



# SOFTWARE INGENIERITZA

*PROIEKTUA: 3.ITERAZIOA*

**Egileak:** Markel Etxabe, Mikel Martin eta Andrea Jaunarena

**Data:** 2022/05/21

<b>WEB ZERBITZUAK ETA INTERNALIZAZIOA</b>	<b>2</b>
<b>SARRERA</b>	<b>3</b>
<b>ESKAKIZUNEN BILKETA</b>	<b>4</b>
<b>DOMEINUAREN EREDUA</b>	<b>4</b>
<b>ERABILPEN KASUEN EREDUA</b>	<b>5</b>
User:	5
Admin:	6
Registered:	11
Registered eta Admin batera:	16
<b>DISEINUA</b>	<b>17</b>
SEKUENTZIA DIAGRAMAK	17
KLASE DIAGRAMA	31
<b>INPLEMENTAZIOA</b>	<b>33</b>
Metodoak:	33
<b>ONDORIOAK</b>	<b>44</b>
<b>BIDEROAREN URL-A</b>	<b>45</b>
<b>KODEAREN URL-A</b>	<b>45</b>



## WEB ZERBITZUAK ETA INTERNALIZAZIOA

Bai, biak inplementatu ditugu.

### SARRERA

Proiektu honen helburua, apustu-etxe baten portaera edukiko duen java proiektua sortzea izan da. Gure kasuan, hainbat funtzionalitate inplementatu dizkiogu batez ere erabiltzaile erregistratuentzat. Apustuen, gertaeren, erabiltzaileen eta klase gehigarrien informazio guztia datu base bertan gordetzen da. Azpimarratzekoa da bi erabiltzaile mota daudela, erregistratua eta administratzaileaa; biek baimen eta erabilpen kasu ezberdinak edukiko dituzte.

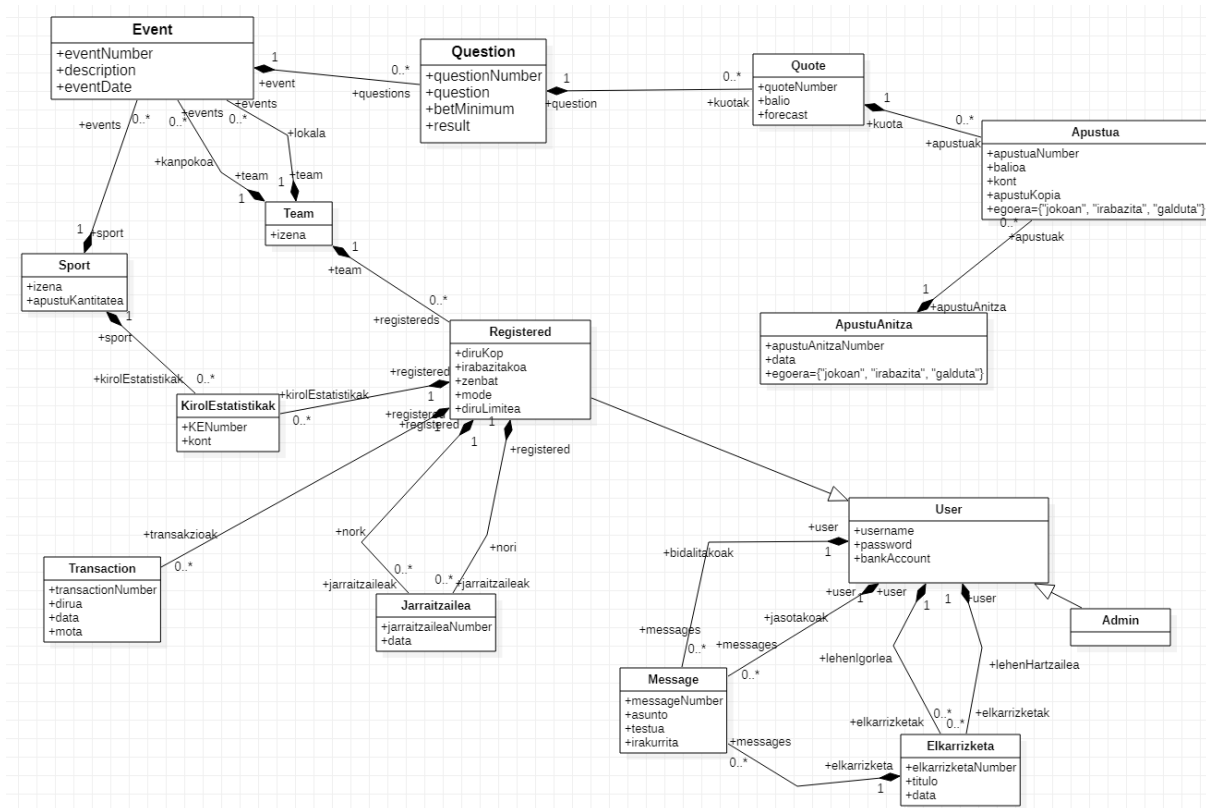
Administratzailearen funtzionalitateak aztertzen baditugu; gertaerak, galderak eta kuotak sortu edo ezabatzeko aukera ematen zaio. Horretaz gain, administratzailea izango da gertaeren gaderen emaitzak jartzeaz arduratuko dena. Hala ere, aukera batzuk kenduko zaizkio, adibidez, amaitu ez den gertaera baten emaitzak jartzea hau ez bada amaitu. Horrela, etikoki zuzenak kontsideratzen ez diren erabakiak artzea galaraziko zaio.

Erregistratutako erabiltzailea izango litzake funtzionalitate gehienak erabil ditzakeen erabiltzaile mota. Login eta erregistratze sistema bat sortu da bezeroak zerbitzuan identifikatzeko. Gainera, gertaeren, galderen eta kouten informazio osoa ikus dezakete. Erabilera garrantzitsuena apustu egiteko aukera izango litzake; hau, anitza izan daiteke erabiltzaileak horrela nahi badu, hau da, kuota bat baino gehiago kontuan hartu ditzake apustua sortzeko. Apustuei dagokionez, hauek ezabatzeko aukera edukiko dute eta mugimendu guztiak ikusteko. Gehigarri bezala, erregistratuak beraien artean jarraitu daitezke eta horrela, jarraitutako erabiltzaileak egindako apustuak automatikoki egin. Erabilpen kasu berezi bezala apustu gomendazio sistema bat prestatu da, kirol poularra, zure gustukoena eta talde gustukoenaren gertaerak erakutsiko dituen.

Erabilzaileak komunikatzeko mezu sistema bat sortu da, non hainbat elkarrizketa eduki daitezkeen beste erabiltzaile ezberdinekin. Esan beharra da, galderak ikusteko erabilpen kasurako ez dela beharrezko erregistratua egotea.

# ESKAKIZUNEN BILKETA

## 1. DOMEINUAREN EREDUA



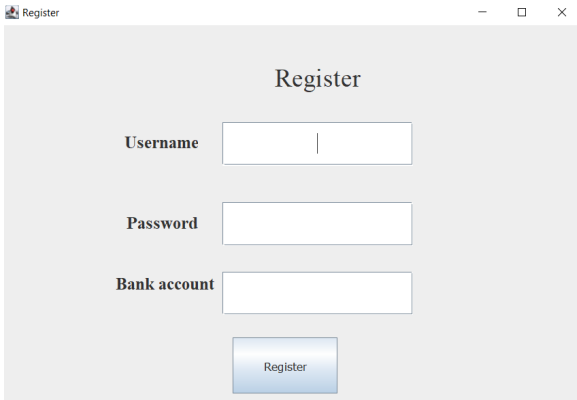
- Gertaera batek galdera asko izan ditzake. Gertaera batek kirol bat dauka. Gertaera batek bi talde dauzka: kanpokoa eta lokala.
- Galdera bat gertaera bat dauka eta kuoten lista bat.
- Kuota batek apustu asko izan ditzake eta galdera bat.
- Apustu batek kuota bat dauka eta apustu anitza bat.
- Apustu Anitza batek apustuen lista bat dauka eta erabiltzaile bat dauka.
- Registered batek talde bat izan dezake, apustu anitz asko egin ditzake, transakzio asko izan ditzake, jarraitzaileen listetan egon daitez (bai nork eta bai nori listetan), kirol estatistika asko izan ditzake eta user klasearen azpi klasea da.
- Transaction batek registered bat dauka.
- Jarraitzailea bi registered ditu.
- Team batek registered asko izan ditzake.
- KirolEstatistikak registered bat du.
- Sport batek gertaera asko izan ditzake eta kirol estatistika asko ere.
- User elkarrizketa asko izan ditzake eta mezu asko ere.
- Admin user klasearen azpiklasea da.
- Mesage bi erabiltzaile izan ditzake.
- Elkarrizketa batek mezu asko izan ditzake eta bi erabiltzaile (hartzailea eta igorlea).

## 2. ERABILPEN KASUEN EREDUA

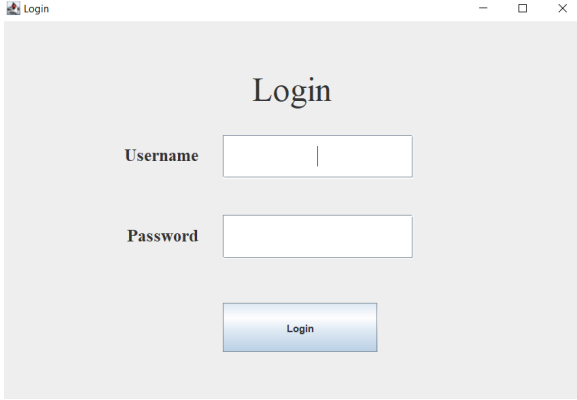
Erabilpen kasuei dagokienez, hiru aktore ditugu gure proiektuan (user, registered eta admin). Userrek egiten dituen erabilpen kasuak registered eta adminak egin dezakete.

### User:

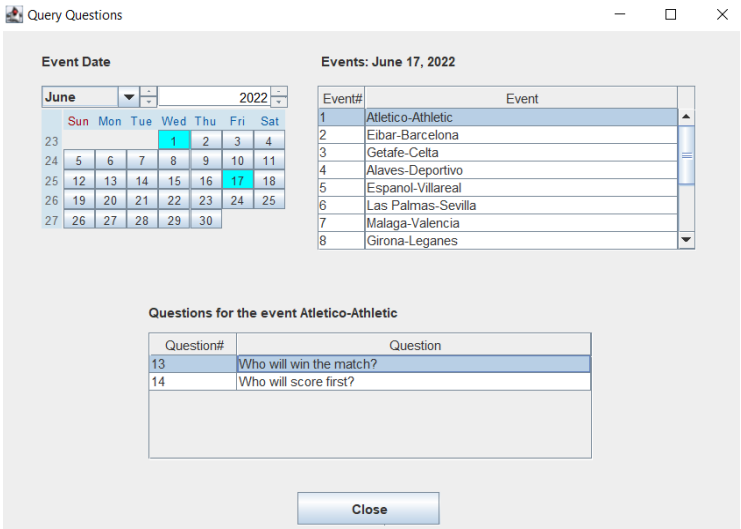
- Erregistratu:

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> shows a window with textbox for <b>username</b>, <b>password</b> and <b>bank account</b></li> <li>2. <i>User</i> enters the data required</li> <li>3. <i>System</i> saves the <b>user</b> data</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. The username is already used. The user cannot be registered.</li> </ol>	

- Login:

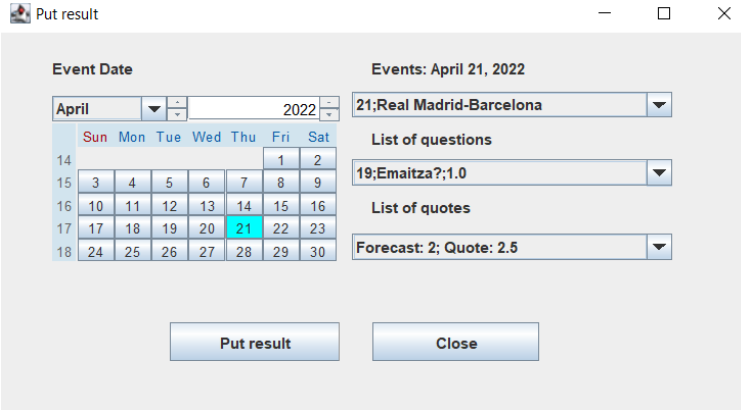
Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> shows a window with textbox for <b>username</b> and <b>password</b></li> <li>2. <i>User</i> enters the data required</li> <li>3. <i>System</i> check if the <b>user</b> data is correct and logs in his account</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. The username or the password are not correct. The user cannot be logged.</li> </ol>	

- Query Questions:

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> shows a Calendar where days with events are highlighted</li> <li>2. <i>User</i> selects a <b>Date</b> in a Calendar</li> <li>3. <i>System</i> displays the <b>events</b> of this <b>date</b></li> <li>4. <i>User</i> selects an <b>event</b></li> <li>5. <i>System</i> displays the <b>questions</b> associated to the selected <b>event</b></li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. There are no events on this date. Events cannot be shown.</li> <li>2. There are no questions associated to the event. Questions cannot be shown.</li> </ol>	

### Admin:

- Emaizta ipini:

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> shows a Calendar where days with events are highlighted</li> <li>2. <i>Admin</i> selects a <b>Date</b> in a Calendar</li> <li>3. <i>System</i> displays the <b>events</b> of this <b>date</b></li> <li>4. <i>Admin</i> selects an <b>event</b></li> <li>5. <i>System</i> displays the <b>questions</b> associated to the selected <b>event</b></li> <li>6. <i>Admin</i> selects a <b>question</b></li> <li>7. <i>System</i> displays the <b>quotes</b> associated to the selected <b>question</b></li> <li>8. <i>Admin</i> selects a <b>quote</b> and clicks emaitzaklpini.</li> <li>9. <i>System</i> gives the forecast of the selected quote to its questions result.</li> <li>10. <i>System</i> gives money to the users that have bet to the winner forecast.</li> <li>11. <i>System</i> creates a <b>transaction</b> and delete the <b>bet</b></li> </ol>	

### ## Alternative flow

1. There are no events on this date. Quote cannot be added.
2. There are no questions associated to the event. Quote cannot be added.
3. There are no quotes associated to the event. Result cannot be put.
4. There are no bets associated to the quote. Bets cannot be deleted.

### ● Gertaera ezabatu

#### Gertaera Fluxua

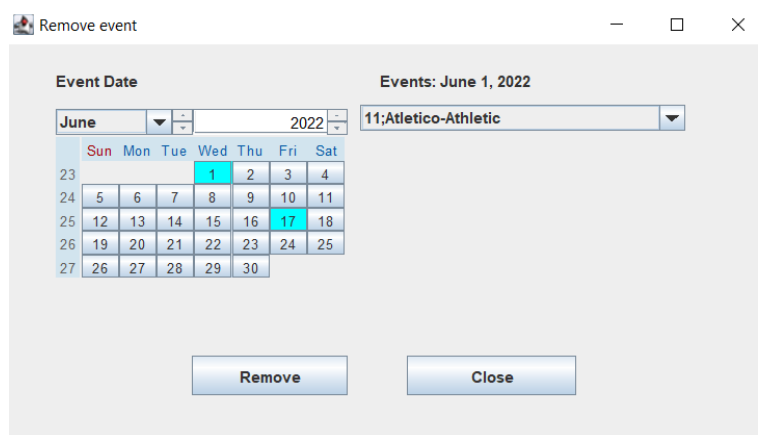
### ## Basic Flow

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** and clicks in delete button.
5. *System* deletes from user the bets associated to the selected event and gives their money back.
6. *System* delete the selected event and their associated questions, quotes and bets.

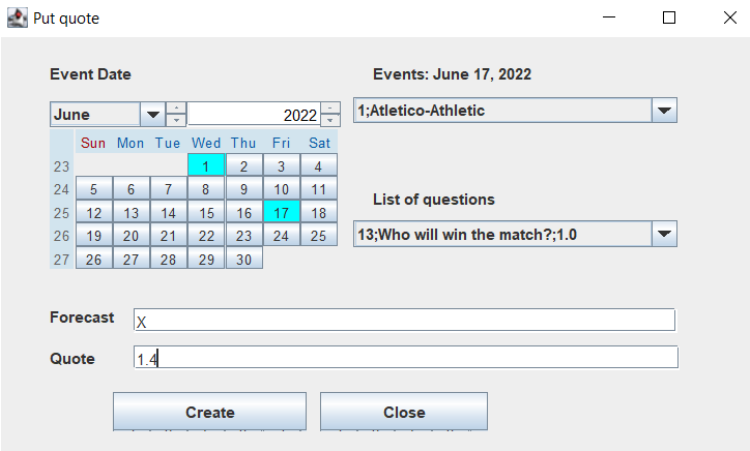
### ## Alternative flow

1. There are no events to be deleted.
2. The event has finished but results haven't been decided yet. The event cannot be deleted.
3. The event has been finished and the results have been decided, so there is no money to give back.

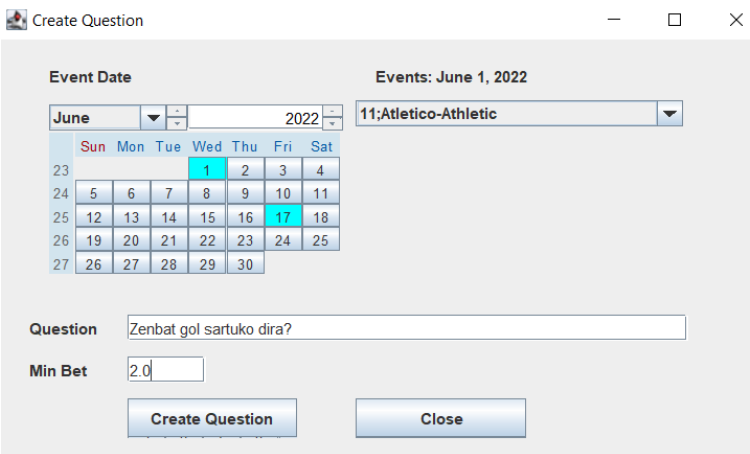
#### GUI



- Kuotak ipini:

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> shows a Calendar where days with events are highlighted</li> <li>2. <i>Admin</i> selects a <b>Date</b> in a Calendar</li> <li>3. <i>System</i> displays the <b>events</b> of this <b>date</b></li> <li>4. <i>Admin</i> selects an <b>event</b></li> <li>5. <i>System</i> displays the <b>questions</b> associated to the selected <b>event</b></li> <li>6. <i>Admin</i> selects a <b>question</b></li> <li>7. <i>Admin</i> introduces a <b>forecast</b> and <b>value</b> which represent a <b>quote</b></li> <li>8. <i>System</i> adds the new <b>quote</b> (the <b>forecast</b> and the <b>value</b>) to the selected <b>question</b></li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. There are no events on this date. Quote cannot be added.</li> <li>2. There are no questions associated to the event. Quote cannot be added.</li> <li>3. The quote field is empty or it is not a positive number. Quote cannot be added.</li> <li>4. The quote is already created for that question. Quote cannot be added.</li> </ol>	

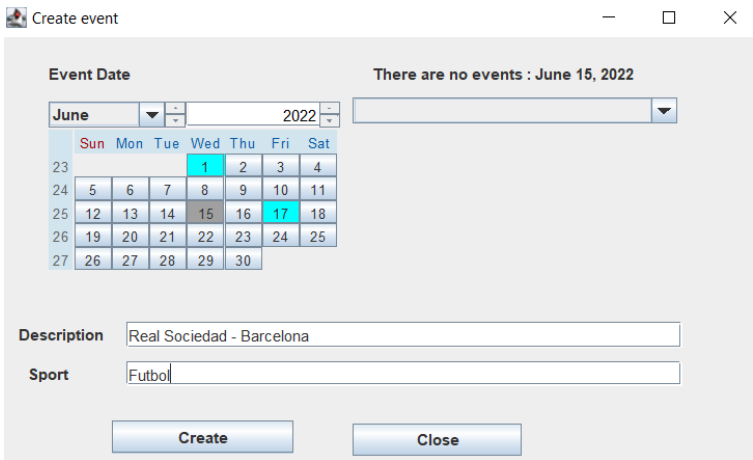
- Create question:

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> shows a Calendar where days with events are highlighted</li> <li>2. <i>Admin</i> selects a <b>Date</b> in a Calendar</li> <li>3. <i>System</i> displays the <b>events</b> of this <b>date</b></li> <li>4. <i>Admin</i> selects an <b>event</b></li> <li>5. <i>Admin</i> introduces a <b>question</b> and a minimum <b>betting price</b></li> <li>6. <i>System</i> adds the new <b>question</b> with a minimum <b>betting price</b> to the selected <b>event</b></li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. There are no events on this date. Question cannot be added.</li> </ol>	

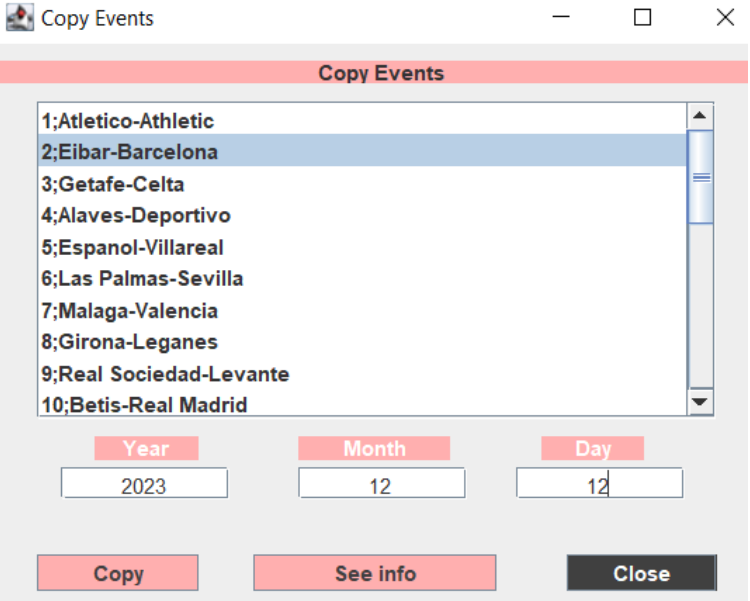


2. The question field is empty. Question cannot be added.
3. The minimum betting price is empty or it is not a number. Question cannot be added.
4. Event date has already finished (event day is before current day). Question cannot be added.

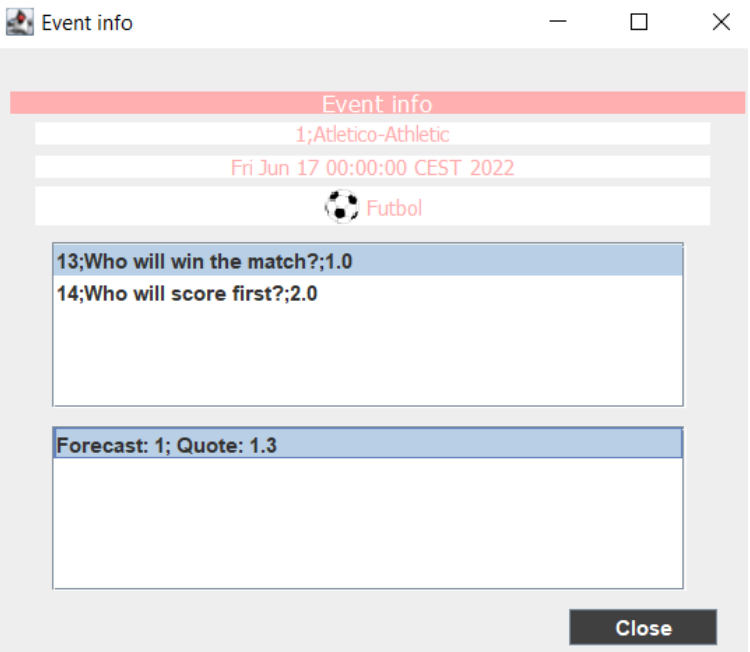
● Gertaerak sortu:

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> shows a Calendar where days with events are highlighted</li> <li>2. <i>Admin</i> selects a <b>Date</b> in a Calendar</li> <li>3. <i>System</i> displays the <b>events</b> on this <b>**date**</b></li> <li>4. <i>Admin</i> introduces an <b>event</b></li> <li>5. <i>System</i> adds the new <b>event</b> to the selected <b>date</b></li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. There are no events on this date. Event cannot be added.</li> <li>2. The event field is empty. Event cannot be added.</li> <li>3. Date has already passed (event day is before current day). Event cannot be added.</li> <li>4. The event is already created. Event cannot be added.</li> </ol>	

- **Gertaerak kopiaitu:**


Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> <b>gertaera</b> guztiak erakusten ditu.</li> <li>2. <i>Registered</i> <b>gertaera</b> bat aukeratzen du.</li> <li>3. <i>Registered</i> <b>data</b> egoki bat sartzen du gertaera hori kopiaitzeko.</li> <li>4. <i>Registered</i> kopiaitu botoiari ematen dio.</li> <li>5. <i>System</i> <b>gertaeren</b> lista eguneratzen du; beraz, kopiaitu den <b>gertaera</b> berria bertan agertzen da.</li> <li>6. <i>Registered</i> <b>gertaeraren</b> informazioa ikusi nahi badu extends GertaeraInfo.</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. Hilabetea ez dago 0 eta 12 tartean. Ezin da gertaera kopiaitu.</li> <li>2. Eguna ez dago 0 eta 31 tartean. Ezin da gertaera kopiaitu.</li> <li>3. Data gaur baino lehen da. Ezin da gertaera kopiaitu.</li> <li>4. Gertaera jadanik existitzen da edo/ eta kirola ez da existitzen. Gertaera ezin da kopiaitu.</li> </ol>	

- **Gertaera info:**

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>Registered</i> lehengo GUI-tik ikusi nahi duen <b>gertaera</b> aukeratzen du.</li> <li>2. <i>System</i> aukeratutako <b>gertaera</b> erakusten du.</li> <li>3. <i>System</i> <b>gertaera</b> horri lotutako galderak erakusten ditu.</li> <li>4. <i>Registered</i> <b>galdera</b> bat erakusten du.</li> <li>5. <i>System</i> <b>galdera</b> horri lotutako kuotak erakusten ditu.</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. Gertaera horrek ez ditu galderarik. Ez ditu galderak erakusten.</li> <li>2. Aukeratutako galdera ez ditu kuotik. Ez ditu kuotak erakusten.</li> </ol>	

## Registered:


- Dirua sartu

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> shows a field to enter a <b>value</b>.</li> <li>2. <i>Registered</i> fields the gap with an amount of money.</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. The amount of money needs to be a positive number.</li> </ol>	

- Mugimenduak ikusi

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> shows the <b>transactions</b> of the <b>user</b>.</li> <li>2. <i>System</i> show the <b>money</b> of the <b>user</b>.</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. The user hasn't done any transactions.</li> </ol>	

- Apustua ezabatu

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> shows a list with the bets of the user.</li> <li>2. <i>Registered</i> selects a <b>bet</b> to be deleted.</li> <li>3. <i>System</i> generates a <b>Transaction</b></li> <li>4. <i>System</i> gives the money back to the user.</li> <li>5. <i>System</i> deletes the bet from the user and the database.</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. There are no bets to be deleted.</li> </ol>	

- Apustu anitza egin

## Gertaera Fluxua

## GUI

### ## Basic Flow


1. *System* shows a Calendar where days with events are highlighted
2. *Registered* selects a **Date** in a Calendar
3. *System* displays the **events** of this **date**
4. *Registered* selects an **event**
5. *System* displays the **questions** associated to the selected **event**
6. *Registered* selects a **question**
7. *System* displays the **quotes** associated to the selected **question**
8. *Registered* selects one or more than one **quote** and puts a value to the bet.
9. *System* creates a **Transaction** and a **Apustua** with the given values.
10. *System* add the **transaction** and the bet to the user.

### ## Alternative flow

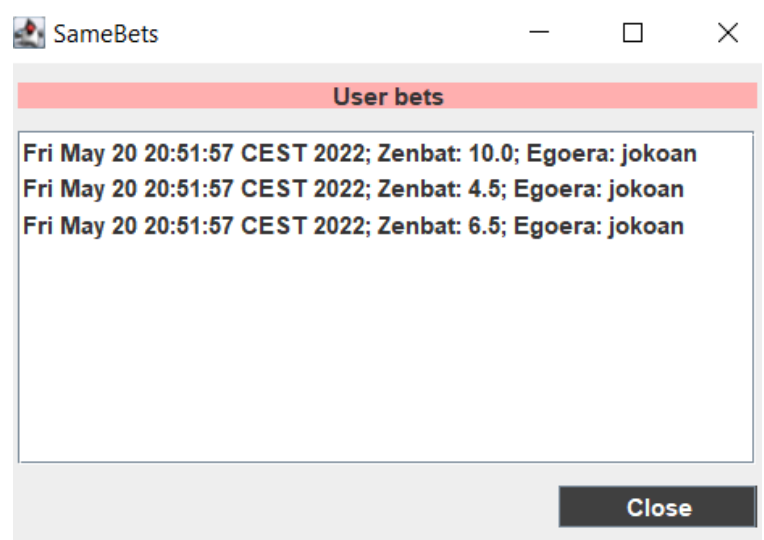
1. There are no events on this date. Quote cannot be added.
2. There are no questions associated to the event. Quote cannot be added.
3. There are no quotes associated to the event. Result cannot be put.
4. The value must be a positive number.
5. The value must be higher than the min bet.
6. The user must have more money than the bet value.
7. The bet cannot be done if the date has already expired.

The screenshot shows a web application titled 'Bet'. It features a calendar for June 2022 on the left. The main area displays the selected date, 'Events: June 17, 2022', and a list of events. The first event is '1;Atletico-Athletic'. Below this is a 'List of questi...' section with a dropdown menu showing '13:Who will win the match?:1.0'. Underneath is a 'List of quotes' section with a dropdown menu showing 'Forecast: 1; Quote: 1.3'. At the bottom, there is an 'Amount of money' input field with a value of '5'. Three buttons are at the bottom: 'Bet' (pink), 'Finish bet' (orange), and 'Close' (black). On the right side, there is a 'Your bet' panel showing 'Forecast: 2; Quote: 100.0' and 'Forecast: 1; Quote: 1.3'.

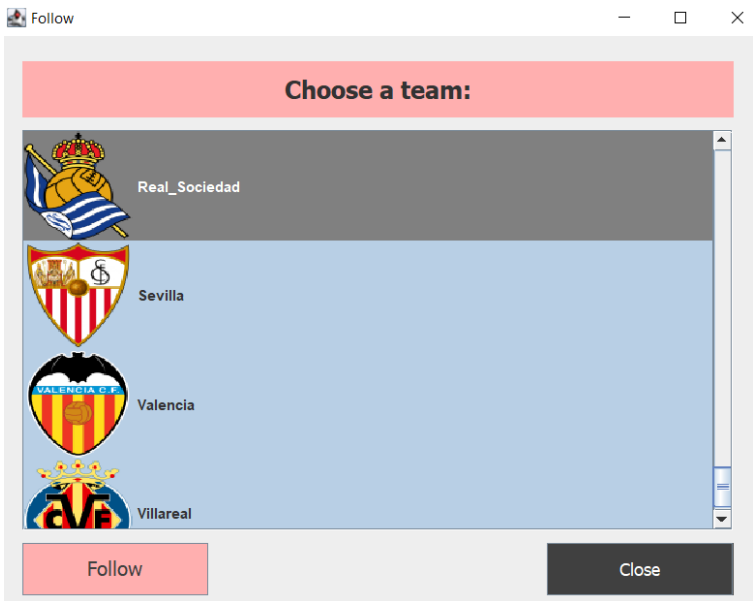
- Erabiltzailea jarraitu

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> shows a list with all the users <b>registered</b> in order of the money they win</li> <li>2. <i>Registered</i> enters the max <b>bet</b> amount he want and select a register <b>user</b> to follow.</li> <li>3. if <i>Registered</i> want to see the <b>bets</b> of the other user clicks see bet and extends <b>ApustuakIkusi</b></li> <li>4. <i>Registered</i> click the follow button</li> <li>5. <i>System</i> store in the database that <b>registered</b> is following the user with the max bet</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. You already follow that user. You cannot follow the user twice.</li> <li>2. You can't follow yourself.</li> <li>3. There is no max bet entered. You cannot follow the user.</li> </ol>	

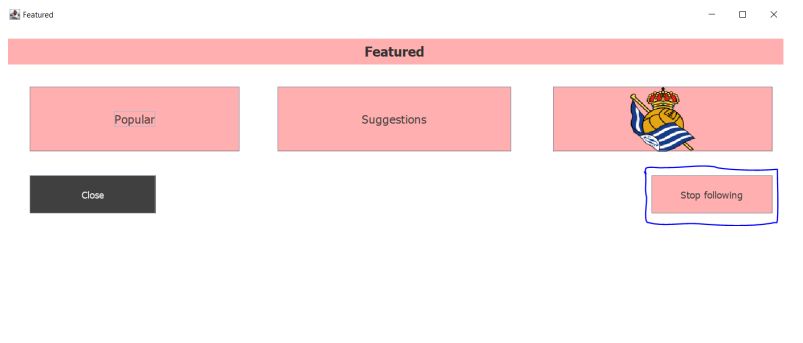
- Apustuak ikusi

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> shows a list with all the <b>bets</b> of the <b>user</b></li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. The user has not bet. No bets showed.</li> </ol>	

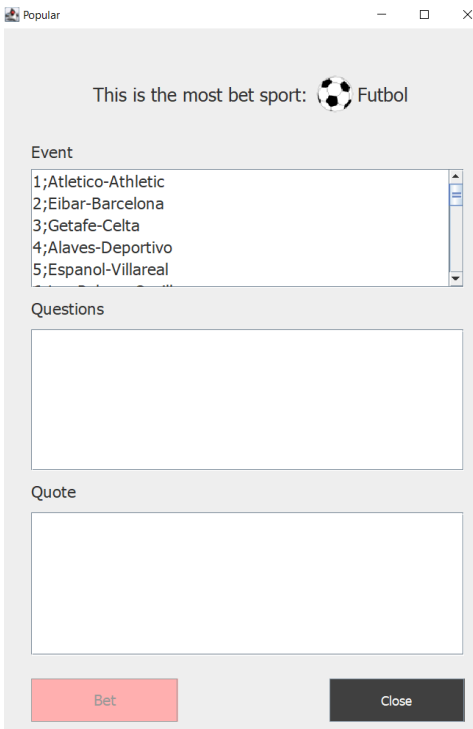
- Gustokoen taldea aukeratu

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> shows a list with all the teams</li> <li>2. <i>Registered</i> selects a <b>team</b> and click the button to follow.</li> <li>3. <i>System</i> store in the database that registered is following that <b>team</b>.</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. There are no teams created. Registered cannot follow a team.</li> </ol>	

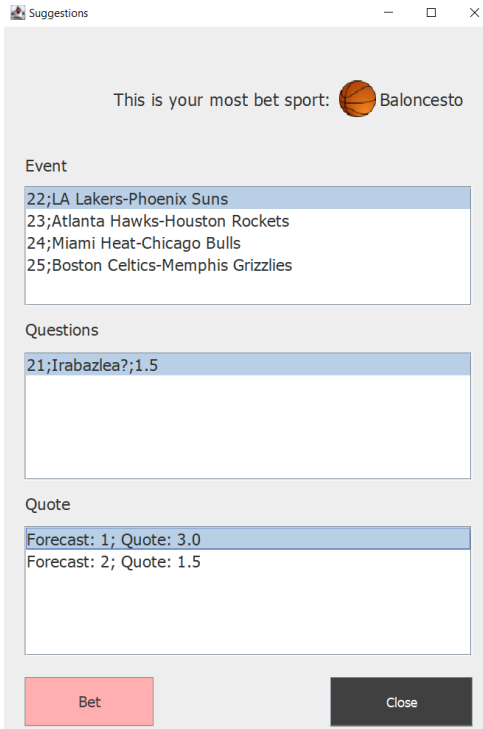
- Gustokoen taldea ezabatu

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>Registered</i> <b>taldea</b> kentzeko botoia sakatzen du.</li> <li>2. <i>System</i> registered-aren gustoko <b>taldea</b> ezabatzen dio.</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. Registered-ak ez du gustoko talderik.</li> </ol>	

- Kirol popularrenaren informazioa

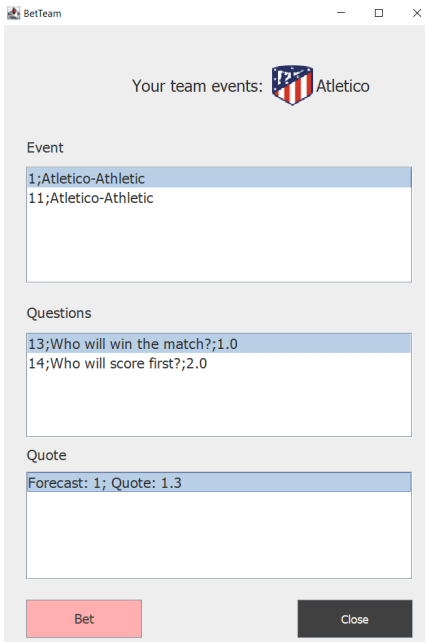
Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> zein den <b>kirol</b> popularrena kalkulatzeko du.</li> <li>2. <i>System</i> hari lortutako <b>gertaerak</b> inprimatzeko dira pantailan.</li> <li>3. <i>Registered</i> gertaera bat aukeratzeko du.</li> <li>4. <i>System</i> gertaera horri lotutako <b>galderak</b> erakusteko ditu.</li> <li>5. <i>Registered</i> <b>galdera</b> bat aukeratzeko du.</li> <li>6. <i>System</i> <b>galdera</b> horri lotutako <b>kuotak</b> erakusteko ditu.</li> <li>7. <i>Registered</i> <b>kuota</b> bat aukeratzeko du.</li> <li>8. <i>Registered</i> nahi badu <b>apustua</b> egin dezake, extends ApustuaEginGUI.</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. Ez daude kirol horren gertaerak.</li> <li>2. Gertaera batek ez dauka galderarik.</li> <li>3. Galdera batek ez dauka kuotarik.</li> </ol>	

- Kirol sugerentzia pertsonalaren informazioa

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> zein den erabiltzaile horren gehien apostatu duen <b>kirola</b> kalkulatzeko du.</li> <li>2. <i>System</i> hari lortutako gertaerak inprimatzeko dira pantailan.</li> <li>3. <i>Registered</i> <b>gertaera</b> bat aukeratzeko du.</li> <li>4. <i>System</i> <b>gertaera</b> horri lotutako <b>galderak</b> erakusteko ditu.</li> <li>5. <i>Registered</i> <b>galdera</b> bat aukeratzeko du.</li> <li>6. <i>System</i> <b>galdera</b> horri lotutako kuotak erakusteko ditu.</li> <li>7. <i>*Registered*</i> <b>kuota</b> bat aukeratzeko du.</li> <li>8. <i>*Registered*</i> nahi badu apustua egin dezake, extends ApustuaEginGUI.</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. Ez daude kirol horren gertaerak.</li> <li>2. Gertaera batek ez dauka galderarik.</li> </ol>	

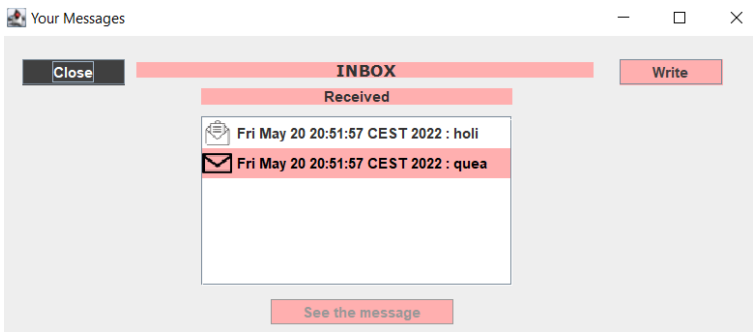
3. Galdera batek ez dauzka kuotarik.

- Talde gustokoenaren informazioa

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> gustoko <b>taldeari</b> lotutako gertaerak inprimatzen dira pantailan.</li> <li>2. <i>Registered</i> <b>gertaera</b> bat aukeratzen du.</li> <li>3. <i>System</i> gertaera horri lotutako galderak erakusten ditu.</li> <li>4. <i>Registered</i> <b>galdera</b> bat aukeratzen du.</li> <li>5. <i>System</i> galdera horri lotutako <b>kuotak</b> erakusten ditu.</li> <li>6. <i>Registered</i> <b>kuota</b> bat aukeratzen du.</li> <li>7. <i>Registered</i> nahi badu <b>apustua</b> egin dezake, extends ApustuaEginGUI.</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. Ez daude talde horren gertaerak.</li> <li>2. Gertaera batek ez dauka galderarik.</li> <li>3. Galdera batek ez dauzka kuotarik.</li> </ol>	

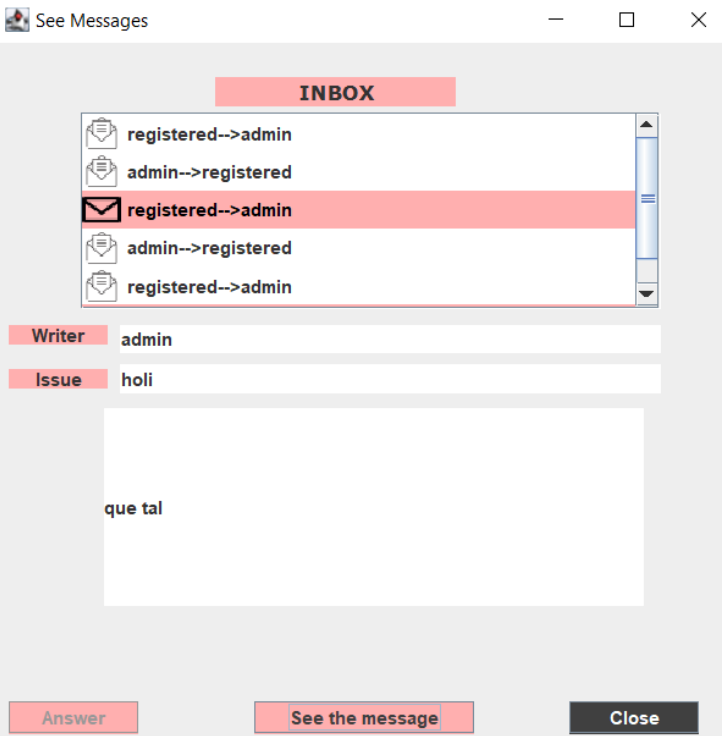
Registered eta Admin batera:

- Sarrera ontzia ikusi

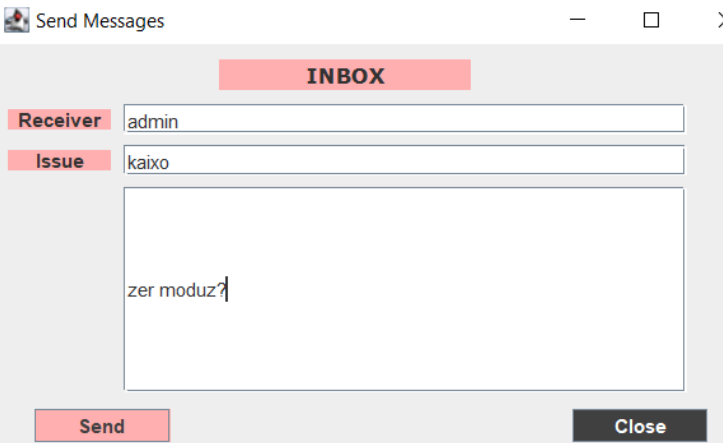
Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> erabiltzaile horren <b>elkarriketak</b> erakusten ditu.</li> <li>2. <i>Registered</i> <b>mezu</b> berri bat idatzi nahi badu extends MezuakBidali.</li> <li>3. <i>Registered</i> elkarriketa baten <b>mezuak</b> ikusi nahi baditu extends MezuakIkusi.</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. Ez ditu elkarriketarik. Ezin ditu elkarriketa baten mezuak ikusi.</li> </ol>	



- Mezuak ikusi

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>System</i> igorleak aukeratu duen elkarrizketa pantailan agertzen da bere mezuekin.</li> <li>2. <i>Registered</i> <b>mezu</b> bat aukeratzen du.</li> <li>3. <i>Registered</i> aukeratu dezake <b>mezua</b> ikustea edo erantzutea.</li> <li>4. <i>System</i> erabiltzaileak <b>mezua</b> ikusten badu, mezu hori ikusita bezala markatzen da.</li> <li>5. <i>Registered</i> <b>mezua</b> erantzun dezake, extends MEZUA BIDALI GUI.</li> </ol>	

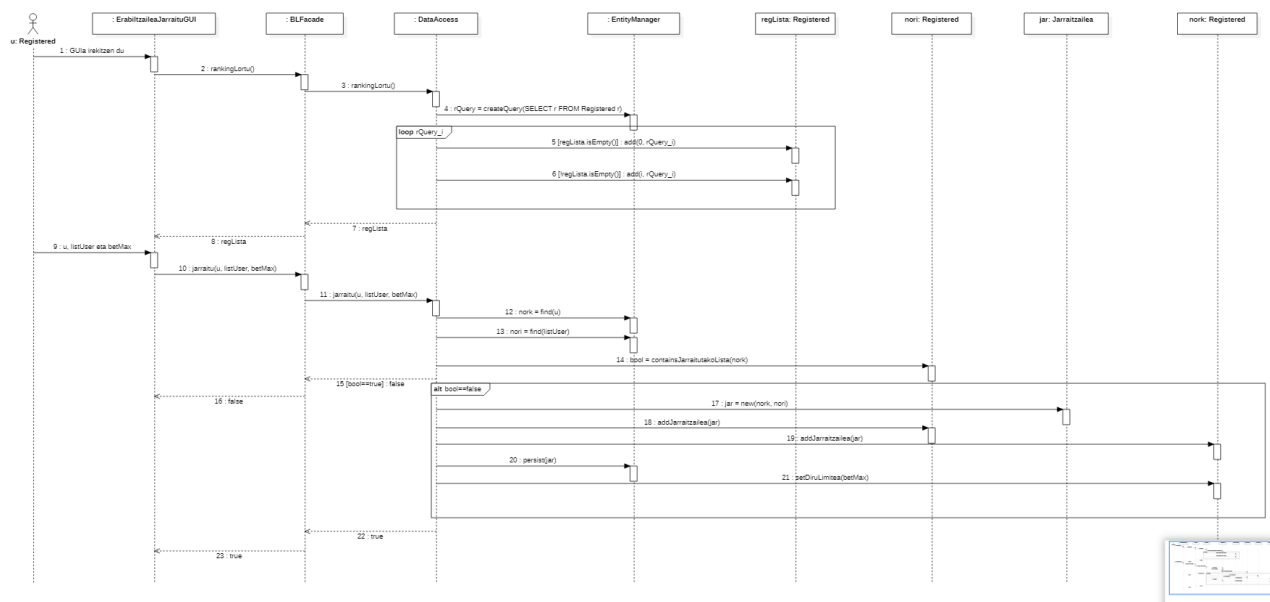
- Mezuak bidali

Gertaera Fluxua	GUI
<p>## Basic Flow</p> <ol style="list-style-type: none"> <li>1. <i>Registered</i> hartzailea, <b>elkarrizketaren</b> burua eta mezuaren testua idazten du.</li> <li>2. <i>Registered</i> <b>mezua</b> bidali botoiari ematen dio.</li> <li>3. <i>Registered</i> sarrera ontzira bueltatu daiteke.</li> </ol> <p>## Alternative flow</p> <ol style="list-style-type: none"> <li>1. Hartzailea ez da existitzen. Mezua ezin da bidali.</li> <li>2. Hartzailea zu zara. Mezua ezin da bidali.</li> </ol>	

## DISEINUA

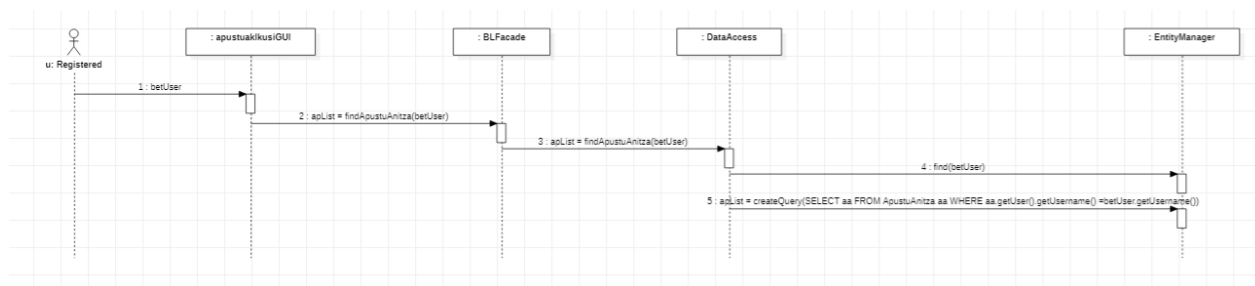
## SEKUENTZIA DIAGRAMAK

### ERABILTZAILE JARRAITU



Erabiltzailea GUI-an sartzen den momentuan, sistemak erabiltzaile guztiak bilatuko ditu datu basean eta ordenatutako lista batean bueltatuko ditu irabazitako diru kopuruaren arabera. Hau egiteko query bat egingo du eta gero loop batean ordenatuko ditu. Erabiltzaileak zeini jarraitu nahi dion aukeratu du eta apustatu nahi duen diru maximoa. Sistemak, erabiltzailea bilatuko du eta dagoeneko jarraituko beharreko erabiltzailea jarraitzen duen bilatuko du. Hala bada, abisua emango du eta ez du jarraituko. Bestela, jarraitzaile motako objektu bat sortuko du eta bi erabiltzaileak gehituko dizkio. Azkenik hau gorde eta diru limitea ezarriko dio.

### APUSTUAK IKUSI



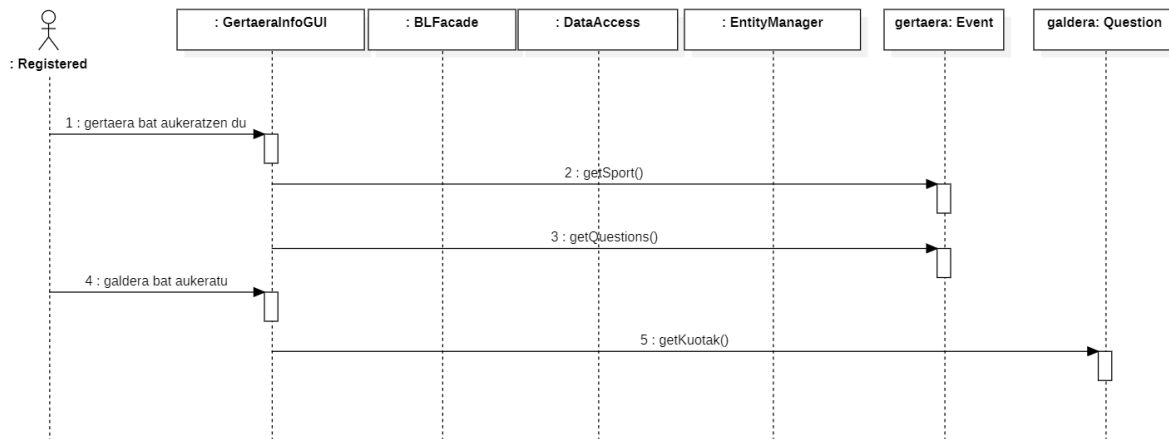
ApustuakIkusiGUI-an erabiltzailearekin findApustuAnitza metodoari dei egingo diogu. Guia BFacadera dei egingo dio eta BLFacade-ak DataAccess-era. DataAccess-ak erabiltzaile hau bilatu egingo du. Query bat egingo da erabiltzaile horrek dituen apustuak lortzeko. Lortzen den lista itzuli egingo da. Horrela GUI-an apustu lista hau ikusi ahal izango zen.



berriak sortu ditugu (new bidez). Question-ek duen listaraGehitu metodoaren bidez, kuota txertatzen diogu.

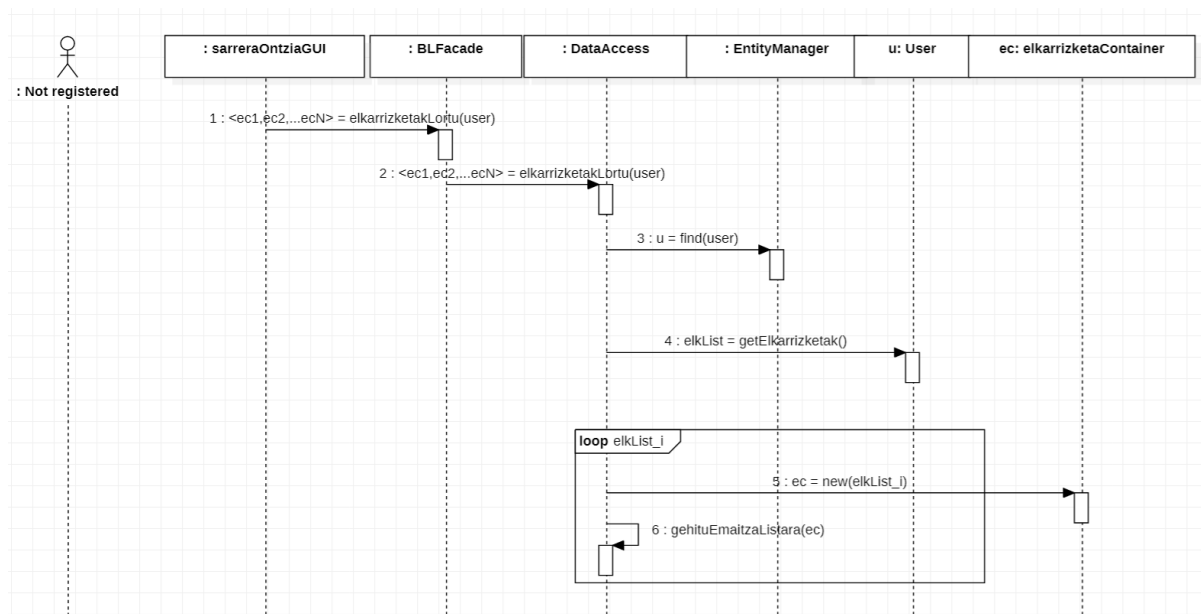
Bi loop-ak bukatzean DataAccess-ek boolearra bueltatuko dio BLFacaderi eta BLFacadek GUIari.

## GERTAERA INFO



Erabiltzaileak gertaera bat aukeratu ondoren, sistemak bere informazioa hartuko du eta galdera bat hartzean bere kuota.

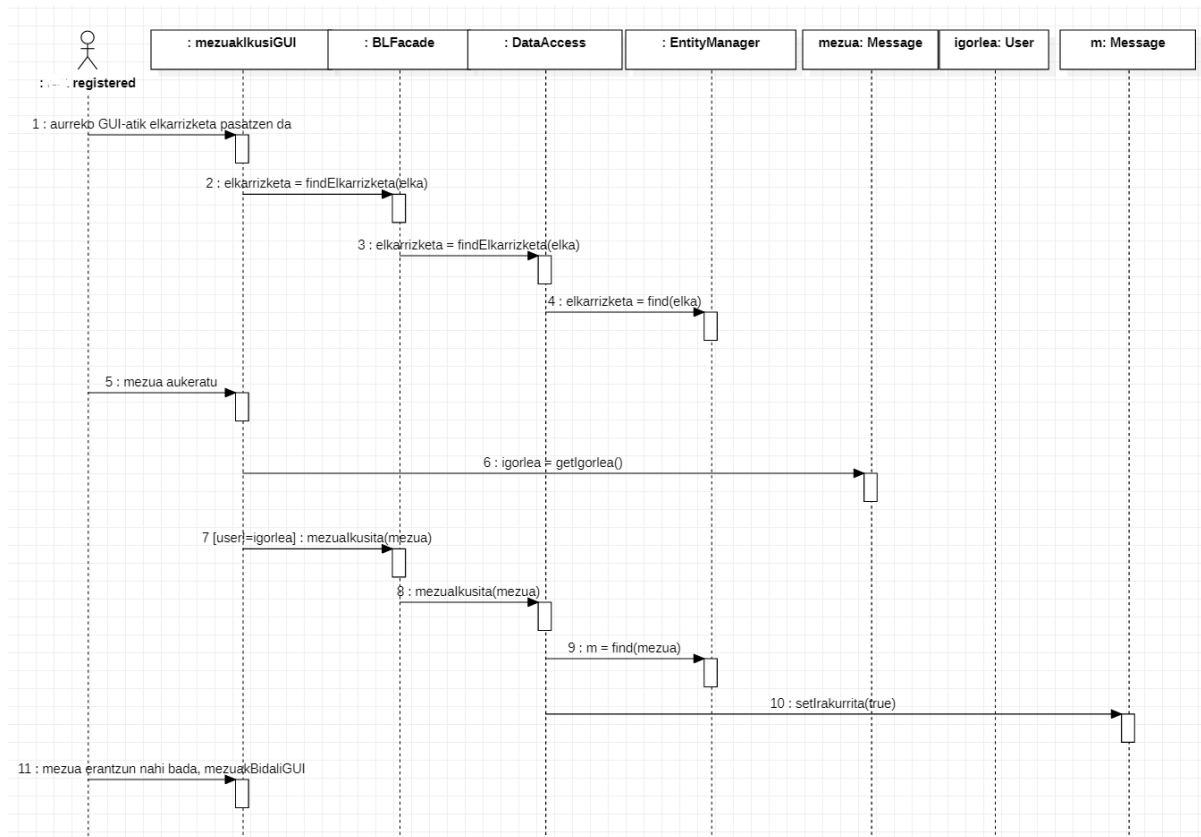
## SARRERA ONTZIA IKUSI



SarreraOntziaGUI izeneko GUI bat sortu egin dugu. Hoenk BLFacaderi dei egingo dio elkarriketakLortu metodoarekin. Metodoari erabiltzailea pasatuko zaio. BLFacade-ak DataAccess-ari dei egingo dio elkarriketakLortu(user) metodoarekin dei egingo dio. DataAccess-ak EntityManager-ari dei egingo dio erabiltzailea lortzeko. Erabiltzaileaz baliatuz erabiltzaile horrek dituen elkarriketak lortu egingo dira. Elkarriketa Container-rak

sortzen joango gara gero GUI-an web zerbitzuekin elkarriketak inprimatu ahal izateko. Azkenik lista hau bueltatu egingo da.

## MEZUAK IKUSI



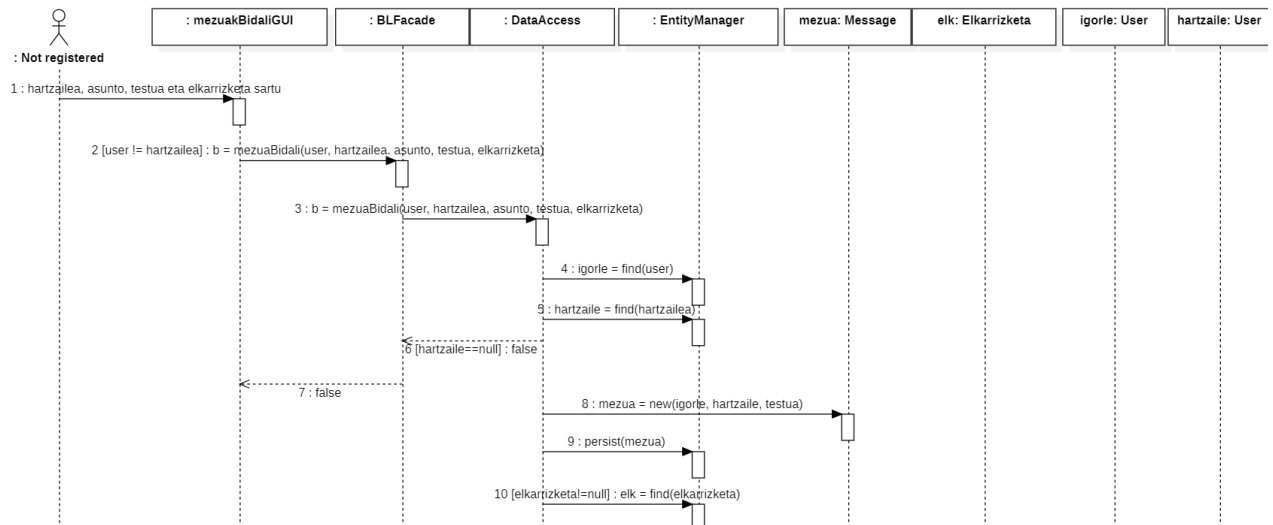
Aurreko GUI-tik pasatzen diogu elkarriketa bat mezuakIkusiGUIari. GUI horrek BLFacadeko findElkarriketa metodoa bidez dei egiten dio elkarriketa pasatuz. Metodoak datu baseko elkarriketa bueltatuko dio GUIari. BLFacadek berdina egiten du DataAccess-ekin. DataAccessek Entity managerreko find metodoa bidez elkarriketa bilatu eta bueltatzen du.

Orduan behin elkarriketa edukita, erabiltzaileak (admin edo registered) mezu bat aukeratzen du. GUIak getIgorlea() mezuko metodoa bidez mezuaren igorlea lortzen du.

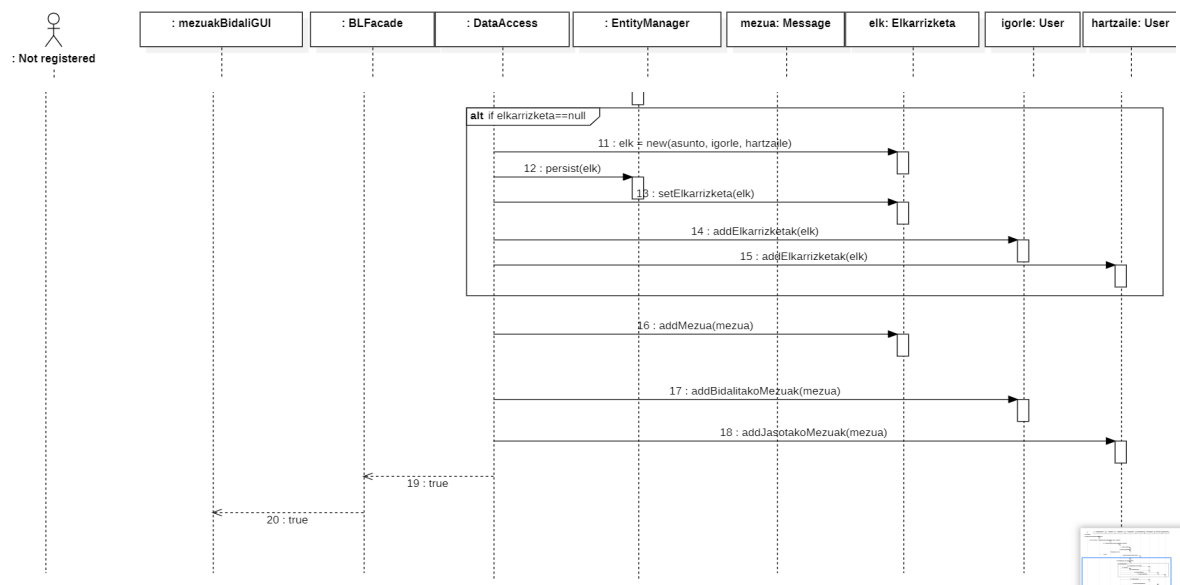
GUIak mezualkusita metodoari dei egiten dio mezua pasatuz (beti ere lehengo igorlea ez bada erabiltzailea). BLFacadek berdina egingo du DataAccessekin. DataAccessek lehen bezala mezua bilatuko du Entity Managerren dagoen find bidez. Behin mezua edukita setIrakurrita egiten dugu().

Erabilpen kasu honek mezuakBidali erabilpen kasua extends mezuak bidali nahi bada.

## MEZUAK BIDALI

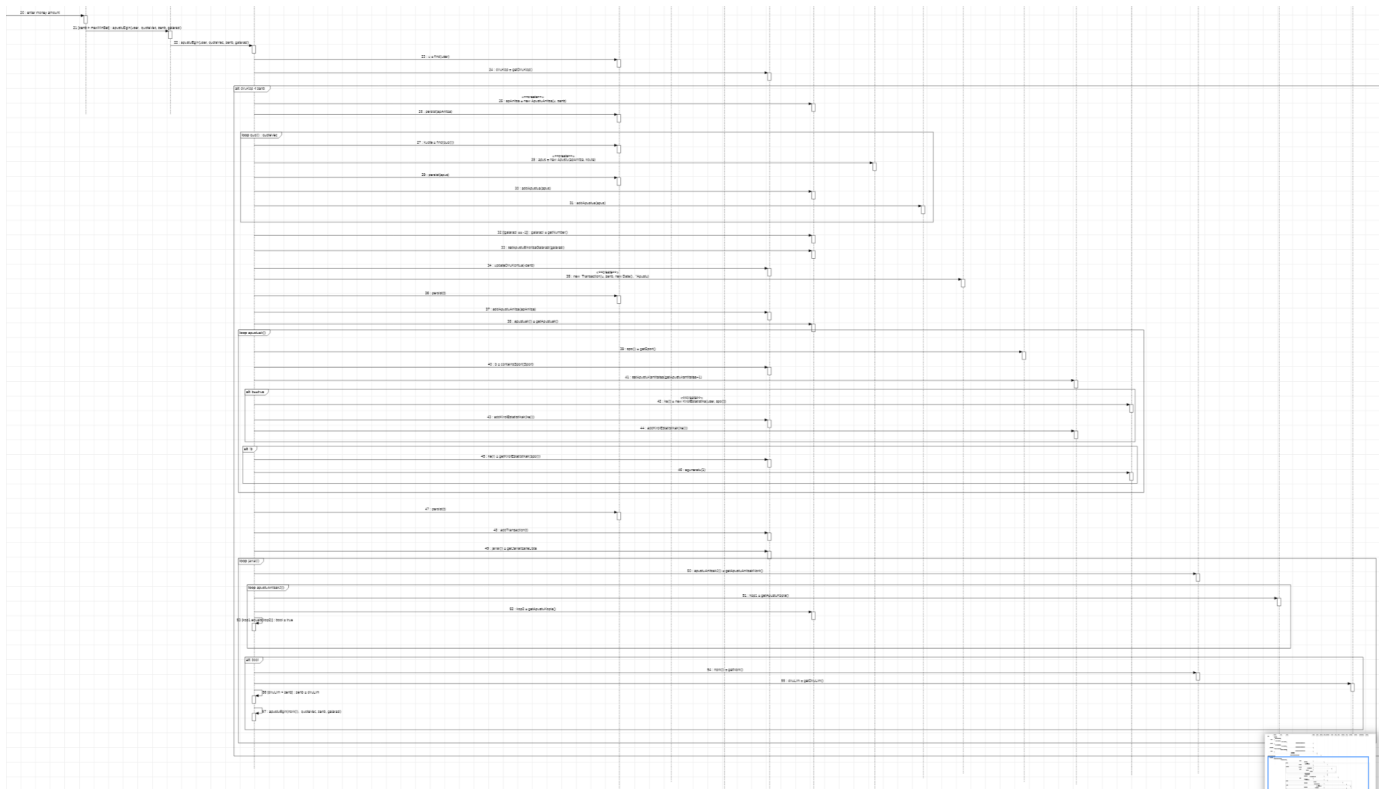


Erabiltzaileak, hartzailearen izena, testua, gaia eta elkarrizketa sartuko ditu. Erabiltzailea eta hartzailea berdinak ez direla frogatuko du sistemak, mezua bildali baino lehen. Gero, bi erabiltzaileak bilatu eta hartzaile existitzen dela frogatuko du. Ondoren, mezua sortu eta elkarrizketa bilatuko du.



Mezua guztiz berria bada, elkarrizketa berria sortu beharko da. Honi, mezua gehituko zaio eta baita ere hartzaileari.





Erabiltzaileak diru kopuru sartu egingo du apustu bat egiteko. `apustuaEgin(user, quoteVec, zenb, galarazi)` metodoarekin GUI-tik BLFacade-ra, BLFacade-tik DataAccess-era joango gara. Horrela DataAccess-ek EntityManager-rak dei egingo dio pasatu diogun erabiltzailea datu basean bilatzeko.

Erabiltzaile horrengana joango gara zenbat diru daukan jakiteko. Erabiltzaile horrek duen diru kantitatea apustatu nahi duen baino handiagoa bada; hau da, diru nahiko baldin badu apustua egiteko, programarekin jarraitu egingo dugu bestela ez.

Apustu anitza berri bat sortuko egingo dugu, erabiltzailearekin eta erabiltzaile horrek apustatu nahi duen diru kantitatearekin. Apustu anitz hau datu basean persist egingo dugu.

Parametro bezala pasatako kuota bekoerearen for bat egingo dugu. Apustu berriak sortzen joango gara, kuota hori jarritz eta apustu anitza ere esleituz. Apustu berri hau datu basean gordeko dugu. Apustu anitzari apustu berri hau gehitu egingo diogu.

Galarazi zenbakia -1 bada, apustu anitzaren zenbakia hartuko egingo dugu eta galarazi atributuari esleitu egingo diogu. Gainera, apustu anitzari galarazi zenbaki hori jarriko diogu.

Erabiltzaileari diru kantitatea eguneratu egingo diogu. Metodoari zenbakia negatiboan sartuko egingo diogu, diru kontutik kendu behar diogulako diru hori erabiltzaileari.

Erabiltzaile horren apustuak lortu egingo ditugu eta for bat egingo dugu hauen gainean. Apustu bakoitzaren kirola lortuko ditugu. Kirolaren apustu kantitatea atributuan bat gehitu egingo dugu. Erabiltzaileak kirol hori duen edo ez ikusiko dugu. Kirol hori badu, kirol





Kuota dagokion kirola lörtzen dugu getSport metodoa bidez, eta kirol horri apustu kantitatea gutxitu behar diogu, setApustuKantitatea metodoari dei eginez. Berdin kirol estatistikekin, userrari getKirolEstatistikak eginez (kirola parametro bezala pasatuz) lortu dugu kirolaren estatistikak eta apustu kantitatea gutxi behar zaio, setKont eginez.

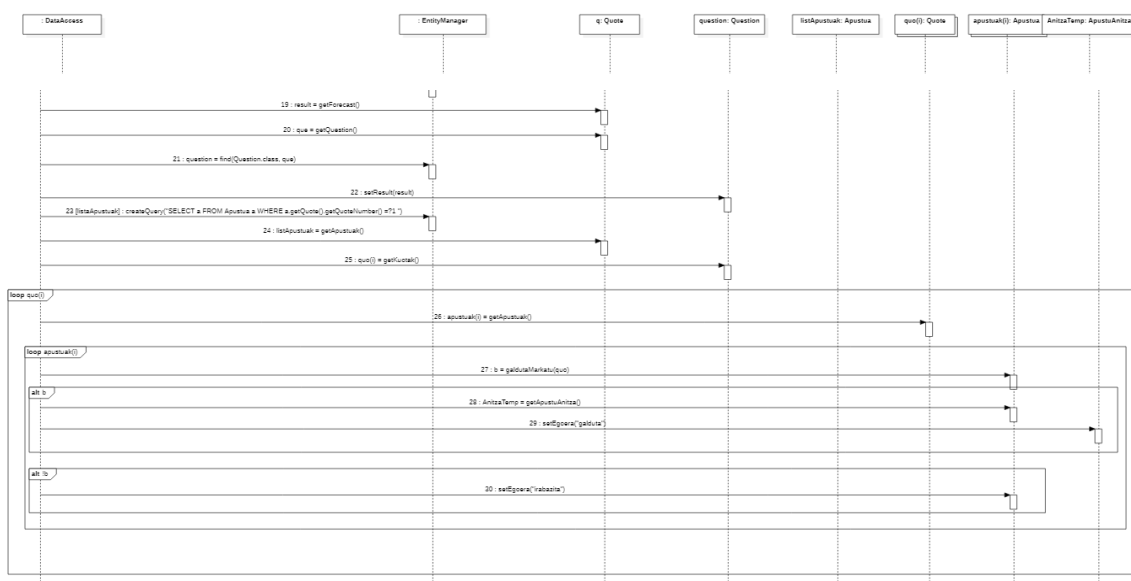
Bukatzeko apustua osorik ezabatzeko `removeApustua` egiten du `DataAccess`ek `Entity Manager`rek duen metodoa baliatuz.

## EMAITZAK IPINI



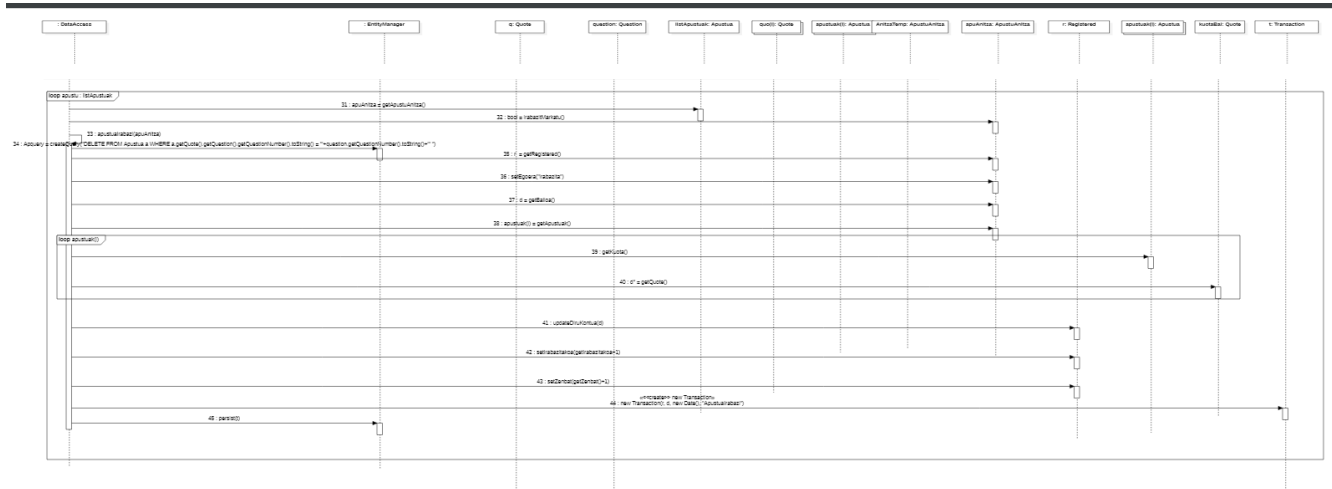
Hasieran, sistemak egutegiko egunak koloreztatuko ditu gertaeraren bat badute. Administratzaileak data bat aukeratu beharko da eta sistema datu baseraino joango da data horretako gertaera guztiak atzitzera. Gertaerak bueltatzean, administratzaileak gertaera bat aukeratu beharko du eta sistemak aurreko prozesua jarraituko du baina orain “question”-ekin. Prozesu bera errepikatuko da kuotentzako.

Administratzaileak kuota aukeratzean, emaitza jartzearen prozesua hasiko da. Lehenik eta behin, kuota datu basea bilatuko du.



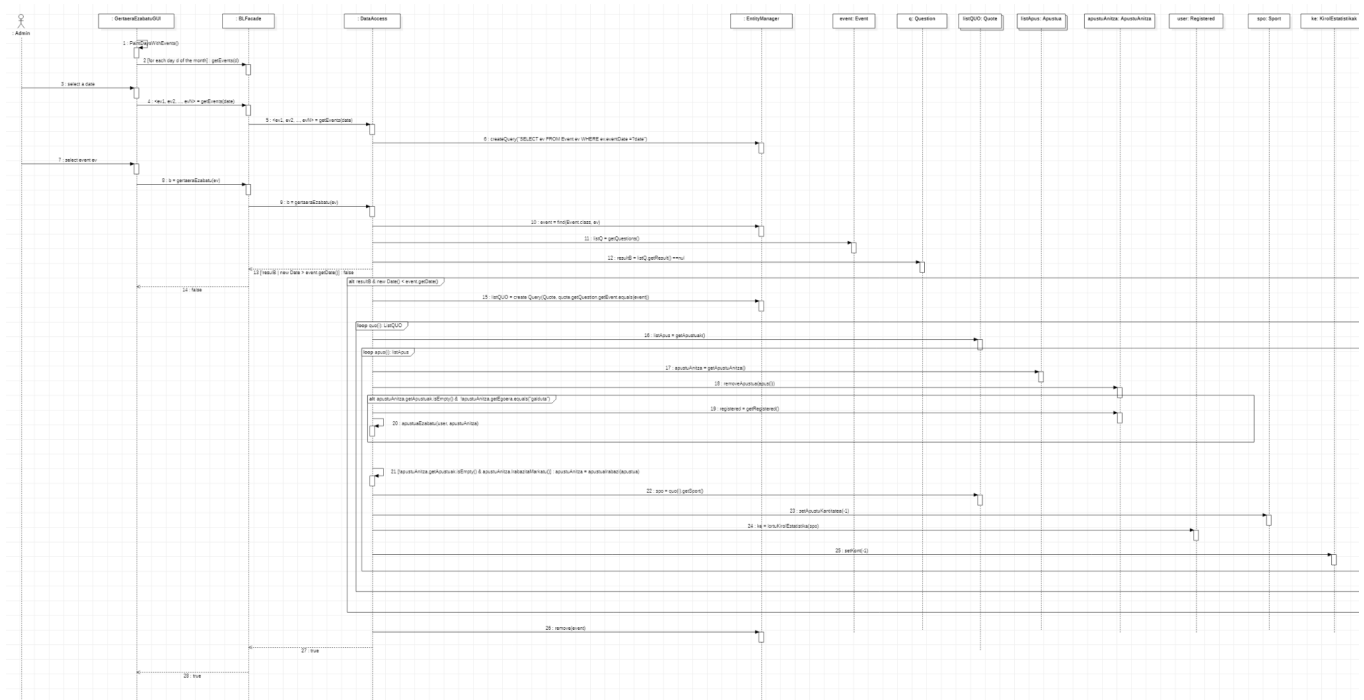
Kuotatik, forecast-a atzituko da result bezala ezartzeko eta galdera ere atzituko da. Galderari result-a esleituko zaio. Behin hau eginda, galdera horren apsutu guztiak lortuko ditu query baten bidez eta galdera orien kuota guztiak lortuko dira. Ondoren, lortutako kuota bakoitzeko

iteratuko da eta bakoitzeko apustua lortuko berriz ere, bere gainean iteratzeko. Loop horretan, begiratuko da ea apustu hori galduta egongo zen kouta hori emaitza bezala erabakitzen bada. Balio hori, b boolearran ezarriko da. True, balioa badu bere apustu anitza atzitu eta galduta bezala ezarriko du. Bestela, apustua bera irabazita jarriko du.



Azkenik, aurretik atzitutako apustu listaren loop bat egingo du. Hasieran, bere apustu anitza lortuko du eta ea bere apustu guztiak irabazita dauden begiratuko du, metodo baten bidez. Hala bada, irabazitatzat emango da. Hau gertatzean, irabazita bezala utziko da apustu anitza eta erabiltzaileari eman behar zaion diru kopurua kalkulatu eta loop baten bidez. Azkenik, diru hori erabiltzaileari gehituko zaio eta transakzio bat sortu eta gordeko da adierazten apustua irabazi duela.

## GERTAERA EZABATU



Erabiltzaileak data bat aukeratzen du. `getEvents(date)` metodoarekin GUI-tik BLFacade-ra, BLFacade-tik DataAccess-era joango gara. Horrela DataAccess-ek EntityManager-rak query bat egingo du data horretan dauden gertaerak lortzeko. Gertaera hauek GUI-an erakutsi egingo dira.

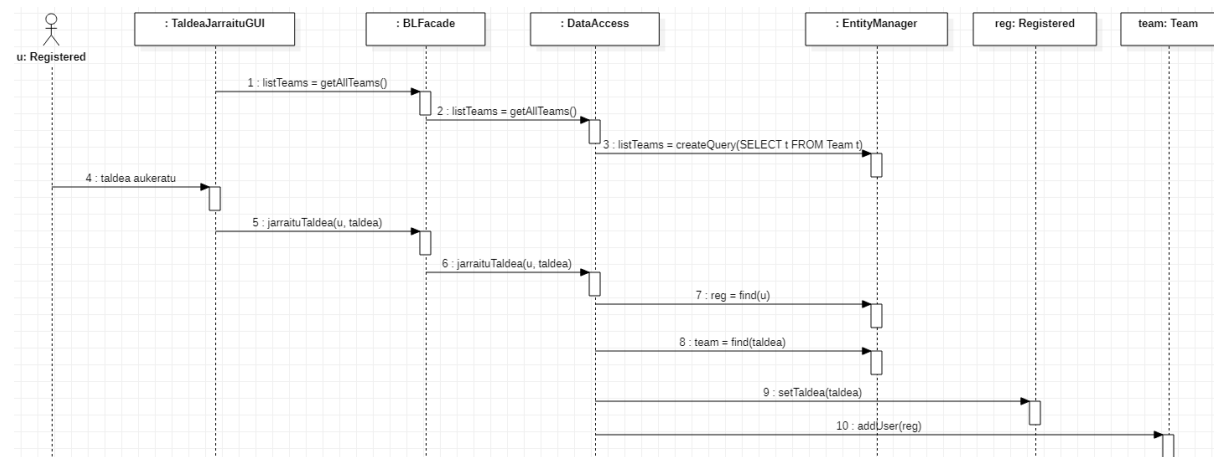
Erabiltzaileak gertaera bat aukeratzen du. `gertaeraEzabatu(ev)` metodoarekin GUI-tik BLFacade-ra, BLFacade-tik DataAccess-era joango gara. Horrela DataAccess-ek EntityManager-rak gertaera hau bilatu egingo du datu basean.

Gertaera horren galderak lortu egingo ditu. Lista horrek baldin badu galdera bat emaitza ipini gabe edo gaurko data gertaera horren data baino txikiagoa bada, metodoak false itzuli egingo du.

Query bat egingo dugu gertaera horri lotutako kuotak lortzeko. Lista honen gainean for bat egingo dugu. Kuota bakoitzaren apustu listak lortu egingo ditugu eta for bat egingo dugu. Apustu bakoitzaren apustu anitza lortu egingo dugu eta apustu anitz honi apustua ezabatu egingo diogu listatik. Lista hori hutsa badago eta apustu anitzaren egoera galdua ez bada, berriro apustua ezabatu metodoari dei egingo diogu.

Apustua irabazi metodoari dei egingo diogu apustua pasata. Kirolaren apustu kantitateari bat kenduko diogu eta kirol estatistikaren ere bat kenduko diogu haren kontagailuari. Azkenik, gertaera ezabatu egingo dugu eta true bueltatu egingo dugu.

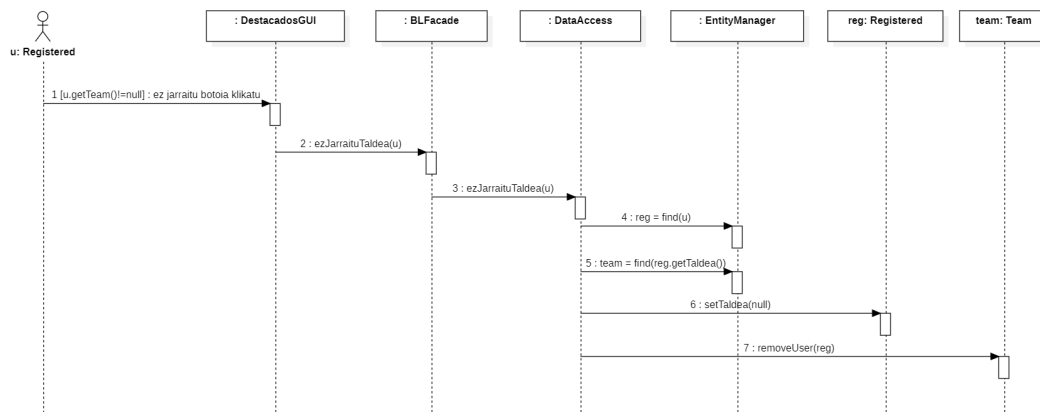
## GUSTOKOEN TALDE AUKERATU



GUIak `getAllTeams` metodoaren bidez talde guztien lista bat lortzen du, BLFacadek berdin. DataAccess-ek query bat sortzen du eta talde guztien lista bat bueltatzen du.

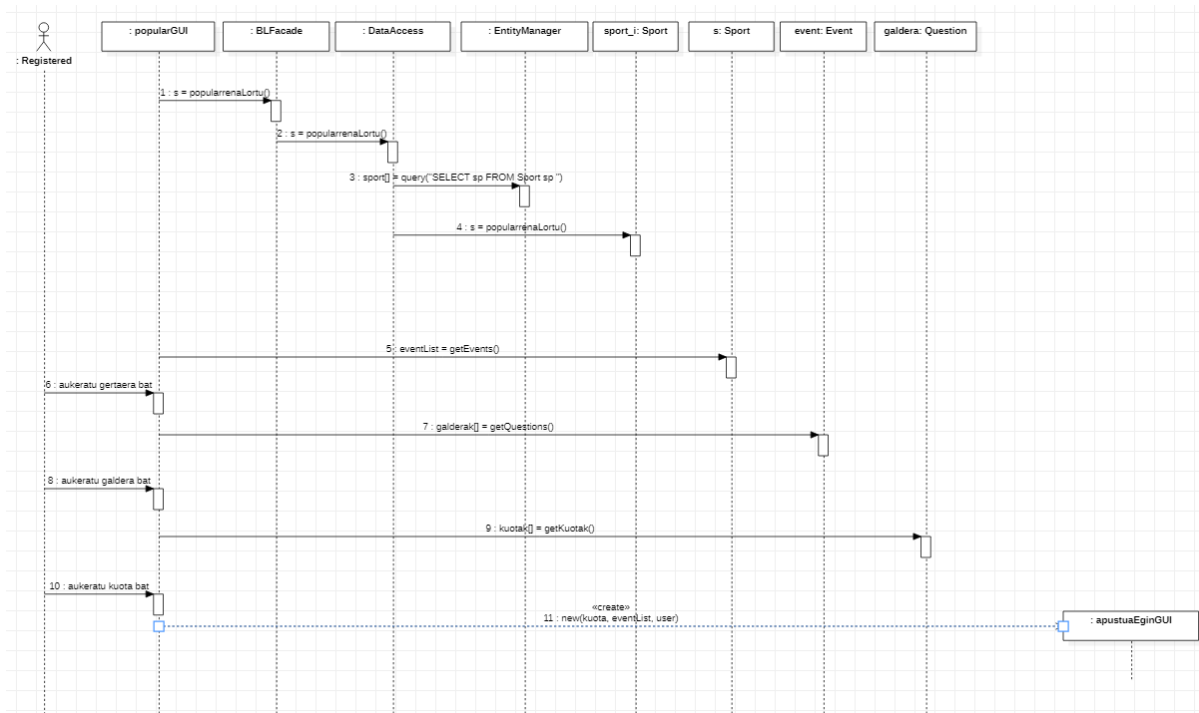
Registered-ek talde bat aukeratzen du eta GUIak `jarraituTaldea` metodoari egiten dio dei user hori eta taldea parametro bezala pasatuz, BLFacadek berdin. DataAccess-ek bi `find`-en bidez usera eta taldea bilatzen ditu. Eta user horri `setTaldea()` metodoa bidez gustoko taldea esleitzen dio. Alderantziz berdin, taldeari `addUser()` egiten dio, listara gehitzeko.

## GUSTOKOEN TALDEA EZABATU



Erabiltzaileak talde bat jarraitzen badu kentzeko aukera edukiko du. Sistemak, datu basean erabiltzaileari talde kenduko dio eta taldeari erabiltzailea

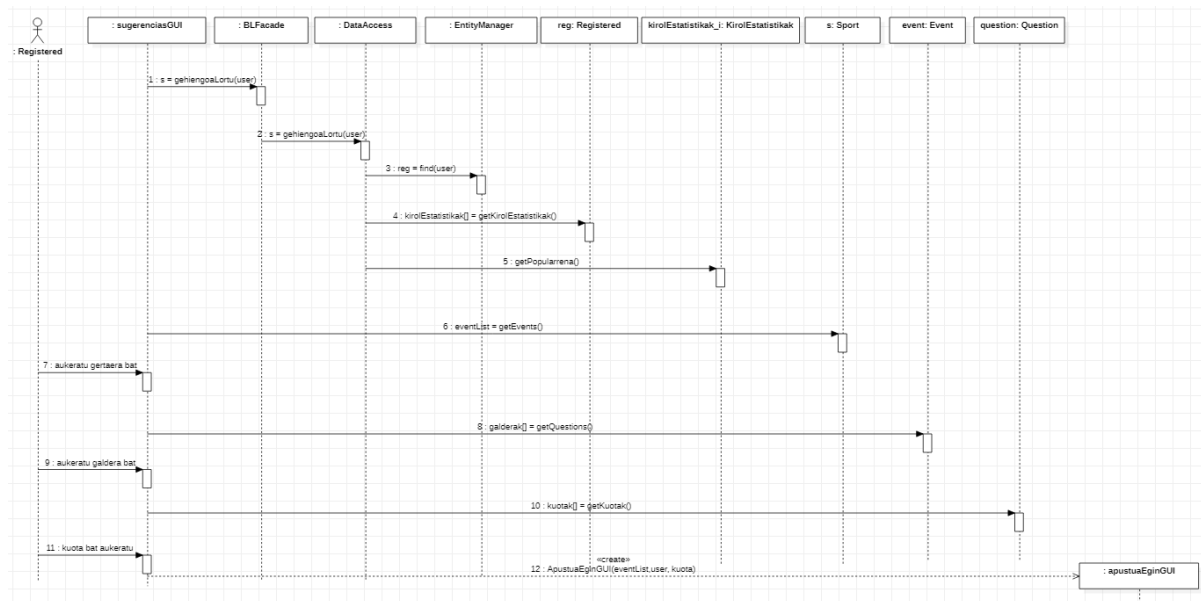
## KIROL POPULARRENA INFO



popularrenaLortu() metodoarekin GUI-tik BLFacade-ra, BLFacade-tik DataAccess-era joango gara. Horrela DataAccess-ek EntityManager-rak query bat egingo du datu basean dauden kirol guztiak lortzeko. Kirol guztiak begiratzuz jakingo dugu zein den popularrena.

Kirol horri lortutako gertaerak lortu egingo ditugu. Erabiltzaileak gertaera bat aukeratu egingo du. Gertaera horren galderak lortu egingo ditugu. Erabiltzaileak galdera bat aukeratu egingo du. Galdera horrek kuotak lortu egingo ditugu. Erabiltzaileak kuota bat aukeratu egingo du. Kuota horrekin apustuaEgin GUI-ra joango gara, gertaeren lista, erabiltzailea eta kuota pasatuko diogu.

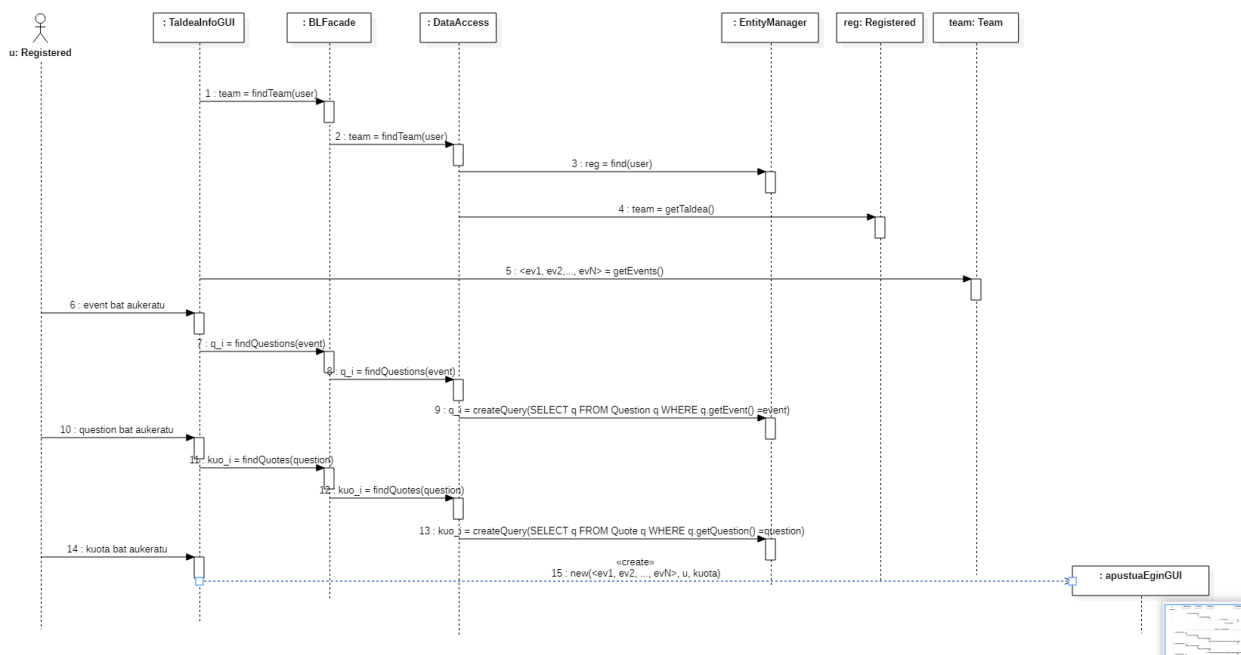
## KIROL SUGERENTZIA INFO



gehiengoLortu() metodoarekin GUI-tik BLFacade-ra, BLFacade-tik DataAccess-era joango gara. Horrela DataAccess-ek EntityManager-rak find bat egingo du datu basean userra dagoen begiratzeko. getKirolEstatistikak metodoa bidez userraren kirolen bektorea lortzen dugu. Bektore horri getPopularrena() metodoa bidez userrak gehien apustu duen kirola lortzen dugu, KirolEstatistikak duen kont-a baliatuz.

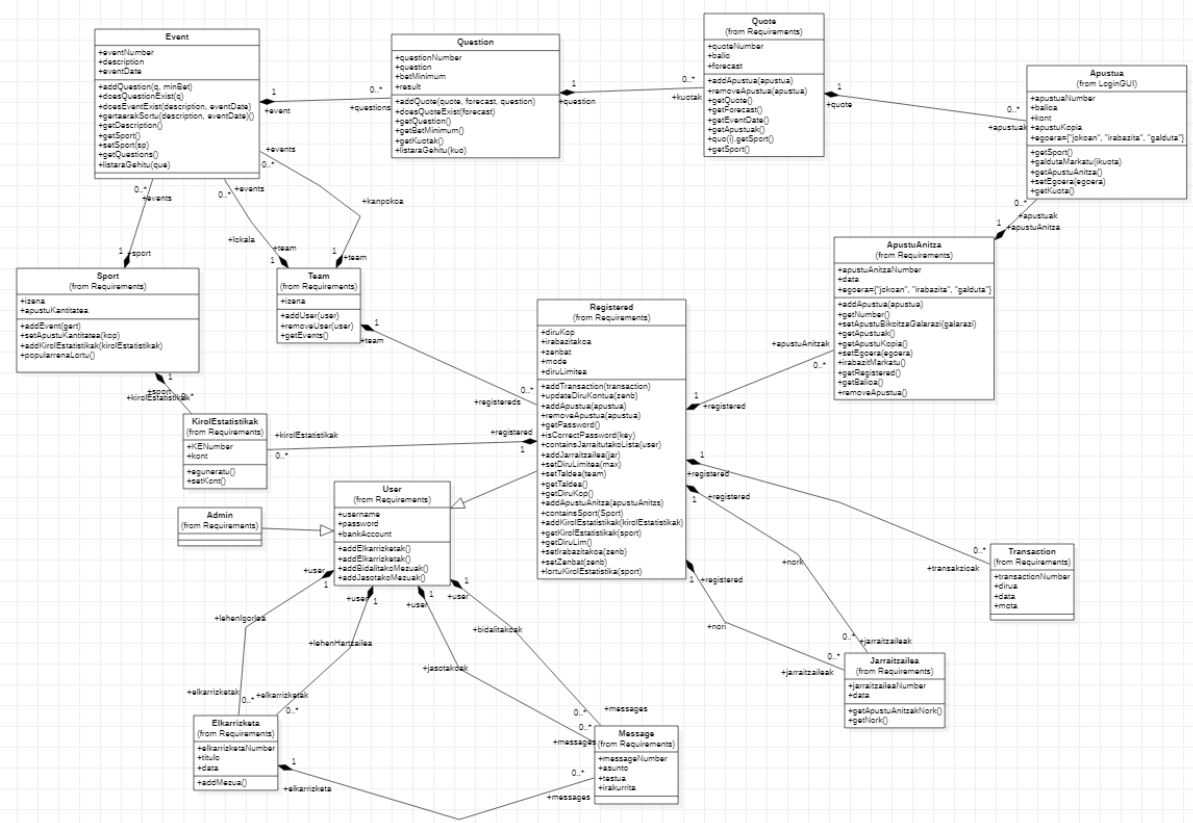
GUIak kirol horren gertaera guztiak getEvents-i esker lortuko ditu. Erabiltzaileak gertaera bat hautatuko du eta getQuestions egin dio gertaera horri GUIak. Erabiltzaileak berriz galdera bat aukeratuko du eta berriz, GUIak getQuotes egingo du eta erabiltzaileak kuota bat aukeratu egingo du. Kuota horrekin apustuaEgin GUI-ra joango gara, gertaeren lista, erabiltzailea eta kuota pasatuko diogu.

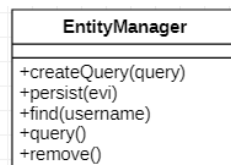
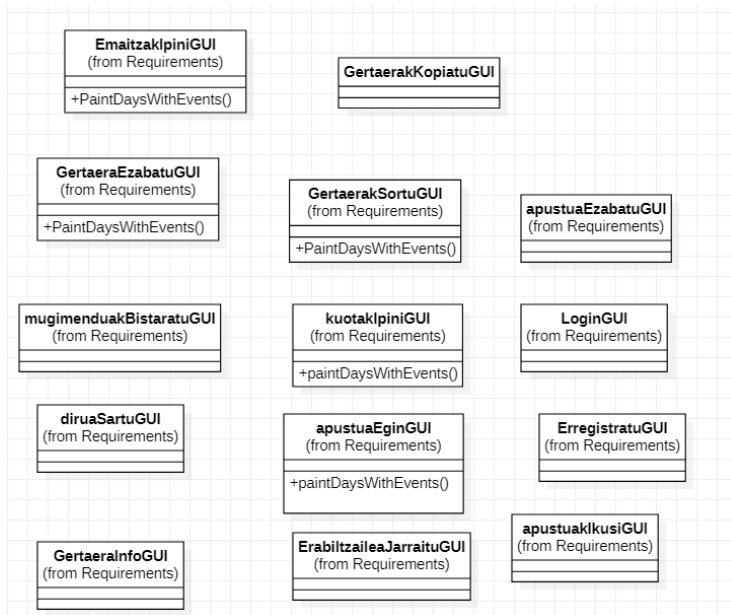
## TALDE GUSTOKOENA INFO



Sistemak erabiltzailearen taldea bilatuko du datu basean eta hortik bere gertaerak lortuko ditu. Erabiltzaileak gertaera bat aukeratu ondoren, sistemak bere galderak datu basean bilatuko ditu (batzuetan Jlist-ek arazoak eman dizkigu zuzenean bere lista atzitzean). Berdina gertatuko erabiltzaileak galdera hautatzean baina kuota ekarrita. Erabiltzaileak kuota bat aukeratu badu, sistemak apustuEginGUI-ra bidaliko du kuota horrekin.

## KLASE DIAGRAMA







## IMPLEMENTAZIOA

### Metodoak:

#### createQuestion(Event event, String question, float betMinimum)

Metodo honek galdera bat sortzen dut gertaera baterako. Galdera hau bere testuarekin eta bere apostatzeko balio minimoarekin sortzen da.

1. Parametro bezala pasatako gertaera datu basean bilatzen du.
2. Jadanik galdera hori duen gertaera bat existitzen bada, salbuespen bat jaurti egingo da QuestionAlreadyExist deitzen dena.
3. Transakzio bat ireki egingo dugu, bere barruan honako hauek egingo dira:
  - a. Datu basean bilatu dugun gertaerari sortzen ari garen galdera gehituko diogu.
  - b. Galdera datu basean gordeko dugu persist eginez.
4. Transakzioa bukatuko dugu.
5. Sortu dugun galdera bueltatuko dugu.

#### getEvents(Date date)

Metodo honek, data bat pasata, data horretan dauden gertaerak itzuliko ditu.

1. Query bat egin dugu data horretan dauden gertaerak lortzeko.
2. Gertaera horiek pantailan inprimatzen ditugu eta metodoak lista hori itzultzen du.

#### getEventsMonth(Date date)

Metodo honek, data bat pasata, hilabete horretako zein egunetan dauden gertaerak itzuliko ditu.

1. Query bat egingo du hilabete horretako datekin
2. Banan-banan query-ko datak lista batean sartu eta lista hori bueltatuko du.

#### isLogin(String username, String password)

Metodo honi user baten izena eta pasahitza pasatuko diogu. Hauek izanda, metodoak esango digu erabiltzaile hori loggeatuta dagoen edo ez. User hori loggeatuta badago user bera bueltatuko dugu, bestela null.

1. Datu basean bilatuko dugu erabiltzailea parametro bezala pasa dugun string-arekin.
2. If bat egingo dugu erabiltzailea null den edo ez jakiteko.
  - a. Null ez bada:

- i. Parametro bezala pasatako pasahitza eta datu basean bilatu dugun erabiltzailearen pasahitzak berdinak diren baieztatzen dugu. Berdinak badira, user bera bueltatu egingo da.
3. If barruan sartu ez bada; hau da, erabiltzailea null bada, metodoak null itzuli egingo du.

## isRegister(String username)

Erabiltzaile baten izena pasata, erabiltzaile hori erregistratuta dagoen edo ez esaten du.

1. Pasa diogun izenaz baliatuz, erabiltzaile hori datu basean bilatu egingo dugu.
2. True edo false itzuli egingo du erabiltzailea datu basean aurkitu den edo ez jakinda.

## storeRegistered(String username, String password, Integer bankAccount)

Izen, pasahitza eta banku kontua(zenbakia) pasata erabiltzaile berri bat sortuko du erregistratu motakoa.

1. Registered motako objektua sortzen du (username, password, bankAccount)-ekin
2. Datu basean persisteatzen du.

## gertaerakSortu(String description, Date eventDate, String sport)

Metodo honi parametro bezala hiru gauza pasako dizkiogu: gertaeraren deskripzioa, gertaeraren data eta gertaera horrek izango duen kirola. Metodo honen izenak esaten duen bezala gertaera bat sortuko du.

1. Parametro bezala pasatako kirola datu basean bilatu egingo dugu
2. Kirol hori null ez bada:
  - a. Query bat egingo da parametro bezala pasatako data horretan dauden gertaera guztiak lista batean gordetzeko.
  - b. For bat egingo dugu egiaztatzeko ea lista horretan dagoen parametro bezala pasatako deskripzio bera duena.
  - c. Deskripzioak berdinak badira, boolear bat false jarriko dugu.
3. Boolear hori true bada; hau da, sortu nahi dugun gertaera ez badago jadanik sortua.
  - a. Deskripzioa zatitu egingo dugu “-” karaktereaz. Horrela talde lokala eta kanpoko taldea lortuko ditugu.
  - b. Gertaera berri bat sortuko dugu deskripzioa, gertaera data, talde lokala eta kanpoko taldearekin.

- c. Berri sortu dugun gertaerari kirola jarriko diogu.
- d. Kirolari ere gertaera hori esleituko diogu.
- e. Gertaera datu basean gordeko dugu persist eginez.
- 4. Kirol hori null bada; hau da ez bada existitzen, false itzuli egingo da.
- 5. Azkenik boolearra itzuli egingo da, gertaera ondo sortu den edo ez jakiteko.

## storeQuote(String forecast, Double Quote, Question question)

Forecast, quota zenbakia eta galdera pasata, storeQuote metodoak kuota bat gordetzen du datu basean existitzen ez bada.

- 1. Parametro bezala pasa diogun galdera datu basean bilatu egin dugu.
- 2. Galdera horrek parametro bezala pasa diogun forecast hori duen kuota bat baldin badu, QuoteAlreadyExists salbuespena jaurti egingo da.
- 3. Galderari kuota gehitu egingo diogu.
- 4. Kuota datu basean gordeko dugu.
- 5. Metodoak kuota itzuli egingo du.

## findQuestion(Event events)

Gertaera baten galderen lista atzitzen du datubasetik.

- 1. Query bat egiten du gertaera hori duten galderekin eta lista bueltatzen du.

## DiruaSartu(User u, Double dirua, Date data, String mota)

Dirua sartu metodoak, bere izenak esaten duen bezala, erabiltzaile bati dirua sartzeko balio du. Parametro bezala, erabiltzailea, diru kantitatea, data eta zergatik sartu den dirua pasako dira.

- 1. Parametro bezala pasatako erabiltzailea, Registered motakoa izango dena datu basean bilatu egingo dugu.
- 2. Transaction klaseko transaction berri bat egingo dugu erabiltzailearekin, diru kantitatearekin, datarekin eta transakzio motarekin (zergatik egiten ari garen transakzio hori).
- 3. Erabiltzaileari transakzioa gehituko diogu.
- 4. Erabiltzaileari diru kantitatea eguneratuko diogu.
- 5. Transakzioa datu basean sartuko dugu, persist eginez.

## findQuote(Question question)

Parametro bezala galdera bat pasata, galdera horri dagokion kuoten lista itzuli egingo du findQuote metodoak.

1. Query bat egingo dugu galdera horren kuotak lortzeko.
2. Metodoak kuoten lista itzuli egingo du.

## ApustuaEgin(User u, Vector<Quote> quote, Double balioa, Integer apustuBikoitzaGalarazi)

Erabiltzailearentzat apustua egiten du, kuotak, eta balioa jakinda. Gainera, apustua ondo egin den bueltatuko da.

1. Erabiltzailea datu basean bilatu eta apust egiteko nahiko diru duen begiratzeko.
  - a. Ez badu nahiko diru false bueltatuko du.
  - b. Nahiko diru bada apustuAnitza sortzen du.
  - c. Listako kuota bakoitza datu basean bilatu eta apustu berri bati sartzen du.
  - d. Apustua apustu anitzari gehitzen dio eta datu basean gordetzen du.
  - e. Galarazi balioa -1 bada (hasierako apustua da) honi apustu berriaren zenbakia esleitzen dio eta apustu anitzari esleitzen dio.
  - f. Erabiltzailearen dirua eguneratzen du.
  - g. Transakzio berria sortzen du.
  - h. Kuota bakoitzeko, bere kirolean apustu kantitatea eguneratzen da.
  - i. Kuota bakoitzeko, bere kirolean erabiltzailearen apustu kantitatea eguneratzen da, kirolEstatistikak erabiliz.
  - j. Transakzioa erabiltzaileari gehitu eta datu basean gordetzen da.
  - k. Apustu baten kopia ez bada, bere jarraitzaileek apustu egingo dute.
  - l. Apustuaren balioak, jarraitzaileen diru maximoa gainditzen badu, diru maximoarekin egingo da apustua.
  - m. true bueltatuko du.

## findApustuAnitza(User u)

Metodo hau erabiltzaile bat pasata, erabiltzaile horrek dituen apustu anitzak lortzeko erabiltzen dugu. Esan den bezala, parametro bezala erabiltzailea pasako diogu.

1. Parametro bezala pasatzen dugun erabiltzaile datu basean bilatu egingo dugu.
2. Query bat egiten dugu erabiltzaile horrek dituen apustu anitzak lortzeko.
3. Lista hori itzultzen du metodo honek.

## findEvent(Quote q)

Parametro bezala kuota bat pasata, kuota horrek duen gertaera bueltatuko du finEvent metodoak.

1. Parametro bezala pasa dugun kuota datu basean bilatu egingo dugu.
2. Kuota horren galdera lortu egingo dugu.
3. Galdera horren gertaera lortu egingo dugu.
4. Gertaera hori bueltatu egingo du metodoak.

## findQuestionFromQuote(Quote q)

Datu basean bilatzen du pasatako kuotaren question-a (serializazio arazoengatik erabilia).

1. Find bat egiten du datu basean kuotarentzat.
2. Kuotaren galdera bueltatzen du.

## findEventFromApustua(Apustua q)

Apustu bat parametro bezala pasata, apustu horrek duen gertaera itzultzen du.

1. Parametro bezala pasatzen diogun apustua datu basean bilatzen du.
2. Apustu horren kuota lortzen dugu.
3. Kuota horren galdera lortzen dugu.
4. Galdera horren gertaera lortzen dugu.
5. Gertaera hori itzultzen dugu.

## apustuaEzabatu(User user1, ApustuAnitza ap)

Parametro bezala pasatako apustu anitza ezabatzen du, eta balioak eguneratzen ditu(Kirolak eta erabiltzailearen dirua).

1. Erabiltzailea eta apustu anitza datu basean bilatzen ditu.
2. Apustuaren dirua erabiltzaileari bueltatzen dio.
3. Transakzio berria sortzen du eta erabiltzaileari gehitzen dio.
4. Erabiltzailearen listatik apustu anitza ezabatzen du.
5. Gainera, apustu anitzaren apustu bakoitzeko.
  - a. Apustua kuotari kentzen dio
  - b. Kuotaren kirolaren apustu kopuruari 1 kentzen dio.
  - c. Erabiltzailearen kirolaeriko kirolEstatistikari 1 kentzen dio
6. Apustu anitza datu basetik ezabatzen du.

## findTransakzioak(User u)

Erabiltzaile bat parametro bezala pasata, erabiltzaile horrek dituen transakzioak itzultzen ditu.

1. Parametro bezala pasatzen diogun erabiltzailea datu basean bilatzen du eta Registered bezala gordetzen dugu.
2. Query bat egin dugu. Bertan erabiltzaile horrek dituen transakzioak lortzen ditugu.
3. Transakzioen lista hori itzultzen du metodo honek.

## EmaitzakIpini(Quote quote)

Quota bat pasata, emaitzak ipini metodoak, kuota horri dagozkion apustuei galduta egoera jarriko zaie.

1. Parametro bidez pasatako kuota bilatu egingo dugu datu basean.
2. Kuota izanda, galdera lortu egingo dugu.
3. Galdera izanda, gertaera lortu egingo dugu.
4. Gertaeraren data lortu egingo dugu.
5. Gaurko data gertaeraren data baino txikiago bada, EventNotFinished salbuespena jaurti egingo da.
6. Kuotaren apustuak lortu egingo ditugu.
7. Galdera datu basean bilatu egingo dugu.
8. Galderari emaitza jarri egingo diogu.
9. Apustu guztiak korritu egingo ditugu:
  - a. Apustu bakoitza galduta egoera jarriko diogu, eta galduta markatuta metodoari dei egingo diogu.
10. Apustu bakoitzaren apustu anitzak ere irabazita markatu egingo ditugu.

## gertaeraEzabatu(Event ev)

Pasatako gertaera ezabatzen du eta horri erlazionatutako apustuak ezabatzen ditu.

1. Gertaera datu basean bilatu eta bere galderen emaitzak jarrita dauden begiratzen.
2. Galdera baten emaitza jarrita ez badago zuzenean false bueltaten du.
3. Bestela gertaera ez bada oraindik gertatu
  - a. Gertaeraren kuota guztiak bilatzen ditu.
  - b. Kuota bakoitzeko
    - i. Kuota duten apustu bakoitzeko
      1. Bere apustu anitza biltzen da.
      2. Apustu ezabatzen da.
      3. Apustu anitza apustu gabe geratzen bada eta galduta badago apustu anitza ezabatzen da-

4. Apustu anitzaren geratzen diren apustu guztiak irabazita badaude apustua irabazitzat ematen da.
5. Apustuaren kirolaren apustu kopuruari 1 kentzen zaio.
6. Erabiltzailearen kirolarekiko kirolEstatistika-ri 1 kentzen zaio.
4. Gertaera datu basetik ezabatzen da eta “cascade” moduan bere galdera eta kuotak.

## saldoaBistaratu(User u)

Erabiltzaile bat parametro bezala pasata, erabiltzaile horren diru kopurua itzultzen dugu string bezala.

1. Parametro bezala pasatzen dugun erabiltzailea datu basean bilatzen dugu eta Registered bezala gordeko dugu.
2. Registered horrek diru kopurua lortzen dugu.
3. Diru kopuru hori string bihurtzen dugu eta itzultzen dugu.

## elkarriketakLortu(User u)

Erabiltzaile bat pasata, erabiltzaile horrek dituen elkarriketak itzuli egingo ditu.

1. Parametro bezala pasatako erabiltzailea datu basean bilatu egingo dugu.
2. Erabiltzaile horren elkarriketak lortu egingo ditugu.
3. Elkarriketen for bat egingo dugu:
  - a. Elkarriketa bakoitzerako Elkarriketa Container bat sortu egingo dugu.
4. Elkarriketa container-ren lista bueltatu egingo dugu.

## mezuaBidali(User igorle, String hartzailea, String titulo, String test, Elkarriketa m)

Mezu bat sortzen du bi erabiltzailearekin eta mezuarentzako informazioarekin.

1. Hartzailea eta igorlea datu basean bilatuko ditu find bidez.
2. Hartzailea null bada(hau da ez da existitzen) zuzenean false bueltatuko du eta metodoa amaituko da.
3. Mezu berria sortuko du igorle hartzaile eta testuarekin eta datu basean sartuko du.
4. Pasatako elkarriketa ez bada “null” datu basean bilatuko du.
5. Pasatako elkarriketa “null” bada esan nahi du ez dela existitzen eta kasu horretan:
  - a. Elkarriketa berria sortuko da titulo eta bi erabiltzaileekin.
  - b. Mezua gehituko zaio elkarriketari eta mezuari elkarriketa.
  - c. Bi erabiltzaileak gehituko zaizkio elkarriketari.
6. Erabiltzaileei mezua gehituko zaie.

## rankingLortu()

Registered guztien ranking bat egingo du metodo honek, erabiltzaile bakoitzak irabazi duen diru kopuru kantitatearen arabera ordenatuta.

1. Query bat egiten dugu datu baseko registered guztiak lortzeko.
2. Query-tik lortzen dugun listarekin for bat egiten dugu.
3. Lista berri bat sortzen dugu, bertan erabiltzaileak ordenean sartzen joango gara.
  - a. Sortu dugun lista berria hutsa badago
    - i. Lista horren lehen posizioa sartuko dugu erabiltzailea
  - b. Bestela
    - i. Begiratuko dugu ea listan dagoen beste erabiltzaile bat diru kopuru gutxiagorekin. Horrela jakingo dugu uneko erabiltzailea non sartu.
4. Azkenik sortu dugun lista itzuliko du metodoak.

## getEventsAll()

Datu basean dauden gertaera guztiak itzuliko ditu metodo honek.

1. Query bat egingo dugu datu basean dauden gertaera guztiak lortzeko.
2. Gertaeren lista bueltatu egingo du metodoak.

## mezualkusita(Message m)

Mezua parametro bezala emanda ikusita bezala markatuko da.

1. Mezua datu basean bilatuko du find bidez.
2. Bere ikusita egoera “true” bezala ezartzen du.

## gertaerakKopiatu(Event e, Date date)

Metodo honek, gertaera eta data bat pasata, gertaera berri bat sortuko du pasatako gertaeraren kopia bat izango dena baina parametro bezala pasa dugun datan sortuko da.

1. Parametro bezala pasatako gertaera datu basean bilatu egingo dugu.
2. Query bat egingo dugu. Query honek parametro bezala pasatako gertaerak duen deskripzioa eta data berdina duen gertaeren lista bat itzuli egingo digu.
3. Boolear bat false bezala hasieratu egingo dugu.
4. Lista hau hutsa badago:
  - a. Boolearra true jarriko dugu.
  - b. Pasa dugun gertaeraz baliatu egingo gara bere deskripzioa hartzeko. Deskripzioarekin talde lokala eta kanpoko taldea lortuko dugu.



- c. Gertaera berri bat sortuko dugu. Pasatako gertaeraren deskripzioa berdina, parametro bezala pasatako data, talde lokala eta kanpoko taldea jarriko diogu.
- d. Gertaera berri honi lehengo gertaerak duen kirola esleituko diogu.
- e. Kirol horri gertaera berri hau ere esleituko diogu.
- f. Gertaera berri hau datu basean gordeko egingo dugu persit eginez.
- g. Lehengo gertaeraren galderak hartuko ditugu eta for bat egingo dugu.
  - i. Galdera berri bat sortu egingo dugu
  - ii. Gertaera berriari galdera hau gehituko diogu.
  - iii. Galdera hori datu basean gordeko dugu.
  - iv. Galdera horrek dituen kuotak hartuko ditugu eta for bat egingo dugu.
    - 1. Kuota berri bat sortuko dugu.
    - 2. Galdera berriari kuota hau esleituko diogu.
    - 3. Kuota berri hau datu basean sartuko egingo dugu.
- 5. Azkenik hasieran sortutako boolearra itzuli egingo dugu.

## jarraitu(Registered jabea, Registered jarraitua, Double limit)

Jabea eta jarraitu pasata, diru limite batekin, jarraitzailea erabiltzaile hori jarraitu egingo du.

- 1. Jarraitu datu basean bilatu egingo dugu eta jarraitu izenarekin gorde.
- 2. Jabea datu basean bilatu egingo dugu eta harpideduna izenarekin gorde.
- 3. Harpidedunaren jarraitutako listan jarraitu ez badago:
  - a. Jarraitzaile berri bat sortu dugu harpideduna eta jarraitu erabiltzaileekin.
  - b. Harpidedunari jarraitzailea gehituko diogu.
  - c. Jarraituri jarraitzailea gehitu diogu.
  - d. Jarraitzailea datu basean gordeko dugu.
  - e. Harpidedunari diru limitea jarriko diogu.
  - f. Boolear bat true-n jarriko dugu.
- 4. Boolearra itzuli egingo da.

## gehiengoaLortu(User reg)

Erabiltzaileak gehien apostatu duen kirola bueltatuko du.

- 1. Erabiltzailea datu basean bilatuko du.
- 2. Max balio berri bati min\_value esleituko zaio.
- 3. Kirol baten objektua sortuko da eta "null" esleituko zaio.
- 4. Erabiltzailaren kirolEstatistika bakoitzarekiko
  - a. Bere kontagailua max baino handiagoa bada, kirol bezala bere kirola gordeko da eta max bezala kontagailua
- 5. Bukaera geratutako kirola bueltatuko da.

## popularrenaLortu()

Metodo honek zein izan den gehien apustatu egin den kirola esango digu.

1. Sport bat hasieratuko dugu null balioarekin.
2. Query bat egingo dugu datu basean dauden kirol guztiak lortzeko.
3. Query horrek itzultzen digun kirol lista korritu egingo dugu for bat eginez:
  - a. Kirol batek badu apustu kantita gehiago, lehen hasieratu dugun sport aldagaian gordeko dugu kirol hori.
4. Azkenean, listatik korritzetik bukatzen dugunean, apustu gehien dituen kirola bueltatuko du metodo honek.

## ezJarraituTaldea(User u)

Erabiltzaile bat pasata, erabiltzaile horrek jarraitzen duen taldea jarraitzen utziko dio.

1. Parametro bezala pasa dugun erabiltzailea datu basean bilatu egingo dugu.
2. Erabiltzaile horrek duen taldea bilatu egingo dugu datu basean.
3. Taldeari erabiltzailea ezabatu egingo diogu.
4. Erabiltzaileari taldea null bezala jarriko diogu.

## getAllTeams()

Datu baseko kirol guztien lista bueltatuko du.

1. Query bat egiten da kirol guztientzako
2. Query-tik kirol lista hartzen da eta bueltatzen da.

## jarraituTaldea(User u, Team t)

Metodo honek, erabiltzaile bat eta talde bat pasatzen badiogu, erabiltzaile horrek talde bat jarraitu ahal izango du.

1. Parametro bezala pasatako erabiltzailea datu basean bilatu egingo dugu.
2. Parametro bezala pasatako kirola datu basean bilatu egingo dugu.
3. Kirol horri erabiltzaile hori gehitu egingo diogu.
4. Erabiltzaile horri kirol hori gehitu egingo diogu.

## findUser(User user)

Erabiltzaile baten izena pasata, erabiltzailea itzuliko du metodo honek.

1. Parametro bezala jaso den izenaz baliatuz, datu basean bilatuko dugu erabiltzaile hori.
2. Erabiltzaile hori bueltatu egingo du metodoak.

## getEventsTeam(Team t)

Parametro bezala pasatako talderen geratera guztiak bueltatzen ditu.

1. Gertaera Query bat egiten da non taldea lokala edo kanpokoia izan daiteken.
2. Query-aren gertaera lista bueltatzen da.

## findElkarrizketa(Elkarrizketa elk)

Elkarrizketa bat pasata, elkarrizketa hori datu basean bilatu egingo dugu eta itzuli egingo dugu.

1. Parametro bezala pasatutako elkarrizketa datu basean bilatu egingo dugu.
2. Elkarrizketa hori itzuli egingo du metodoak.

## mezuakLortu(Elkarrizketa e)

Elkarrizketa bat pasata, elkarrizketa horretan dauden mezuak itzuliko ditu metodo honek.

1. Parametro bezala pasatako elkarrizketa bilatu egingo dugu datu basean.
2. Elkarrizketa horien mezuak lortu egingo ditugu.
3. For bat egingo dugu mezu horren gainean:
  - a. Mezua Container sortu egingo dugu mezu bakoitzarekin.
4. Mezu Container lista bat itzuliko du metodo honek.

## findTeam(User u)

Datu basean bilatuta erabiltzailearen taldea lortzen da.

1. Erabiltzailea registered bezala bilatzen du datu basean.
2. Erabiltzailearen taldea bueltatzen du.

## findSport(Event q)

Gertaera bat pasata, gertaera hori datu basean bilatu egingo dugu eta hari dagokion kirola bueltatu egingo du metodo honek.

1. Parametro bezala pasatuako gertaera datu basean bilatu egingo dugu.
2. Gertaera horri lotuta dagoen kirola lortu egingo dugu.
3. Kirol hori itzuli egingo du metodoak.

## ONDORIOAK

Gure iritziz ikasi duguna proiektua egiteko garaian oso baliagarria da edozein web aplikazio egiteko, bai kode aldetik eta baita interfazeen aldetik. Gainera erabili dugun SCRUM teknika proiektua arintzen laguntzen du; izan ere, iterazioka egin dugunez proiektua zatika egitea eraginkorragoa izan da. Dena batera bidali izan bazen seguruenaz ez genuen jakingo gauza batzuk egiten; izan ere, sekuentzia diagramak, domeinu ereduak... egiten proiektuaren parean praktikatzeko joan gara, gaiak gehiago sakonduz.

Gure proiektua garatzeko modua presentzialki denak batera izan da eta nahiko komenigarria dela uste dugu; izan ere, lanaren egoera egunean eramateko egokiena da.

Hala ere, dena ez da izan hain erraza, batzuetan arazo batzuk izan ditugu serializatzerakoan batez ere. Hasieran hainbat ziklo zirkular genituen klaseen artean, adibidez, registeredek registered lista bat zuen jarraitzaileak irudikatzeko... Baina lortu ditugu konpontzea beste klase batzuk sortuz.

Pozik gaude proiektuaren emaitzarekin; izan ere, nahiko beteta dago. Erabilpen kasu asko egin ditugu eta gure erabilpen kasu berezia ekarpen berriak ematen dizkio web apustuari.

Hurrengo urteko ikasleei gomendatzen dieguna bereziki honakoa da: proiektua egunean eramatea horrela entregatzeko garaian ez dute arazorik pasako. Gainera, denek proiektuaren egoera egunean jakitea komenigarri dela ikusten dugu.

Laburbilduz, hiruroi asko gustatu zaigu ikasgai honetan landutakoa, gure etorkizunerako baliagarria ikusten baitugulako.

## BIDEROAREN URL-A

URLa: <https://youtu.be/8BE1Ap9p1mA>

## KODEAREN URL-A

GitHub-en igotzeko arazoak izan ditugu, tamaina dela eta. Orduan entregatu dugun karpetaen erroan data eta Equipos karpetak daude, eclipsen probatu nahi baduzu horiek horrela txertatu:

GitHub-eko esteka (bertan view raw edo download-eri eman):

[https://github.com/mikel0912/Bets21/blob/main/Bets21\\_MarkelEtxabe\\_AndreaJaunarena\\_MikelMartin.zip](https://github.com/mikel0912/Bets21/blob/main/Bets21_MarkelEtxabe_AndreaJaunarena_MikelMartin.zip)

