flow
1. *Admin* selects a **Date**
2. *System* displays the **events** of this **date**
3. *Admin* selects an event and clicks delete button
4. *System* return money to all *Users* that has a **Bet** in that event
5. *System* deletes all cuotes (for that event)
6. *System* delete all questions (for that event)
7. *System* deletes the selected event
```

- Gertaera bikoitzu GUI



```
# Flow of events

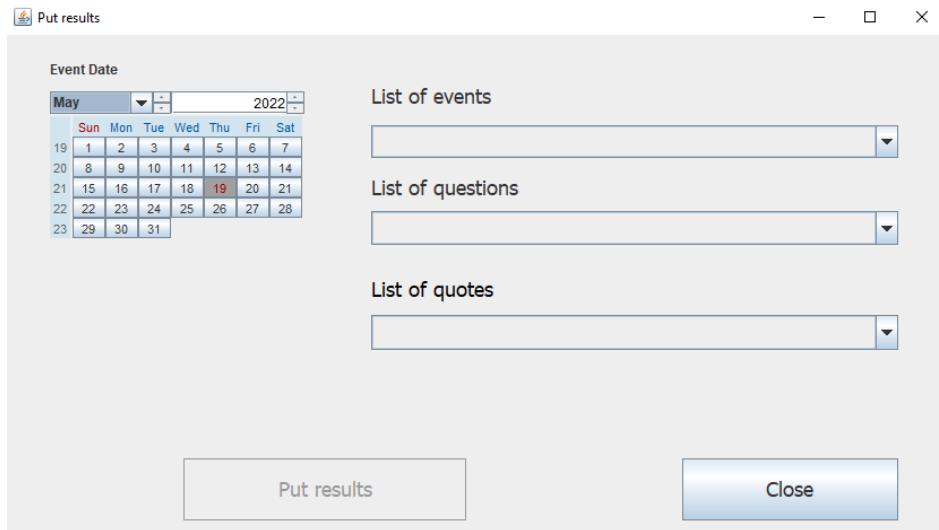
## Basic Flow

1. *System* shows a Calendar
2. *Admin* selects a **Date** in a Calendar
3. *System* displays the **events** of this **date**
4. *Admin* selects a **Event**
5. *Admin* enters the day to copy the **Event**
6. *System* creates a new **Event** that is equal to the selected.

## Alternative Flow

1. Introduced *Date* is not valid.
```

- Emaitzak ipini GUI

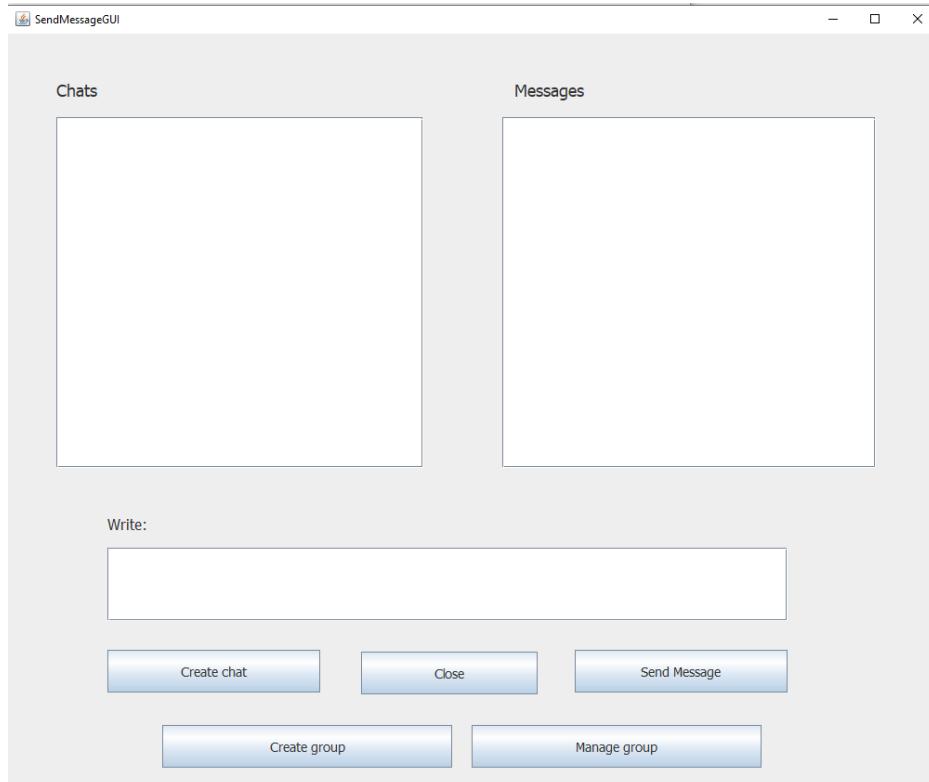


```
# Flow of events
## Basic Flow
1. *System* shows a Calendar
2. *Admin* selects a **Date** in a Calendar
3. *System* displays the **events** of this **date**
4. *Admin* selects a **Question** for that **Event**
5. *Admin* selects the winner quote.
```

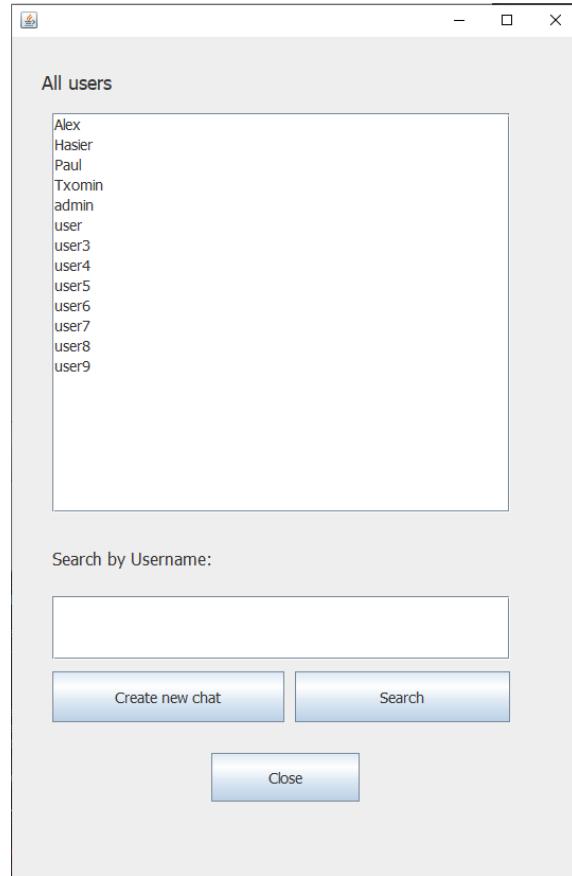
## Software ingeniaritza

Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

- Mezua bidali GUI



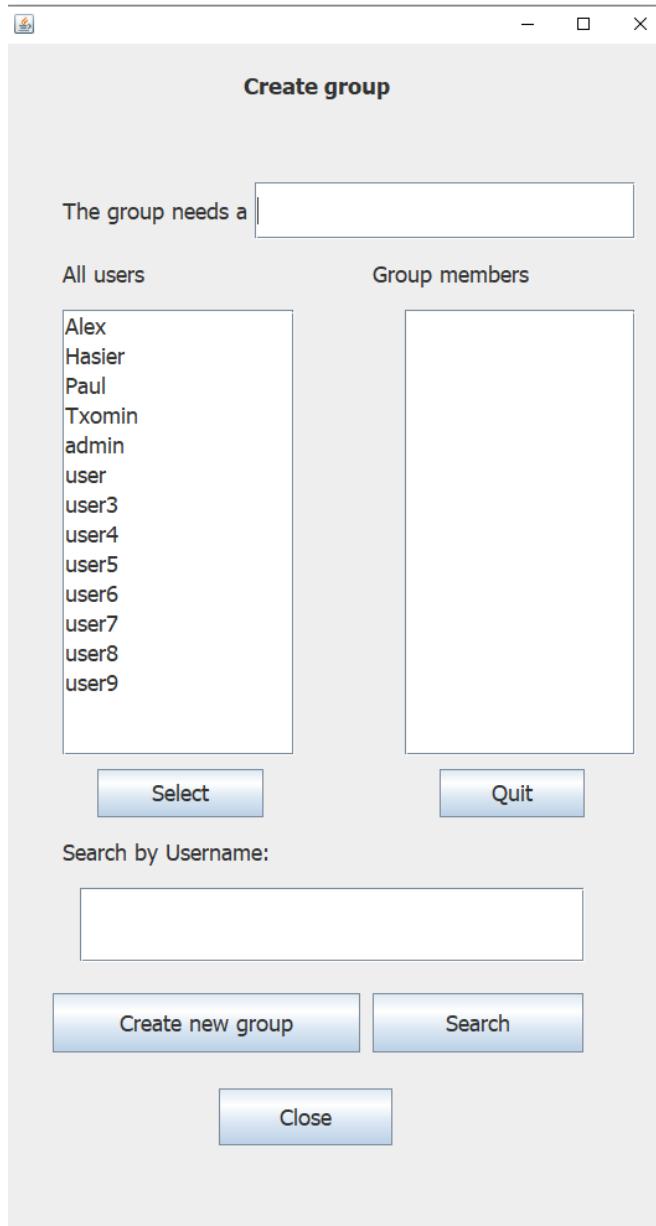
- Txata sortu GUI



## Software ingeniaritza

Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

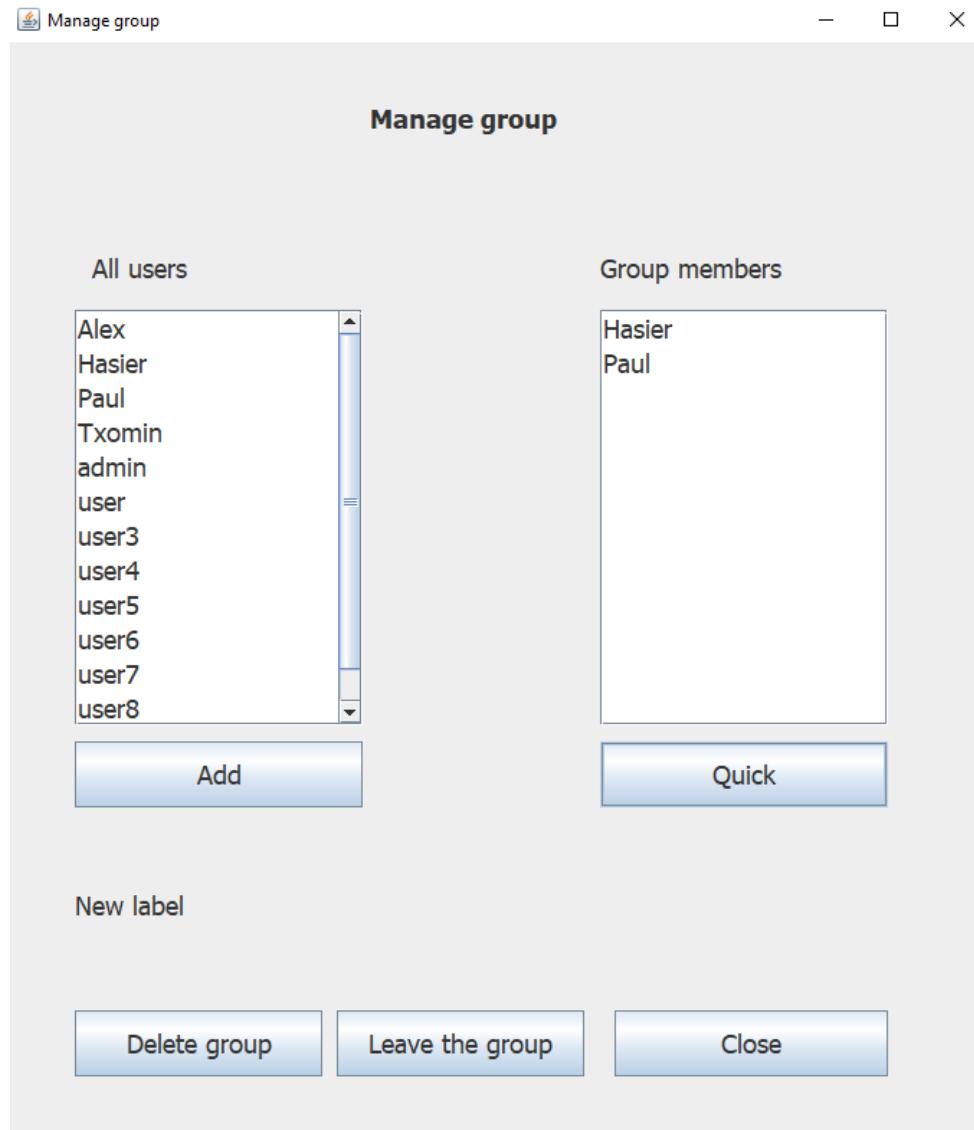
- Taldea sortu



```
# Flow of events
## Basic Flow
1. *System* shows a Interface where the user see his **Chats** and can press the button to create group
2. *User* press the button to create group
3. *System* System redirects **User** to another GUI
4. **User** enters the name of the group and the usernames of the people that will be in the group
5. *System* creates the group.

## Alternative flow
1. *User* enters an amount of money = 0 or lower
2. *System* sends a message showing what is wrong
3. *User* enters amount again
```

- Taldea administratu GUI



```
# Flow of events
## Basic Flow
1. *User* clicks to manage the selected group
2. *User* press the button to leave the group
3. *System* deletes the group from *User* group list.
4. if the *User* was the group admin, *system* will set the first *User* of the group as group admin
```

```
# Flow of events
## Basic Flow
1. *User* press the button to delete the group
2. *System* will delete the group from all *Users* group list.
3. *System* deletes the group from the data base

## Alternative flow
1. *User* need to be the group admin
```

```
# Flow of events
## Basic Flow
1. *User* clicks the button to manage group
2. *User* clicks to quick one of the members
3. *System* deletes *User* from group members list
4. *System* deletes *Group* from *User* group list
```

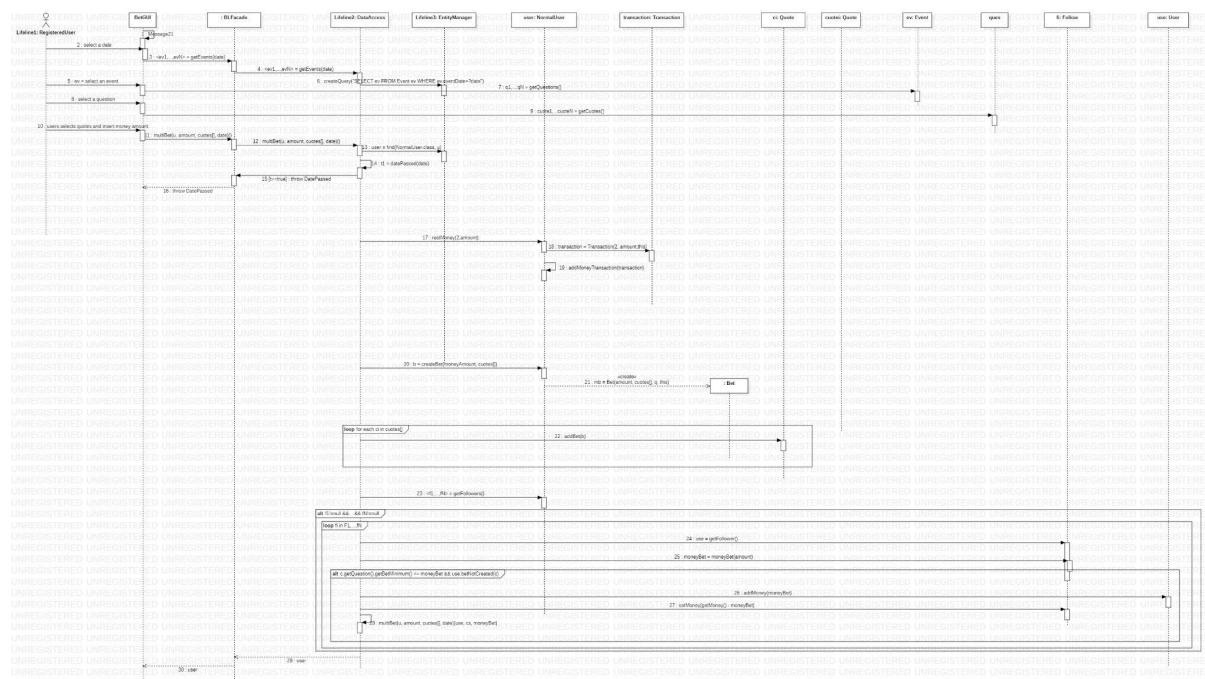
```
# Flow of events
## Basic Flow
1. **User** selects a group and press the button to manage it
2. **System redirects User to another GUI
3. **User** enters the name of the **User** he wants to invit to the group
4. *System* add the **User** to the group

## Alternative flow
1. *User* need to be the group admin to add people
```

## Diseinua

- Sekuentzia diagramak

  - CreateMultiBet

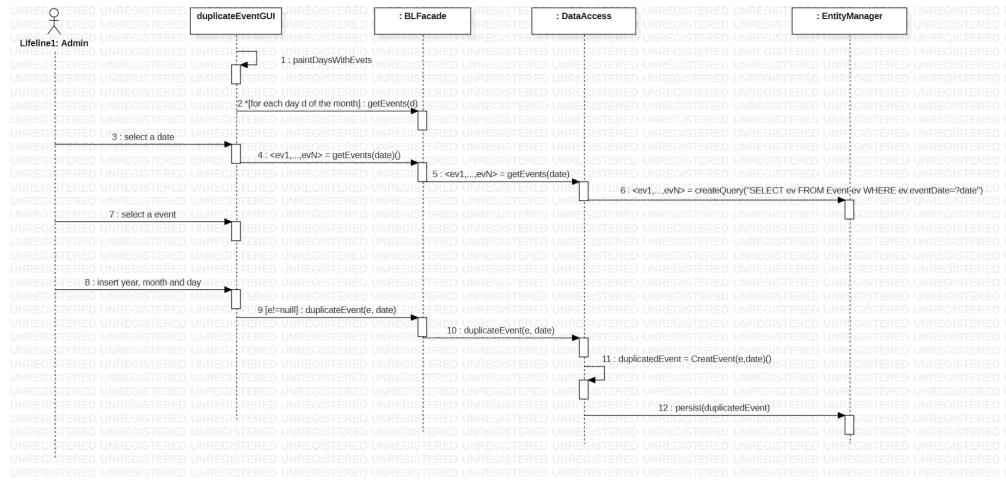


CreateMultiBet erabilpen kasua egiteko, BetGUI-tik abiatu gera, honen funtzionamendua pixka bat aldatuz. Lehenego Gui-an egutegi bat erakusten zaigu eventuak dauzkaten egunekin urdinez markatuak, auekeratzten dugu data bat eta honen gertaerak erakusten zaizkigu, ondoren honen gertaeeren artean galdera bat aukeratu dezkegu eta honen galderen cuoten artean apostu egin nahi dugun kuota aukeratu dezakegu eta apostatuko dugun diru kantitatea sartu. Gero botoia sakatuz Facade-ko dei bat egingo dugu Multibet funtzioa exekutatz. Bet-aren kuota baten data pasatu bada errore bat bueltatuko da. Data ez bada pasatu exekuzioa jarraituko da, lehenengo apostatutako dirua kenduko zaio erabiltzaileari. Ondoren user barruan sortuko da multibet-a sartuz kuotak, diru kantitatea, eta erabiltzailea. Gero Multibetaren bet bakoitzagatik, honen kuotara gehituko zaio bet-a. Bukatzeko

## Software ingeniaritzak

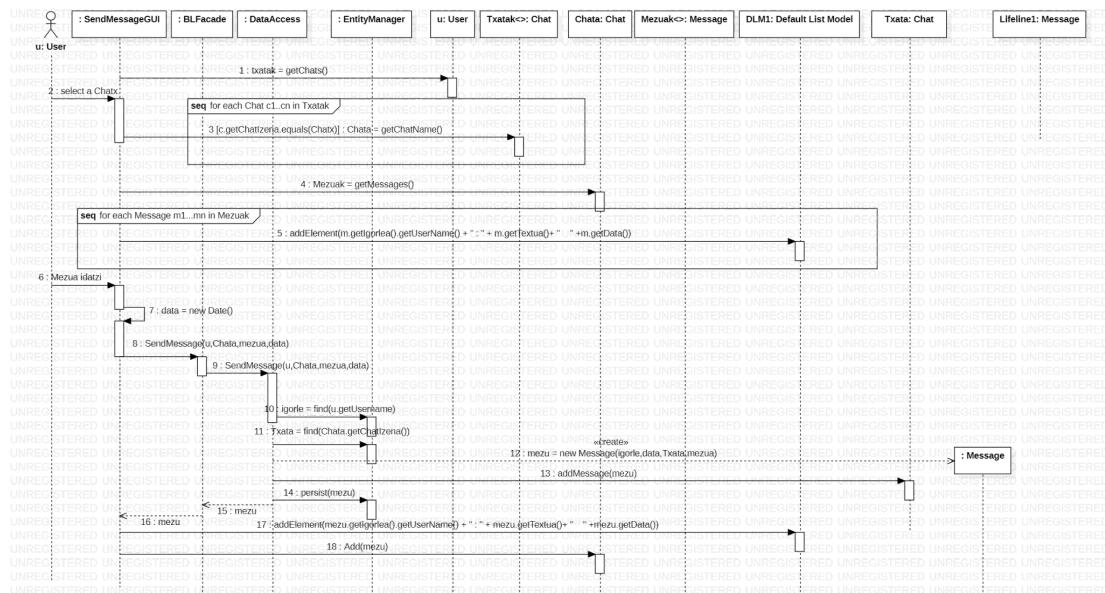
Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

- DuplicateEvent



Lehenengo GUI-an existitzen diren gertaerak kargatzen dira eta gertaerak dituzten egunak urdinez margotzen dira. Ondoren erabiltzaileak aukeratzen du duplikatu nahi duen gertaeraren eguna eta sistemak kargatzen ditu egin honen gertaerak. Bikoitztutako gertaera eta bikoitztuaren data aukeratuz eta botoia sakatuz Facade-ko `duplicateEvent` exekutatzentz da. Ondoren gertaera bat sortzen da aukeratutako gertaera eta data berria pasatz parametro bezala, ondoren gertaera honenen persist-a egiten da.

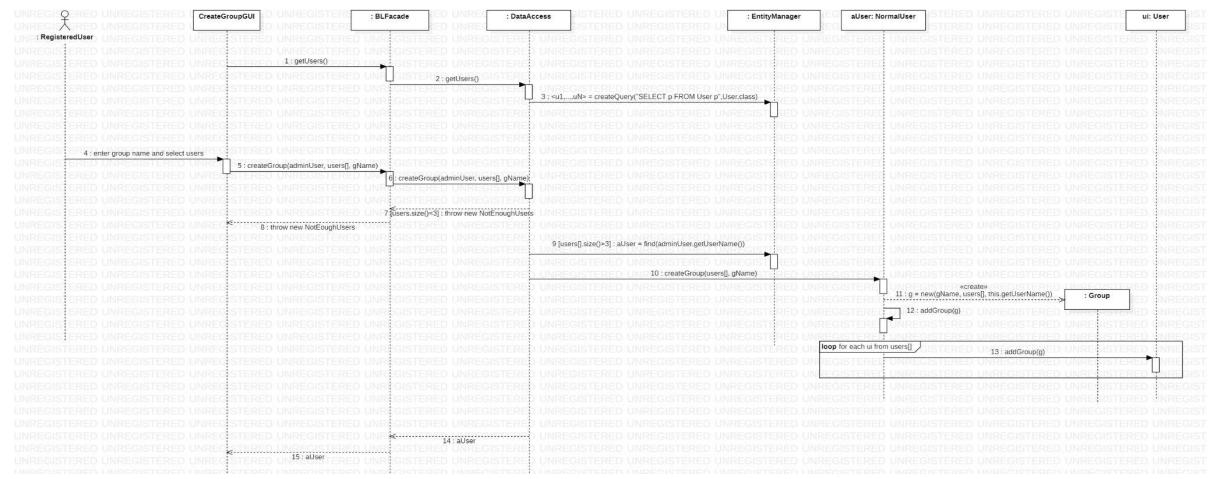
- **SendMessages**



Erabilpen kasu honetan lehenengo erbiltzailearen txatak kargatzen dira JList batean, honen ondoren txat bat aukeratuta badago honen mezuaak beste JList batean. Dena kargatuta dagoenean idazten da bidali nahi den mezua, mezua bidali botoia sakatzean momentuko data jasotzen da, data, erabiltztailea(iqorela), aukeratutako Txat-a eta mezuaaren testua

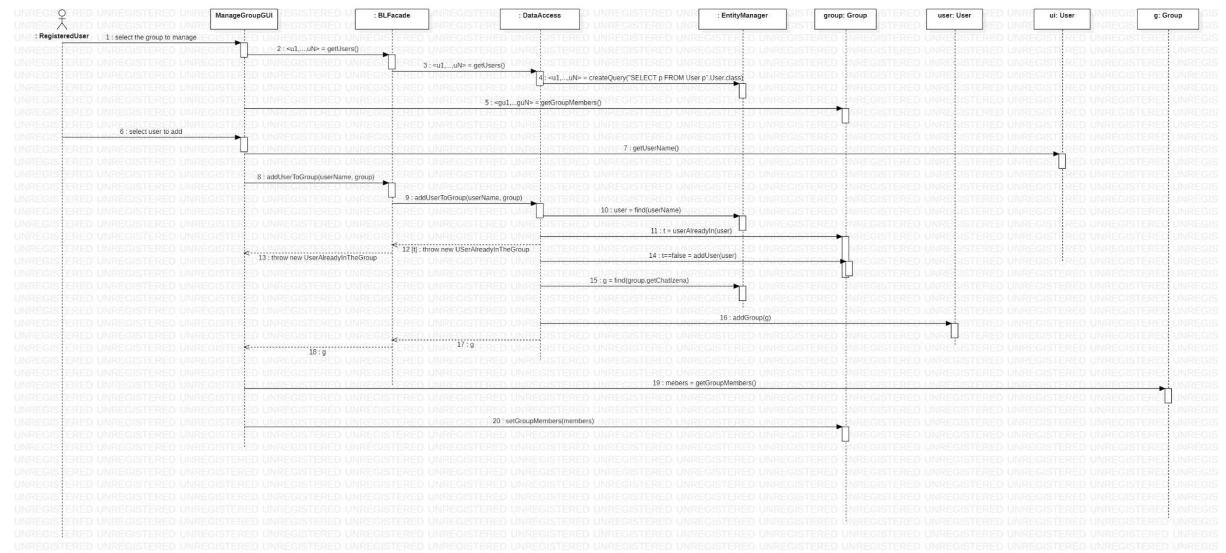
pasatuz Facade-ko SendMessageGUI funtzioa exekutatzen da. Dataacces mailan emandako txata eta igorlea bilatzen dira datubasean. Datu hauekin sortzen da Mezu bat, mezu hau datubasean aurkitutako Txat-ari gehitzen zaio. Gero mezu honen persista egiten da, SendMessages sortutako mezua bueltatzen du eta GUI-ko erabiltzailera zein Mezuetako DLM-ari gehitzen zaio mezua.

- CreateGroup



CreateGroup erabilpen kasuan lehenengo datubaseko erabiltzaile guztia kargatzen dira , Facade-ko getUsers funtzioa erabiliz. Ondoren kargatutako erabiltzaileak aukeratzen dira nahi direnak taldean egotea, honez gain taldearen izena aukerazten da. Datu hauekin Facade-ko CreateGroup funtzioa exekutzten da , Dataacces klasean konprobatzeko da ea taldeak 3 erabiltzaile dituen gutxienez, gutxiago baditu , NotEnoughUsers errorea jaruti egiten du , 3 erabiltzaile baingo gutxiago baditu txat pribatua izango zelako. Errore hau ez badu jaurti funtzioa exekutazten jarraitzen du, taldeko erabiltzaileak biltazen dira datubasean eta gordetzen dira ArrayListbatean. ArrayList hau eta taldeko izena pasatuz parametro bezala NormalUser klaseko CreateGroup funtzioa exekutatzen da, talde berri bat sortuz (g) eta taldeko erabiltzaile guztiei taldea gehituz. Bukatzean Admin(Taldea sortu duen erabiltzailea) bueltatzen da.

- AddUserToGroup

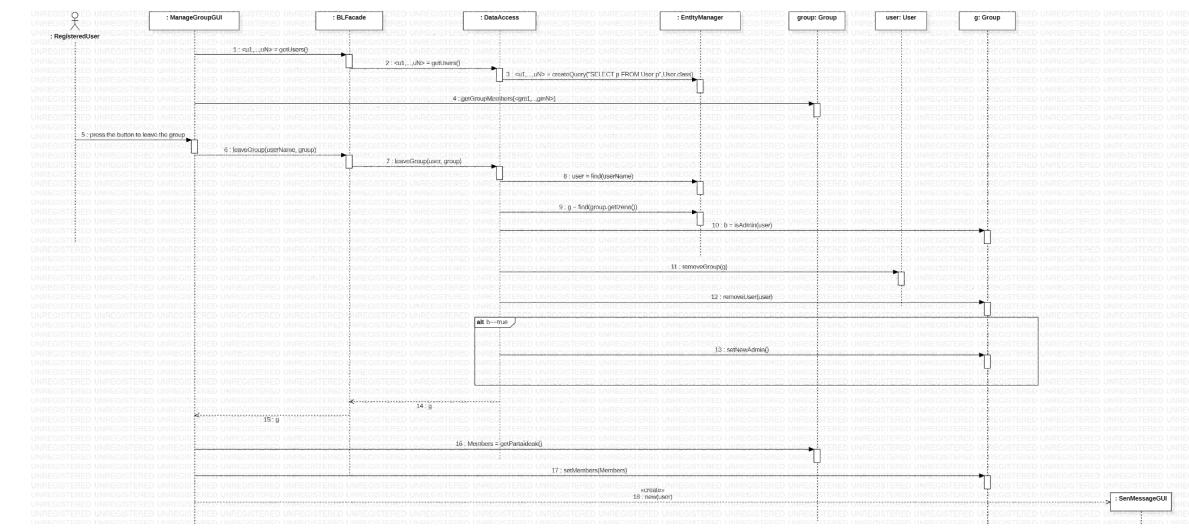


Erabiltzaile bat gehitzeko Talde batera lehenengo aukeratu egin behar da kudeatu nahi den taldea, hau aukeratzean datubasetik kargatzen dira databaseko erabiltzaile guztiak , eta bereztein da beste bigarren lista bat bakarrik taldeko erabiltzaileekin. Gero aukeratzne da erabiltzaileen artean taldera gehitu nahi den erabiltzailea. Botoia sakatuz facadeko addUserToGroup funtzioa exekutazen da, pasatuz taldea eta erabiltzaile berria parametro bezala. Sartu nahi den erabiltzailea datubasetik kargatzen da eta konprobatzan da ea hau taldean dagoen, taldean badago USerAlreadyInTheGroup errorea jarutitzen da. Bestela exekuzioa jarraitzen da. Taldea datubasetik kargatzen da eta honi gehitzen zaio erabiltzaile berria eta erabiltzaileari taldea. Bukatzeko funtzioa taldearen return bat egiten du. Bukatzeko aktualizatzen dira GUI-an dagoen group objektuaren partaideak.

# Software ingeniaritza

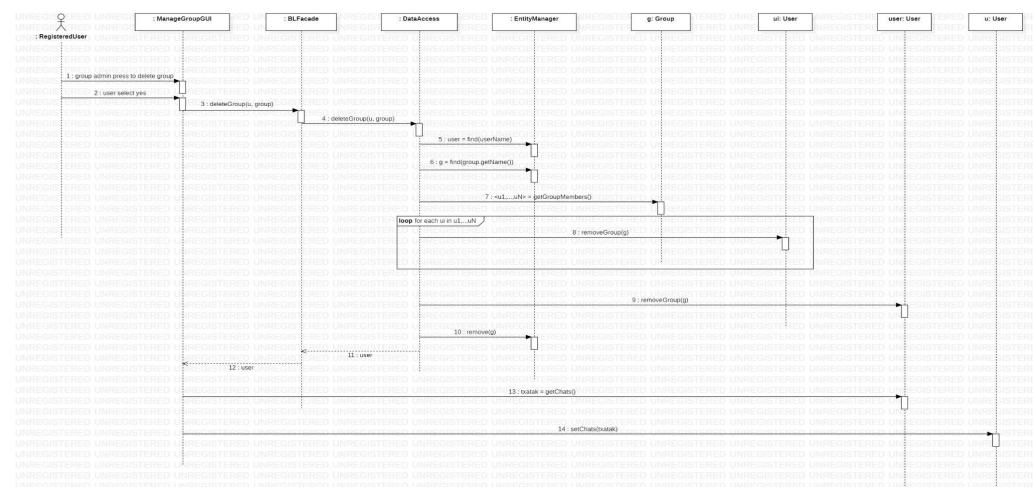
Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

- LeaveGroup



Erabilpen kasu honetan aurrekoan bezala , ManageGroupGUI-tik deia egiten denez aurreko erabilpen kasuan bezala, databaseko erabiltzaileak kargatzen dira. Leave group botoia sakatzean, facadeko leaveGroup funtzioa exekutatzen da. Dataacces-en erabiltzailea eta taldea kargatzen dira datubasetik. Konprobatzeko da ea erabiltzailea taldearen administratzailea den ala ez, ondoren erabiltzaileari taldea ezabatzen zaio eta taldetik erabiltzailea. Erabiltzailea taldearen admina baten, setNewAdmin exekutatzen da, taldearen administratzaile berri bezala erabiltzaile zaharrena ezarriz. Bukatzeko taldearen return-a egiten da eta GUI-ko group objektuaren partaideak berruitzen dira.

- DeleteGroup



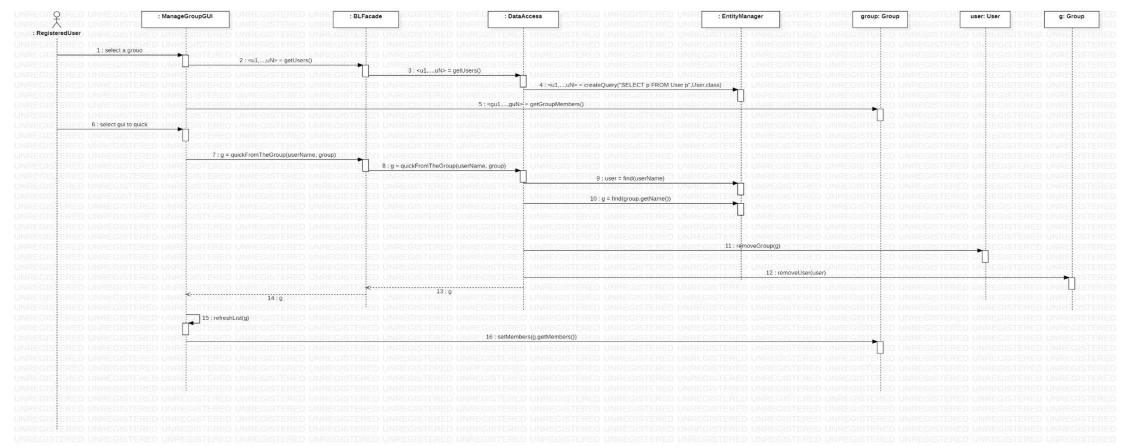
Talde bat borratzeko lehenengo honen admin-a taldea aukeratzen du eta botoia sakatzen du, gero erabakia baieztatzen du "yes" botoia sakatu. Ondoren deleteGroup exekutatzen da , gero borratu nahi den taldea eta borratzen duen erabiltzailea datubasetik kargatzen dira. Taldearen erabiltzaile bakoitzean talde hau ezabatzen da eta ondoren taldea ezabatzen

# Software ingeniaritza

Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

da taldea datubasetik. Bukatzeko erabiltzailea bueltatzen da eta hoen txatak esartzen dira GUI-ko txatak bezala, deleteGroup exekutazean Chat-ak egunerezko.

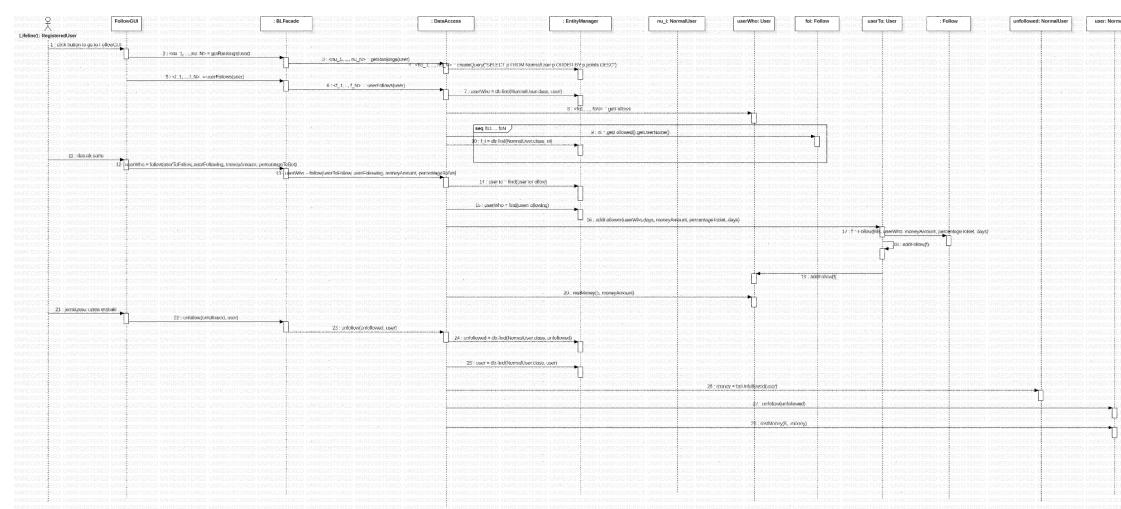
- Quick From Group



Lehnengo ManageGroupGUI-an erabilizaleak argatzen dira, ondoren user-a aukerazten du taldeko zein erabiltzaile bota nahi duen taldetik eta botoia sakatzen du. Parametro bezala taldetik botako duen erabiltzailearen izena eta taldea pasaz exekutatzen da QuickFromTheGroup funtzioa. Parametro hauek erabiliz, datubasetik ekartzen dira erabiltzaliea eta Taldea, erabiltzaileari taldea ezabatzen zaio taldeko listatik eta taldeko partaideetatik erabiltzaile hau ezabatzen da. Funtzioak taldea bueltazen dio GUI-ari eta hau erabiltzen da taldearen partaideak berriztatzeko.

- FollowUser

○



Follow GUI exekutazean sistema Facade-ko GetRankings eta userFollows erabiliz, kargatzen dira puntu gehiago duten erabiltzailen ranking bat, ranking hau apostu irabazien arabera dijoa, eta erabiltzaileak jarraitzen dituen erabiltzaileen lista bat. GUI-an datuak kargatuta eukita user-a datuak sartzen ditu( Zein zerabiltzaile jarraitu, zenbat dirua erreserbatuko da pertsona honen apostuak jarraitzea eta apostuen kopiak egitean hauem

## **Software ingeniaritza**

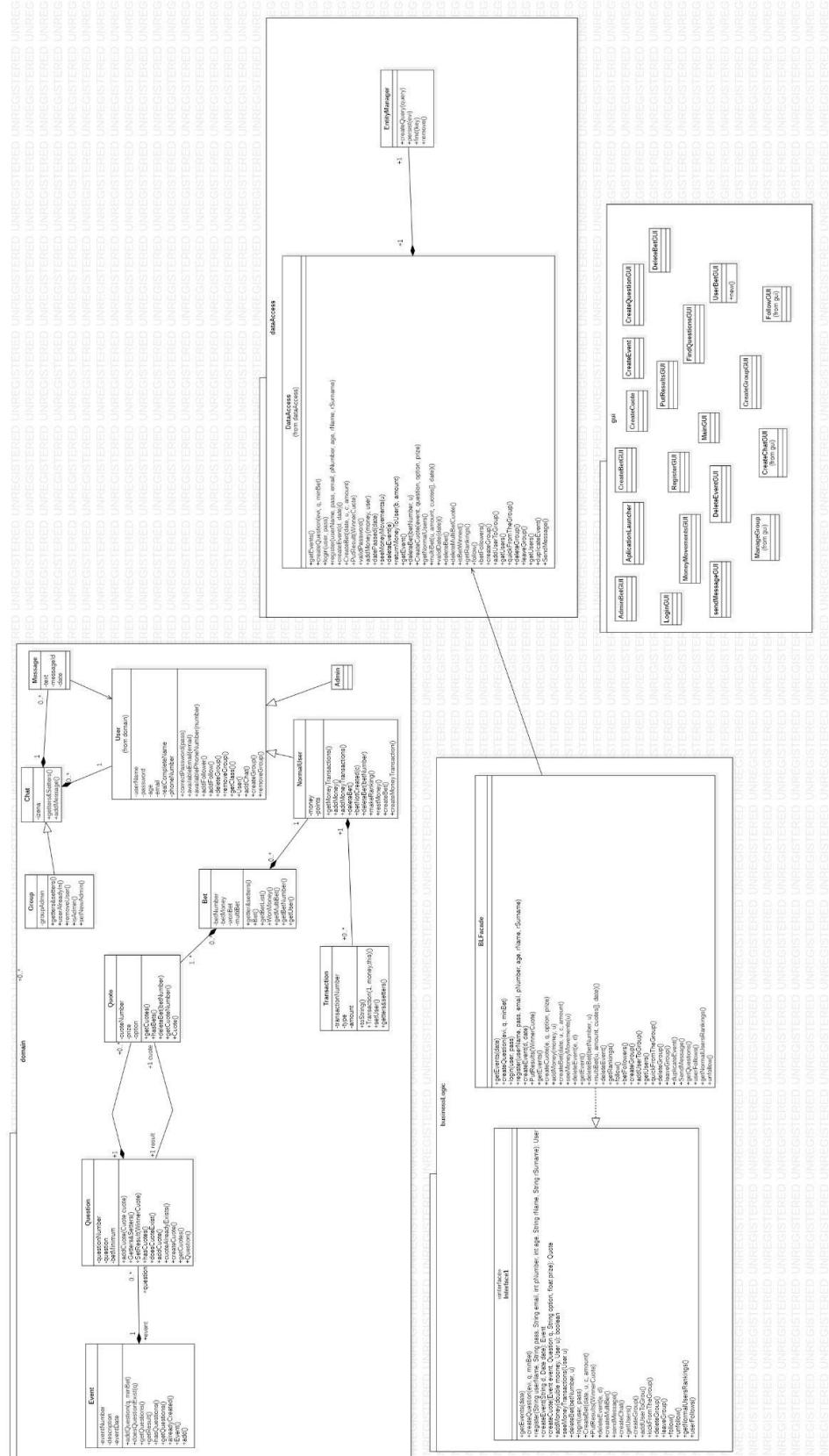
Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar

---

diruaren ze portzentaila apostatuko da). Datu hauek eta aplikazioa erabiltzen duen user-a pasatuz exekutatzen da facade-ko follow funtzioa: Datacesen datubaetik erabiltzaile jarritzaile eta jarraitua kargatzen dira, ondoren User jarraituari Follower bat gehitzen zaio addFollower erabiliz. AddFollower bueltatzen duen Follow a jarritzaileari gehitzen zaio ere. Beste funtzio bat implementatuta dago Unfollow, botoia sakatuz jarraitzen uztea erbaki dugun erabiltzailea eta exekuzioan dagoen erabiltzailea pasatuz unfollow metodoa exekutatzen da. Metodo honek perstona bat Jarraitzeari uzten dio, erabiltzaile jarritzailea zein jarraitua kargatzen dira datubastik, gero jarraipenari lotutako dirua jasotzen da, jarraitzeari uzten zaio eta jarraitzaileri erabiltzaile hau jarraitzeko orduan sartutako diruaren geratzen kantitatea bueltatzen zaio.

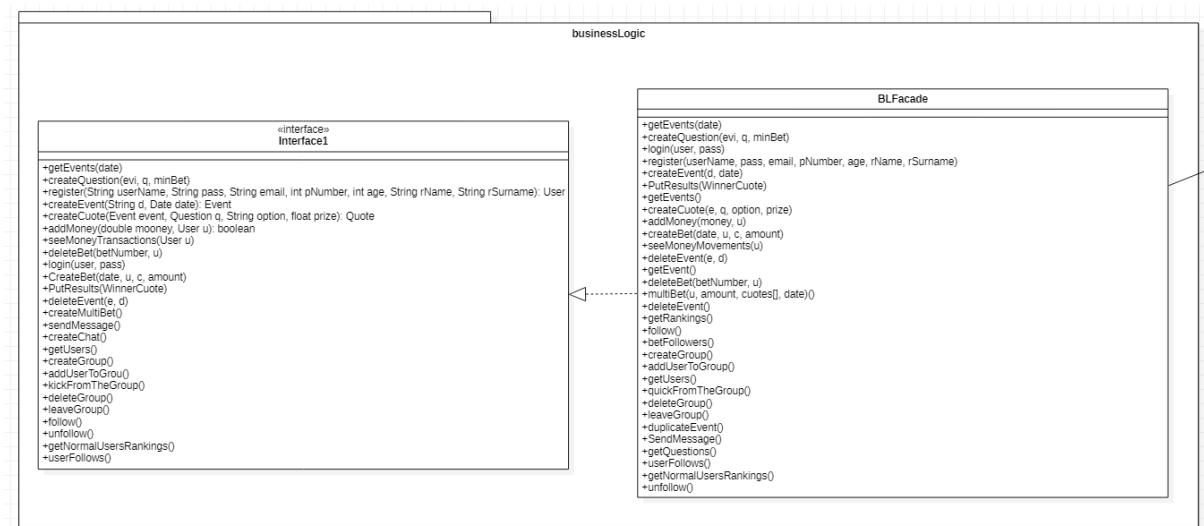
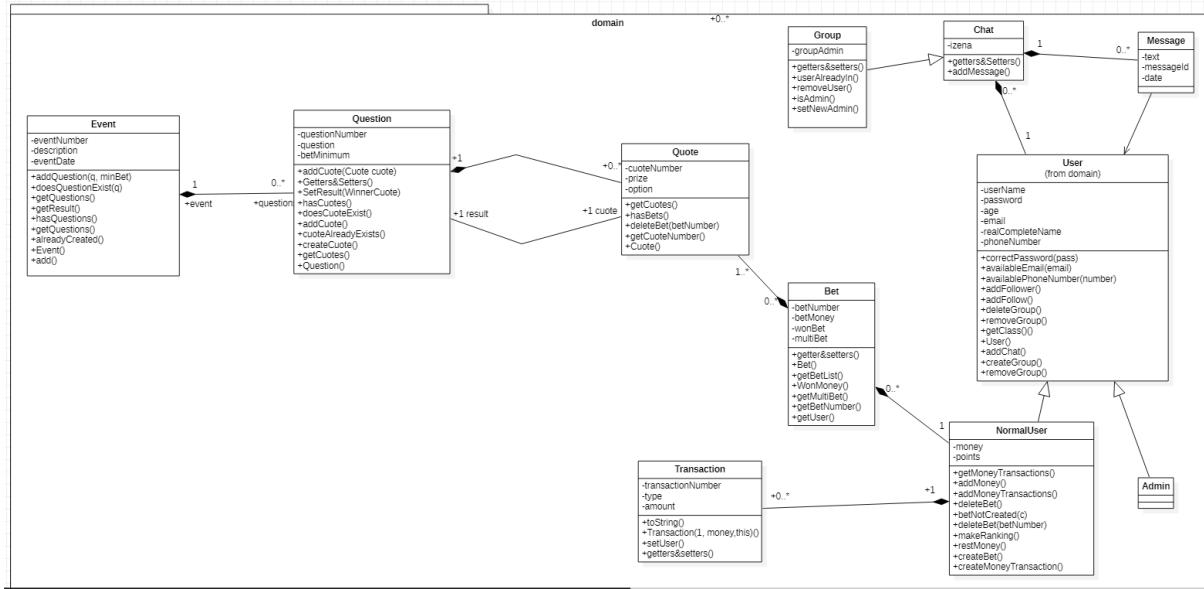
+

# Klase diagrama



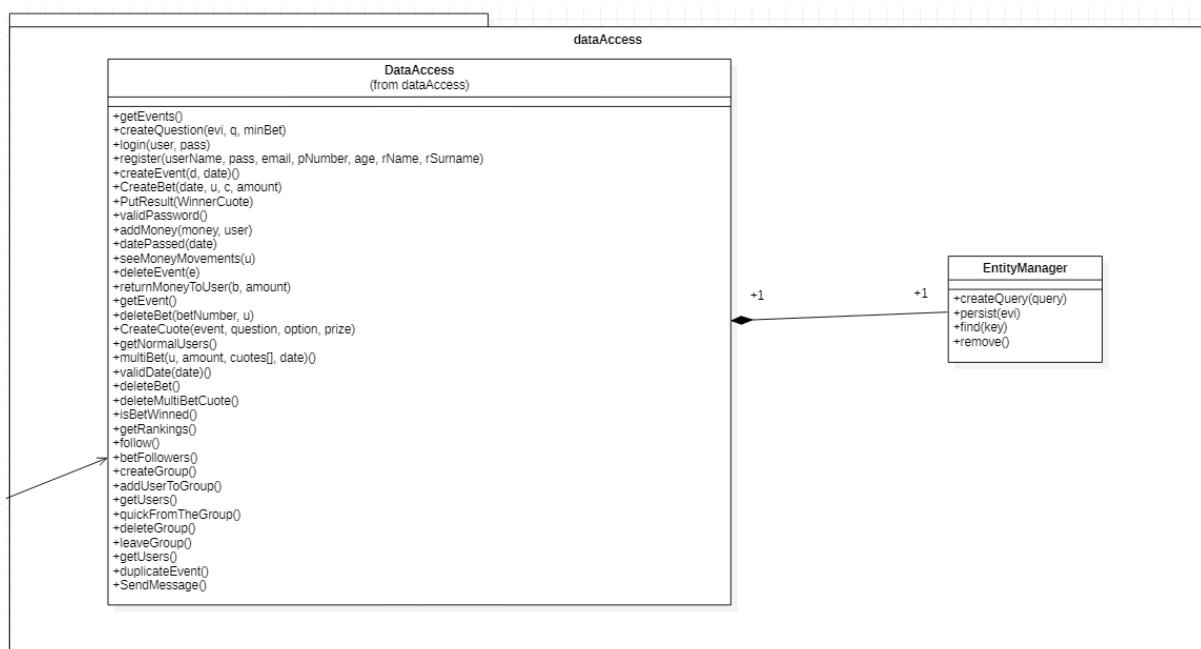
# Software ingeniaritza

Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar



## Software ingeniaritza

Paul Martinez, Hasier Zaldua, Txomin Errasti eta Alex Amenabar



## **Implementazioa**

```
/**  
 * Metodo hau erabiltzaileak saioa hasteko erabiltzen da  
 * @param user erabiltzailearen izena  
 * @param pass erabiltzailearen pasahitza  
 *  
 * @throws IsEmpty aldagietako bat null bada  
 * @throws UserNameAlreadyUsed erabiltzailearen izena dagoeneko erabilita  
 badago  
 * @throws WrongPassword pasahitza okerra bada  
 * @throws UserNotExist erabiltzaile hori ez bada existitzen  
 */
```

```
public User login(String u, String p) throws  
IsEmpty,WrongPassword,UserNameAlreadyUsed,UserNotExist;
```

```
/**  
 * Metodo hau erabiltzaile bat erregistratzeko erabiltzen da  
 * @param userName erabiltzaileak erabiliko duen izena  
 * @param pass erabiltzaileak erabiliko duen pasahitza  
 * @param email erabiltzaileak erabiliko duen email helbidea  
 * @param pNumber erabiltzailearen telefono zenbakia  
 * @param age erabiltzailearen adina  
 * @param rName erabiltzailearen izena  
 * @param rSurname erabiltzailearen abizena  
 *  
 * @throws IsEmpty aldagaietako baten balioa null bada  
 * @throws WrongPassword pasahitza ez bada zuzena (luzera <3)  
 * @throws UserNameAlreadyUsed izena dagoeneko erabilita badago  
 * @throws AgeException adina 18 baino txikiagoa bada  
 * @throws PhoneNumberUsed telefono zenbakia dagoeneko erabilita badago  
 * @throws EmailUsed email helbidea dagoeneko erabilita badago  
 *  
 */
```

```
public NormalUser register(String userName, String pass, String email, int  
pNumber, int age, String rName, String rSurname) throws  
IsEmpty,WrongPassword,UserNameAlreadyUsed,AgeException,PhoneNumber  
Used,EmailUsed;
```

```
/**  
 * Metodo honek gertaera bat sortzen du  
 * @param date gertaeraren data  
 * @param d gertaeraren deskribapena  
 *  
 * @throws DatePassed data dagoeneko pasata badago  
 * @throws EventAlreadyCreated gertaera hori sortuta badago  
 */  
  
public Event createEvent(String d, Date date) throws  
DatePassed,EventAlreadyCreated;  
/**  
 * Metodo honek kuota bat sortzen du  
 * @param event kuota gehituko zaion gertaera  
 * @param q kuota gehituko zaion galdera  
 * @param option kuota mota  
 * @param prize kuotaren saria  
 *  
 * @throws CuoteAlreadyCreated kuota dagoeneko sortuta badago  
 * @throws EventFinished gertaera bukatuta badago  
 * @throws NoValidMoneyException sariaren zenbakia gaizki badago  
 *  
 */  
  
public Quote createCuote(Event e, Question q, String option, float prize)  
throws CuoteAlreadyCreated,EventFinished,NoValidMoneyException;  
  
/**  
 * Metodo honek erabiltzailearen kontuan dirua gehitzen du  
 * @param money sartu nahi den diru kantitatea  
 * @param u dirua gehitu behar zaion erabiltzailea  
 *  
 * @throws NoValidMoneyException sartutako diru kantitatea okerra bada  
 * @throws IsEmpty aldagietako bat null bada  
 */  
  
public NormalUser addMoney(double money, NormalUser u) throws  
NoValidMoneyException, IsEmpty;  
/**  
 * Metodo honek erabiltzailearen diru mugimenduak erakusten ditu  
 * @param u erabiltzailea  
 */
```

```
public ArrayList<Transaction> seeMoneyMovements(NormalUser u);
/** 
 * Metodo honek apostu bat sortzen du
 * @param u apostua egin duen erabiltzailea
 * @param c apostatu den kuota
 * @param amount apostatu den diru kantitatea
 * @param date apostuaren eguna
 *
 * @throws NotEnoughMoneyException erabiltzaileak nahiko diru ez badu
 * @throws BetAlreadyCreatedException apostua dagoeneko sortuta badago
 * @throws IsEmpty aldagietako bat null bada
 * @throws DatePassed apostuaren eguna pasata badago
 */

public NormalUser createBet(NormalUser u, Quote c, double amount, Date
date) throws
NotEnoughMoneyException,BetAlreadyCreatedException,IsEmpty,DatePassed;

/** 
 * Metodo honek gertaera bat ezabatzen du
 * @param e ezabatu nahi den gertaera
 * @param d gertaeraren eguna
 *
 * @throws IsEmpty e edo d null badira
 */

public void deleteEvent(Event e, Date d) throws IsEmpty;

/** 
 * Metodo honek apostu bat borratzen du
 * @param betNumber apostuaren IDa
 * @param u apostua egin duen erabiltzailea
 *
 * @throws NonPositiveNum apostu zenbakia negatiboa bada
 * @throws IsEmpty erabiltzailea null bada
 */

public User deleteBet(int betNumber, NormalUser u) throws NonPositiveNum,
IsEmpty;
/** 
 * Metodo honek galderen emaitzak jartzen ditu
 * @param WinnerCuote galderaren emaitza asmatu duen kuota
 *
 * @throws CuoteHasNotQuestion kuotak ez badu galderarik
 */
```

```
 * @throws QuestionHasResult galdera horrek dagoeneko emaitza jarrita badu
 */
```

```
 public Quote PutResults(Quote WinnerCuote) throws
CuoteHasNotQuestion,QuestionHasResult;
```

```
 /**
 * Metodo honek apostu anitzak sortzen ditu
 * @param user apostua egin duen erabiltzailea
 * @param cuotes apostuan gehitu dituen kuota guztiak
 * @param moneyAmount apostatu duen diru kantitatea
 * @param date apostuaren eguna
 *
 * @throws NotEnoughMoneyException erabiltzaileak nahiko diru ez badu
 * @throws BetAlreadyCreatedException apostua dagoeneko eginda badago
 * @throws IsEmpty aldagaietako bat null bada
 * @throws DatePassed apostuaren data pasata badago
 *
 */
```

```
 public void duplicateEvent(Event e, Date date) throws IsEmpty;
```

```
 /**
 * Metodo honek gertaerak bikoizten ditu
 * @param e bikoitzu nahi den gertaera
 * @param data gbikoiztutako gertaerari jarri nahi zaion data
 *
 * @throws IsEmpty e edo date aldagaiak null badira
 */
```

```
 public NormalUser createMultiBet(NormalUser user, ArrayList<Quote> cuotes,
double moneyAmount, Date date) throws
NotEnoughMoneyException,BetAlreadyCreatedException,IsEmpty,DatePassed;
```

```
 /**
 * Metodo hhau mezuak bidaltzeko erabiltzen da
 * @param igorle mezia bidali duen erabiltzailea
 * @param txata bi erabiltzaileen artean sortutako txata
 * @param textua bidalitako mezia
 * @param data mezia bidalitako eguna
 */
```

```
 public Message SendMessage(User igorle,Chat txata,String textua, Date data);
```

```
 /**
```

```
* Metodo honek txat bat sortzen du
* @param Izena txataren izena
* @param Erabiltzaileak txatean parte hartzen duten erabiltzaileak
*/
public Chat CreateChat(String Izena,ArrayList<User> Erabiltzaileak) ;

/**
* Metodo honek erabiltzaile bat lortzen du datu basetik
*/
public ArrayList<User> getUsers();

/**
* Metodo honek txateatzeko taldeak sortzen ditu
* @param adminUserName taldeko administratzailea, taldea sortu duena
* @param users erabiltzaile normalak
* @param gName taldearen izena
*
* @throws GroupAlreadyCreated taldea dagoeneko sortuta badago
* @throws NotEnoughUsers taldea sortzeko nahiko erabiltzaile ez badaude(2 edo
gutxiago)
*
*/
public User createGroup(String adminUserName, ArrayList<String> users,
String gName)throws NotEnoughUsers,GroupAlreadyCreated;

/**
* Metodo honek erabiltzaile bat gehitzen du taldera
* @param group taldea
* @param userName erabiltzailearen izena
*
* @throws UserAlreadyInTheGroup erabiltzailea dagoeneko taldean badago
*
*/
public Group addUserToGroup(String userName, Group group)throws
UserAlreadyInTheGroup;

/**
* metodo honek erabiltzaile bat taldetik botatzen du
* @param userName erabiltzailearen izena
* @param group taldea
*/

```

```
public Group quickFromTheGroup(String userName, Group group);

/**
 * Metodo honek taldea borratzen du
 * @param u taldea borratuko duen erabiltzailea, administratzailea izan behar da
 * @param group taldea
 */
public User deleteGroup(User u, Group group);

/**
 * Metodo hau taldetik irteteko da
 * @param user taldetik aterako den erabiltzailea
 * @param group taldea
 */
public Group leaveGroup(User user, Group group);

/**
 * Metodo hau erabiltzaileak jarraitzeko erabiltzen da, hau da jarraitzaileak
jarraitutakoaren apostu berdinak egindo
 * ditu automatikoki eta apostuak egiteko diru porzentai bat adieraziko du.
 * @param userToFollow jarraitu nahi den erabiltzailea
 * @param userFollowing jarraitzailea
 * @param money jarraitutako pertsona horren apostuetan erabili nahi den dirua
 * @param percentageToBet jarraitutako pertsonaren apostuetako diru kantitateren
zein porzentai apostatu nahi den.
 *
 * @throws notFoundException erabiltzaileak nahiko diru ez badu
 * @throws invalidPercentage sartutako porzentaia gaizki badago(x>=1 edo x=<0)
 * @throws NonPositiveNum diru kantitateak negatiboak badira
 */
public NormalUser follow(String userToFollow, NormalUser userFollowing,
double money, double percentageToBet) throws NotEnoughMoneyException,
invalidPercentage, NonPositiveNum;

/**
 * Metodo hau erabiltzailea jrraitzeaz uzteko erabiltzen da
 * @param unfollowedUser jrraitzeaz utzi nahi den erabiltzailea
 * @param mainUser jarraitzailea
 */
public NormalUser unfollow(String unfollowedUser, NormalUser mainUser);
```

```
* @param user erabiltzailea
*/
public List<NormalUser> getNormalUsersRankings(NormalUser user);

/**
 * Metodo honek erabiltzaile batek jarraitzen dituen erabiltzaile lista erakusten du
 * @param user erabiltzailea
*/
public List<NormalUser> userFollows(NormalUser user);
```

## **Ondorioak**

Hasteko, proiektuaren inguruko nondik norakoak aipatuko ditugu. Proiektuaren garapen prozesua eta erabilitako metodologia oso egokiak iruditu zaizkigu. Lanketa iterazioaten banatzeak lana modu hobean antolatzeko aukera eman digu, edukiak pixkanaka landu bitartean. Hasieratik proiektu osoa aurkeztuzkero, arazo handiak izango genituen lana modu egoki batean antolatzeko, eta ondorioz garatzeko.

Halere, garapen prozesuan arazo nagusia klasean gaiak lantzeko erritmoa izan da. Hau da, diseinuaren lanketa (sekuentzi diagramena batez ere) 3. iterazioaren aurretik landu ez izanak eragin negatiboa izan du, aurretik egindako sekuentzi diagramak ez baitziten zuzenak, eta ondorioz, horiek zuzentzen denbora dexente pasatu behar izan dugu azken iterazio honetan. Hasieratik sekuentzi diagramen inguruan lanketa sakonagoa eginik ez genituen horrenbeste aldaketa egin beharko. Halere, ulertzen dugu klasean lanketa modu honetara egiteak, kontzeptuak barneratzea errazten duela, diseinuko fase guztiak hasieran landuzkero ez baikenituen horren ondo barneratuko.

Hori dela eta, egokia iruditzen zaigu hurrengo urteetarako sekuentzi diagramen lanketa txiki bat hasieratik egitea, eta amaieran sakontzea.

Klaseko dinamikari dagokionez, klase teorikorik ez egotea oso onuragarria iruditu zaigu, irakasgaia praktikan oinarritzea oso lagungarria baita, batez ere, horren izaera praktikoa duen irakasgai batean. Teoria asko emanda ere, diseinua lantzeko modurik onena gure proiektuan aplikatzea da.

Amaitzeko, hainbat gomendio hurrengo urteetako ikasleentzako:

- Proiektua denborarekin hasi garatzen, momentu askotan zailtasunak sortzen baitira eta hauek konpontzea konplikatua izan daiteke denbora gutxirekin.
- Diseinua nola egin ondo ulertzea eta duen garrantzia ematea. Nahiz eta hasieran ez dirudien horren garrantzitsua (lehenengo iterazioan batez ere), garatu behar diren erabilpen kasuak konplikatu ahala argiago ikusten da zergatik den garrantzitsua diseinu egoki bat egitea, denbora asko aurrezten baitu.
- Proiektuaren hasieratik gauzak ondo eta txukun egiten saiatu, bestela amaieran infernu bat da txorakeriak zuzentzen egotea (salbuespenak...).

## **Bideoa**

[https://www.youtube.com/watch?v=kee2vwMKomk&ab\\_channel=AlexAmenabar](https://www.youtube.com/watch?v=kee2vwMKomk&ab_channel=AlexAmenabar)