eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Fakultatea

SOFTWARE INGENIARITZA

PROIEKTUAREN DOKUMENTAZIOA

Egileak:
Urtzi Espinosa
Unax Amunarriz
Zuriñe Behobide

AURKIBIDEA

- **Web Zerbitzua eta Internalizazioa**

Internalizazioa garatu dugu, web zerbitzuak ez.

- **Sarrera**

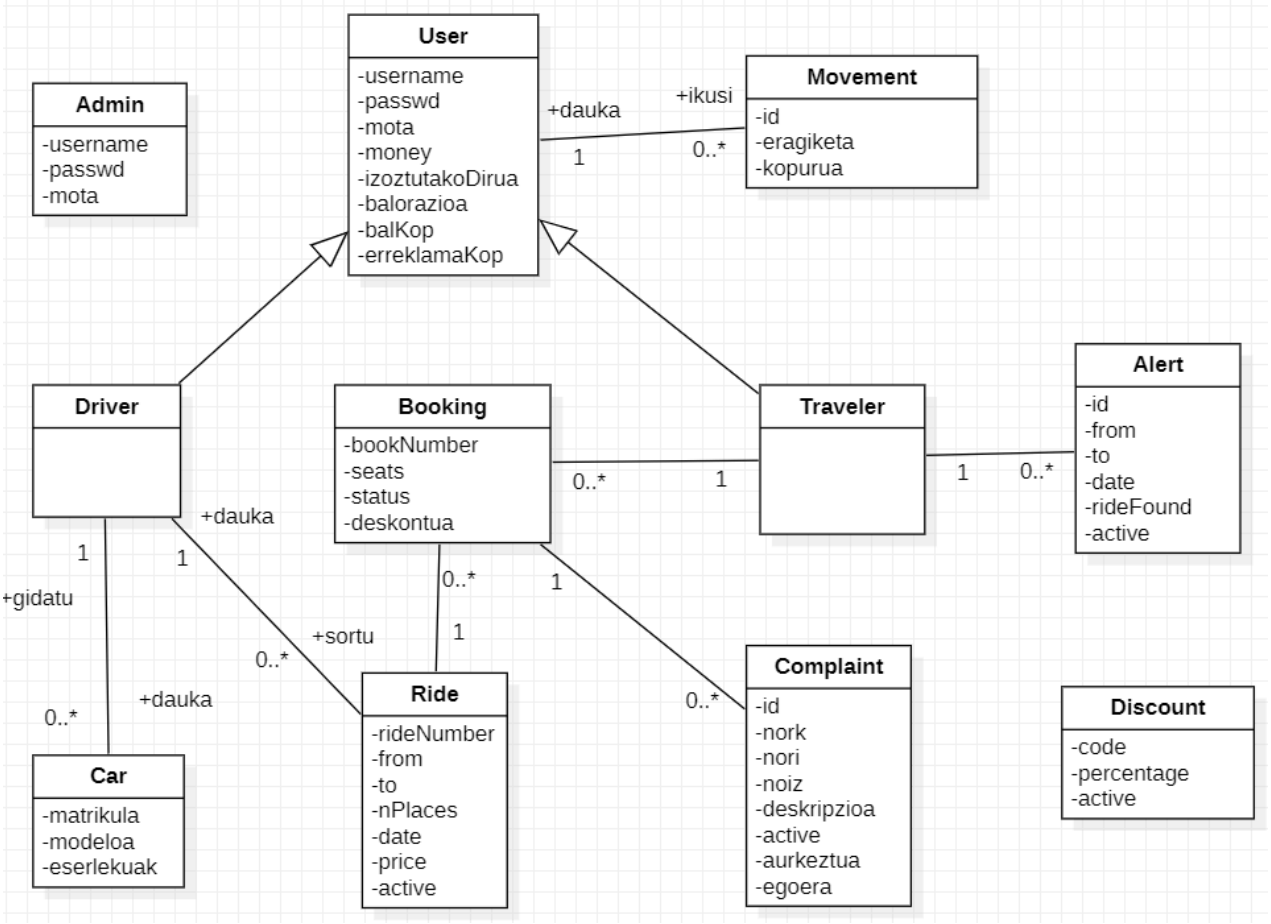
Hiru mailako software arkitektura batean diseinatutako bidaiak konpartitzeko aplikazio bat garatu dugu.
Aplikazio honetan erabiltzaile erregistratuentzako hainbat funtzionalitate agertzen dira, baina honako funtzio nagusi hauek azpimarratuko ditugu:

  - Bidaiak sortzea.
  - Bidaiak erreserbatzea.
  - Bidaiak bilatzea.

- **Eskakizun bilketa eta diseinua**
  - Entitateak

Guztira lau entitate definitu ditugu: *Ez-erregistratua*, *User*, *Traveler* eta *Driver*.
Lehenengo entitatea, aplikazioan erregistratu ez diren erabiltzaileei dagokio. Entitate hau ez dago datu basuan gordeta, beraz, aplikazioaren barruan ez du inongo eraginik egingo erregistratua izan arte. Beraz, hiru entitate nagusiak *User, Traveler* eta *Driver* dira. *Traveler* eta *Driver* klaseek atributu berdinak konpartitzen dituztenez*, User* klasearen umeak dira eta hainbat funtzionalitate konpartitzen dituzte.

- Domeinuaren eredua eta bere erlazioak



Diseinatu dugun domeinuaren ereduan, 11 klase definitu ditugu: *Admin, User, Movement, Driver, Booking, Traveler, Alert, Car, Ride, Complaint* eta *Discount.*

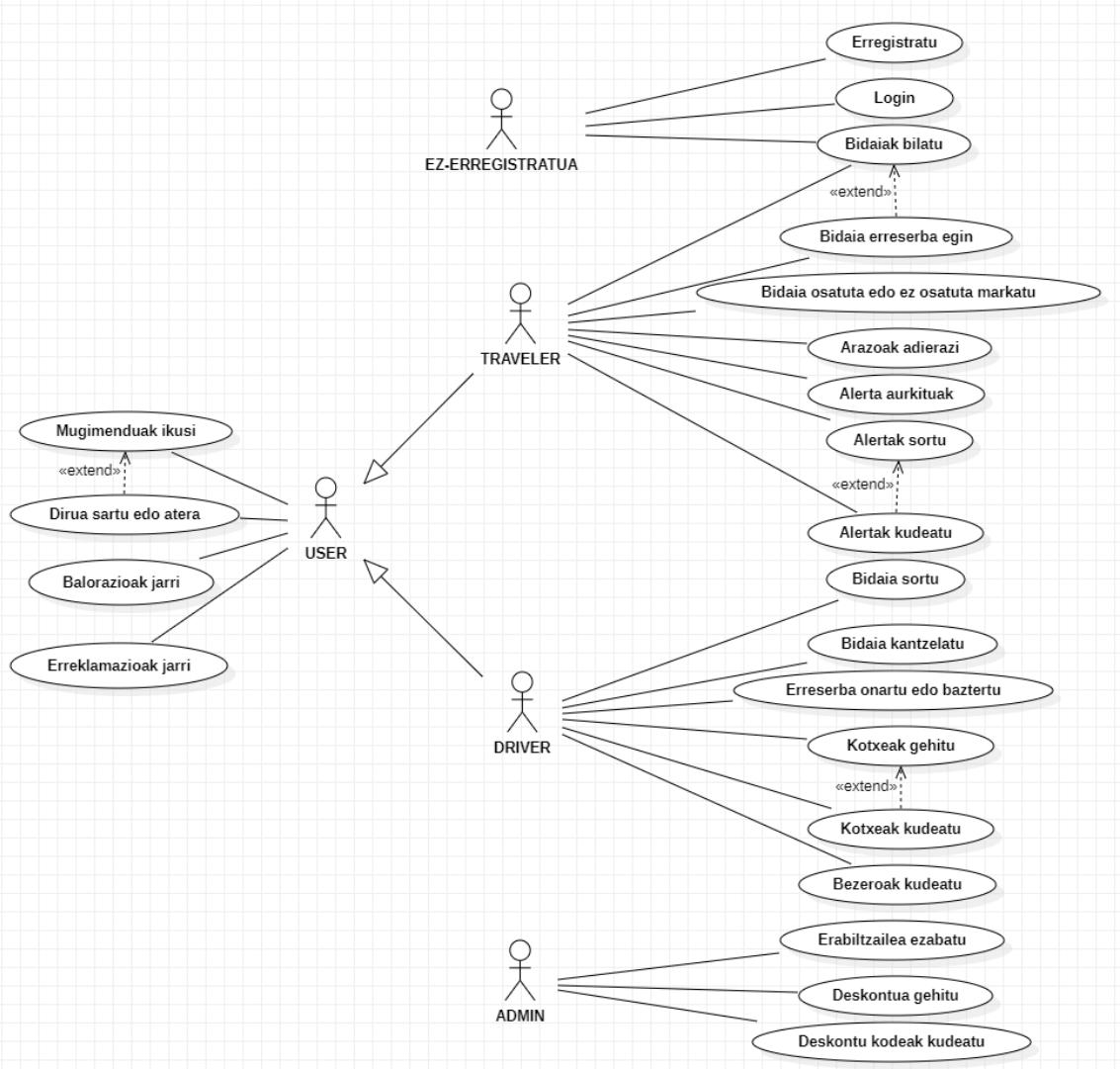
User klaseak erabiltzaile erregistratuak biltzen ditu eta bi klase ume dauzka: *Driver* eta *Traveler*. *User* klasea definitu dugu aurreko bi klase hauek atributu berdinak konpartitzen dituztelako. *User* klaseak *Movement* klaseko lista bat dauku, bertan, diruarekin egindako mugimendu eragiketa guztiak agertuko dira eta *Movement* klase bakoitza *User* bakar bati lotuta egongo da.

Driver klaseak *Car* motako lista bat dauka eta *Car* bakoitzak *Driver* bakarra gidatzeko aukera izango du. Horretaz gain, *Driverrak* sortutako bidaien lista bat dauka, *Ride* izenarekin definitutakoa.

Traveler klaseak hainbat *Alert* kudeatzeko lista bat dauka eta *Alert* bakoitza Traveler bakar bati lotuta dago. *Traveler* klaseak baita *Booking* klaseari lotuta dago, erreserbatutako bidaia guztiak gordetzeko. *Booking* klase honek beste bi klaseekin lotuta dago. Alde batetik, *Ride* klasearekin, erreserba bakoitza bidaia bati esleituta egoteko eta *Ride* batek hainbat erreserba desberdinak edukitzeko. Bestetik, *Booking* bakar bat hainbat erreklamazio eduki ditzakeenez, *Complaint* klaseari lotuta dago.

Azkenik, loturik gabeko bi klase gelditzen dira: *Admin* eta *Discount. Admin* klaseak aplikazio barruan administratzaile bezala jokatzen du eta funtzio nagusiak jasatzen ditu. *Discount* klasea, *Travelerrek* bidaia baten erreserbaren prezioari aplikatu diezaioketen deskontua da.

- Erabilpen kasuen eredua



- Erabilpen kasu bakoitzeko bere gertaera fluxua eta interfaze grafikoa

Erregistratu

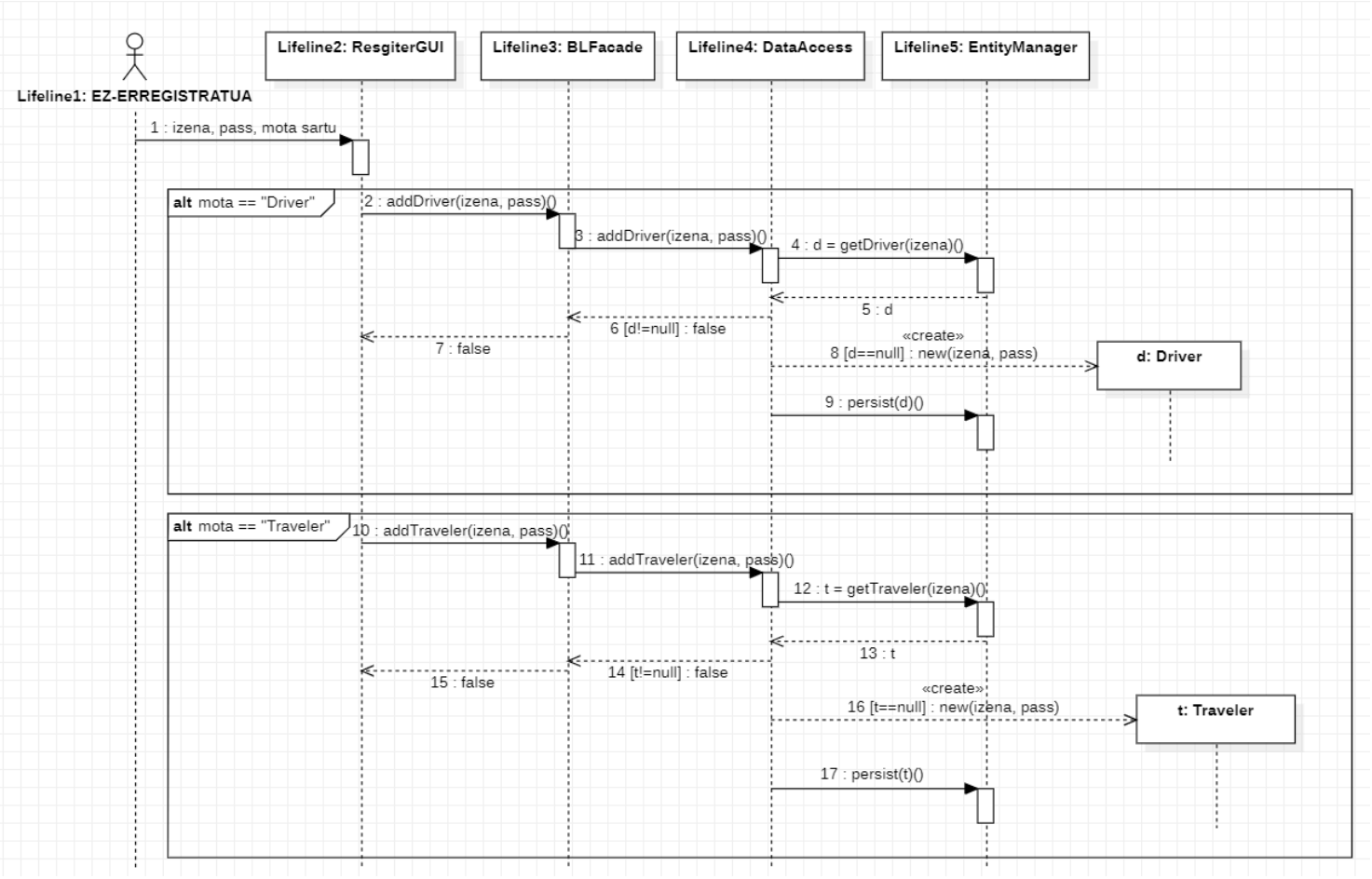
Gertaera fluxuak

Basic flow

1. *Ez-erregistratua* **izena** eta **pasahitza** sartuz, bidaiaria edo gidaria aukeratu eta sartu.
2. *Sistemak* aukeratutako rollaren arabera erabiltzaile hori dagoeneko existitzen den frogatzen du. Ez bada existitzen erabiltzailerik, kontu bat sortzen du.

Alternative flow

1. Datuak falta dira. Ez da erregistratzen. Sistemak erabiltzailea abisatzen du.
2. Dagoeneko erabiltzailea existitzen da. Sistemak erabiltzailea abisatzen du.



Login

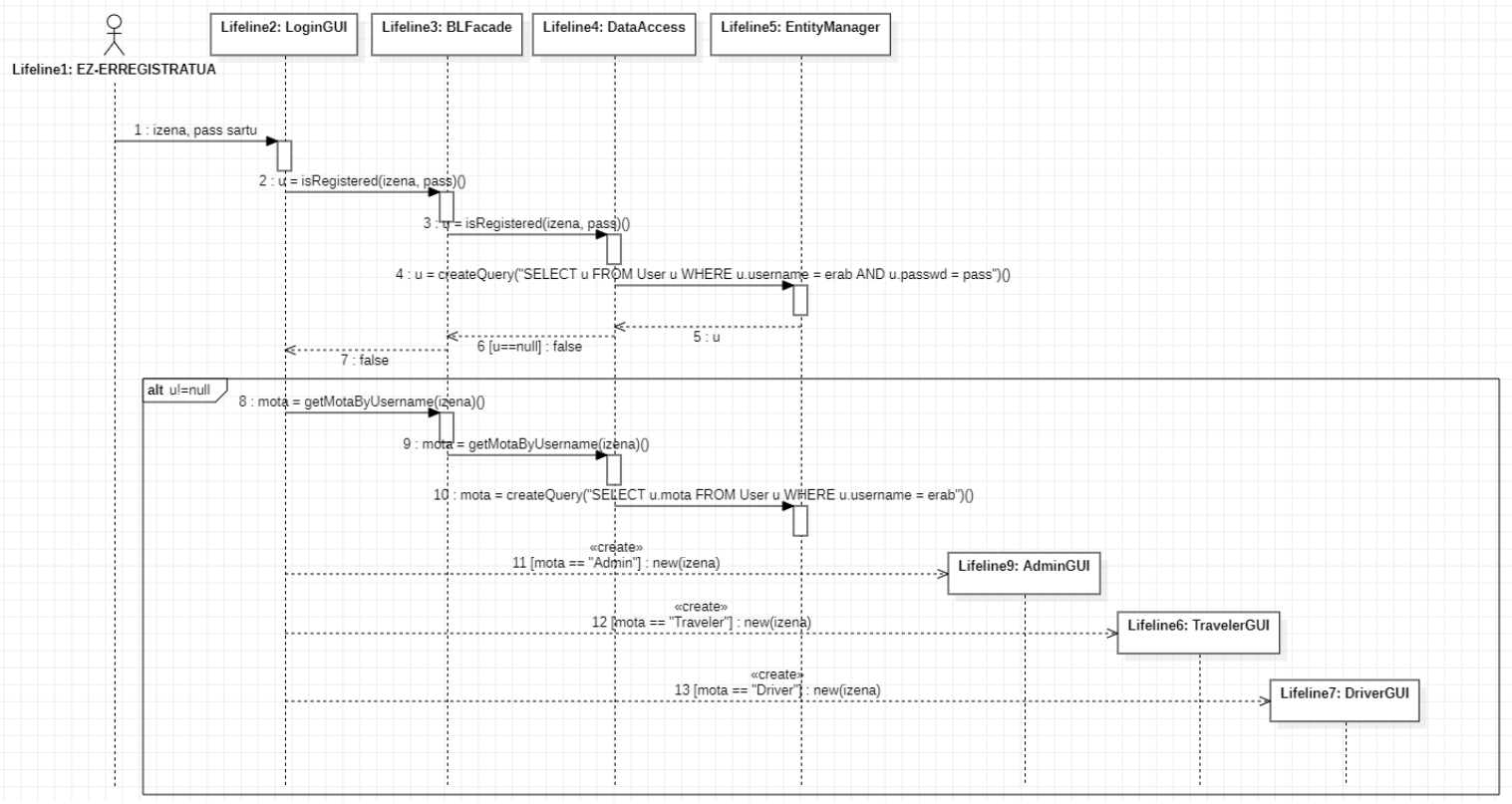
Gertaera fluxuak

Basic flow

1. *Ez-erregistratua* **izena** eta **pasahitza** datuak ipiniz, aplikaziora sartu.
2. *Sistemak* jarritako datuen arabera kontu bat existitzen bada frogatzen du. Jadanik existitzen bada, erabiltzailaren rollaren arabera aplikaziora sartzen da.

Alternative flow

1. Datuak falta dira. Sistemak erabiltzailea abisatzen du.
2. Ez da existitzen erabiltzailerik. Sistemak erabiltzailea abisatzen du.



Bidaiak bilatu

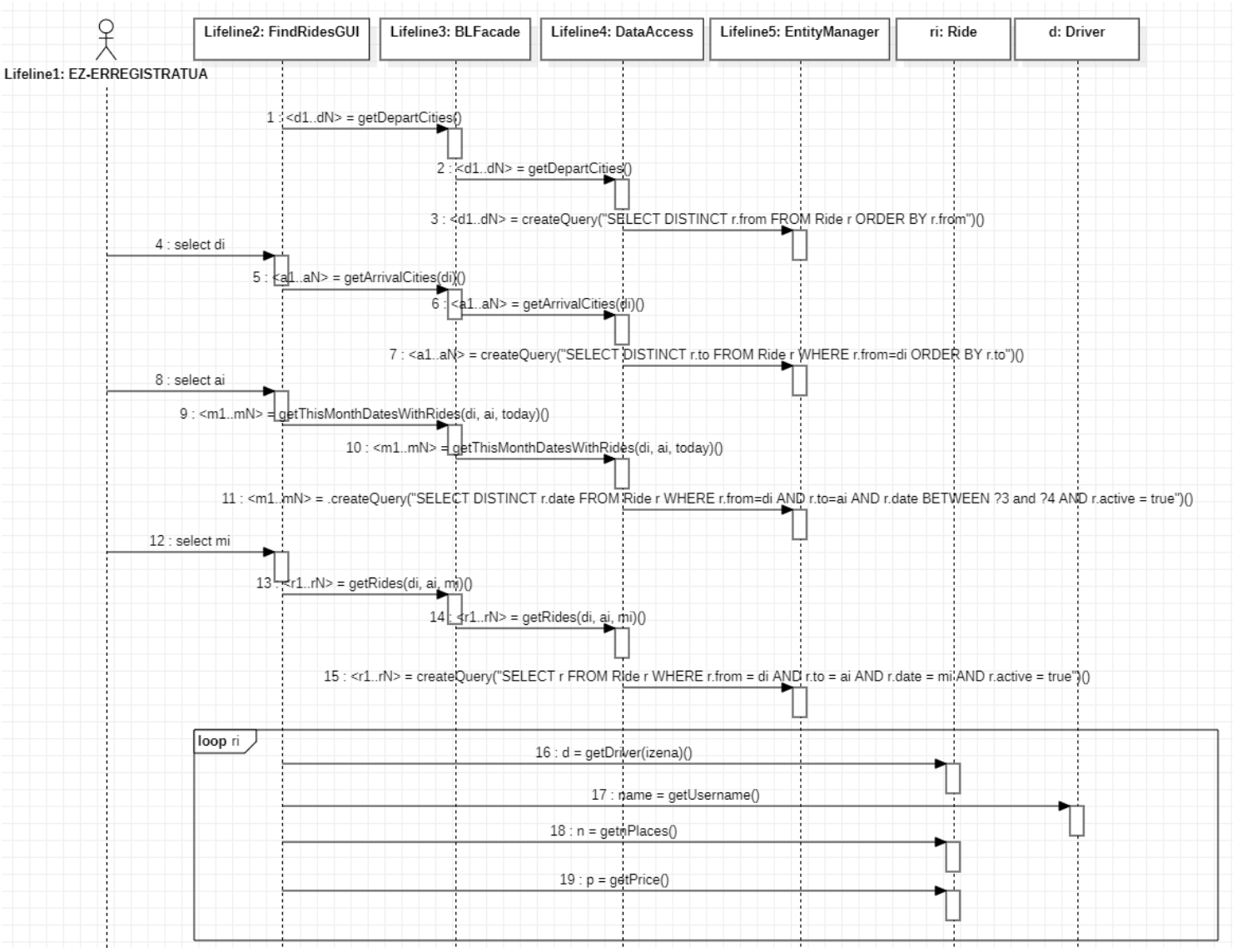
Gertaera fluxuak

Basic flow

1. *Sistema* **nondik norako** hiri guztiak erakutsi eta **egutegia** ipini.
2. *Bidaiaria edo ez-erregistratua* sistemak jarritako jatorrien artean bat aukeratu.
3. *Sistema* jatorriko hiriaren arabera helmuga aukerak eguneratu.
4. *Bidaiaria edo ez-erregistratua* helmuga aukeratu.
5. *Sistema* nondik norako bidaiaren aukeratutako hirien arabera eskuragarri dauden egunak irudikatu.
6. *Bidaiaria edo ez-erregistratua* eskuragarri dauden egunetatik bat aukeratu.
7. *Sistema* aukeratutako eguna kolore batekin markatu eta egun horretako bidaien datuak (gidariaren izena, eserlekuak eta prezioa) erakutsi.

Alternative flow

1. Gidariak aukeratutako egunean ez dago bidaiarik. Bidaien lista hutsa dago.



Bidaia erreserba egin

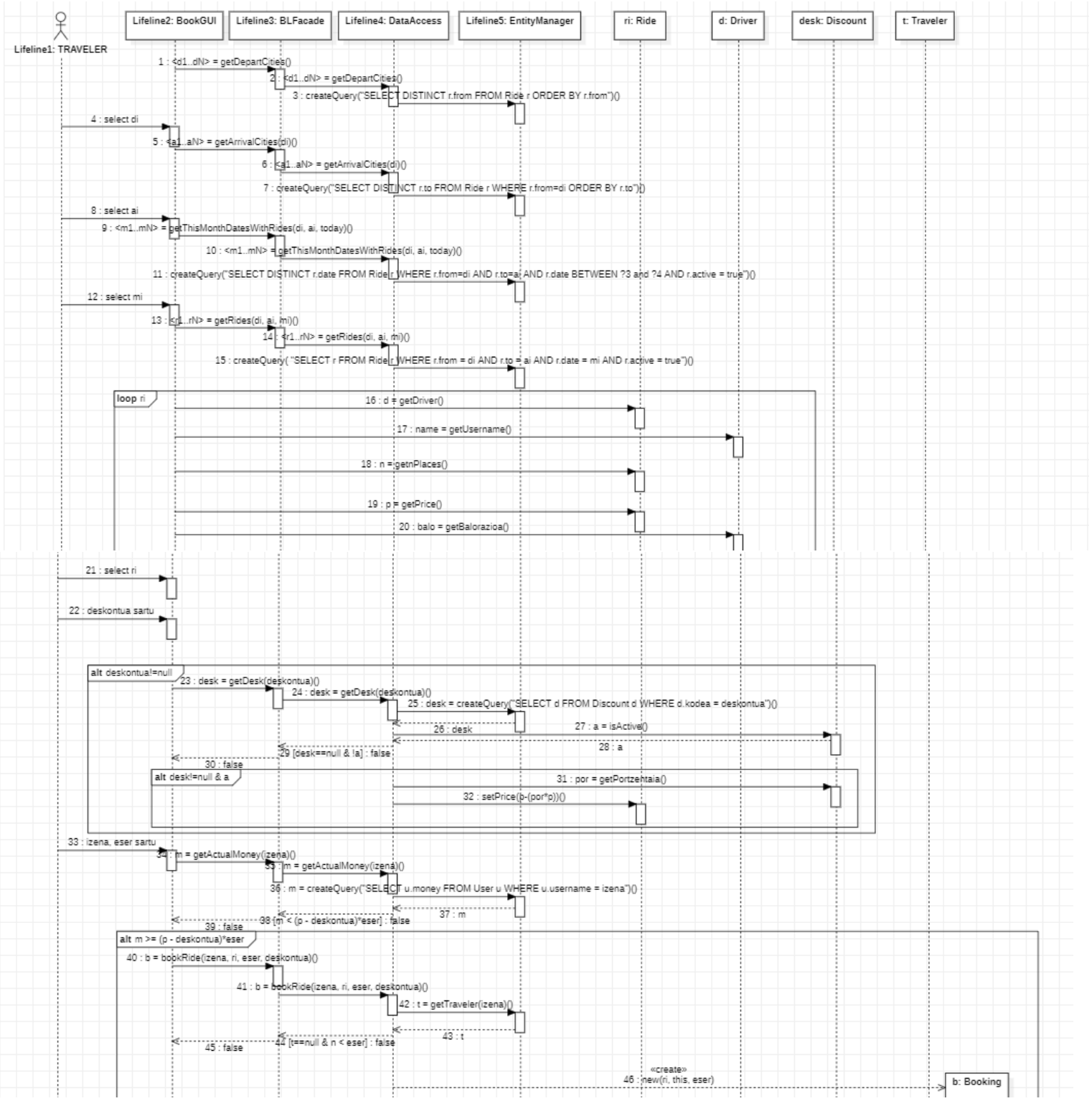
Gertaera fluxuak

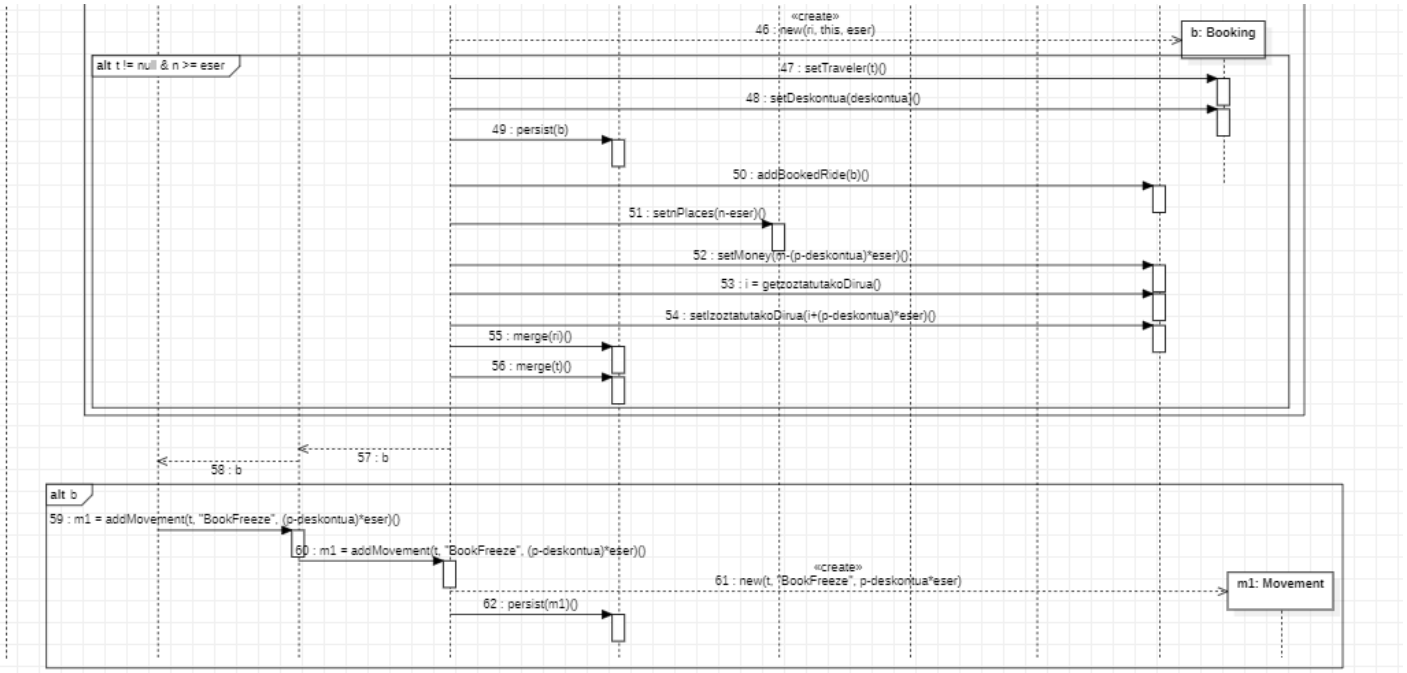
Basic flow

1. *Sistema* **nondik norako** hiri guztiak erakutsi eta **egutegia** ipini.
2. *Bidaiaria* sistemak jarritako jatorrien artean bat aukeratu.
3. *Sistema* jatorriko hiriaren arabera helmuga aukerak eguneratu.
4. *Bidaiaria* helmuga aukeratu.
5. *Sistema* nondik norako bidaiaren aukeratutako hirien arabera eskuragarri dauden egunak irudikatu.
6. *Bidaiaria* eskuragarri dauden egunetatik bat aukeratu.
7. *Sistema* aukeratutako eguna kolore batekin markatu eta egun horretako bidaien datuak (gidariaren izena, eserlekuak, gidariaren balorazioa eta prezioa) erakutsi.
8. *Bidaiariak* bidaia bat aukeratu.
9. *Bidaiaria* deskontu bat sartzeko aukera dauka.
10. *Sistemak* bidaiaria deskonturen bat sartu duen konprobatzen du eta aktiboa edo ez dagoen begiratzen du. Deskontua existitzen bada eta aktiboa badago, bidaiaren prezioa eguneratzen du.
11. *Bidaiaria* izena eta eserlekuak sartu.
12. *Bidaiariak* aukeratutako bidaiaren erreserba egiten du.
13. *Sistemak* bidaiaria existitzen den zihurtatzen du.
14. *Sistemak* erreserba sisteman gordetzen du eta bidaiariari esleitzen dio.
15. *Sistemak* bidaiaren eserlekuak eguneratzen ditu eta bidaiariari dirua kentzen zaio.
16. *Sistema* erreserba ondo burutu bada, bidaiariaren kontuan diruarekin sortutako mugimendua gordetzen du.

Alternative flow

1. Ez daude bidaiak aukeratutako egunean, beraz ezin da erreserba egin.
2. Deskonturik sartzen ez badu, 0 balioa hartzen du.
3. Deskontuaren kodea ez da existitzen. Sistemak erabiltzailea abisatzen du.
4. Ez dago diru nahikoa erreserba egiteko.
5. Libre dauden eserleku baino gehiago erreserbatzen saiatzen ari da. Sistemak erabiltzailea abisatzen du.
6. Erreserba ez da ondo gorde. Sistemak erabiltzailea abisatzen du.


Lifeline1: TRAVELER
Lifeline2: BookGUI
Lifeline3: BLFacade
Lifeline4: DataAccess
Lifeline5: EntityManager
ri: Ride
d: Driver
desk: Discount
t: Traveler
1 : <d1..dN> = getDepartCities()
2 : <d1..dN> = getDepartCities()
3 : createQuery("SELECT DISTINCT r.from FROM Ride r ORDER BY r.from")()
4 : select di
5 : <a1..aN> = getArrivalCities(di)()
6 : <a1..aN> = getArrivalCities(di)()
7 : createQuery("SELECT DISTINCT r.to FROM Ride r WHERE r.from=di ORDER BY r.to")()
8 : select ai
9 : <m1..mN> = getThisMonthDatesWithRides(di, ai, today)()
10 : <m1..mN> = getThisMonthDatesWithRides(di, ai, today)()
11 : createQuery("SELECT DISTINCT r.date FROM Ride r WHERE r.from=di AND r.to=ai AND r.date BETWEEN ?3 and ?4 AND r.active = true")()
12 : select mi
13 : <r1..rN> = getRides(di, ai, mi)()
14 : <r1..rN> = getRides(di, ai, mi)()
15 : createQuery("SELECT r FROM Ride r WHERE r.from = di AND r.to = ai AND r.date = mi AND r.active = true")()
loop ri
16 : d = getDriver()
17 : name = getUsername()
18 : n = getnPlaces()
19 : p = getPrice()
20 : balo = getBalorazioa()
21 : select ri
22 : deskontua sartu
alt deskontua!=null
23 : desk = getDesk(deskontua)()
24 : desk = getDesk(deskontua)()
25 : desk = createQuery("SELECT d FROM Discount d WHERE d.kodea = deskontua")()
26 : desk
27 : a = isActive()
28 : a
29 [desk==null & !a] : false
30 : false
alt desk!=null & a
31 : por = getPortzentaia()
32 : setPrice(p-(por*p))()
33 : izena, eser sartu
34 : m = getActualMoney(izena)()
35 : m = getActualMoney(izena)()
36 : m = createQuery("SELECT u.money FROM User u WHERE u.username = izena")()
37 : m
38 [m < (p - deskontua)*eser] : false
39 : false
alt m >= (p - deskontua)*eser
40 : b = bookRide(izena, ri, eser, deskontua)()
41 : b = bookRide(izena, ri, eser, deskontua)()
42 : t = getTraveler(izena)()
43 : t
44 [t==null & n < eser] : false
45 : false
«create»
46 : new(ri, this, eser)
b: Booking



Bidaia osatuta edo ez osatuta markatu

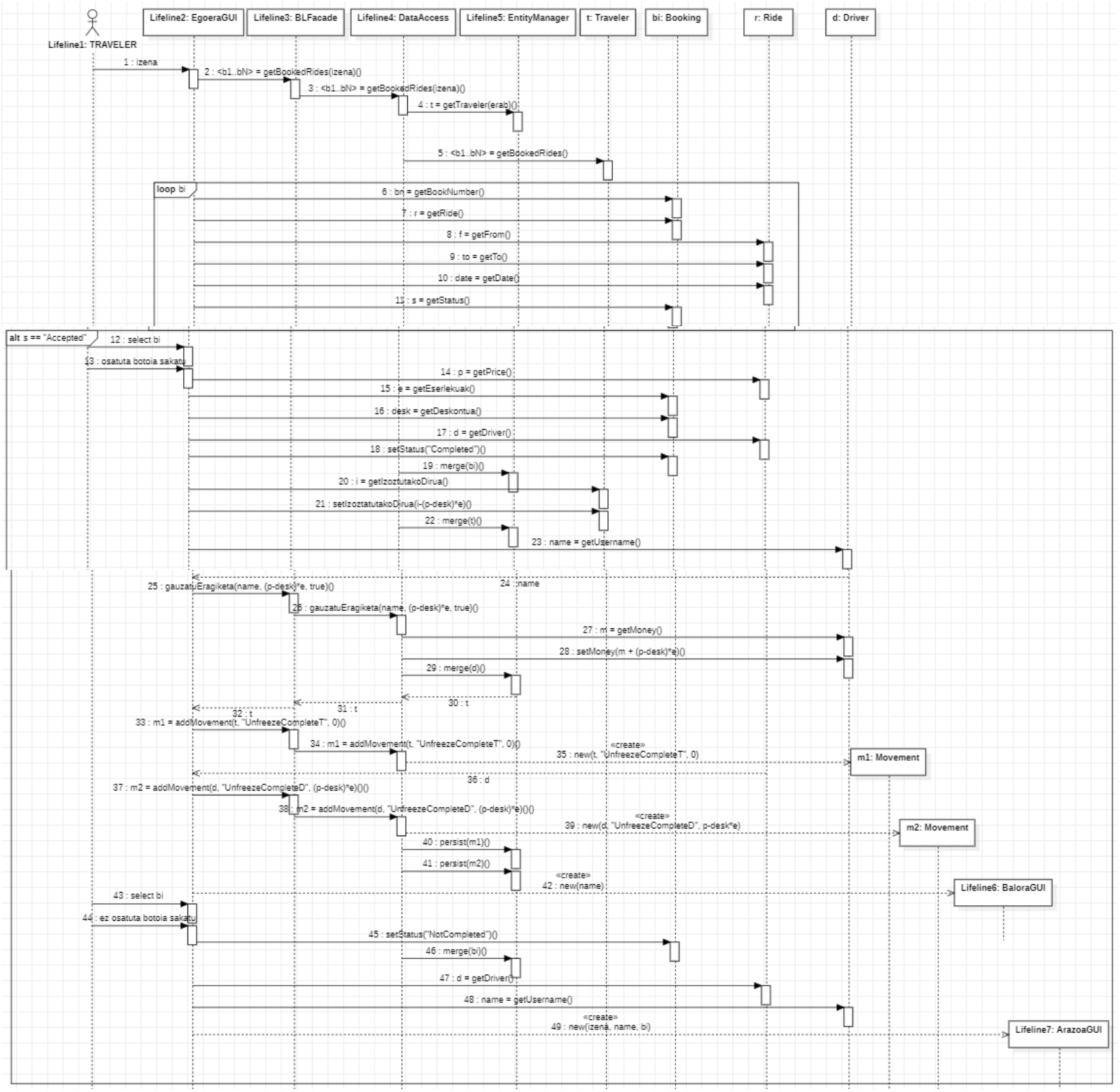
Gertaera fluxuak

Basic flow

1. *Bidaiariak* **izena** eta **egoera** sartzen ditu.
2. *Sistemak* bidaiariari dagokion erreserben lista erakusten du, hain zuzen ere, erreserben zenbakia, nondik nora bidaiak, data eta erreserbaren egoera.
3. *Bidaiaria* "Accepted" egoeran dagoen erreserba eskaera bat aukeratu eta **osatuta** edo **ez osatuta** dagoen markatzen du.
4. *Sistemak* erreserbaren egoera eguneratzen du.
5. *Sistemak* **osatuta** markatzerakoan, gidariari bidaiaren prezioa kontuan sartu eta bidaiariari dirua kontutik atera. Aurreko bi diru-mugimenduak sisteman go
   BaloraGUI pantailara bideratu.
6. *Sistemak* **ez osatuta** markatzerakoan, *ArazoaGUI* pantailaratu bideratu.

Alternative flow

1. Bidaiariak erreserbarik ez du aukeratu. Sistemak erabiltzailea abisatzen du.
2. Aukeratutako erreserbaren egoera ez dago zehaztuta. Sistemak erabiltzailea abisatzen du.
3. Bidaia gaur baino lehenagokoa izan behar da. Sistemak erabiltzailea abisatzen du.

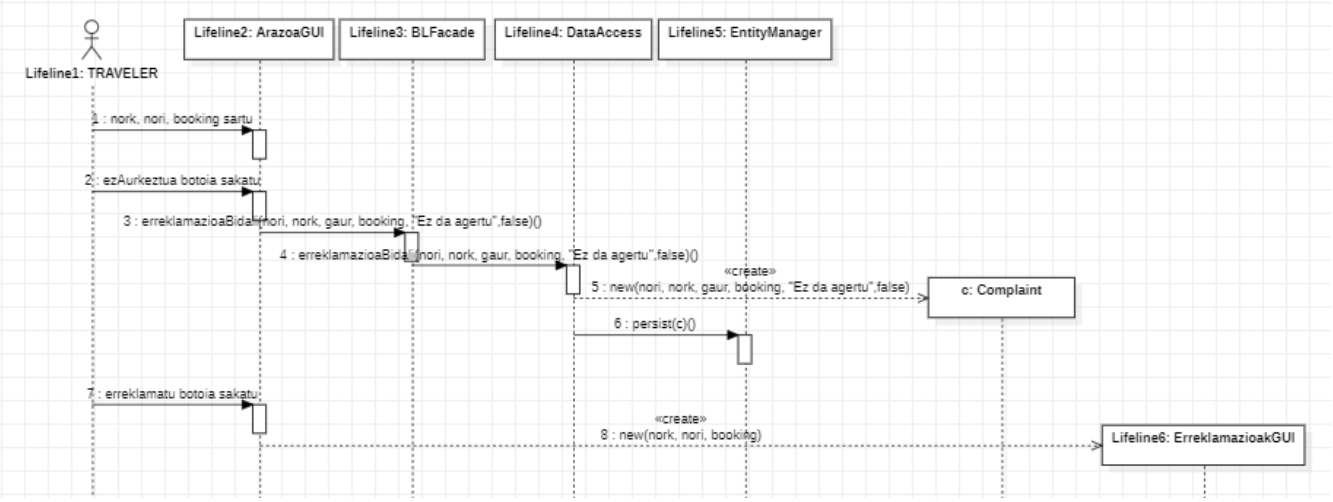

Lifeline1: TRAVELER
Lifeline2: EgoeraGUI
Lifeline3: BLFacade
Lifeline4: DataAccess
Lifeline5: EntityManager
t: Traveler
bi: Booking
r: Ride
d: Driver
1 : izena
2 : <b1..bN> = getBookedRides(izena)()
3 : <b1..bN> = getBookedRides(izena)()
4 : t = getTraveler(erab)()
5 : <b1..bN> = getBookedRides()
loop bi
6 : bn = getBookNumber()
7 : r = getRide()
8 : f = getFrom()
9 : to = getTo()
10 : date = getDate()
11 : s = getStatus()
alt s == "Accepted"
12 : select bi
13 : osatuta botoia sakatu
14 : p = getPrice()
15 : e = getEserlekuak()
16 : desk = getDeskontua()
17 : d = getDriver()
18 : setStatus("Completed")()
19 : merge(bi)()
20 : i = getIzoztutakoDirua()
21 : setIzoztatutakoDirua(i-(p-desk)*e)()
22 : merge(t)()
23 : name = getUsername()
24 : name
25 : gauzatuEragiketa(name, (p-desk)*e, true)()
26 : gauzatuEragiketa(name, (p-desk)*e, true)()
27 : m = getMoney()
28 : setMoney(m + (p-desk)*e)()
29 : merge(d)()
30 : t
31 : t
32 : t
33 : m1 = addMovement(t, "UnfreezeCompleteT", 0)()
34 : m1 = addMovement(t, "UnfreezeCompleteT", 0)()
«create»
35 : new(t, "UnfreezeCompleteT", 0)
m1: Movement
36 : d
37 : m2 = addMovement(d, "UnfreezeCompleteD", (p-desk)*e)()()
38 : m2 = addMovement(d, "UnfreezeCompleteD", (p-desk)*e)()()
«create»
39 : new(d, "UnfreezeCompleteD", p-desk*e)
m2: Movement
40 : persist(m1)()
41 : persist(m2)()
«create»
42 : new(name)
Lifeline6: BaloraGUI
43 : select bi
44 : ez osatuta botoia sakatu
45 : setStatus("NotCompleted")()
46 : merge(bi)()
47 : d = getDriver()
48 : name = getUsername()
«create»
49 : new(izena, name, bi)
Lifeline7: ArazoaGUI

Arazoak adierazi

Gertaera fluxuak

Basic flow

1. *Bidaiariak* **nork**, **nori** eta **erreserba** sartu.
2. *Bidaiariak* ezAurkeztua botoia sakatu, bidaiako egunean ez delako agertu.
3. *Sistemak* erreklamazio bat bidaliko du gidariaren izenean adierazteko bidaiaria ez dela agertu eta sortutako erreklamazioa gorde.
4. *Bidaiariak* erreklamatu botoia sakatu, erreklamazioa jartzeko.
5. *Sistemak ErreklamazioakGUI* pantailara bideratu.



Alerta aurkituak

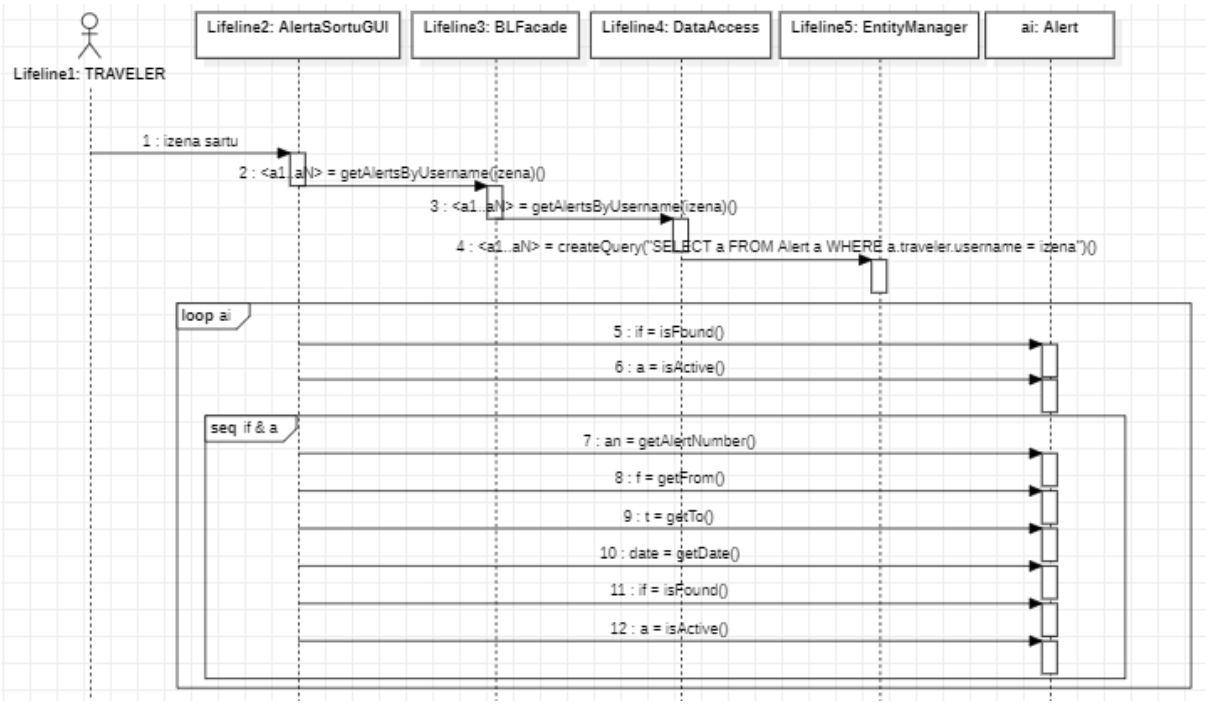
Gertaera fluxuak

Basic flow

1. *Bidaiariak* izena sartzen du.
2. *Sistemak* bidaiariari dagokion alerten lista erakusten du, hain zuzen ere, alerta zenbakia, nondik, nora, data, aurkituta dagoen eta aktibatuta dagoen.

Alternative flow

1. Alertarik ez da agertzen. Sistemak erabiltzailea abisatzen du.



Alertak sortu

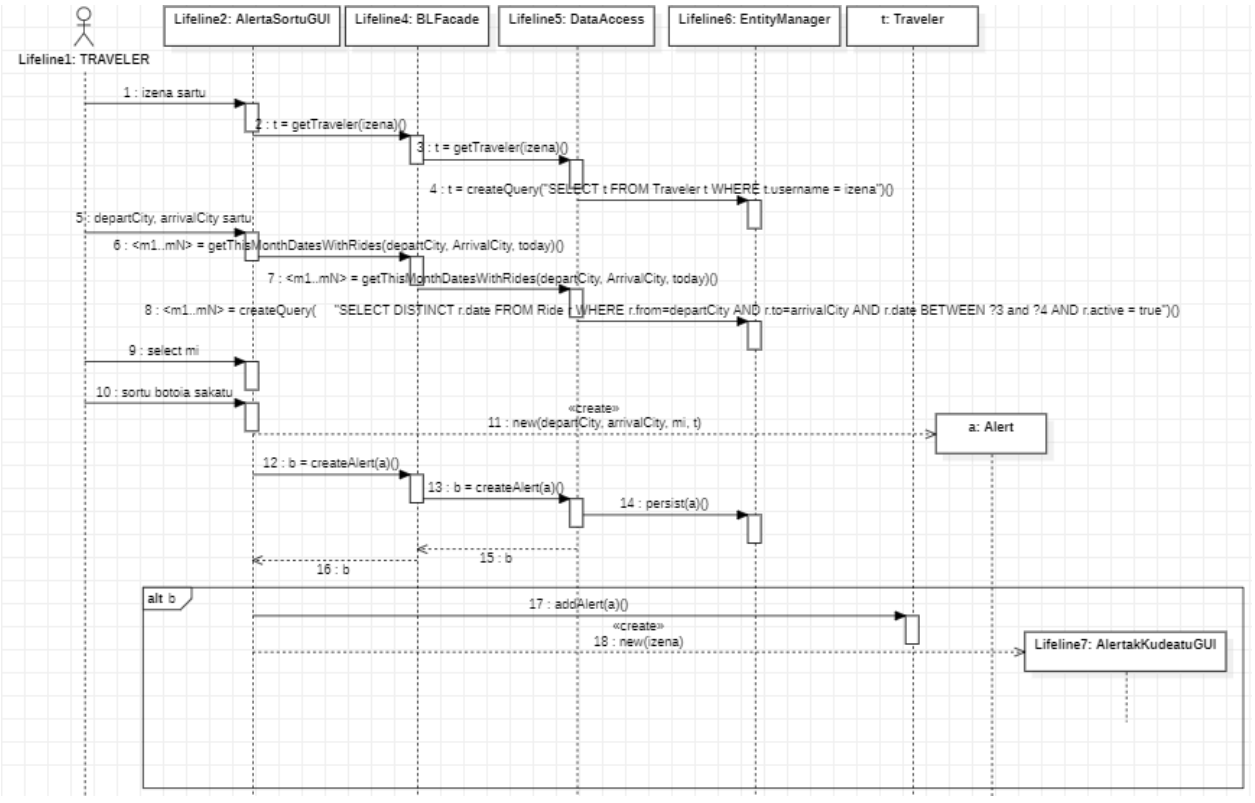
Gertaera fluxuak

Basic flow

1. *Bidaiariak* nondik, norako bidaia sartzen du.
2. *Sistemak* egutegia erakusten du.
3. *Bidaiariak* egun bat aukeratu.
4. *Bidaiariak* izena sartzen du.
5. *Bidaiariak* sortu botoia sakatzen du.
6. *Sistemak* alerta berria sortu eta gordetzen du.
7. *Sistemak* alerta ondo sortuta konprobatzen du eta bidaiariari esleitu.
8. *Sistemak AlertakKudeatuGUI* pantailara bideratu.

Alternative flow

1. Alerta berria ez da sortu. Sistemak erabiltzailea abisatzen du.



Alertak kudeatu

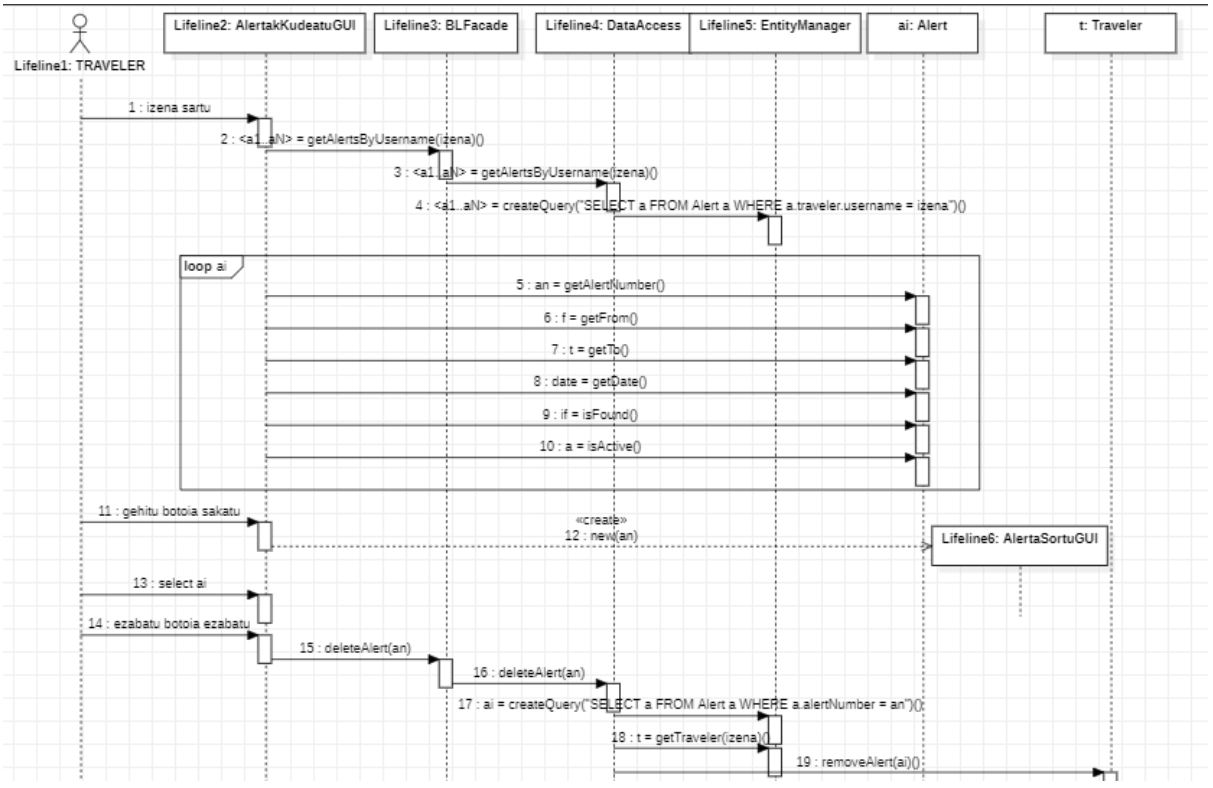
Gertaera fluxuak

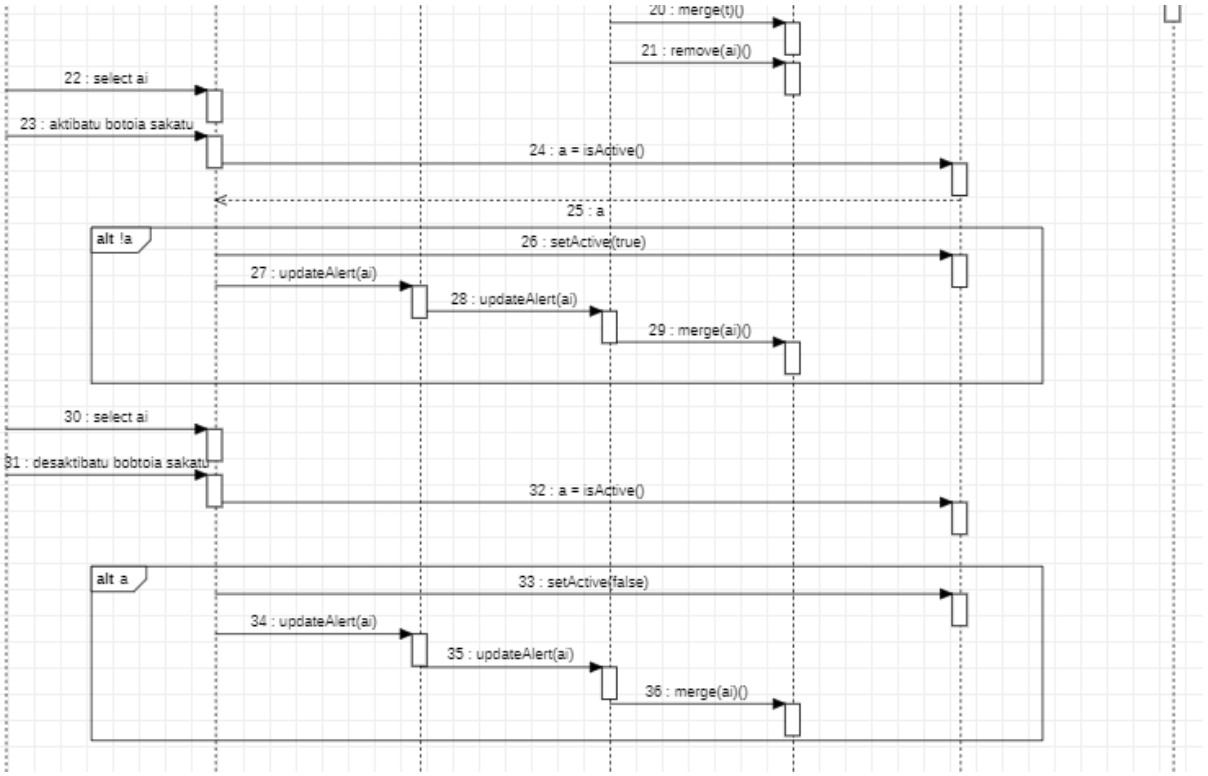
Basic flow

1. *Bidaiariak* izena sartzen du.
2. *Sistemak* bidaiariari dagokion alerten lista erakusten du, hain zuzen ere, alerta zenbakia, nondik, nora, data, aurkituta dagoen edo ez eta aktibatuta dagoen edo ez.
3. *Bidaiariak* gehitu botoia sakatu.
4. *Sistemak AlertaSortuGUI* pantailara bideratu.
5. *Bidaiariak* alerta bau aukeratu eta ezabatu botoia sakatu.
6. *Sistemak* bidaiariari alerta kendu, eguneratu eta sistematik kendu.
7. *Bidaiariak* alerta bau aukeratu eta aktibatu botoia sakatu.
8. *Sistemak* aukeratutako alertaren egoera konprobatu, aktibatuta ez badago, aktibatu eta eguneratu.
9. *Bidaiariak* alerta bau aukeratu eta desaktibatu botoia sakatu.
10. *Sistemak* aukeratutako alertaren egoera konprobatu, aktibatuta badago, desaktibatu eta eguneratu.

Alternative flow

1. Bidaiariak erreserbarik ez du aukeratu. Sistemak erabiltzailea abisatzen du.




20 : merge(t)()
21 : remove(ai)()
22 : select ai
23 : aktibatu botoia sakatu
24 : a = isActive()
25 : a
alt !a
26 : setActive(true)
27 : updateAlert(ai)
28 : updateAlert(ai)
29 : merge(ai)()
30 : select ai
31 : desaktibatu bobtoia sakatu
32 : a = isActive()
alt a
33 : setActive(false)
34 : updateAlert(ai)
35 : updateAlert(ai)
36 : merge(ai)()

Bidaia sortu

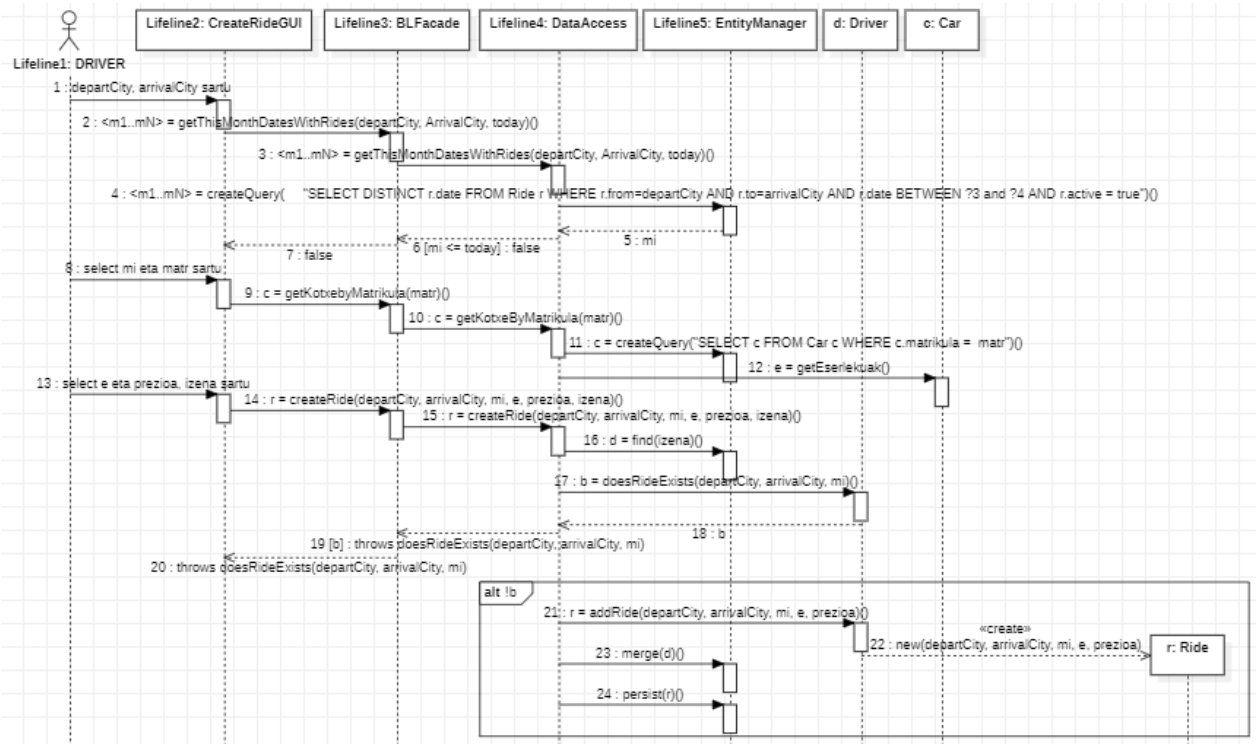
Gertaera fluxuak

Basic flow

1. *Gidaria* **jatorria**, **helmuga**, **prezioa** eta **izena** sartu.
2. *Sistema* egutegia erakutsi.
3. *Gidaria* **bidaia eguna** aukeratu.
4. *Gidaria* kotxearen **matrikula** sartu.
5. *Sistema* matrikula hori dagokion kotxea erakutsi.
6. *Gidaria* **kotxea** eta **eserleku kopurua** aukeratu.
7. *Sistemak* bidaia sortzen du eta gidariari esleitzen dio.

Alternative flow

1. Datuak falta dira. Amaitu.
2. **eserleku kopurua** edo **prezioa** ez dira zenbakiak. Sistemak erabiltzailea abisatzen du.
3. **bidaia eguna** pasa da. Sistemak erabiltzailea abisatzen du.
4. **bidaia** dagoeneko existitzen da **gidari** honentzat. Bidaia ez da sortzen. Sistemak erabiltzailea abisatzen du.



Bidaia kantzelatu

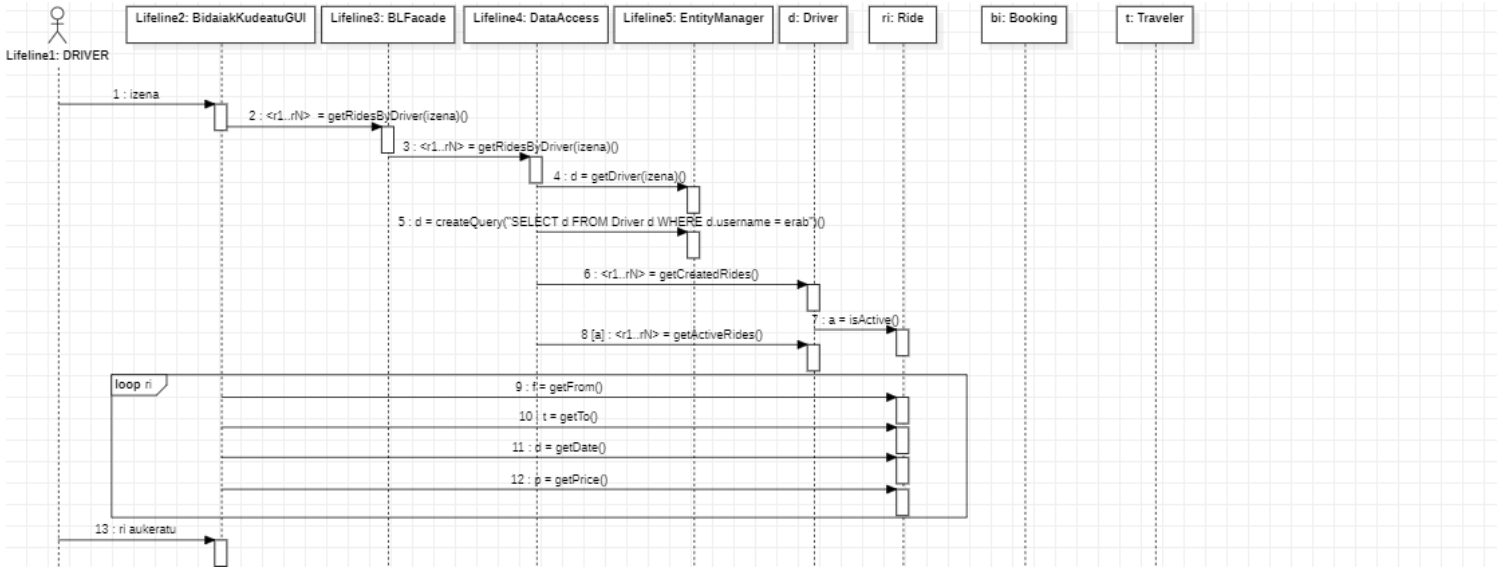
Gertaera fluxuak

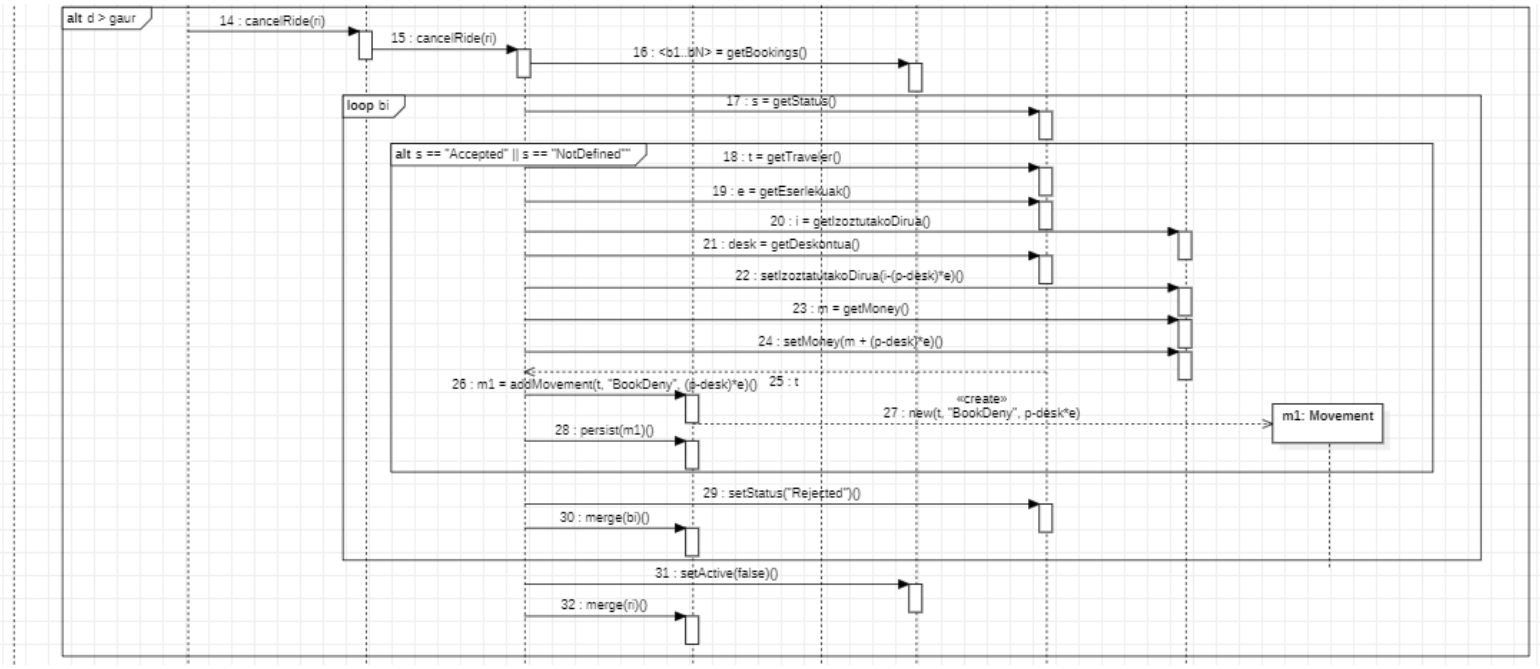
Basic flow

1. *Gidaria* **izena** sartzen du.
2. *Sistemak* gidariari dagokion bidaien lista erakusten du, hain zuzen ere, nondik norako bidaiak, data eta prezioa.
3. *Gidaria* bidaia aukeratu.
4. *Sistemak* dagokion bidaia kantzelatzen du. Horretarako bidaiaren erreserben lista aztertu behar du.
5. *Sistemak* "Accepted edo "NotDefined" egoeran dauden erreserbak aukeratu eta bidaiariari bidaiaren prezioa kontuan sartu eta sortutako diru-mugimendua gorde.
6. *Sistemak* erreserbaren egoera "Rejected" egoerara aldatu eta eguneratu.
7. *Sistemak* bidaiaren egoera false egoerara eguneratu.

Alternative flow

1. Gidariak bidaiarik ez du aukeratu. Sistemak erabiltzailea abisatzen du.
2. Aukeratutako bidaiaren egoera jadanik zehaztuta dago. Sistemak erabiltzailea abisatzen du.
3. Dagoeneko bidaiaren data pasa da. Sistemak erabiltzailea abisatzen du.





Erreserba onartu edo baztertu

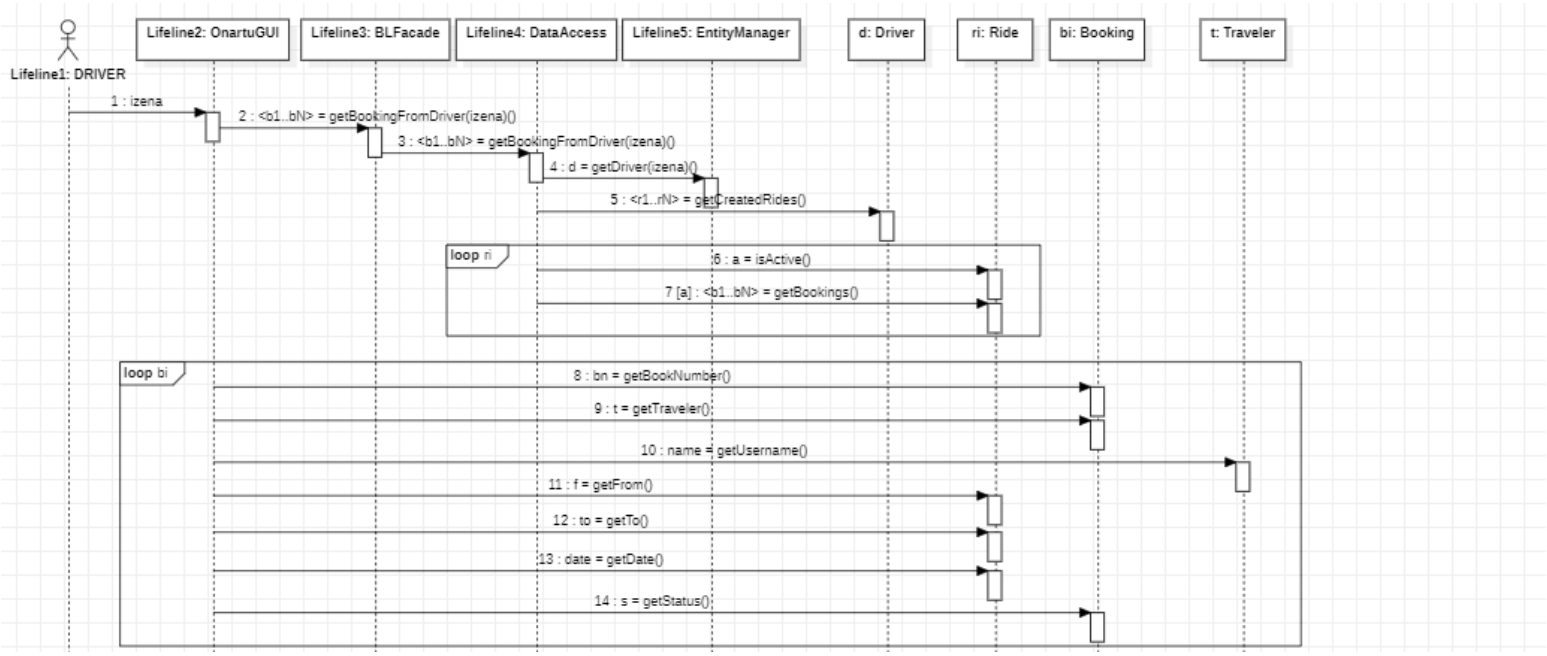
Gertaera fluxuak

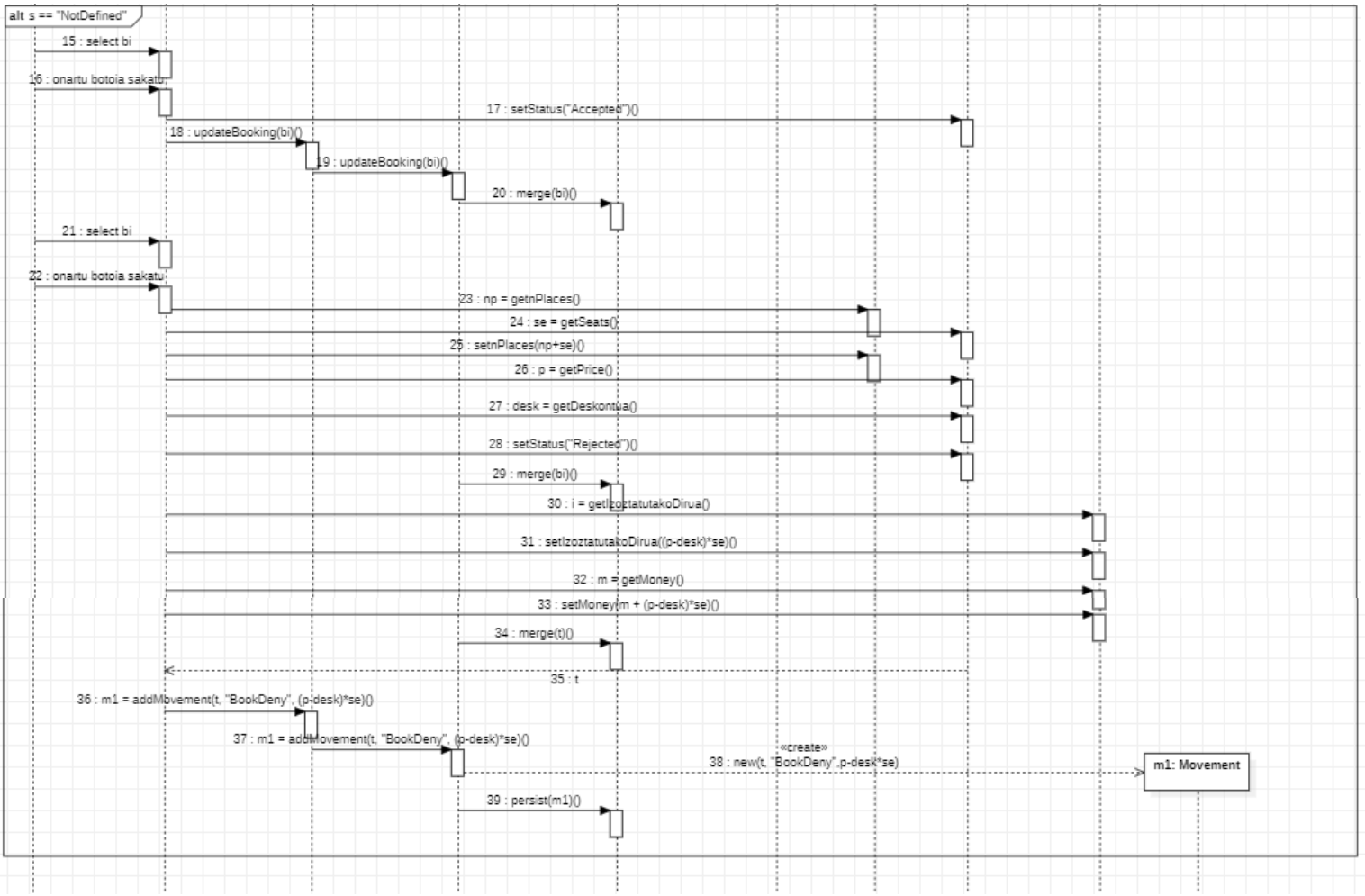
Basic flow

1. *Gidaria* **izena** eta **egoera** sartu
2. *Sistemak* gidariari dagokion erreserben lista erakusten du, hain zuzen ere, erreserben zenbakia, bidaiariaren izenak, nondik nora bidaiak, data eta erreserbaren egoera.
3. *Gidaria* erreserba eskaera bat aukeratu eta **onartu** edo **baztertzen** du.
4. *Bidaiaria* "NotDefined" egoeran dagoen erreserba eskaera bat aukeratu eta **onartu** edo **ez onartu** markatzen du.
5. *Sistemak* **onartu** markatzerakoan, erreserbaren egoera "Accepted" egoerara aldatu eta eguneratu.
6. *Sistemak* **ez onartu** markatzerakoan, erreserbaren egoera "Rejected" egoerara aldatu eta eguneratu. Bidaiaren eserlekuak eguneratu. Bidaiariari bidaiaren prezioa kontuan sartu eta sortutako diru-mugimendua sisteman gorde.

Alternative flow

1. Gidariak erreserbarik ez du aukeratu. Sistemak erabiltzailea abisatzen du.
2. Aukeratutako erreserbaren egoera jadanik zehaztuta dago. Sistemak erabiltzailea abisatzen du.




alt s == "NotDefined"
15 : select bi
16 : onartu botoia sakatu
17 : setStatus("Accepted")()
18 : updateBooking(bi)()
19 : updateBooking(bi)()
20 : merge(bi)()
21 : select bi
22 : onartu botoia sakatu
23 : np = getnPlaces()
24 : se = getSeats()
25 : setnPlaces(np+se)()
26 : p = getPrice()
27 : desk = getDeskontua()
28 : setStatus("Rejected")()
29 : merge(bi)()
30 : i = getIzoztatutakoDirua()
31 : setIzoztatutakoDirua((p-desk)*se)()
32 : m = getMoney()
33 : setMoney(m + (p-desk)*se)()
34 : merge(t)()
35 : t
36 : m1 = addMovement(t, "BookDeny", (p-desk)*se)()
37 : m1 = addMovement(t, "BookDeny", (p-desk)*se)()
«create»
38 : new(t, "BookDeny" p-desk*se)
m1: Movement
39 : persist(m1)()

Kotxea gehitu

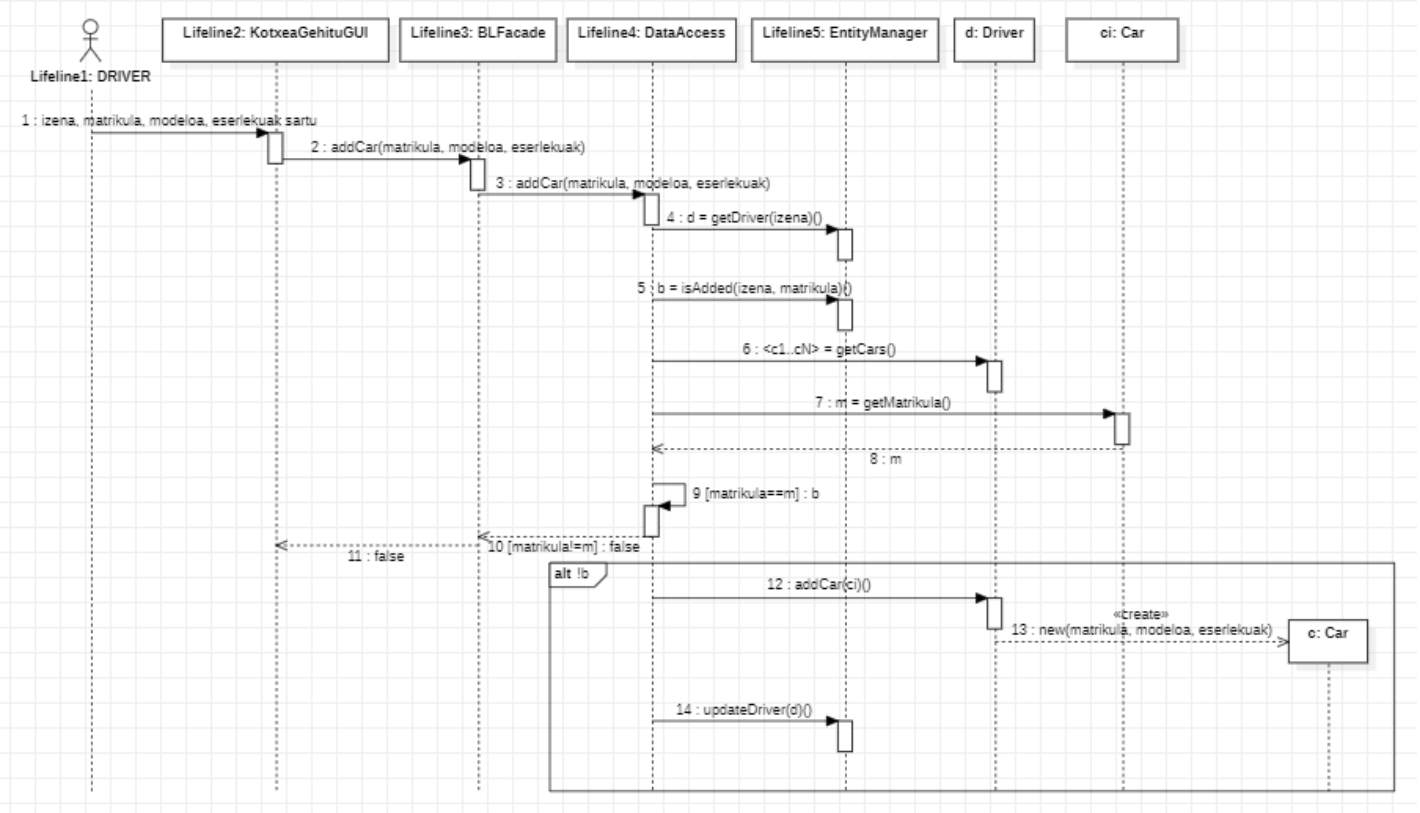
Gertaera fluxuak

Basic flow

1. *Gidaria* **matrikula, modeloa** eta **eserleku kopurua** ipini.
2. *Sistemak* aukeratutako datuen arabera kotxea gehitzen du.
3. *Sistemak* kotxea gehitzeko dagoeneko sortuta badago konprobatzen du. Sortuta ez badago gidariari kotxea esleitu eta eguneratu.

Alternative flow

1. Datuak falta dira. Sistemak erabiltzailea abisatzen du.
2. **eserleku kopurua** ez da zenbakia. Sistemak erabiltzailea abisatzen du.
3. **kotxea** jada existitzen da **gidari** honentzat. Kotxea ez da sortzen. Sistemak erabiltzailea abisatzen du.



Kotxeak kudeatu

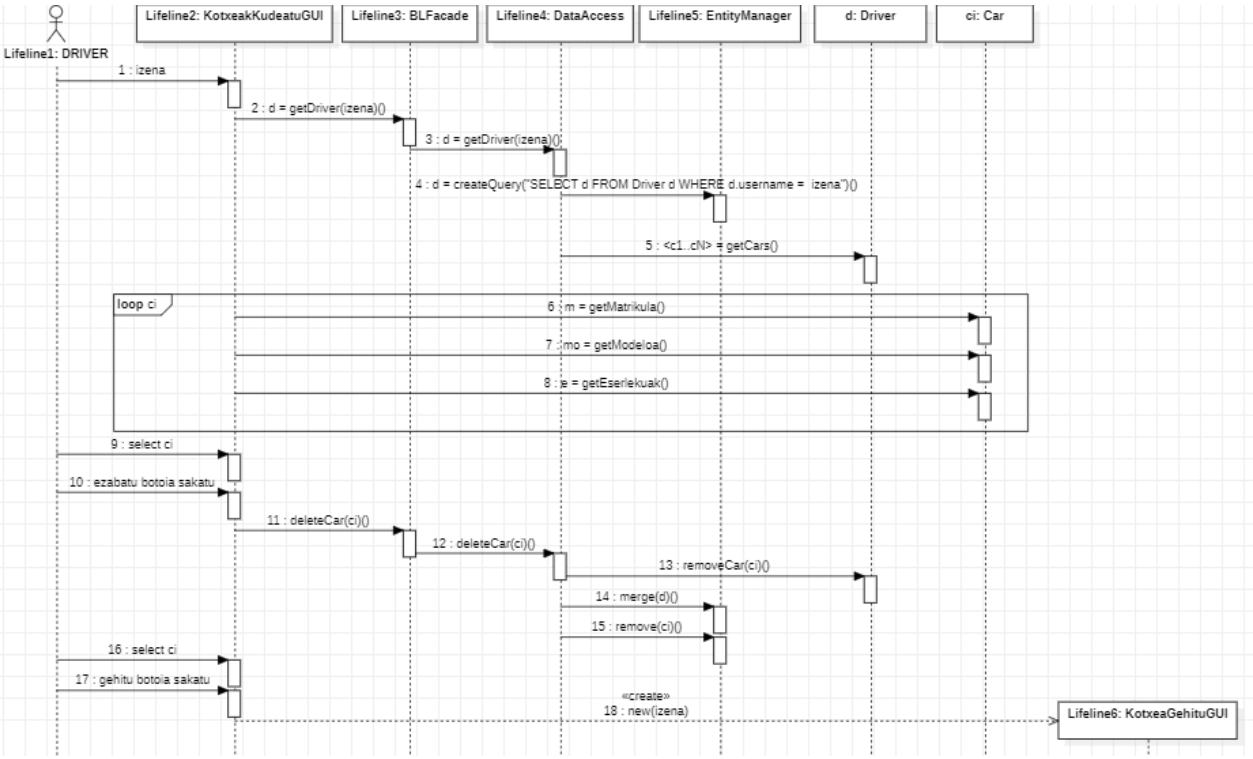
Gertaera fluxuak

Basic flow

1. *Gidariak* **izena** sartzen du.
2. *Sistemak* gidariari dagokion kotxeen lista erakusten du, hain zuzen ere, matrikula, modeloa eta eserlekuak.
3. *Gidariak* kotxe bat aukeratu eta ezabatu botoia sakatu.
4. *Sistemak* kotxea ezabatzen du.
5. *Gidariak* gehitu botoia sakatu eta *KotxeaGehituGUI* pantailara bideratu.

Alternative flow

1. Kotxeen lista hutsa da. Ez da kotxerik agertzen.



Bezeroak kudeatu

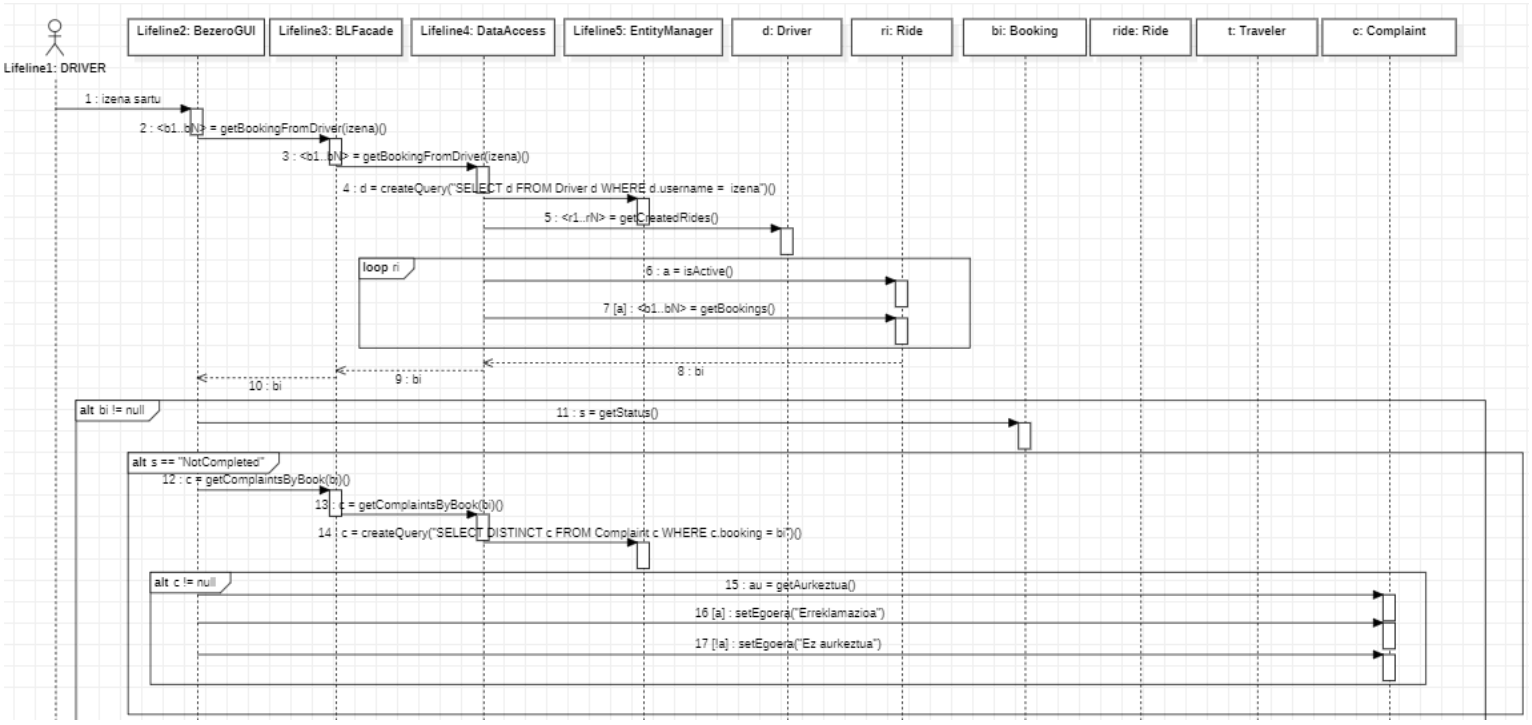
Gertaera fluxuak

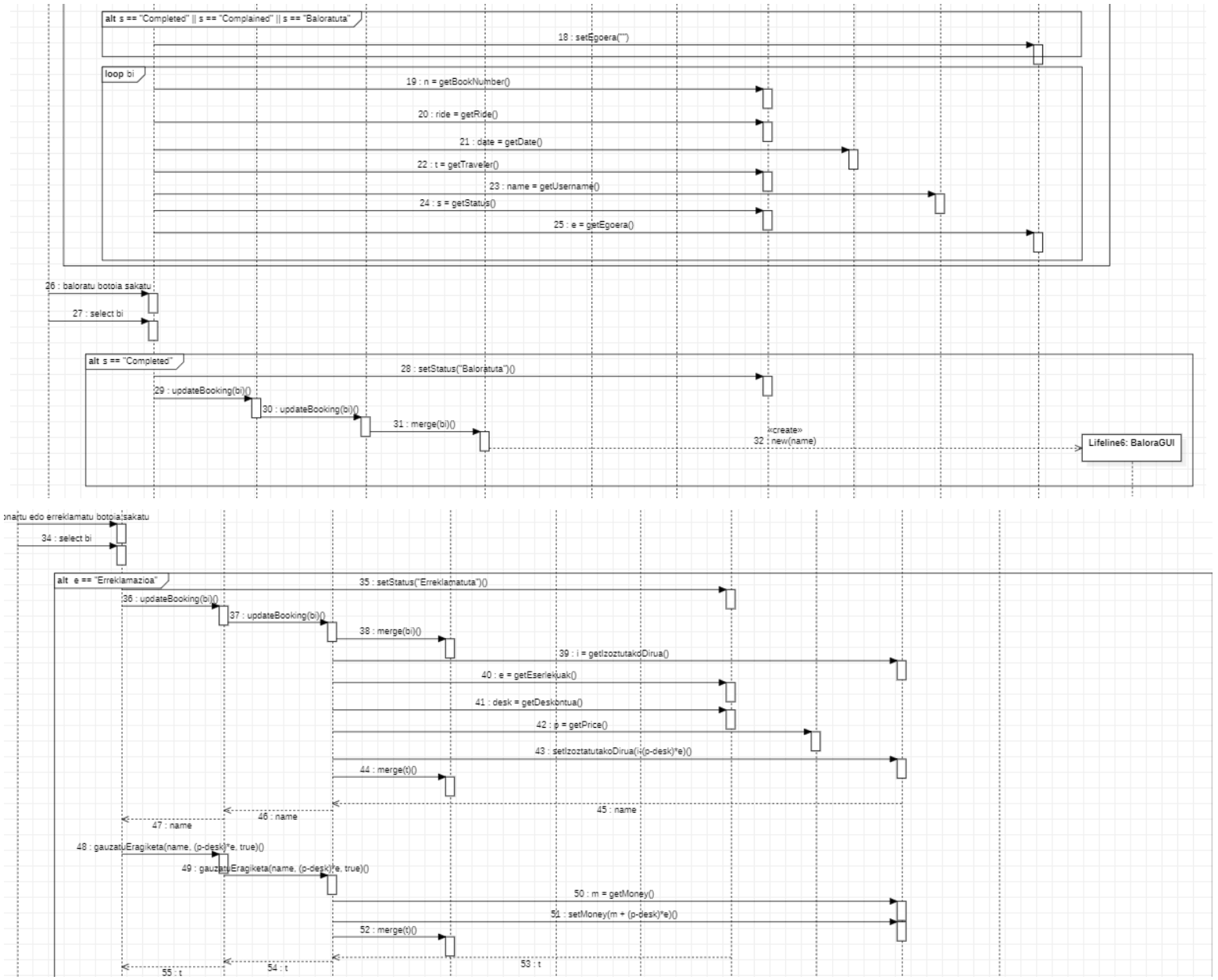
Basic flow

1. *Gidariak* **izena** sartzen du.
2. *Sistemak* gidariari dagokion erreserben lista eskuratzen du eta gidariak sortutako bidaia guztietatik aktibo egoeran daudenen erreserbak hartu.
3. *Sistemak* erreserben egoera eskuratu.
4. *Sistemak* "NotCompleted" egoeran dauden kasuetan, erreklamazioak hartu eta aurkeztua edo ez dagoen konprobatu. Aurkeztua izan den kasuetan, erreklamazioaren egoeran "Erreklamazioa" jarri, bestela "Ez aurkeztua".
5. *Sistemak* "Completed", "Complained" edo "Baloratu" egoeran daudenei, erreklamazioaren egoeran ezer ez jarri.
6. *Sistemak* erreserben datu hauek erakutsi: erreserba zenbakia, bidaiaren data, bidaiariaren izena, erreserbaren egoera eta erreklamazioaren egoera.
7. *Gidariak* erreserba bat aukeratu
8. *Gidariak* baloratu botoia sakatu.
9. *Sistemak* erreserbaren egoeran "Completed" badago konprobatu, **BaloraGUI** pantailara bideratu eta erreserbaren egoera "Baloratuta" egoerara eguneratu.
10. *Gidariak* onartu/erreklamatu botoia sakatu.
11. *Sistemak* konprobatu erreklamazioaren egoera.
12. *Sistemak* erreklamazioaren egoera "Erreklamazioa"-n badago, erreserbaren egoera eguneratu, bidaiariari dirua bueltatu eta sortutako diru-mugimenduak gorde. Gidariaren erreklamazio kopuruari bat gehitu.
13. *Sistemak* erreklamazioaren egoera "Ez aurkeztua"-n badago, erreserbaren egoea eguneratu, bidaiariari dirua kendu, gidariari dirua bueltatu eta sortutako diru-mugimenduak gorde. Bidaiariaren erreklamazio kopuruari bat gehitu.

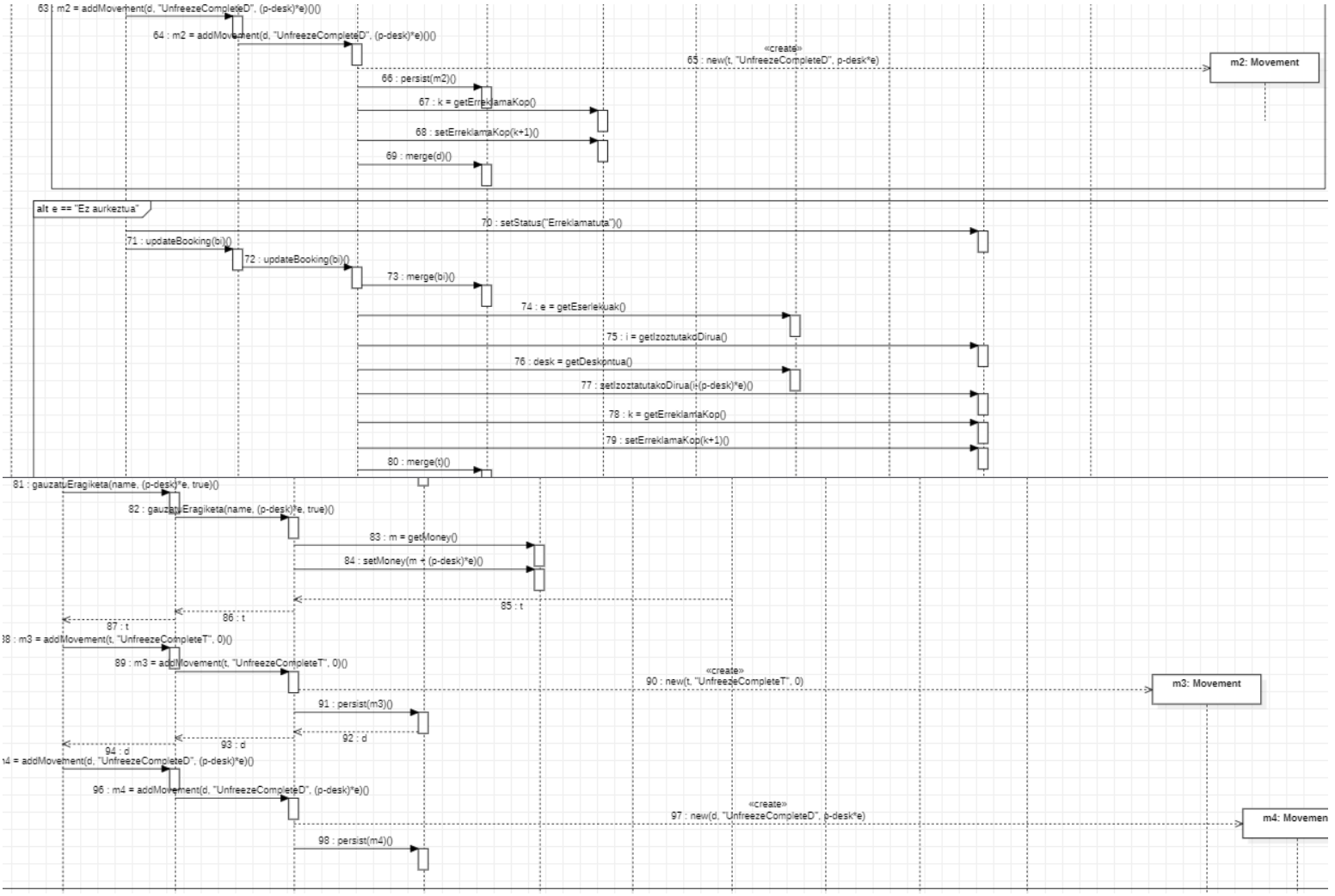
Alternative flow

1. Erreserben lista hutsa da. Ez da errserbarik agertzen.
2. Ez dago erreklamaziorik. Ez da agertzen.




alt s == "Completed" || s == "Complained" || s == "Baloratuta"
18 : setEgoera("")
loop bi
19 : n = getBookNumber()
20 : ride = getRide()
21 : date = getDate()
22 : t = getTraveler()
23 : name = getUsername()
24 : s = getStatus()
25 : e = getEgoera()
26 : baloratu botoia sakatu
27 : select bi
alt s == "Completed"
28 : setStatus("Baloratuta")()
29 : updateBooking(bi)()
30 : updateBooking(bi)()
31 : merge(bi)()
«create»
32 : new(name)
Lifeline6: BaloraGUI

34 : select bi
alt e == "Erreklamazioa"
35 : setStatus("Erreklamatuta")()
36 : updateBooking(bi)()
37 : updateBooking(bi)()
38 : merge(bi)()
39 : i = getIzoztutakoDirua()
40 : e = getEserlekuak()
41 : desk = getDeskontua()
42 : p = getPrice()
43 : setIzoztatutakoDirua(i-(p-desk)*e)()
44 : merge(t)()
45 : name
46 : name
47 : name
48 : gauzatuEragiketa(name, (p-desk)*e, true)()
49 : gauzatuEragiketa(name, (p-desk)*e, true)()
50 : m = getMoney()
51 : setMoney(m + (p-desk)*e)()
52 : merge(t)()
53 : t
54 : t
55 : t


63 : m2 = addMovement(d, "UnfreezeCompleteD", (p-desk)*e)()()
64 : m2 = addMovement(d, "UnfreezeCompleteD", (p-desk)*e)()()
«create»
65 : new(t, "UnfreezeCompleteD", p-desk*e)
m2: Movement
66 : persist(m2)()
67 : k = getErreklamaKop()
68 : setErreklamaKop(k+1)()
69 : merge(d)()
alt e == "Ez aurkeztua"
70 : setStatus("Erreklamatuta")()
71 : updateBooking(bi)()
72 : updateBooking(bi)()
73 : merge(bi)()
74 : e = getEserlekuak()
75 : i = getIzoztutakoDirua()
76 : desk = getDeskontua()
77 : getIzoztatutakoDirua(i-(p-desk)*e)()
78 : k = getErreklamaKop()
79 : setErreklamaKop(k+1)()
80 : merge(t)()
81 : gauzatuEragiketa(name, (p-desk)*e, true)()
82 : gauzatuEragiketa(name, (p-desk)*e, true)()
83 : m = getMoney()
84 : setMoney(m + (p-desk)*e)()
85 : t
86 : t
87 : t
88 : m3 = addMovement(t, "UnfreezeCompleteT", 0)()
89 : m3 = addMovement(t, "UnfreezeCompleteT", 0)()
«create»
90 : new(t, "UnfreezeCompleteT", 0)
m3: Movement
91 : persist(m3)()
92 : d
93 : d
94 : d
m4 = addMovement(d, "UnfreezeCompleteD", (p-desk)*e)()
96 : m4 = addMovement(d, "UnfreezeCompleteD", (p-desk)*e)()
«create»
97 : new(d, "UnfreezeCompleteD", p-desk*e)
m4: Movemen
98 : persist(m4)()

Erabiltzailea ezabatu

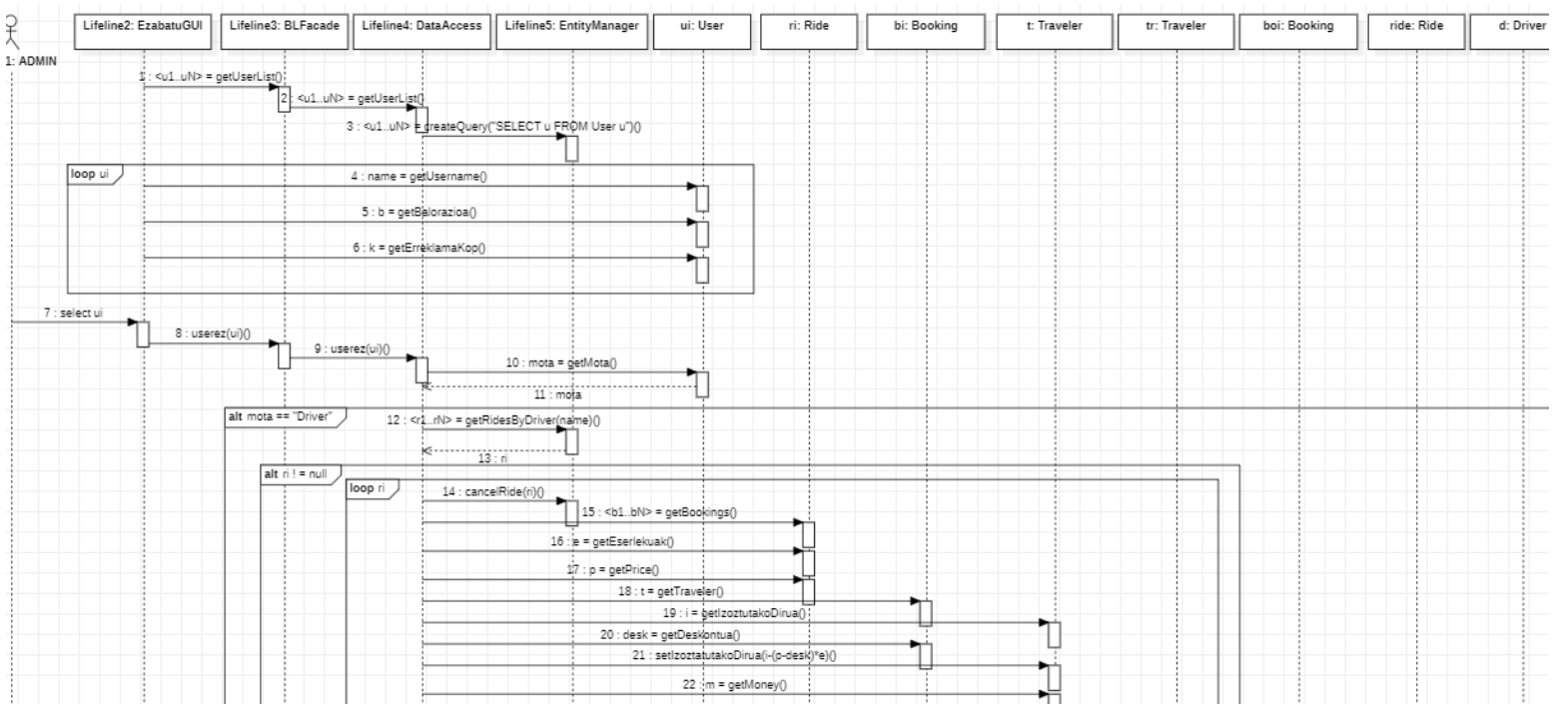
Gertaera fluxuak

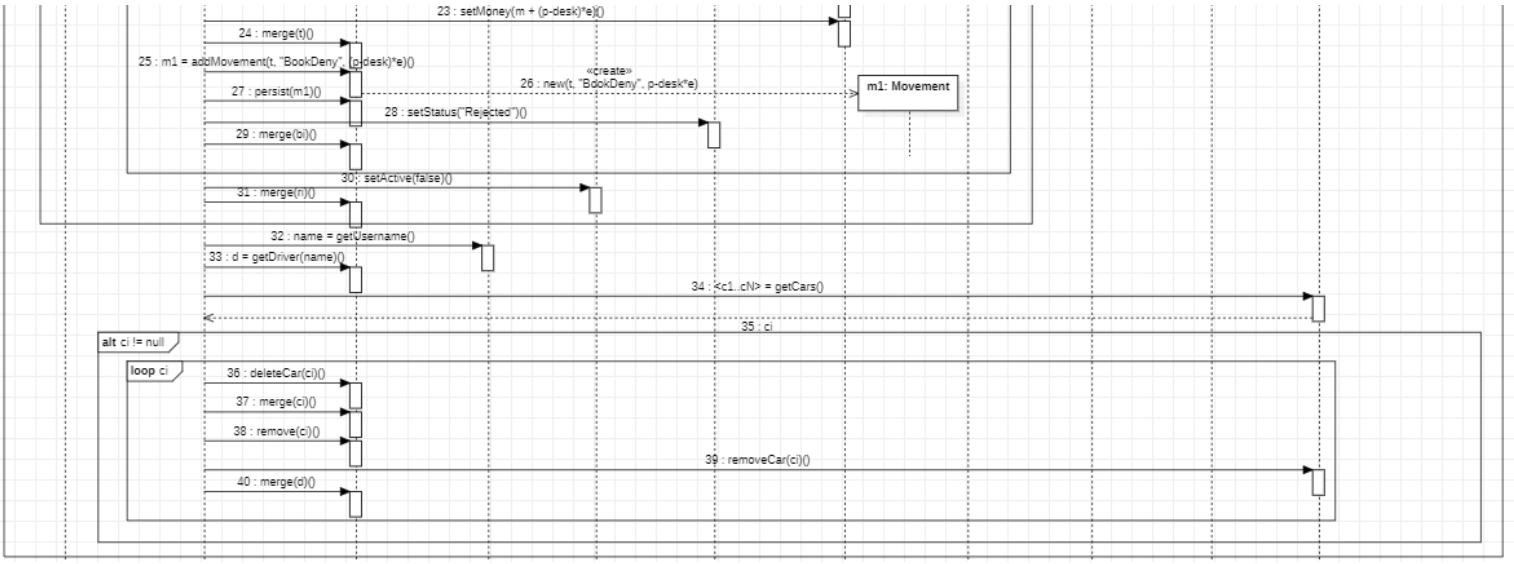
Basic flow

1. *Sistemak* erabiltzaileen lista erakusten du, hain zuzen ere, izena, balorazioa eta erreklamazio kopurua.
2. *Adminak* erabiltzaile bat aukeratu.
3. *Sistemak* erabiltzailea ezabatzeko, mota konprobatu.
4. *Sistemak* gidaria bada, bidaia kantzelatzen du. Horretarako, bidaiaren erreserba egin duten bidaiariei dirua bueltatu eta bidaiaria eguneratu. Sortutako diru-mugimendua gorde eta erreserba eguneratu.
5. *Sistemak* gidariak kotxerik duen konprobatu eta ezabatu.
6. *Sistemak* bidaiaria bada, erreserbatutako bidaiak aukeratu eta erreserbaren egoera eguneratu. Azkenik erreserbatutako bidaiaren eselerkuak bidaiari gehitu.
7. *Sistemak* erabiltzailea eguneratu eta ezabatu.

Alternative flow

1. Erabiltzaileen lista hutsa da. Ez da kotxerik agertzen.
2. Gidariaren bidaien lista hutsa da. Ez da listarik agertzen.
3. Kotxeen lista hitsa da. Ez da listarik agertzen.
4. Bidaiariaren erreserben lista hutsa da. Ez da listarik agertzen.





Deskontua gehitu

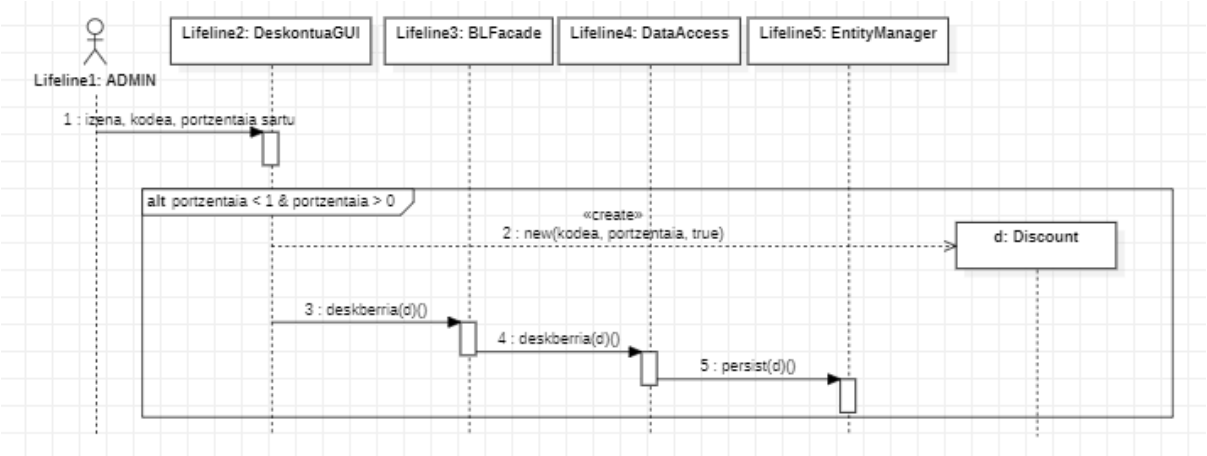
Gertaera fluxuak

Basic flow

1. *Admin* **izena**, **kodea** eta **portzentaia** sartu.
2. *Sistemak* sartutako datuekin deskontua gorde.

Alternative flow

1. Datuak falta dira. Sistemak erabiltzailea abisatzen du.
2. Portzentaia 0 eta 1 arteko balioa izan behar da.



Deskontu kodeak kudeatu

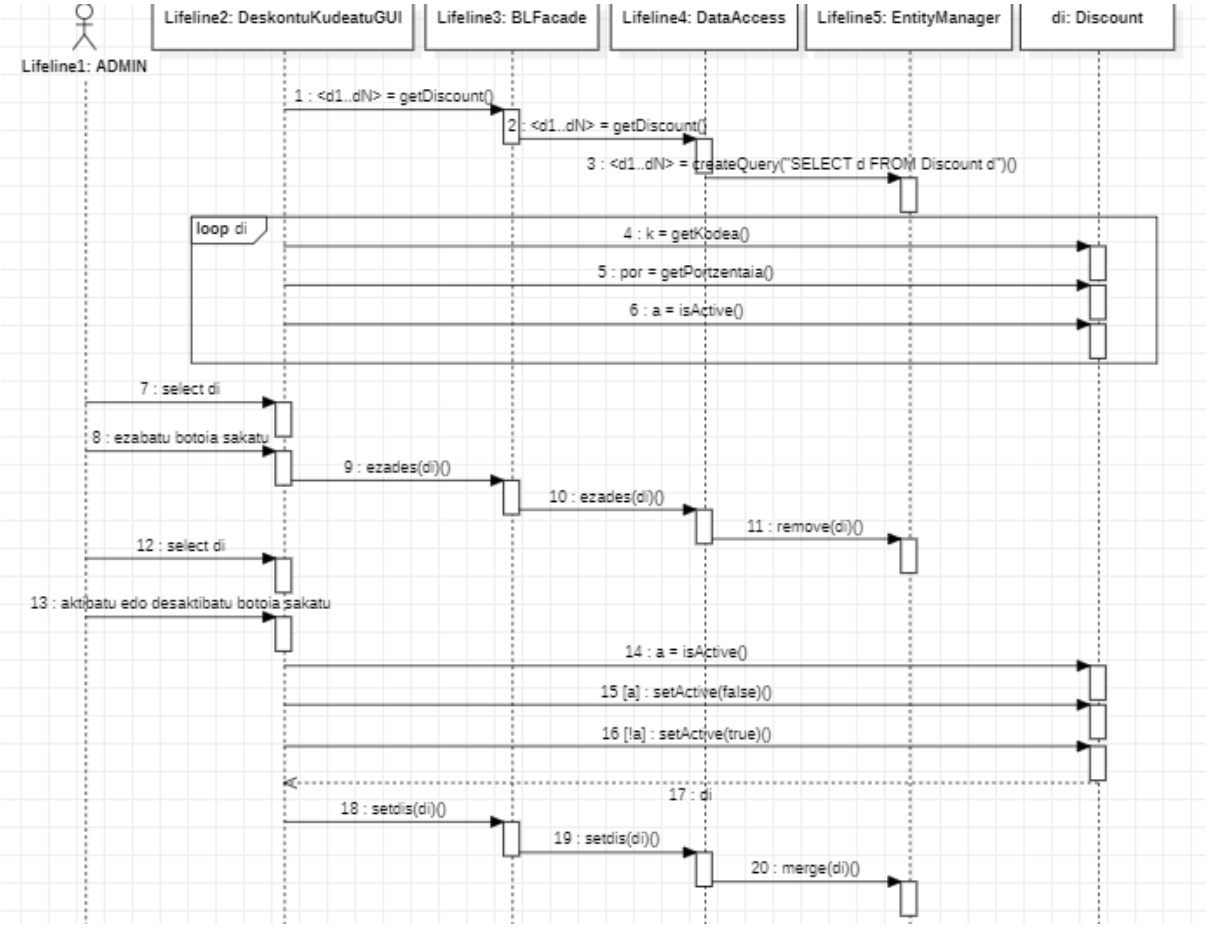
Gertaera fluxuak

Basic flow

1. *Sistemak* deskontuen lista erakusten du.
2. *Adminak* deskontu bat aukeratu eta botoi bat sakatu.
3. *Sistemak* ezabatu botoia sakatzerakoan, deskontua ezabatu.
4. *Sistemak* aktibatu/desaktibatu botoia sakatzerakoan, deskontuaren egoera konprobatzen du, aktiboa badago desaktibatzen du eta desaktibatuta badago aktibatzen du. Azkenik, deskontua eguneratzen du.

Alternative flow

1. Deskontuen lista hutsa da. Ez da kotxerik agertzen.

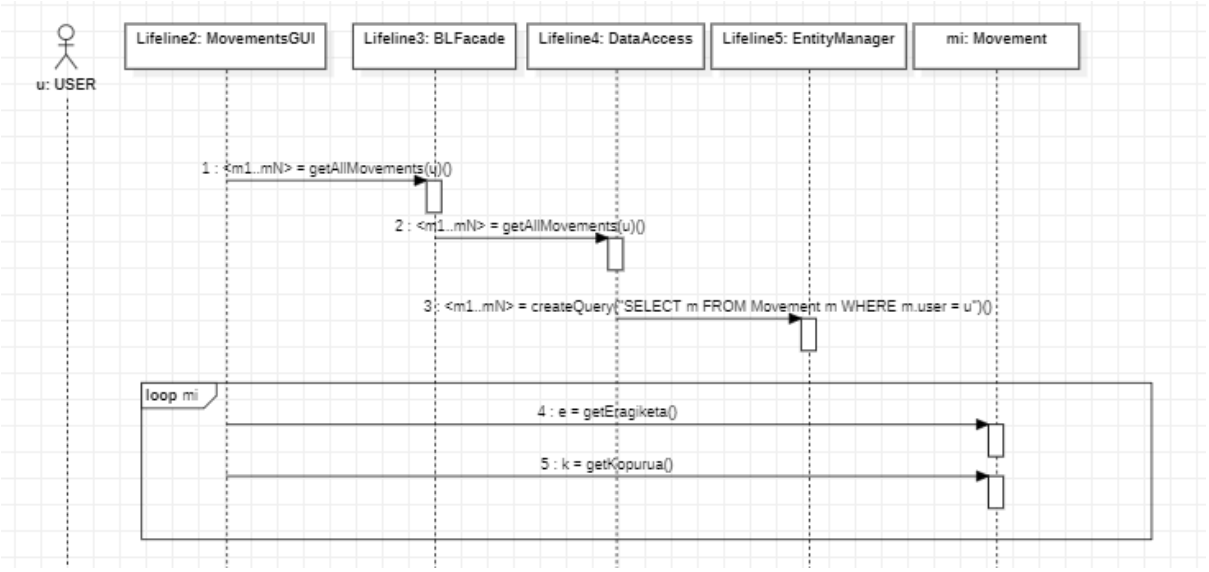


Mugimenduak ikusi

Gertaera fluxuak

Basic flow

1. *Sistema* **eragiketa mota** eta **diru kopuruaren** arabera lista bat erakusten du.
2. *Erregistratua* bere mugimenduak ikusten ditu.



Dirua sartu edo atera

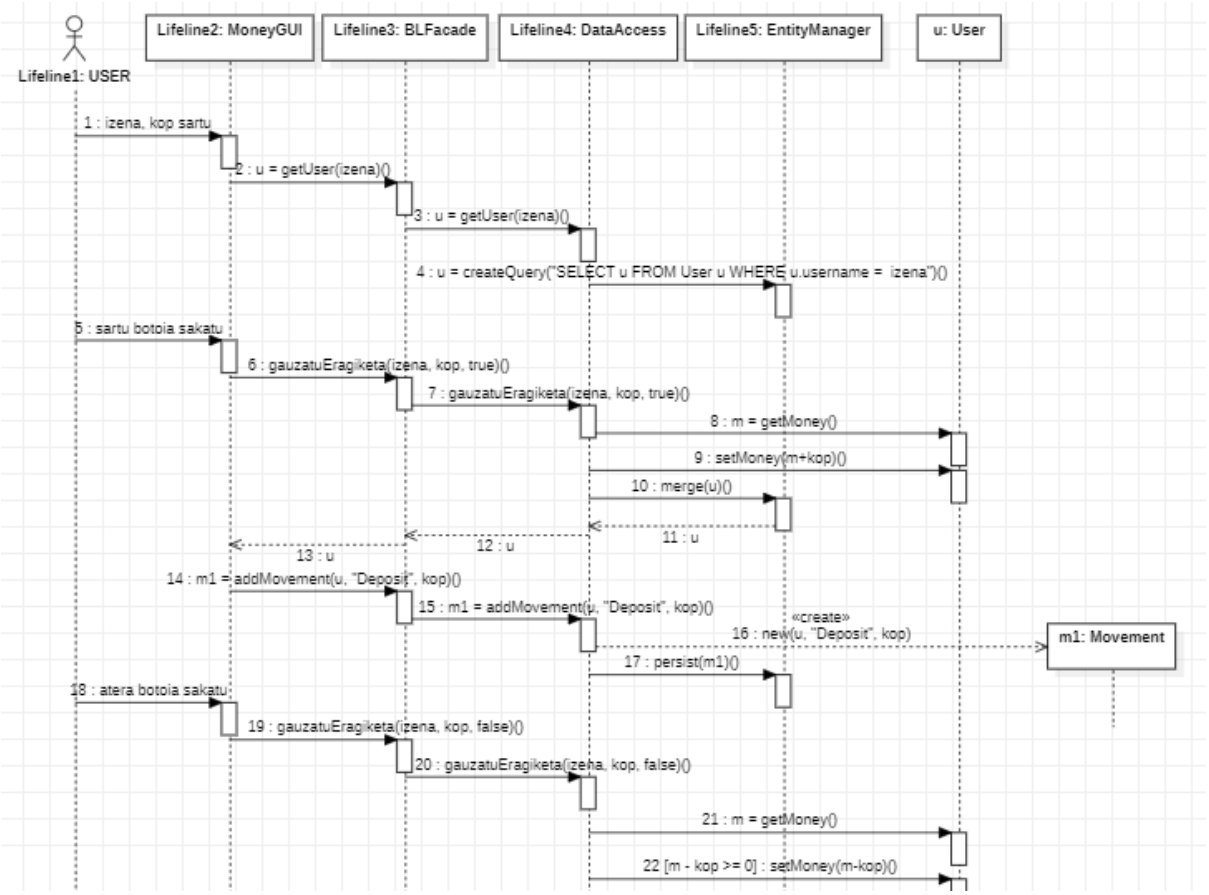
Gertaera fluxuak

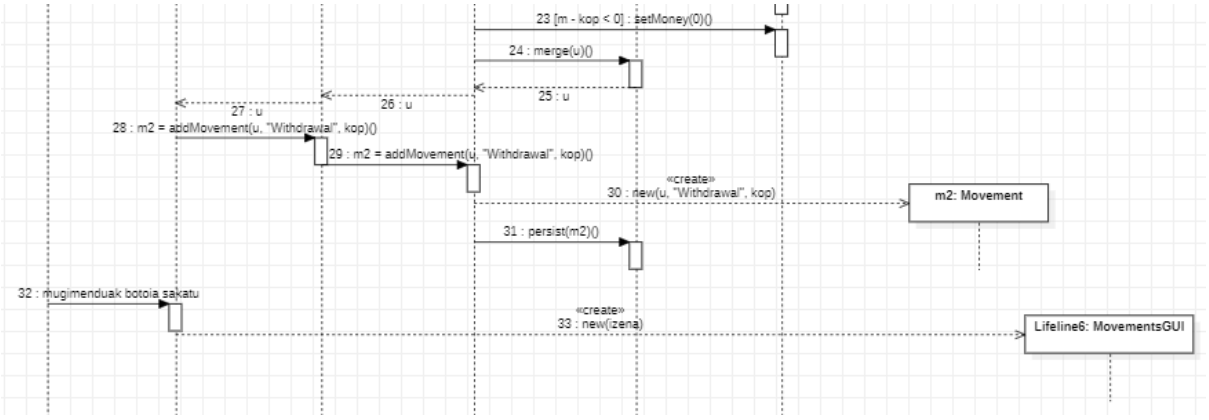
Basic flow

1. *Erregistratuak* **izena** eta **diru kopuru bat** sartzen du.
2. *Erregistratuak* sartu edo atera botoia sakatu.
3. *Sistemak* sartu botoia sakatzerakoan, sartutako diru kopurua erabiltzaileari gehitu eta sortutako diru-mugimendua gorde.
4. *Sistemak* atera botoia sakatzerakoan, sartutako diru kopurua erabiltzaileari kendu eta sortutako diru-mugimendua gorde.
5. *Erregistratuak* mugimenduak botoia sakatu.
6. *Sistemak MovementsGUI* pantailara bideratu.

Alternative flow

1. Diruaren laukian ez da zenbaki egoki bat sartu. Sistemak abisua ematen du.
2. Atera nahi den diru kopurua orain dagoena baina handiagoa da. Sistemak abisua ematen du.



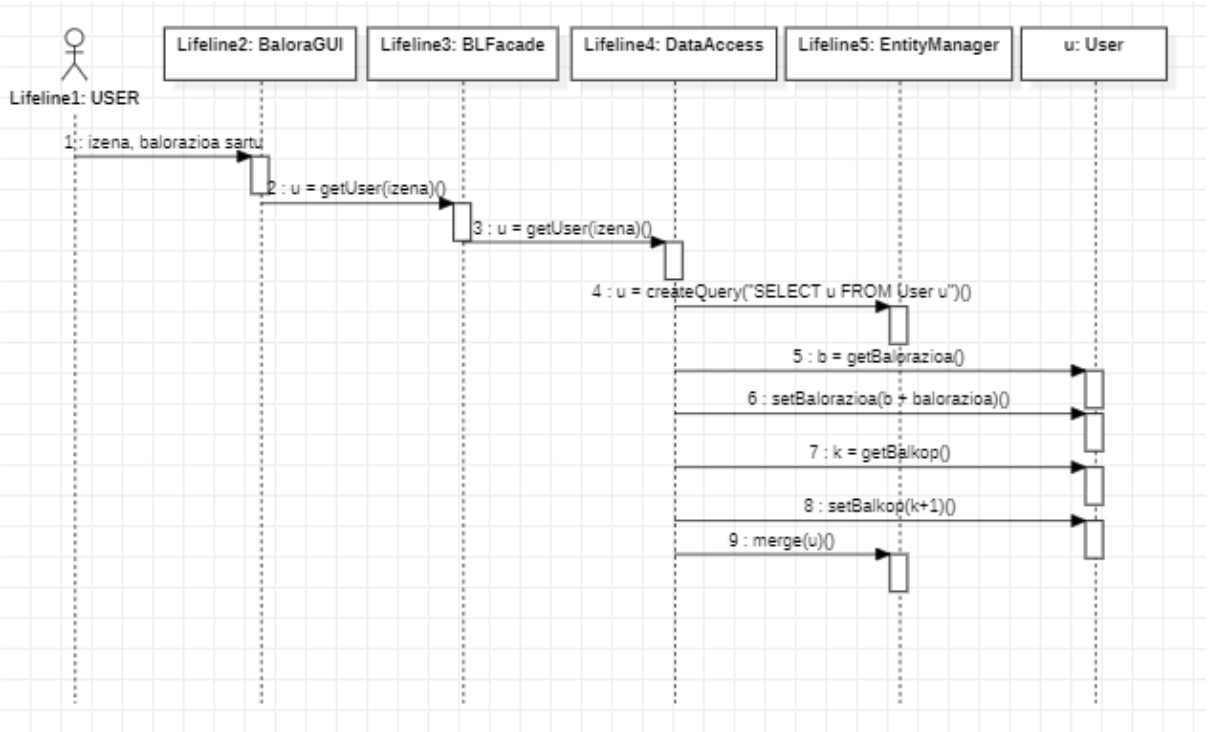


Balorazioak jarri

Gertaera fluxuak

Basic flow

1. *Erregistratua* **izena** eta **balorazioa** sartzen du.
2. *Sistemak* sartutako izenaren erabiltzaileari balorazioa gehitu eta eguneratu.

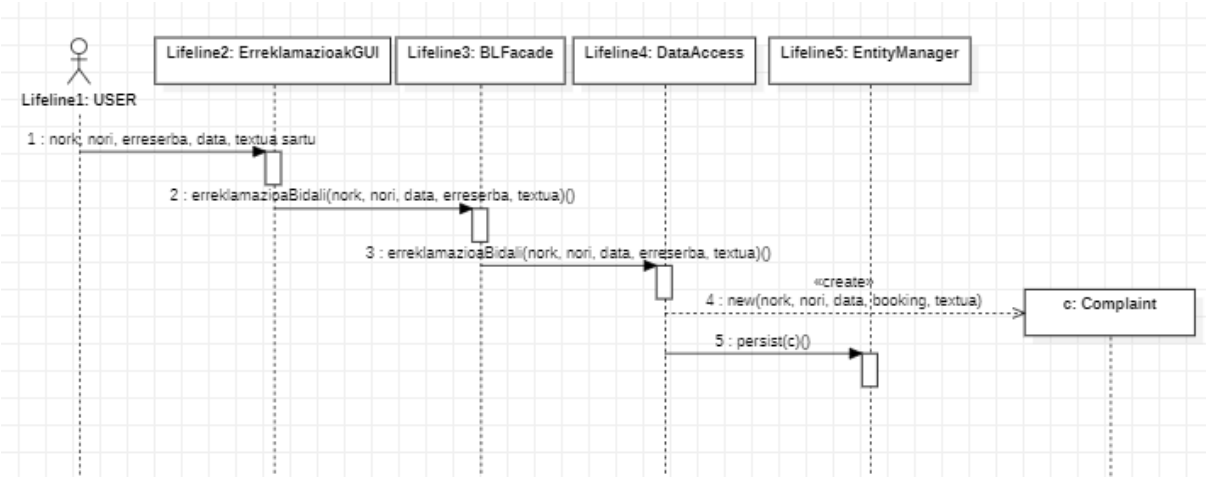


Erreklamazioak jarri

Gertaera fluxuak

Basic flow

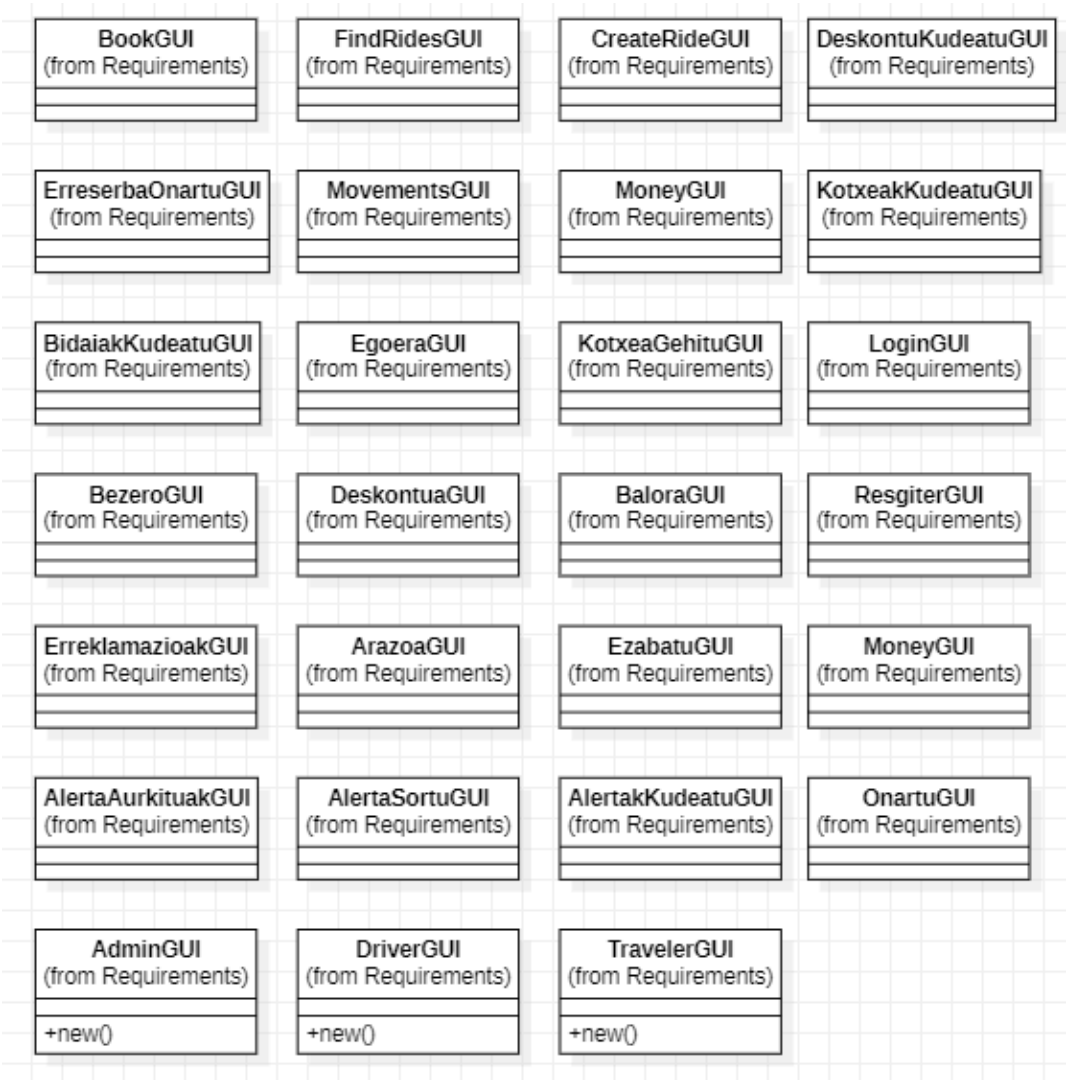
1. *Erregistratua* **nork, nori, gaur, booking** eta **textua.**
2. *Sistemak* sartutako datuen arabera erreklamazioa sortu eta gorde.

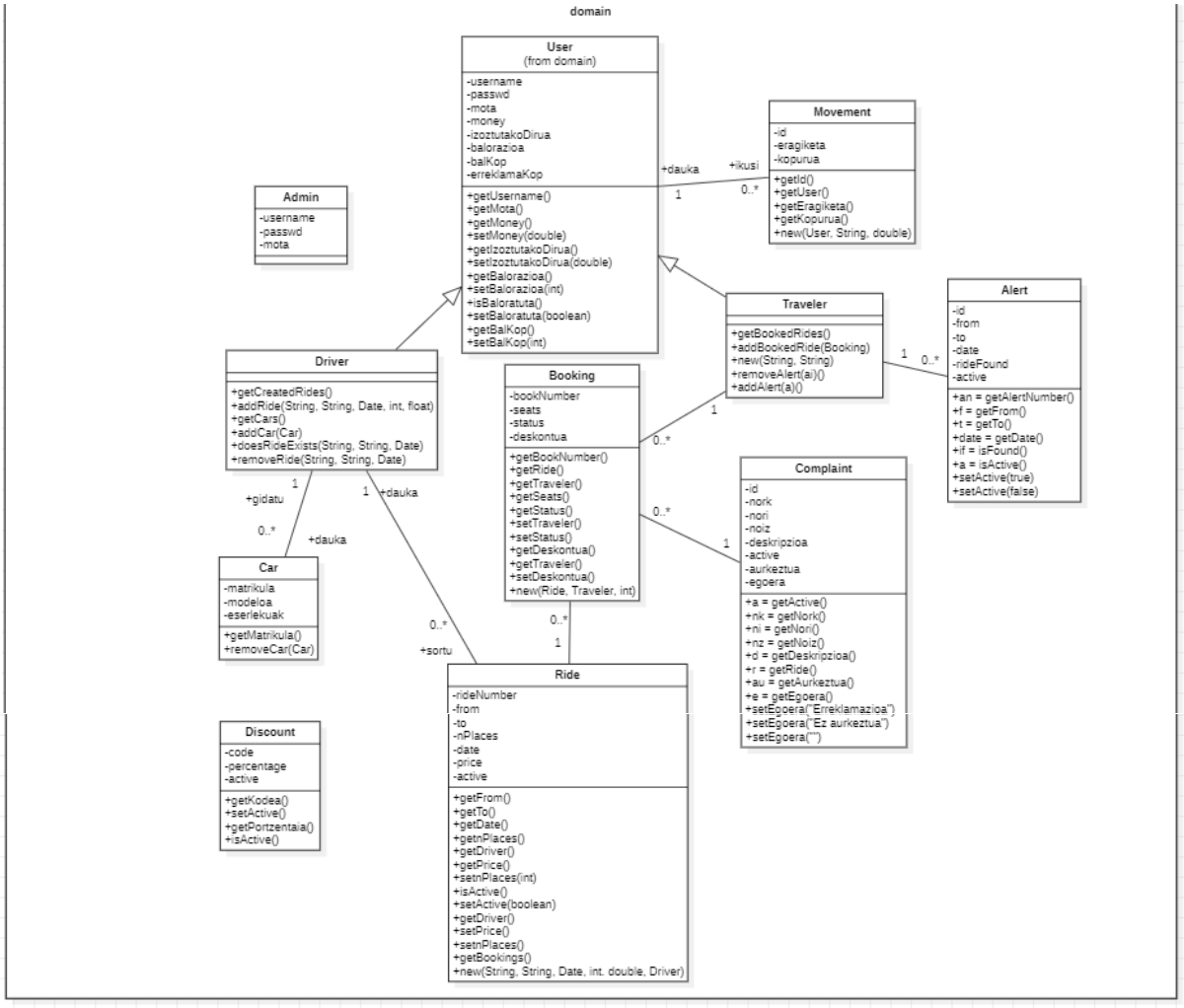


- Klase diagrama osoa

BLFacade
+<d1..dN> = getDepartCities() +<a1..aN> = getArrivalCities(di)() +<m1..mN> = getThisMonthDatesWithRides(di, ai, today)() +<r1..rN> = getRides(di, ai, mi)() +m = getActualMoney(izena)() +gauzatuEragiketa(izena, kop, auk) +addCar(matrikula, modeloa, eserlekuak) +<m1..mN> = getAllMovements(izena)() +cancelRide(ri) +<r1..rN> = getRidesByDriver(izena)() +<b1..bN> = getBookedRides(izena)() +desk = getDesk(deskontua)() +d = getDriver(izena)() +<b1..bN> = getBookingFromDriver(izena)() +erreserbaKudeatu(bi, "Onartu") +erreserbaKudeatu(bi, "Ezeztatu") +erreserbaZehaztu(bi, "Egina") +erreserbaZehaztu(bi, "Bertan Behera") +b = bookRide(izena, ri, eser, deskontua)() +gauzatuEragiketa(name, p, true)() +m1 = addMovement(t, "UnfreezeCompleteT", 0)() +m3 = addMovement(t,"UnfreezeNotComplete",p)() +c = getKotxebyMatrikula(matr)() +<m1..mN> = getThisMonthDatesWithRides(departCity, ArrivalCity, today)() +r = createRide(departCity, arrivalCity, mi, e, prezioa, izena)() +updateBooking(bi)() +m1 = addMovement("BookDeny", p)() +deleteCar(ci)() +<c1..cN> = getAllComplaints() +u = getUser(nk)() +<m1..mN> = getAllMovements(u)() +u = getUser(izena)() +gauzatuEragiketa(izena, kop, true)() +m1 = addMovement(u, "Deposit", kop)() +gauzatuEragiketa(izena, kop, false)() +m2 = addMovement(u, "Withdrawal", kop)() +t = getTraveler(izena)() +erreklamazioaBidali(nork, nori, data, erreserba, textua)() +<u1..uN> = getUserList() +userez(ui)() +deskberria(d)() +<d1..dN> = getDiscount() +ezades(di)() +setdis(di)() +addDriver(izena, pass)() +addTraveler(izena, pass)() +u = isRegistered(izena, pass)() +mota = getMotaByUsername(izena)() +getComplaintsByBook(bi)() +updateBooking(bi)() +m3 = addMovement(t, "UnfreezeCompleteT", 0)() +c = getComplaintsByBook(bi)() +erreklamazioaBidali(nori, nork, gaur, booking, "Ez da agertu",false)() +<a1..aN> = getAlertsByUsername(izena)() +updateAlert(ai) +deleteAlert(an) +b = createAlert(a)() +m1 = addMovement(t, "BookFreeze", (p-deskontua)*eser)() +m1 = addMovement(t, "BookDeny", (p-desk)*se)() +gauzatuEragiketa(name, (p-desk)*e, true)() +m2 = addMovement(d, "UnfreezeCompleteD", (p-desk)*e)()() +m4 = addMovement(d, "UnfreezeCompleteD", (p-desk)*e)()

DataAccess
+<d1..dN> = getDepartCities() +<a1..aN> = getArrivalCities(di)() +<m1..mN> = getThisMonthDatesWithRides(di, ai, today)() +<r1..rN> = getRides(di, ai, mi)() +m = getActualMoney(izena)() +gauzatuEragiketa(izena, kop, auk) +addCar(matrikula, modeloa, eserlekuak) +<m1..mN> = getAllMovements(izena)() +cancelRide(ri) +<r1..rN> = getRidesByDriver(izena)() +<b1..bN> = getBookedRides(izena)() +desk = getDesk(deskontua)() +d = getDriver(izena)() +<b1..bN> = getBookingFromDriver(izena)() +erreserbaKudeatu(bi, "Onartu") +erreserbaKudeatu(bi, "Ezeztatu") +erreserbaZehaztu(bi, "Egina") +erreserbaZehaztu(bi, "Bertan Behera") +b = bookRide(izena, ri, eser, deskontua)() +gauzatuEragiketa(name, p, true)() +m1 = addMovement(t, "UnfreezeCompleteT", 0)() +m3 = addMovement(t,"UnfreezeNotComplete",p)() +c = getKotxeByMatrikula(matr)() +<m1..mN> = getThisMonthDatesWithRides(departCity, ArrivalCity, today)() +r = createRide(departCity, arrivalCity, mi, e, prezioa, izena)() +updateBooking(bi)() +m1 = addMovement("BookDeny", p)() +deleteCar(ci)() +<c1..cN> = getAllComplaints() +u = getUser(nk)() +<m1..mN> = getAllMovements(u)() +u = getUser(izena)() +gauzatuEragiketa(izena, kop, true)() +m1 = addMovement(u, "Deposit", kop)() +gauzatuEragiketa(izena, kop, false)() +m2 = addMovement(u, "Withdrawal", kop)() +t = getTraveler(izena)() +erreklamazioaBidali(nork, nori, data, erreserba, textua)() +<u1..uN> = getUserList() +userez(ui)() +deskberria(d)() +<d1..dN> = getDiscount() +ezades(di)() +setdis(di)() +addDriver(izena, pass)() +addTraveler(izena, pass)() +u = isRegistered(izena, pass)() +mota = getMotaByUsername(izena)() +getComplaintsByBook(bi)() +updateBooking(bi)() +m3 = addMovement(t, "UnfreezeCompleteT", 0)() +c = getComplaintsByBook(bi)() +erreklamazioaBidali(nori, nork, gaur, booking, "Ez da agertu",false)() +<a1..aN> = getAlertsByUsername(izena)() +deleteAlert(an) +updateAlert(ai) +b = createAlert(a)() +m1 = addMovement(t, "BookFreeze", (p-deskontua)*eser)() +m1 = addMovement(t, "BookDeny", (p-desk)*se)() +gauzatuEragiketa(name, (p-desk)*e, true)() +m2 = addMovement(d, "UnfreezeCompleteD", (p-desk)*e)()() +m4 = addMovement(d, "UnfreezeCompleteD", (p-desk)*e)()


BookGUI
(from Requirements)
FindRidesGUI
(from Requirements)
CreateRideGUI
(from Requirements)
DeskontuKudeatuGUI
(from Requirements)
ErreserbaOnartuGUI
(from Requirements)
MovementsGUI
(from Requirements)
MoneyGUI
(from Requirements)
KotxeakKudeatuGUI
(from Requirements)
BidaiakKudeatuGUI
(from Requirements)
EgoeraGUI
(from Requirements)
KotxeaGehituGUI
(from Requirements)
LoginGUI
(from Requirements)
BezeroGUI
(from Requirements)
DeskontuaGUI
(from Requirements)
BaloraGUI
(from Requirements)
ResgiterGUI
(from Requirements)
ErreklamazioakGUI
(from Requirements)
ArazoaGUI
(from Requirements)
EzabatuGUI
(from Requirements)
MoneyGUI
(from Requirements)
AlertaAurkituakGUI
(from Requirements)
AlertaSortuGUI
(from Requirements)
AlertakKudeatuGUI
(from Requirements)
OnartuGUI
(from Requirements)
AdminGUI
(from Requirements)
+new()
DriverGUI
(from Requirements)
+new()
TravelerGUI
(from Requirements)
+new()


domain
User
(from domain)
-username
-passwd
-mota
-money
-izoztutakoDirua
-balorazioa
-balKop
-erreklamaKop
+getUsername()
+getMota()
+getMoney()
+setMoney(double)
+getIzoztutakoDirua()
+setIzoztutakoDirua(double)
+getBalorazioa()
+setBalorazioa(int)
+isBaloratuta()
+setBaloratuta(boolean)
+getBalKop()
+setBalKop(int)
Movement
-id
-eragiketa
-kopurua
+getId()
+getUser()
+getEragiketa()
+getKopurua()
+new(User, String, double)
+dauka
+ikusi
1
0..*
Admin
-username
-passwd
-mota
Traveler
+getBookedRides()
+addBookedRide(Booking)
+new(String, String)
+removeAlert(a)()
+addAlert(a)()
Alert
-id
-from
-to
-date
-rideFound
-active
+an = getAlertNumber()
+f = getFrom()
+t = getTo()
+date = getDate()
+if = isFound()
+a = isActive()
+setActive(true)
+setActive(false)
1
0..*
Driver
+getCreatedRides()
+addRide(String, String, Date, int, float)
+getCars()
+addCar(Car)
+doesRideExists(String, String, Date)
+removeRide(String, String, Date)
Booking
-bookNumber
-seats
-status
-deskontua
+getBookNumber()
+getRide()
+getTraveler()
+getSeats()
+getStatus()
+setTraveler()
+setStatus()
+getDeskontua()
+getTraveler()
+setDeskontua()
+new(Ride, Traveler, int)
1
0..*
Complaint
-id
-nork
-nori
-noiz
-deskripzioa
-active
-aurkeztua
-egoera
+a = getActive()
+nk = getNork()
+ni = getNori()
+nz = getNoiz()
+d = getDeskripzioa()
+r = getRide()
+au = getAurkeztua()
+e = getEgoera()
+setEgoera("Erreklamazioa")
+setEgoera("Ez aurkeztua")
+setEgoera("")
1
0..*
+gidatu
+dauka
1
0..*
Car
-matrikula
-modeloa
-eserlekuak
+getMatrikula()
+removeCar(Car)
+sortu
0..*
1
Ride
-rideNumber
-from
-to
-nPlaces
-date
-price
-active
+getFrom()
+getTo()
+getDate()
+getnPlaces()
+getDriver()
+getPrice()
+setnPlaces(int)
+isActive()
+setActive(boolean)
+getDriver()
+setPrice()
+setnPlaces()
+getBookings()
+new(String, String, Date, int, double, Driver)
Discount
-code
-percentage
-active
+getKodea()
+setActive()
+getPortzentaia()
+isActive()

- **Inplementazioa**

`public void initializeDB()`: Programa hasterakoan objektu desberdinak inizializatzeko erabiliko dugu hau. Modu honetan ez da izango beharrezkoa objektu batzuk sortzea aldiro, adibidez gidariak, bidaiariak, kotxeak…

`public User getUser(String erab)`: Erabiltzaile baten izena parametro bezela pasaz, honek User-a itzuliko du.

`public double getActualMoney(String erab)`: Erabiltzaile baten izena pasaz, honek erabiltzaile horrek duen dirua itzuliko du.

`public boolean isRegistered(String erab, String passwd)`:Erabiltzaile izen bat eta pasahitza pasaz, ea hau erregistratua dagoen bueltatuko du boolean baten bitartez.

`public Driver getDriver(String erab)`: Erabiltzaile baten username pasaz, honek Driver-a itzuliko du.

`public Traveler getTraveler(String erab)`: Lehengo metodoaren berdina egingo du, baina honek Driver-a beharrean Traveler-a itzuliko du.

`public Admin getAdmin(String erab)`: Berriz ere berdina egingo du, baina honek Admin-a itzuliko du.

`public String getMotaByUsername(String erab)`: Erabiltzaile baten username-a pasaz, honek ze erabiltzaile mota den itzuliko du. Gidari, bidaiari edo admin.

`public boolean addDriver(String username, String password)`: Metodo hau erabiliko dugu erabiltzaile berri bat erregistratzerakoan bidaiari bezela. Bidaiari berri bat sortuko du pasa dituen parametroen bidez.

`public boolean addTraveler(String username, String password)`: Lehenagoko metodoaren berdina egiten du, biana kasu honetan Traveler berri bat sortuko du.

`public boolean gauzatuEragiketa(String username, double amount, boolean b)`: Dirua dakarten eragiketeentzakoe rabiltzen da hau. Zein erabiltzaileari eragin, diru kantitatea eta ea dirua sartu edo atera behar den sartuko dugu metodoa gauzatu ahal izateko.

`public boolean bookRide(String username, Ride ride, int seats, double esk)`: Metodo honen bitartez Traveler-ak bidaiak erreserbatu ahal izango du.

`public List<Movement> getAllMovements(User user)`: User bat emanez, honek izan dituen mugimendu guztiak itzuliko ditu lista batean.

`public void addMovement(User user, String eragiketa, double amount)`: Emandako user-ari mugimendu bat gehitu.

`public List<Booking> getBookedRides(String username)`: User batek dituen erreserba guztiak itzuliko ditu lista batean.

`public void updateTraveler(Traveler traveler)`: Traveler batean aldaketa egin ondoren, hau datu basean gorde dadin egiten dugu metodo hau.

`public void updateDriver(Driver driver)`: Lehengo metodo berdina da, baina driver-arekin.

`public void updateUser(User user)`: Lehengo berdina da, baina User batekin.

`public List<Booking> getPastBookedRides(String username)`: Egunerako egindako erreserbak itzuliko ditu lista batean.

`public void updateBooking(Booking booking)`: Bookin bat eguneratzen du aldaketa baten bat egin ondoren.

`public List<Booking> getBookingFromDriver(String username)`: Driver baten bidai desberdinetan dituen erreserbak itzuliko ditu lista batean.

`public List<Ride> getRidesByDriver(String username)`: Gidari batek dituen bidaiak itzuliko ditu lista batean.

`public void cancelRide(Ride ride)`: Bidai bat pasaz, hau ezabatuko du.

`public boolean addCar(String username, Car kotxe)`: Kotxe berri bat gehituko du driver-ari.

`public boolean isAdded(String username, String matr)`: Lehengo metodoarekin zerikusia dauka. Lehengoa erabiltzerakoan honi deitzen zaio ikusteko ea dagoeneko gehitua daukan kotxea itzuliko du.

`public Car getKotxeByMatrikula(String matr)`: Parametro bezela pasa den matrikularen kotxea bueltatuko da.

`public void deleteCar(Car car)`: Kotxe bat ezabatuko du Driver-aren kotxeen listatik.

`public boolean erreklamazioaBidali(String username1, String username2, Date gaur, Booking book, String textua,boolean aurk)`: Erreklamazio berri bat egiteko erabiltzen da metodo hau.

`public void updateComplaint(Complaint erreklamazioa)`: Erreklamazio bat berrituko du aldaketa bat egiterakoan.

`public void createDiscount(Discount di)`: Deskontu bat sortzen duen metodoa da.

`public List<Discount> getAllDiscounts()`: Lista batean itzuliko ditu dauden diskontu guztiak.

`public void deleteDiscount(Discount dis)`: Deskontu bat pasaz, honek deskontuen listatik ezabatuko du.

`public void updateDiscount(Discount dis)`: Deskontu bat aldatu bada, metodo hau aplikatuz aldaketan datu basean gordeko dira.

`public Discount getDiscount(String desk)`: Deskontuaren izenaren bitartez, deskontua itzuliko du.

`public List<User> getUserList()`: User guztiak itzuliko ditu lista batean.

`public void deleteUser(User us)`: Parametro bidez pasa den user-a ezabatuko da.

`public List<Alert> getAlertsByUsername(String username)`: User batek dituen alertak itzuliko ditu lista batean.

`public Alert getAlert(int alertNumber)`: Alerta zenbakiaren bitartez, alerta eskuratu eta bueltatuko da.

`public void updateAlert(Alert alert)`: Alertar batean aldaketa bat egin ondoren, metodo hau erabiliko da alertaren eguneraketa datu basean gordetzeko.

`public boolean updateAlertaAurkituak(String username)`: Alerta aurkitzerakoan metodo hau erabiliko da.

`public boolean createAlert(Alert newAlert)`: Alerta bat sortuko duen metodoa da.

`public boolean deleteAlert(int alertNumber)`: Alerta zenbaki bat parametro bezela pasata, honi dagokion alerta ezabatuko du.

`public Complaint getComplaintsByBook(Booking bo)`: Erreserba batek dituen erreklamaioak itzuliko ditu.

.
-
.
-
.
-
.
-
.
-

.
-
.
-

- **Bideoaren URL-a**

https://youtu.be/JyPL2yxF3aI