

Egileak: Asier Aldai Sanchez, Gorka Aldana Aldecoa eta Jon Andoni Maruri Aspiazu

Web Zerbitzuak: EZ

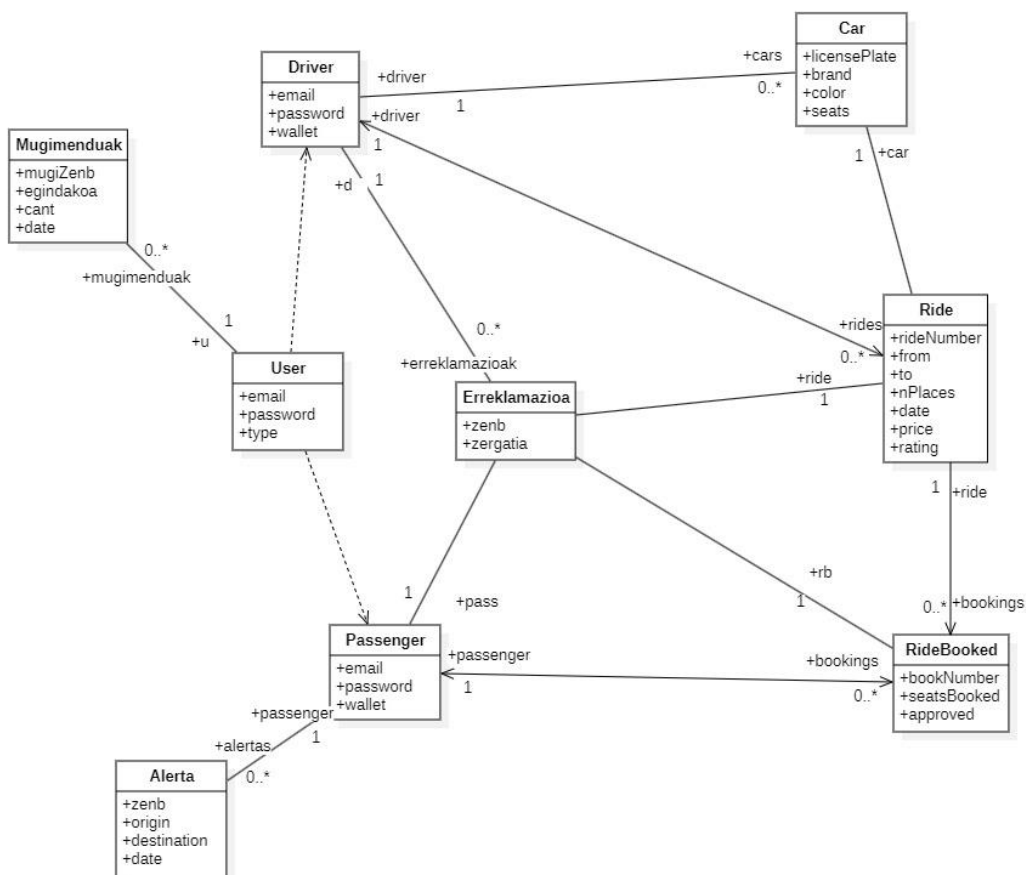
Internalizazioa: BAI

SARRERA

Egin behar izan dugun proiektua auto kompartitzeko aplikazio bat izan da, egin ditugun funtzionalitate nagusienetarikoa Alertak izatearena izan da. Alertak sortzeko, bidaiari kontu bat erabili behar da. Kontu horrek irteera hiria eta helmuga hiria aukeratu beharko ditu, egun batekin batera eta behin gidari batek baldintza horiek betetzen dituen bidaia sortzean, bidaiariaren alerta zerrendara gehituko da. Beste hainbat funtzionalitate egin ditugu, baina beste bat aipatzearren Erreklamazioena nahiko nagusia da.

ESKAKIZUN BILKETA

1.- DOMEINUAREN EREDUA



-**USER** klasea: Klase abstraktu bat da, Passenger eta Driver-ek bere funtzioak erabiltzen dituzte eta User bakoitzak Mugimenduak klaseko Array bat du.

-**MUGIMENDUAK** klasea: Mugimendu bakoitzak zenbaki bat du identifikatzeko, egindakoa (Mugimenduaren azalpen txiki bat), cant (mugitu den diru kantitatea) eta date (mugimendua burutu den data). Mugimenduak klaseak erlazioa du User-ekin, aurreko atalean esan dudan moduan User batek hainbat mugimendu ditu.

Bi kontu mota daude **DRIVER** eta **PASSENGER**:

-**DRIVER** klasea: DRIVER bakoitzak kontu korreo, pasahitza eta diru kantitatea (wallet) gorde egiten ditu. Horrez gain beste klase batzuekin erlazioa du ere. DRIVER bakoitzak erreklamazio zerrenda bat gorde egiten du eta zenbait auto eta bidaia sortu ditzake.

-**PASSENGER** klasea: PASSENGER klaseak DRIVER klasearen atributo berdinak erabiltzen ditu, baina ez ditu erlazio berdinak. Passenger batek hainbat Alerta desberdin izan ditzake, baita ere hainbat Erreserba (RIDEBOOKED).

-**CAR** klasea: CAR klaseak hiru atributu ditu: license (autoren matrikula), color (kolorea) eta brand (autoaren marka). auto batek aldiz bakarrik gidari bakarra izan dezake.

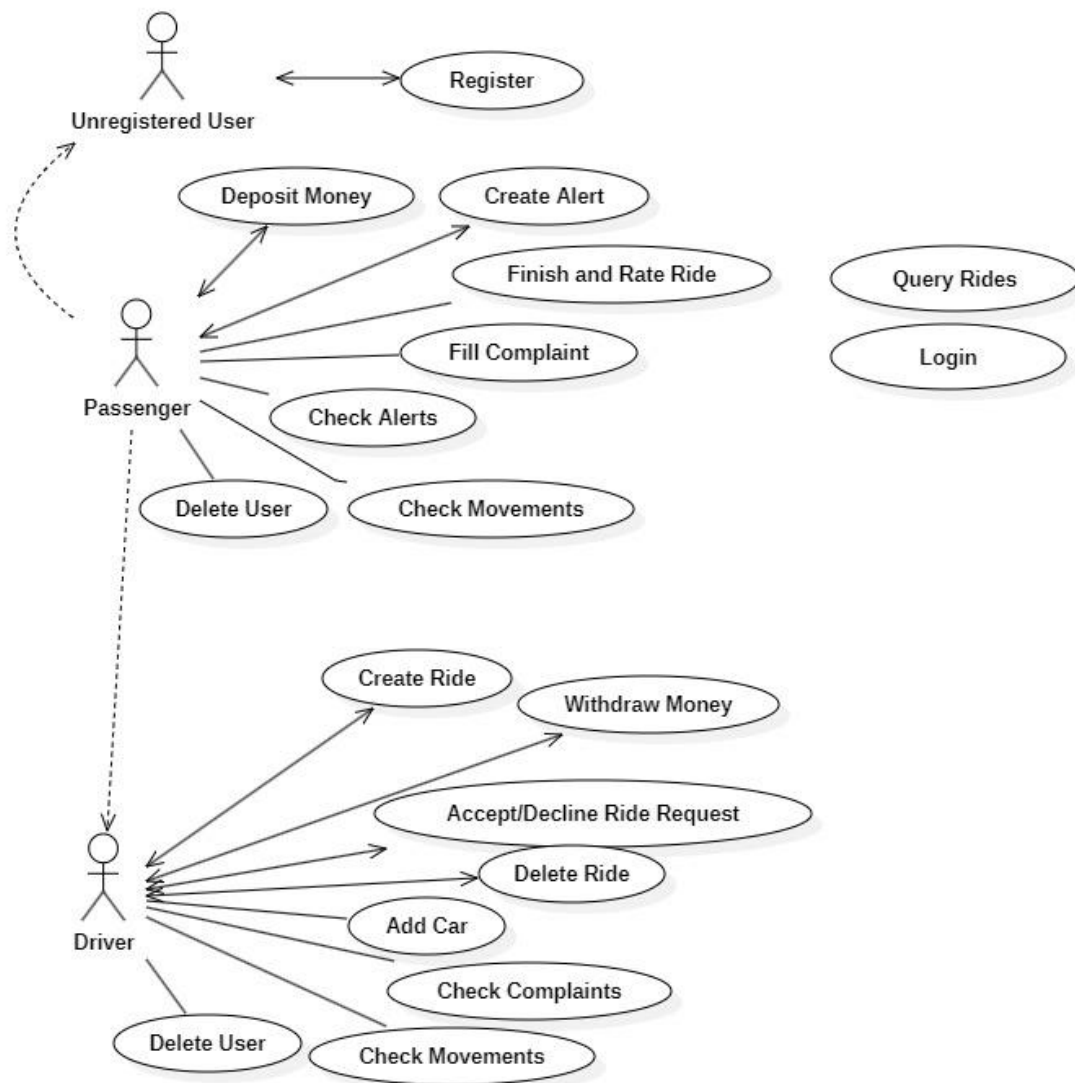
-**RIDE**: RIDE klaseak hamar atributu ditu: rideNumber (bidaia identifikatzeko zenbakia), from (irteera hiria), to (helmuga hiria), nPlaces (zenbat bidaiari sartzen diren), date (bidaiaren eguna), price (bidaiak duen kostua), rating (bidaiak izan duen balorazioa, hau amaieran jartzen da) eta car (bidan erabiliko den autoa). Bidaia batek DRIVER bakarra izan dezake, baina Bidaia bakarrak hainbat erreserba izan ditzake.

-**RIDEBOOKED**: RIDEBOOKED klaseak bost atributu ditu: bookNumber (Erreserba zenbakia), seatsBooked (Erreserbatutako eserleku kopurua) eta approved (Gidariak erreserba onartu duen edo ez jakiteko). Erreserba batek ezin ditu Passenger bat baino gehiago izan, ezta ere Ride bat baino gehiago.

-ALERTA: ALERTA klaseak bost atributu ditu: zenb (Alerta zenbakia), origin (Irteera hiria aukeratzeko), destination (Helmuga hiria) eta date (Bidaiaaren data jartzeko). Alerta batek ezin ditu Passenger bat baino gehiago izan.

-ERREKLAMAZIOA: ERREKLAMAZIOA klaseak sei atributu ditu: zenb (Erreklamazioaren identifikatzailea), zergatia (erreklamazioaren arrazoia), Ride (zein Bidaiaari jarri zaion erreklamazioa), Driver (Bidaiaaren Gidaria), RideBooked (zein erreserbatutako bidaiaari dagokion erreklamazioa) eta Passenger (erreklamazioa jarri duen Bidaiaaria).

2.- ERABILPEN KASUEN EREDUA



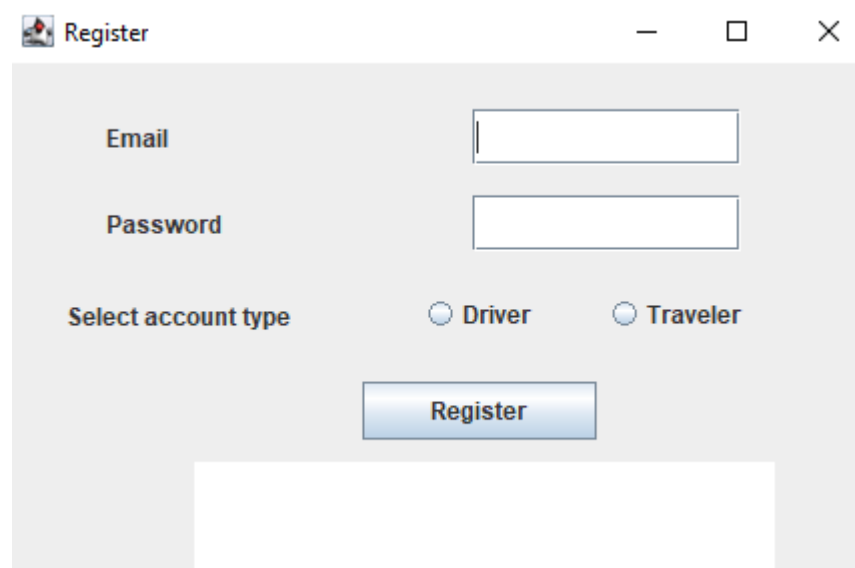
REGISTER:

Basic Flow

1. *Unregistered User* inserts his **email** and **name** once and **password** twice.
2. *Unregistered User* chooses "Bidaia" or "Gidari" option.
3. *System* creates a new user and saves the data in it.

Alternative flow

1. Some fields are empty. System informs the user.
2. email, name and password combination already exists in the system. System informs the user.



PASSENGER AKTOREAREN ERABILPEN KASUAK:

DEPOSIT MONEY:

Basic Flow

1. *Passenger* inserts the amount of money to deposit in *wallet*.
2. *System* adds the inserted amount to *Passenger's* existing *wallet*.
3. *System* updates the numbers displayed on the screen referring to the *Passengers's* wallet.

Alternative flow

1. Some fields are empty. System informs the user.
2. Inserted number is negative. System informs the user.

Deposit Money

This is your actual amount 5.0

Quantity you want to add or retire

Add

CREATE ALERT:

Basic Flow

1. *System* displays all cities where rides depart from
2. *Passenger* selects a departing city
3. *System* displays all destination cities for rides that depart from a selected city.
4. *Passenger* selects an arrival city
5. *System* highlights in a Calendar the days where rides exist from the depart to destination cities in a selected month
6. *Passenger* selects a date in a Calendar
7. *System* displays the rides from the selected departing city to the selected arrival city on that date.
8. *Passenger* selects a ride.
9. *Passenger* inserts the number of seats to book.
10. *Passenger* books the ride.
11. *System* creates a booked ride with inserted data and assigns it to the Passenger.

Alternative flow

1. There are no rides on the selected date. Rides display is empty.
2. number of seat to book is not a number. System informs the user.
3. The inserted number of seats to book exceeds the number of seats available for the selected ride. System informs the user.
4. The booked ride already exist for this passenger. Booked ride is not created. System informs the user.

Create Alert

Depart City

Arrival City

Create

Ride date

May

2025

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
18					1	2	3
19	4	5	6	7	8	9	10
20	11	12	13	14	15	16	17
21	18	19	20	21	22	23	24
22	25	26	27	28	29	30	31

FINISH AND RATE RIDE:

Basic Flow

1. *Passenger* selects a **booked ride** from a list.
2. *Passenger* selects a rating from 0 to 5.
3. *Passenger* clicks "Ride finished".
4. *System* completes transaction of money to *Driver*.
5. *System* deletes RideBooking from *passenger* and ride.

Look Bookings

Donostia/Bilbo/Tue May 13 00:00:00 CEST...

Donostia/Bilbo/Tue May 13 00:00:00 CEST 2025/asier@gmail.com//1/true/jc

Rating

0

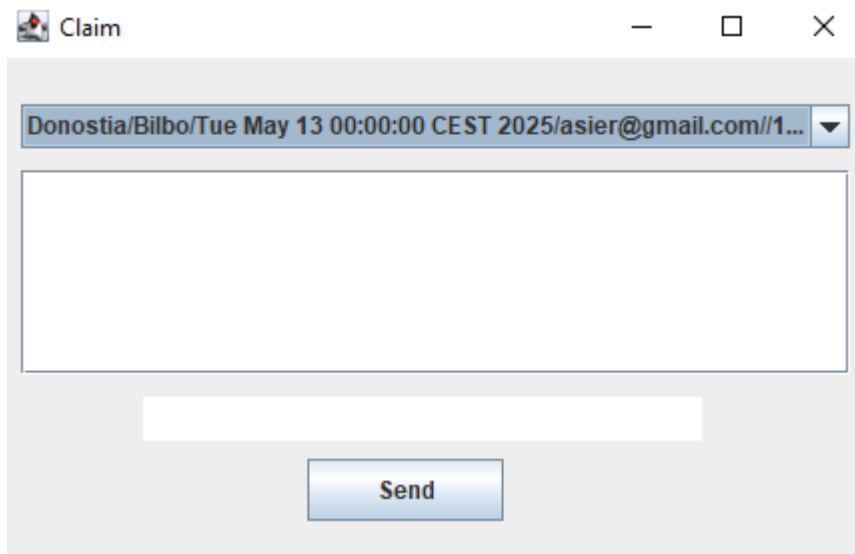
Ride finis...

Claim

FILL COMPLAINT:

Basic Flow

1. *Passenger* selects a **booked ride** from the list.
2. *Passenger* clicks "Claim".
3. *Passenger* selects the desired **ride** again.
4. *Passenger* writes the reason for the complaint.
5. *System* creates a **complaint** and assigns it to the **ride's** assigned **driver**.

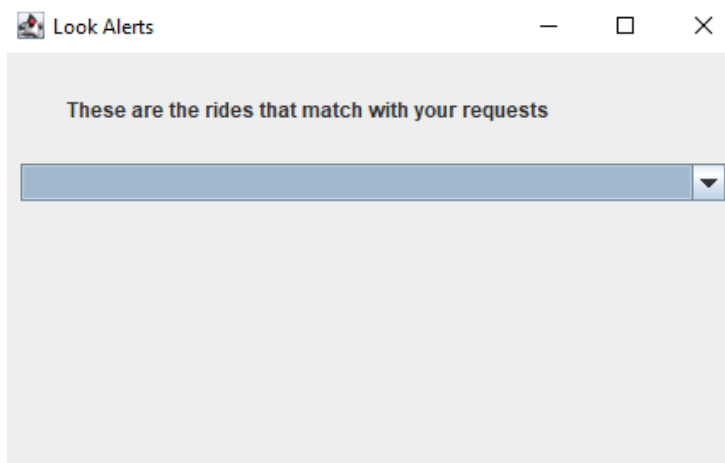


The screenshot shows a window titled "Claim" with standard window controls (minimize, maximize, close). At the top, there is a text field containing the string "Donostia/Bilbo/Tue May 13 00:00:00 CEST 2025/asier@gmail.com//1..." followed by a dropdown arrow. Below this is a large, empty rectangular text area for writing a complaint. At the bottom center, there is a "Send" button.

CHECK ALERTS:

Basic Flow

1. *System* shows a list of **rides** with the same origin, destination and date as existing **alerts**.
2. *Passenger* checks the list.

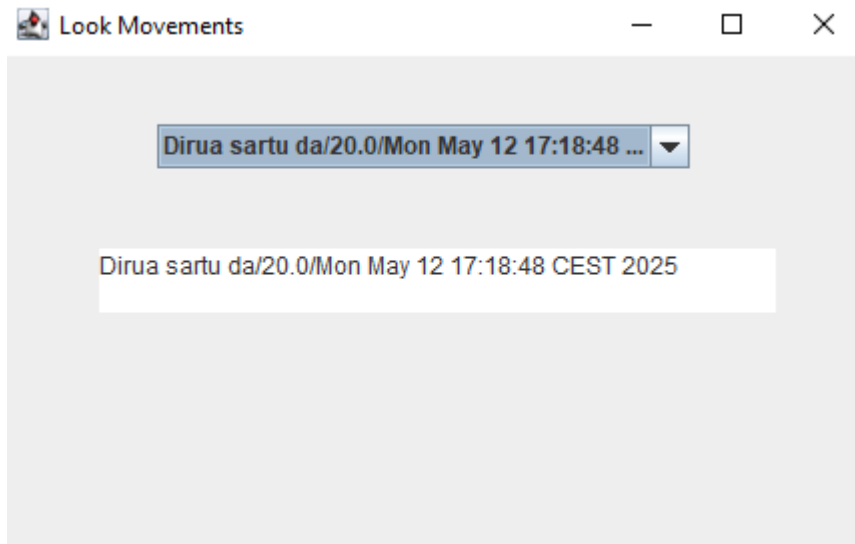


The screenshot shows a window titled "Look Alerts" with standard window controls (minimize, maximize, close). The main content area has a header text "These are the rides that match with your requests". Below the header is a horizontal bar with a dropdown arrow on the right side, likely for selecting a filter or view option.

CHECK MOVEMENTS:

Basic Flow

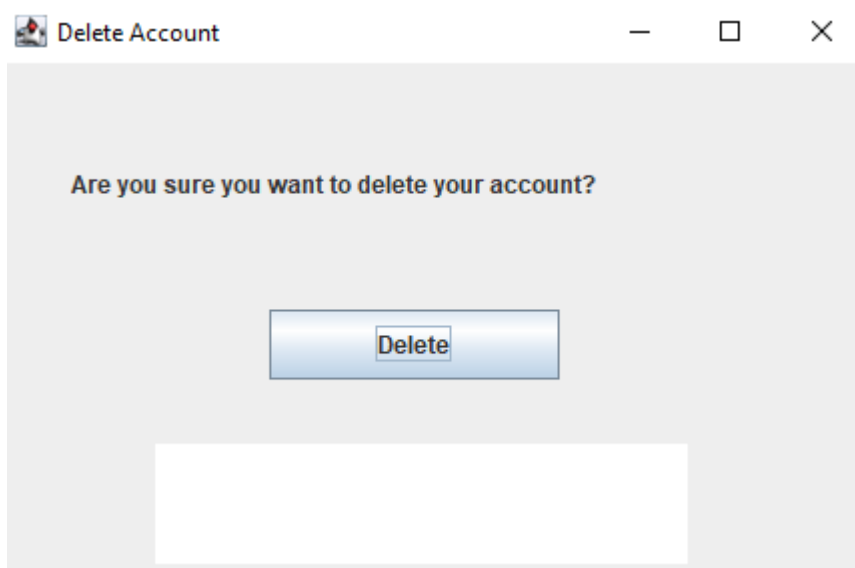
1. *System* shows the list of movements made by the passenger.



DELETE USER (PASSENGER):

Basic Flow

1. *Passenger* clicks "Delete".
2. *System* removes all the booked rides, movements and alerts linked to the passenger from the data base.
3. *System* removes the passenger from the data base.



DRIVER AKTOREAREN ERABILPEN KASUAK:

CREATE RIDE:

Basic Flow

1. *Driver* insert the departing city, arrival city, number of seats and price
2. *Driver* selects a ride date
3. *System* creates a ride with inserted data and assigns to the *Driver*.

Alternative flow

1. Some of fields are empty. Final.
2. number of seat or price is not a number. System informs the user.
3. Ride date is before today. System informs the user.
4. the ride already exist for this passenger. Ride is not created. System informs the user.

Create Ride

Depart City:

Arrival City:

Number of Seats:

Price:

Select a car:

Ride date

May 2025

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
18					1	2	3
19	4	5	6	7	8	9	10
20	11	12	13	14	15	16	17
21	18	19	20	21	22	23	24
22	25	26	27	28	29	30	31

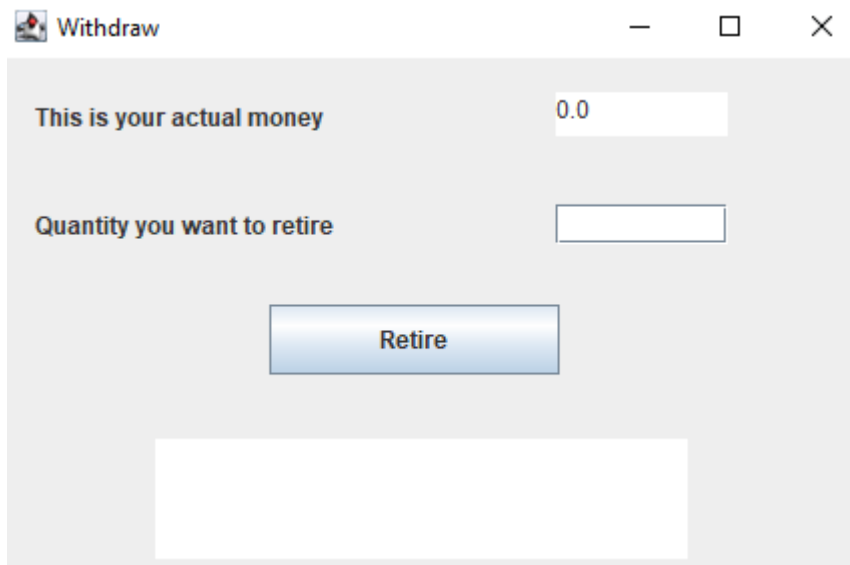
WITHDRAW MONEY:

Basic Flow

1. *Driver* inserts the amount of money to withdraw from *wallet*.
2. *System* extracts the inserted amount from *Driver's* existing *wallet*.
3. *System* updates the numbers displayed on the screen referring to the *Passengers's* wallet.

Alternative flow

1. Some fields are empty. System informs the user.
2. Inserted number is negative. System informs the user.
3. Amount in *wallet* is less than inserted amount. System informs the user.



Withdraw

This is your actual money

Quantity you want to retire

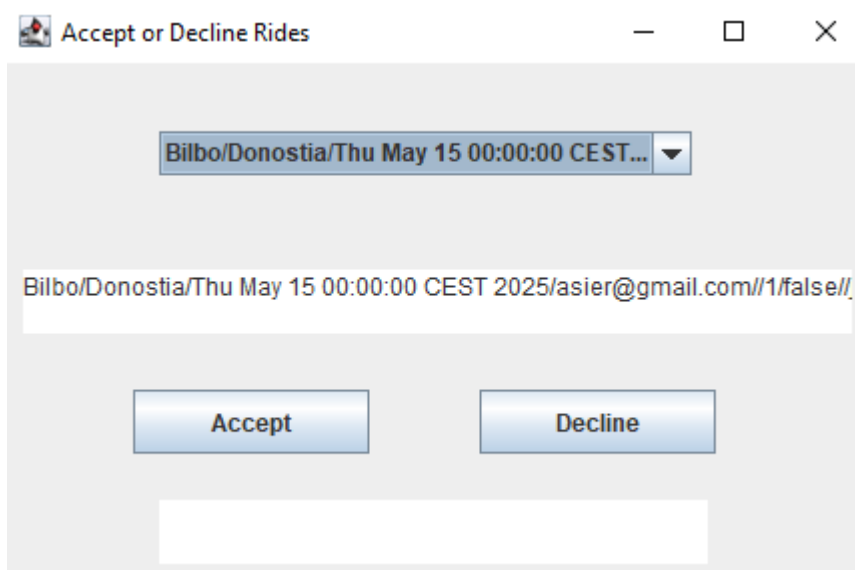
ACCEPT/DECLINE RIDE REQUEST:

Basic Flow

1. *Driver* selects a ride request from the list.
2. *Driver* chooses "Onartu" or "Deusestatu".
3. *System* extracts the amount of booked seats on the booked ride from number of seats in the ride with the same number as the booked ride.
4. *System* removes the ride request from the list.

Alternative flow

1. The booked seats from the ride request exceed the available seats on the ride. System doesn't update the ride's data. System doesn't remove the ride request from the list. System informs the *Driver*



Accept or Decline Rides

▼

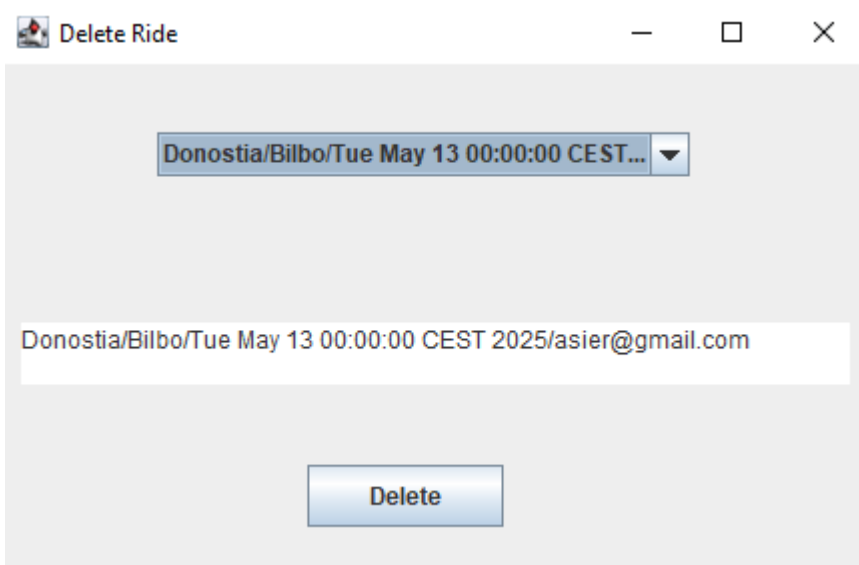
DELETE RIDE:

Basic Flow

1. *Driver* selects a **ride** from the list.
2. *Driver* clicks "Delete".
3. *System* removes all **booked rides** linked to the selected **ride** and returns the money to the **passengers** from the data base.
4. *System* creates a new **movement** and links it to the **driver**.
5. *System* removes the selected **ride** from the data base.
6. *System* notifies the user.

Alternate Flow

1. The selected **ride** doesn't have any **booked ride** linked to it. Procedure stays the same.



Donostia/Bilbo/Tue May 13 00:00:00 CEST...

Donostia/Bilbo/Tue May 13 00:00:00 CEST 2025/asier@gmail.com

Delete

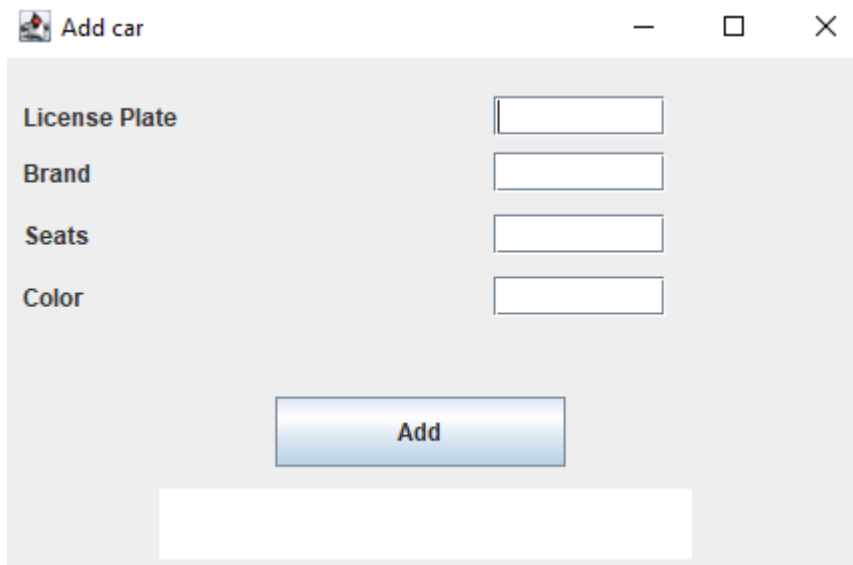
ADD CAR:

Basic Flow

1. *Driver* inserts the license plate, brand, number of seats and color of the car.
2. *Driver* clicks "Add".
3. *System* creates a new **car** and links it to the **driver**.
4. *System* notifies the **driver**.

Alternate Flow

1. The value inserted for **number of seats** is not a number. System notifies the **driver**.
2. The car already exists for thar **driver**. System notifies the **driver**.



License Plate

Brand

Seats

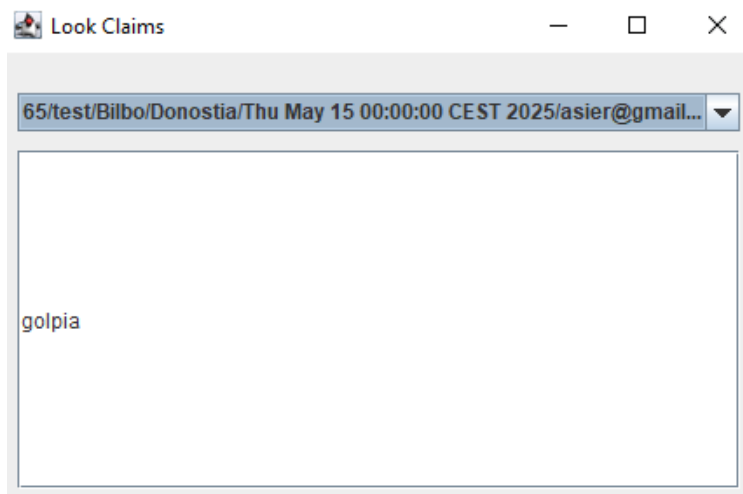
Color

Add

CHECK COMPLAINTS:

Basic Flow

1. System shows the list of complaints linked to the **driver**.



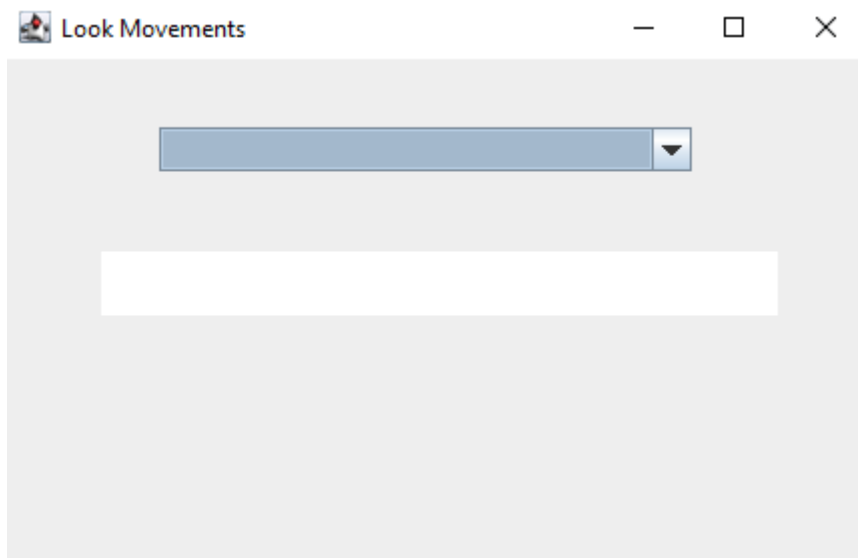
65/test/Bilbo/Donostia/Thu May 15 00:00:00 CEST 2025/asier@gmail... ▼

golpia

CHECK MOVEMENTS:

Basic Flow

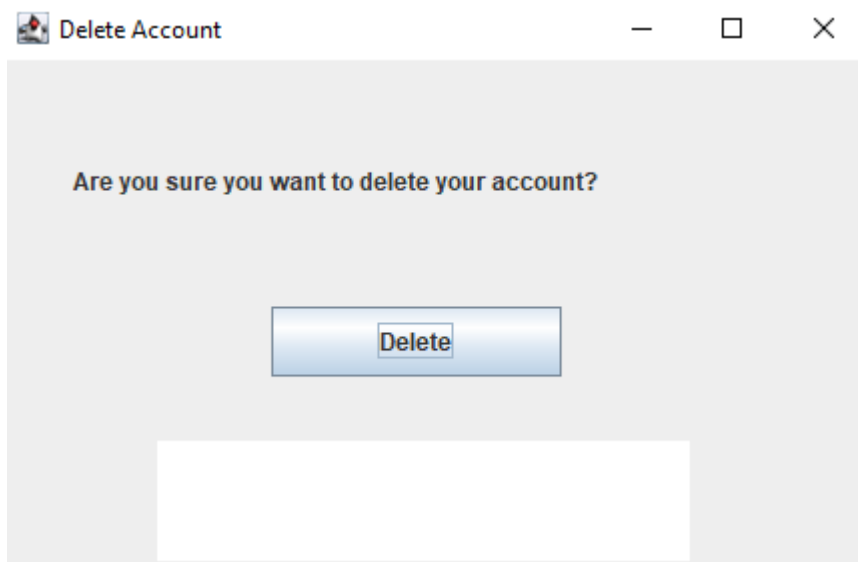
1. System shows the list of movements made by the **driver**.



DELETE USER (DRIVER):

Basic Flow

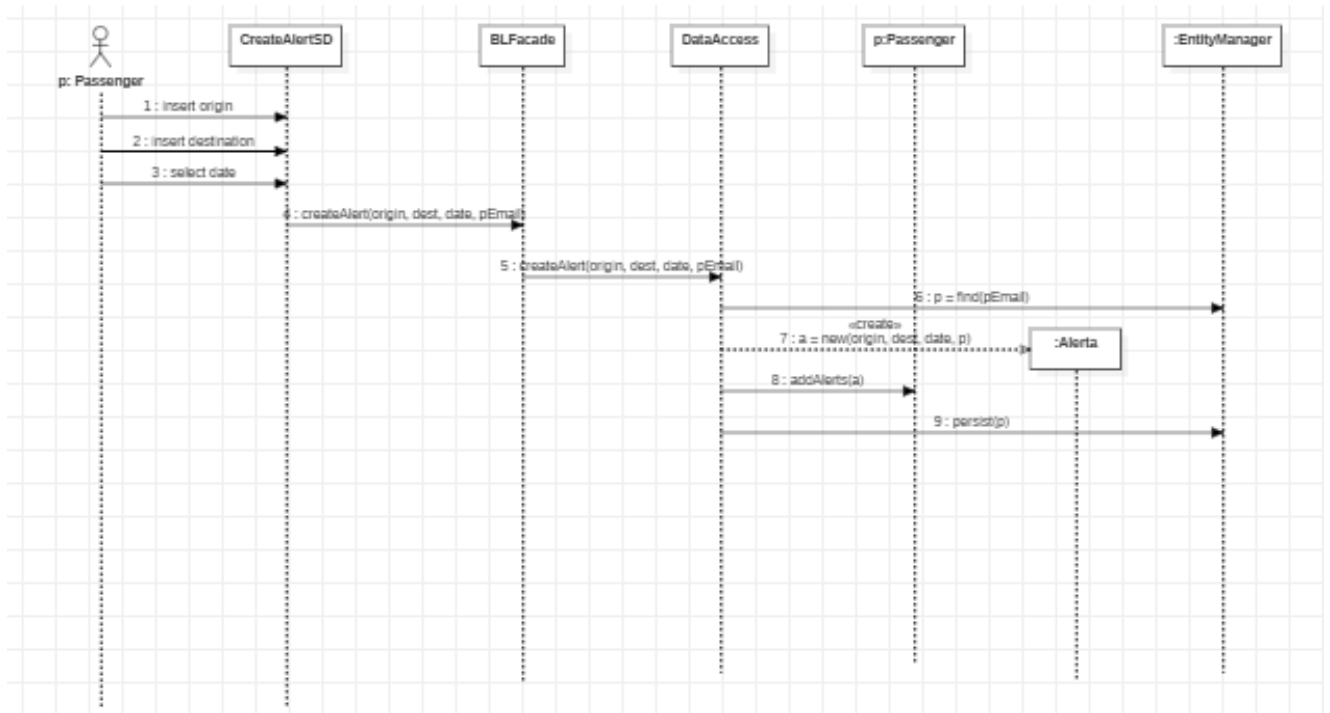
1. *Driver* clicks "Delete".
2. *System* removes all the rides (and the booked rides linked to each ride), cars and movements linked to the driver from the data base.
3. *System* removes the driver from the data base.



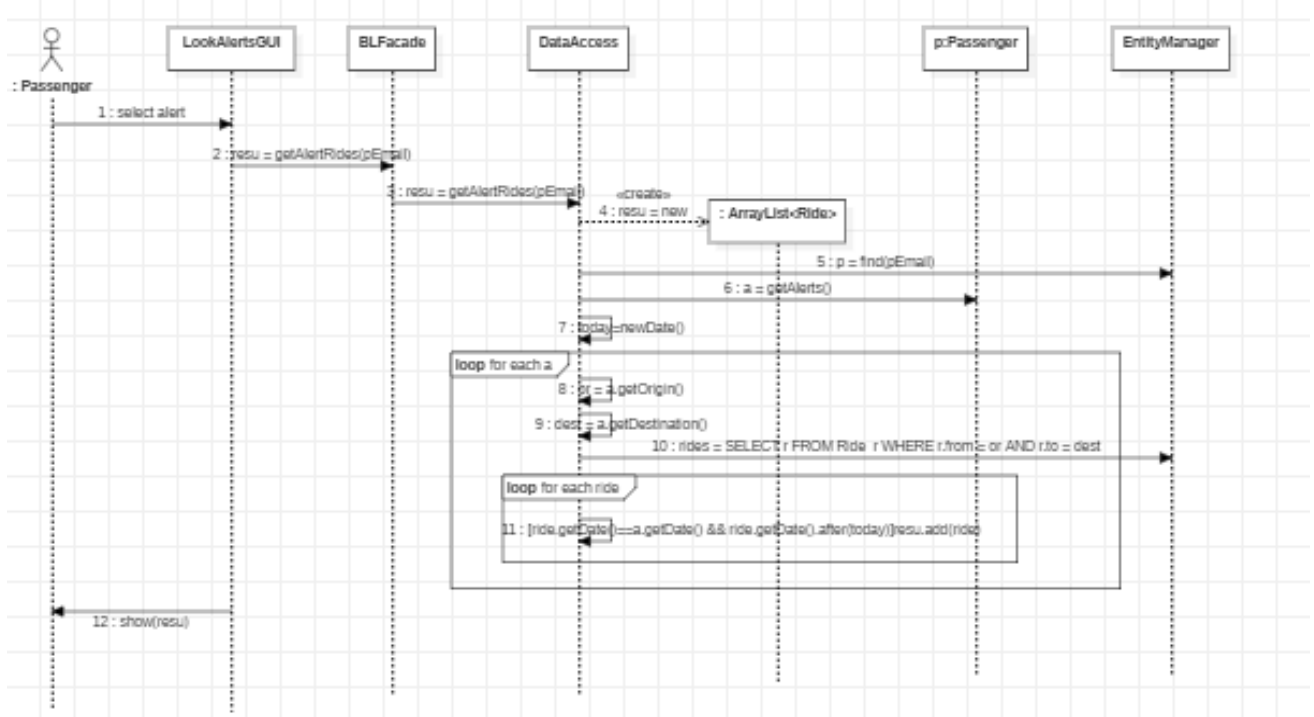
DISEINUA

1.-HIRUGARREN ITERAZIOKO SEKUENTZIA DIAGRAMAK:

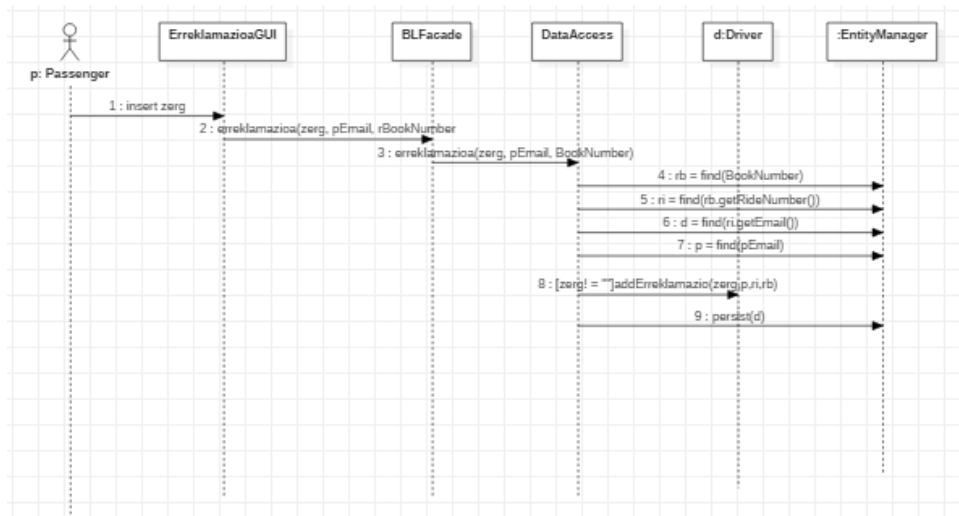
CREATE ALERT:



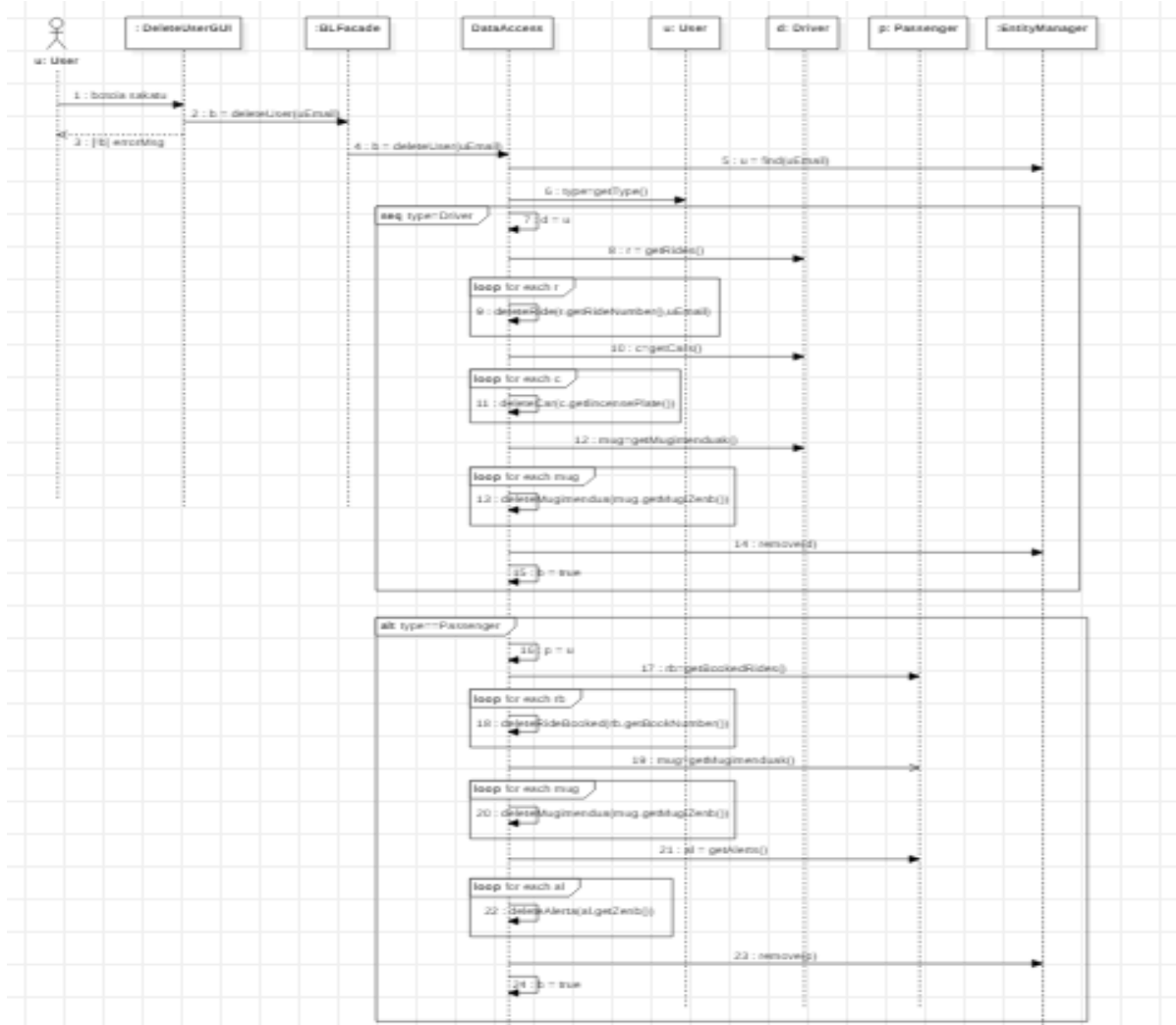
CHECK ALERT:



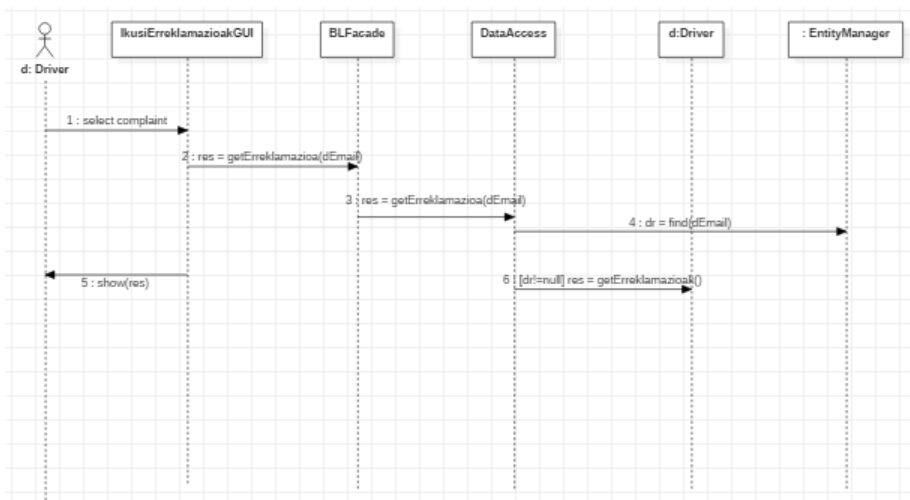
FILL COMPLAINT:



DELETE USER:



CHECK COMPLAINTS:



IMPLEMENTAZIOA:

-getDepartCities: Datu basean dauden bidai bakoitzaren irteera hiri guztiak itzultzen ditu lista batean.

-getDestinationCities: Datu basean dauden bidai bakoitzaren helmuga hiri guztiak itzultzen ditu lista batean.

-createRide: Irteera eta helmuga hiriak, data, eserlekuak, prezioa, autoa eta gidariaren email-a sartuz, bidai bat sortu eta datu basean gordetzen du. Sortutako bidaia itzultzen du.

-getRides: Sartutako irteera eta helmuga hiriak eta datarekin bat datozen bidaiez osatutako lista bat bueltatzen du.

-getThisMonthDatesWithRides: Datu batea abiarazten denean (hau da, config.xml-en “initialize” true deanean), hau bete egiten du hainbat bidaiekin.

-register: Sartutako korreoa, pasahitza duen erabiltzailea bat sortu eta gordetzen du datu basean. Aukeratzen (parametroz sartu) den motaren arabera, gidari bat edo bidaiari bat sortzen du.

-login: Sartzen den korreo, pasahitza eta motarekin bat datorren erabiltzaile bat aurkitzen bada datu basean, erabiltzaileari sarbidea ematen zaio aplikaziora, eta motaren arabera menu bat edo bestea aurkezten zaio.

-withdrawMoney: Korreo eta kopuru bat sartzen da. Sistemak korreo hori dagokion kontuari saldotik sartutako diru kopurua kentzen dio.

-addMoney: Korreo eta kopuru bat sartzen da. Sistemak korreo hori dagokion kontuaren saldoari sartutako kopurua gehitzen dio.

-getBookingsDriver: Korreo bat emanda kontu horretan sartzen da, eta dituen erreserbak bueltatzen ditu.

-bookRide: Passenger bat, gidariaren emaila, erreserbatu nahi dituzun eserlekuak eta bidaiaren id-a pasatuz. Id hori duen bidaia erreserbatzen du.

-accept: Erreserba baten identifikadorea sartuz, erreserba onartu egingo da.

-decline: Erreserba baten identifikadorea sartuz, erreserba ez da onartuko.

-getBookingsPass: Korreo bat emanda kontu horretan sartzen da, eta dituen erreserbak bueltatzen ditu.

-getMugimenduak: User bat emanda, user horrek egin dituen mugimenduak bueltatzen ditu.

-bidaiaEginda: Erreserba baten zenbakia, korreo bat eta balorazio bat sartuta, sistemak zenbaki hori dagokion erreserba datu basetik ezabatzen du, bidaia jada amaitu delako, eta bidaiari sartutako balorazioa ezartzen dio.

-getRidesDriver: Korreo bat sartuta, sistemak erabiltzailea bilatzen du eta honen bidai guztiak itzultzen ditu lista batean.

-deleteRide: Bidai baten zenbaki bat eta gidari baten korreoa sartuta, zenbaki hori dagokion bidaia datu basetik ezabatzen du.

-getDriversWallet: Gidari baten korreoa sartzen da eta gidari horren saldoa bueltatzen du funtzioak.

-getPassengersWallet: Bidaiari baten korreoa sartzen da eta metodoak bidaiari horren saldoa itzultzen du.

-addCar: Matrikula, marka, kolorea, eserleku kopurua eta korreo bat sartuz, auto berri bat sortu eta datu basean gordetzen da eta korreoa dagokion gidariari ezartzen zaio.

-getCars: Sartzen den korreoa dagokion gidariaren auto guztiak itzultzen ditu sistemak zerrenda batean.

-getRideByEmail: Korreo bat, irteera hiria, helmuga hiria eta data bat sartzen dira. Sistemak korreo hori duen gidaria bilatzen du, eta sartutako informazioarekin bat datorren bidaia itzultzen du, datu basean bilatuta.

-erreklamazioa: Testu bat, korreo bat eta erreserba zenbaki bat sartuz, sistemak erreklamazio berri bat sortzen du, korreorekin jabearekin eta erreserba zenbakiarekin bat datorren erreserbarekin lotzen du eta datu basean gordetzen du.

-geterreklamazio: Sartutako korreoa dagokion gidariak ezarrita dituen erreklamazio guztiak lista batean itzultzen ditu sistemak.

-getAllDrivers: Sistemak datu basean dauden gidari guztiak itzultzen ditu zerrenda batean.

-ezabatuErreklamazioa: Erreklamazio baten identifikatzailea sartuz, identifikatzaile horrekin bat datorren erreklamazioa datu basetik ezabatzen da.

-baieztatuErreklamazioa: Sartzen den identifikatzailea duen erreklamazioak lotuta duen bidaiariaren saldoari, erreklamazioari dagokion erreserbaren prezioa gehitzen zaio, hau da, erreklamazioa onartzen da eta dirua itzultzen da.

-deleteUser: Sartutako korreoari dagokion kontua datu basetik ezabatzen da.

-createAlert: Irteera hiria, helmuga hiria, data eta korreo bat sartzen dira. Alerta berri bat sortzen da informazio horrekin, korreo hori duen bidaiaria ezartzen zaio eta datu basean gordetzen da.

-getAlertRides: Korreo bat sartuta, sistemak korreo hori duen bidaiariak sortuta dituen alertekin bat datozen bidai guztiak itzultzen ditu zerrenda batean.

ONDORIOAK

Irakasgai honetan eguneroko lana oinarritzkoa da. Iterazioen bidez lan egitea proiektua burutzeko metodori egokiena dela uste dugu. Metodo honek, proiektua egunera eramatea laguntzen du, pixkanaka gauza berriak inplementatuz. Hiru iterazioetan zehar, zailtasun handienak datu basearekin izan ditugu, nola funtzionatzen duen ulertzean eta atzipenak egitean batez ere. Web-Zerbitzuak gure proiektuan inplementatzen ere zailtasun asko izan ditugu, ondorioz, funtzio hori ez inplementatzen. Nahiko pozik gaude lortutako proiektuarekin, baina GUI diseinuan eta Web-Zerbitzuen inplementazioan hobekuntzarako margina dagoela uste dugu.

Proiektua garatzeko metodologia guztiz egokia da gure ustez baina zenbait gauza hobetu daitezkeela ere pentsatzen dugu. Web-Zerbitzuak ikasgai honeko alderdi garrantzitsua da, baina garrantzia gutxi eman zaio gure ustez. Hainbat erabilpen kasu inplementatu izan beharrean, aplikazio sinpleago bat, baina Web-Zerbitzuetan sakontasun gehiago izatea da espero genuena.

Datorren urteko ikasleei, hasieratik aplikazioaren diseinu ona egitea gomendatuko genieeke, zuzenean programatzen ez hastea. Behin diseinu egokia eginda, pixkanaka-pixkanaka egunero programak inplementatzea da lan egiteko modu egokiena.

BIDEOA:

<https://upvehueus->

my.sharepoint.com/personal/aaldai005_ikarle_ehu_eus/_layouts/15/stream.aspx?id=%2Fpersonal%2Faaldai005_ikarle_ehu_eus%2FDocuments%2FSi_3lterazioa_Bideo%2Emp4&referrer=StreamWebApp%2EWeb&referrerScenario=AddressBarCopied%2Eview%2E8e3ca10f-e53a-4ec9-b881-9e7dce385445