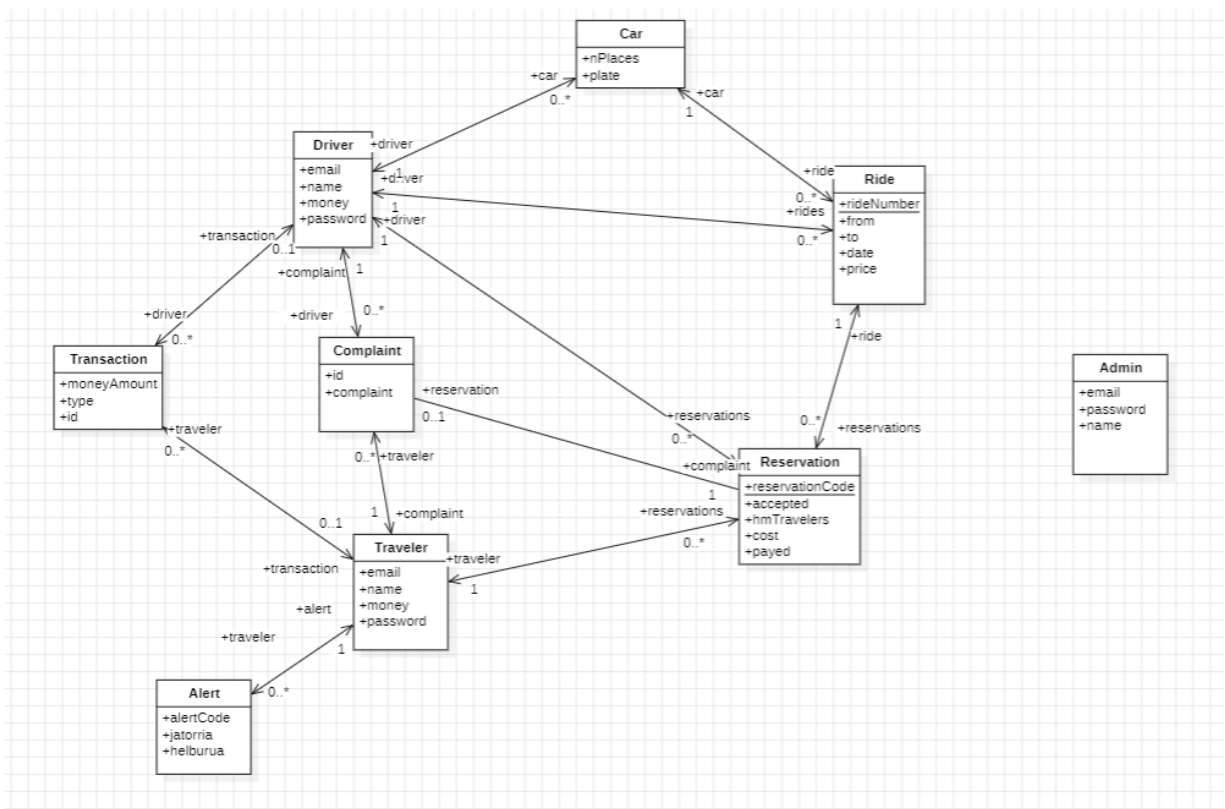


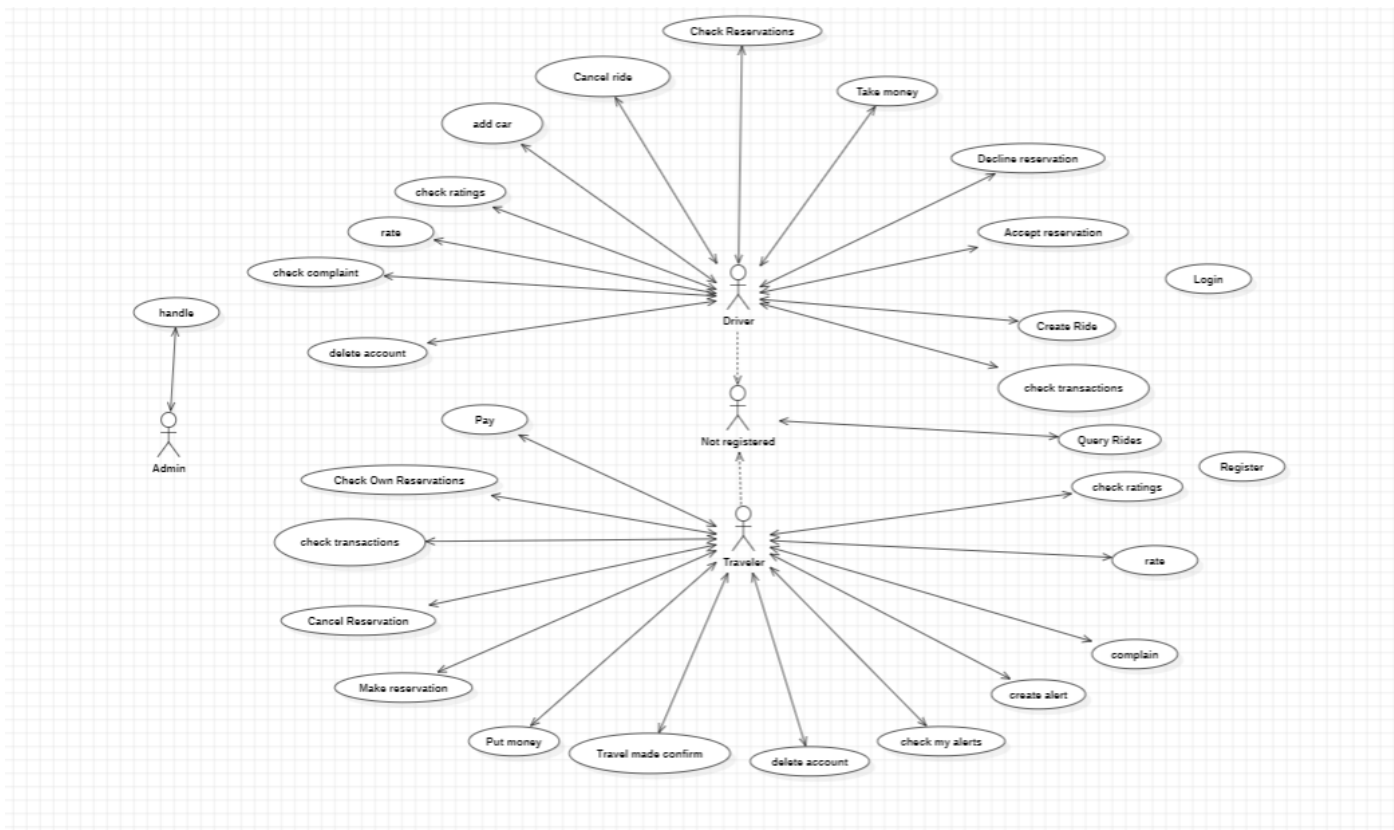
- **Egileak:** Eneko Castro Mata eta Aitzol Letamendi Martinez
- **Web Zerbitzua:** BAI, guztiz inplementatuta
- **Sarrera:**
  - Proiektu hau, bidaiak egiteko kotxea konpartitzeko aplikazio bat da. Aplikazio honetan hiru rol desberdin joka ditzake erabiltzaileak:
    - Administratzaileak bakarrik beste erabiltzaileen arteko kexen kudeaketa egin dezake. Ezin da edozein administratzaile bezala erregistratu, log in egin bakarrik.
    - Gidariak bidaiak eta kotxeak sortu eta bertan behera utzi ditzakete, bidaiariak egindako erreserbak ikusi eta onartu edo ezeztatu, irabazitako dirua beraien kontu korrontera atera, erreserbetako bidaiariak baloratu ditzakete, aplikazioan egondako diru mugimenduak ikusi, beraien balorazioak eta bidaiarien kexak ikusi ditzakete eta nahi izatekotan, kontua ezabatu.
    - Bidaiari bidaiak bilatu ditzakete eta ez existitzekotan interesatzen zaiena, alerta bat sortu dezakete aplikazioak abisatzeko bidai hori sortzen denean. Bidai horietan erreserbak sortu, beraien erreserba edo alerta zerrenda ikusi eta erreserbak ordaindu edo bertan behera utzi ditzakete. Aplikaziora dirua sartu dezakete beraien kontu korrontetik eta aplikazioan egon diren diru mugimenduen zerrenda ikusi dezakete. Bidaia egin eta gero, bidaia egon dela konfirmatu dezakete, gidaria baloratu eta nahi badute, kexa bat jarri. Bidaiariak eman dizkieten balorazioak ikusi ditzakete eta, nahi badute, kontua ezabatu.
- **Eskakizun bilketa:**



### Domeinuaren ereduak:

- **Car:** Driver baten kotxearen informazioa gordetzen duen klasea.
  - Bakoitzak, ride zerrenda bat izango du jakiteko zein ride-etan erabili den kotxe hori eta gidari bat izango du, kotxe horren jabea izango dena.
- **Driver:** Gidari lana egingo duen pertsonaren informazioa gordetzen duen klasea.
  - Bakoitzak hainbat car izan ditzake eta bidai bakoitzerako erabaki ahal izango du zein erabili. Hainbat ride izango ditu, driver horrek sortutako bidaien zerrenda. Hainbat complaint izango ditu, travelerrek jartzen dizkioten kexak. Eta transakzio zerrenda bat izango du, aplikaziotik ateratzen duen dirua, travelerren ordainketak edo administratzaileak dirua kentzea.
- **Ride:** Driver batek, jatorri batetik helmuga batera sortzen duen bidaiaren informazioa gordetzen duen klasea.
  - Bakoitzak car bat izango du, bidai hori egingo duen kotxearen informazioa. Driver bat izango du, kotxea gidatuko duen pertsonaren informazioa. Reservation zerrenda bat izango du, bidaia egongo duten pertsonen informazioa.
- **Complaint:** Traveler batek dirua bueltan nahi duenean, driver baten inguruan sortutako kexa bat gordeko duen klasea.
  - Driver bat izango du, reservation bat, zein erreserbatan eta zein bidaiatan egon den arazoa gordeko dituzten atributuak. Eta traveler bat izango du, kexa sortu duen pertsona.
- **Transaction:** Aplikazioan entitateen artean diru mugimendu desberdinen informazioa gordeko duen klasea:
  - 0 edo 1 driver izango ditu, eta 0 edo 1 traveler izango ditu transakzio motaren arabera. BANKU-TRAVELER, TRAVELER-DRIVER, DRIVER-TRAVELER edo DRIVER-BANKU motakoak izan daitezke.
- **Admin:** Kexen kudeaketa egingo duen pertsonaren informazioa gordeko duen klasea.
  - Ez dauka erlazorik.
- **Reservation:** Traveler batek ride batean egiten duen erreserbaren informazioa gordeko duen klasea.
  - Traveler bat izango du, erreserba egin duen pertsona. 0 edo 1 complain izan ditzake, travelerra kexatu den edo ez. Driver bat izango du, driver bat izango du, bidaia egingo duen pertsona eta ride bat izango du, erreserba egin den bidaiaren informazioa.
- **Traveler:** Bidaiak erreserbatu eta egingo dituen pertsonaren informazioa.
  - Alerta lista bat izango du, erreserba zerrenda bat, complaint zerrenda bat eta transakzio zerrenda bat.
- **Alert:** Traveler batek sortutako alerta baten informazioa gordeko duen klasea.
  - Traveler bat izango du, alerta sortu duen pertsonaren informazioa.

○ Erabilpen kasuen ereduak:



- **Not registered**
  - Login

## Basic Flow

1. System displays login menu
2. User writes their **email**, **password** and **rol**, and clicks the **button** to login
3. System checks the **information** and displays the corresponding menu

## Alternative flow

1. System checks and introduced **information** is wrong, so it asks again for the **information**.

The screenshot shows a web application window titled "Rides". On the left, there is a "Log in" section with labels for "Email" and "Password". Below these are three radio buttons for user roles: "Traveler" (selected), "Driver", and "Admin". At the bottom of this section is the text "Enter your Email and Password". On the right, there are two input fields corresponding to the email and password labels. Above the first input field is a "Back" button, and below the second input field is an "Enter" button.

- Register

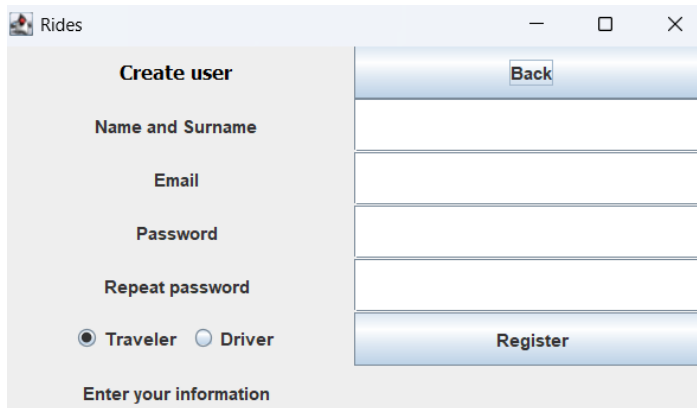
## Register Flow of events

### Basic Flow

1. *System* displays register **menu**
2. *User* writes their **name**, **email**, **password** and **rol**, and clicks the **button** to register
3. *System* checks the **information**, saves the **information** in the **data base** and displays the corresponding **menu**

### Alternative flow

1. *System* checks and introduced **information** is already used, so it asks the **user** to login.



The screenshot shows a window titled 'Rides' with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains a form titled 'Create user'. On the left side of the form, there are labels for 'Name and Surname', 'Email', 'Password', 'Repeat password', and radio buttons for 'Traveler' (selected) and 'Driver'. Below these is the text 'Enter your information'. On the right side, there are four text input fields corresponding to the labels on the left. At the top right of the form area is a 'Back' button, and at the bottom right is a 'Register' button.

- Admin
  - Handle

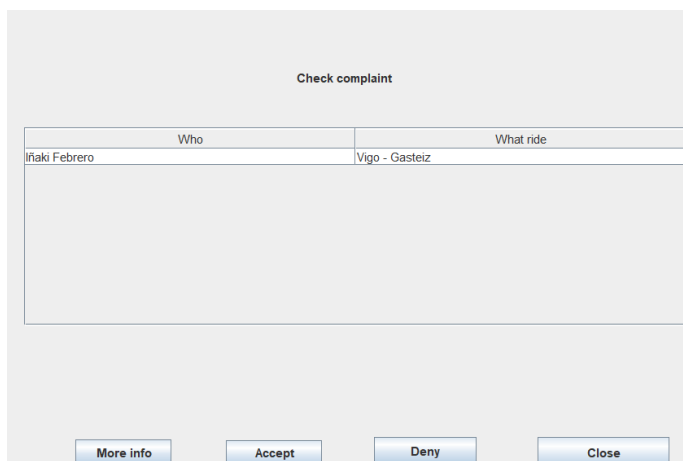
## Handle Flow of events

### Basic Flow

1. *System* displays Handle **menu**

### Alternative flow

1. There are no Complaint, so *System* displays a message saying that.



The screenshot shows a dialog box titled 'Check complaint'. It contains a table with two columns: 'Who' and 'What ride'. The first row of data shows 'Iñaki Febrero' under 'Who' and 'Vigo - Gasteiz' under 'What ride'. Below the table is a large empty rectangular area. At the bottom of the dialog box, there are four buttons: 'More info', 'Accept', 'Deny', and 'Close'.

Who	What ride
Iñaki Febrero	Vigo - Gasteiz

- **Driver**
  - Check transactions

## Check Transactions Flow of events

### Basic Flow

1. *System* displays Check Transactions **menu**.

### Alternative flow

1. There are no transactions, so *System* displays a message saying that.

your transactions

From	To	Amount
Naia Nuñez	Oier Lacalle	5.0
Oier Lacalle	Naia Nuñez	5.0

Close

- Create ride

## Create Ride Flow of events

### 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 **driver**. Ride is not created. System informs the user.

Depart City

Arrival Ci...

Car

Price

Ride date

May 2025

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
18						1	2
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

Create Ride Close

- Accept reservation

## Accept Reservation Flow of events

### Basic Flow

1. *System* displays **reservation menu**.
2. *Driver* selects a **reservation** and clicks the **accept button**.
3. *System* accepts the **reservation** and displays a message saying it has been accepted.

### Alternative flow

1. There are no **reservations**, so *System* displays a message saying that.

The screenshot shows a web interface titled "Rides". It contains a table with the following data:

Traveler	Seat amount	Ride	Ratings
Iñaki Febrero	1	Mordor-Tolosa	2.5
Iñaki Febrero	1	Vigo-Gasteiz	2.5
Naia Nuñez	1	Donostia-Araia	NaN

Below the table are three buttons: "Accept", "Deny", and "Close".

- Decline reservation

## Decline Reservation Flow of events

### Basic Flow

1. *System* displays **reservation menu**.
2. *Driver* selects a **reservation** and clicks the **decline button**.
3. *System* deletes the **reservation** and displays a message saying it has been declined.

### Alternative flow

1. There are no **reservations**, so *System* displays a message saying that.
2. *Driver* declines a **reservation** that has already been **paid**, so *System* removes the **reservation** and gives back the **money** to **traveler**.

The screenshot shows a web interface titled "Rides". It contains a table with the following data:

Traveler	Seat amount	Ride	Ratings
Iñaki Febrero	1	Mordor-Tolosa	2.5
Iñaki Febrero	1	Vigo-Gasteiz	2.5
Naia Nuñez	1	Donostia-Araia	NaN

Below the table are three buttons: "Accept", "Deny", and "Close".

- ## Decline Reservation Flow of events

1. *System* displays **reservation** menu.
2. *Driver* selects a **reservation** and clicks the **decline** button.
3. *System* deletes the **reservation** and displays a message saying it has been declined.

1. There are no **reservations**, so *System* displays a message saying that.
2. *Driver* declines a **reservation** that has already been **paid**, so *System* removes the **reservation** and gives back the **money** to *traveler*.

- Check reservation

## Check Reservations Flow of events

1. System displays Check Reservations menu
2. Driver selects a reservation
3. System displays the reservation's information.

1. There are no reservations, so *System* displays a message saying that.

**Rides**

Traveler	Seat amount	Ride	Ratings
Iñaki Febrero	1	Mordor-Tolosa	2.5
Iñaki Febrero	1	Vigo-Gasteiz	2.5
Naia Nuñez	1	Donostia-Araia	NaN

Accept

Deny

Close

- Cancel ride

## Check Reservations Flow of events

### Basic Flow

1. *System* displays Check Reservations **menu**
2. *Driver* selects a **reservation**
3. *System* displays the **reservation's information**.

### Alternative flow

1. There are no reservations, so *System* displays a message saying that.

**Rides**

From	To	Ride date
Mordor	Tolosa	Wed May 14 00:00:00 ...
Vigo	Gasteiz	Tue May 27 00:00:00 C...
Donostia	Araia	Wed May 14 00:00:00 ...

- Add car

## Create Ride Flow of events

### Basic Flow

1. *Driver* insert the **plate**, **seatsAvailable** and if it is prepared **for people with disabilities**.
2. *System* creates a **car** with inserted data and assigns to the **Driver**.

### Alternative flow

1. Some of fields are empty. Final.
2. **number of seat** is not a number or **plate** is not valid. System informs the user.
3. the **car** already exist for this **driver**. *Car* is not created. System informs the user.

**Plate** 
**Places**

☐ **For people with disabilities**



- Check ratings

## Check Ratings Flow of events

### Basic Flow

1. *System* displays Check Ratings **menu**

### Alternative flow

1. There are no Ratings, so *System* displays a message saying that.

Check own ratings					
1	2	3	4	5	Which rating
0	0	0	1	1	How much

Close

- Rate

## Rate Flow of events

### Basic Flow

1. *System* displays **Rate menu**.
2. *Driver* selects the **Traveler** that it wants to Rate and selects the **rate button**.
3. *System* makes the **rate**.

### Alternative Flow

1. The selected **traveler** is already been rated by this driver.

Rate traveler

☒ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Rate Close

- Check complaint

## Check Complaint Flow of events

### Basic Flow

1. *System* displays Check Complaint **menu**

### Alternative flow

1. There are no Complaint, so *System* displays a message saying that.

The screenshot shows a web interface titled "Check complaint". It features a table with two columns: "Who" and "What ride". The "Who" column contains the text "Iñaki Febrero" and the "What ride" column contains "Vigo - Gasteiz". Below the table, there is a message "Itzuli nire dirua" (Return my money). At the bottom, there are four buttons: "More info", "Accept", "Deny", and "Close".

Who	What ride
Iñaki Febrero	Vigo - Gasteiz

Itzuli nire dirua

More info Accept Deny Close

- Delete account

## Delete account Flow of events

### Basic Flow

1. *Driver* selects a **delete account button**.
2. *System* deletes the **Traveler**.

The screenshot shows a user menu interface. It has a header with "Welcome Oier Lacalle" and "Create Ride". Below the header, there is a table with two columns of menu items. The first column contains "Query Rides", "Create a car", "Check reservations", "Delete account", "Check ratings", and "Check complaint". The second column contains "Take money", "Check own rides", "Check transactions", "Log out", and "Rate travelers". At the bottom, there are three radio buttons for language selection: "Euskara", "Castellano", and "English".

Welcome Oier Lacalle	Create Ride
Query Rides	Take money
Create a car	Check own rides
Check reservations	Check transactions
Delete account	Log out
Check ratings	Rate travelers
Check complaint	

☐ Euskara ☐ Castellano ☐ English

- **Traveler**
  - Check ratings

## Delete account Flow of events

### Basic Flow

1. *Traveler* selects a **delete account button**.
2. *System* deletes the **Traveler**.

The screenshot shows a dialog box titled "Check own ratings". It contains a table with two rows and six columns. The first row is labeled "Which rating" and contains five radio buttons numbered 1 through 5. The second row is labeled "How much" and contains five text input fields, each with the number "0" entered. At the bottom center of the dialog is a "Close" button.

1	2	3	4	5	Which rating
0	0	0	0	0	How much

Close

- Rate

## Rate Flow of events

### Basic Flow

1. *System* displays **Rate menu**.
2. *Traveler* selects the **Driver** that it wants to Rate and selects the **rate button**.
3. *System* makes the **rate**.

### Alternative Flow

1. The selected **Driver** is already been rated by this traveler.

The screenshot shows a dialog box titled "Rate driver". It features five radio buttons labeled 1 through 5, arranged horizontally. Radio button 1 is selected. At the bottom of the dialog are two buttons: "Rate" on the left and "Close" on the right.

Rate driver

☒ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Rate Close

- Complaint

## Complaint Flow of events

### Basic Flow

1. *System* displays **Complaint menu**.
2. *Traveler* selects the **Driver** that it wants to Complaint and selects the **rate button**.
3. *System* makes the **Complaint**.

### Alternative Flow

1. The selected **Driver** is already been Complaint by this traveler.

<b>Make complaint</b>	Back
Complaint	
Write your complaint	Create

- Create alert

## Create alert Flow of events

### Basic Flow

1. *Traveler* insert the **departing city, arrival city, number of seats**
2. *Traveler* selects a **ride date**
3. *System* creates a **alert** with inserted data.

### 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 exists. Ride is not created. System informs the user.

<b>Create alert</b>	Back
From	
To	
Create an alert here	Create

- Check my alerts

## Check alert Flow of events

### Basic Flow

1. *System* displays Check alert **menu**

### Alternative flow

1. There are no alert, so *System* displays a message saying that.

Rides

From	To	Created
Donostia	Araia	true

- Delete account

## Delete account Flow of events

### Basic Flow

1. *Traveler* selects a **delete account button**.
2. *System* deletes the **Traveler**.

Welcome Naia Nuñez	Query Rides
Put money	Check own reservations
Check transactions	Delete account
Check own ratings	Log out
Check alerts	Create alert

☐ Euskara
 ☐ Castellano
 ☐ English

- Travel made confirm

## Travel made confirm Flow of events

### Basic Flow

1. *Traveller* selects the **reservation** it wants to confirm.
2. *System* confirms the ride

### Alternative Flow

1. The **traveller** does not have rides to confirm.
2. The **ride** date has not arrived yet.

Traveler	Seat amount	Ride	Accepted
Naia Nuñez	1	Donostia-Araia	true
Naia Nuñez	1	Bilbo-Donostia	false
Naia Nuñez	1	Vigo-Gasteiz	true

Buttons: Pay, Deny, Confirm, Complaint, Rate, Close

- Put money

## Put Money Flow of events

### Basic Flow

1. *System* displays the **put money** menu.
2. *Traveller* writes their **bank account information** and indicates the **amount of money** they want to put and presses the **accept** button.
3. *System* puts money on **travellers's money**.

### Alternative Flow

1. *Traveller's* bank information is wrong so **system** ask them to write it again.
2. *Traveller* does not have enough **money**.

Put money, you have: 1000000.0 €

Back

Card number

Expire date

CVC

Name and Surname

Amount of money to put in (euros)

Put

- Make reservation

## Make Reservation Flow of events

### Basic Flow

1. *System* displays **make reservation** menu.
2. *Traveller* selects the **ride** that it wants to reserve and selects the **make reservation button**.
3. *System* makes the **reservation**.

### Alternative Flow

1. The selected **ride** is already full.

**Make reservation**

How many travelers

Specify the amount of travelers in the res..

Back

Create

- Cancel resevation

## Cancel Reservation Flow of events

### Basic Flow

1. *System* displays selected **reservation's information**.
2. *Traveller* pushes **cancel button**.
3. *System* cancels the **reservation**.

### Alternative Flow

1. Selected **reservation** is already paid so **traveller** cannot cancel the **reservation**.

**Rides**

Traveler	Seat amount	Ride	Accepted
Naia Nuñez	1	Donostia-Araia	true
Naia Nuñez	1	Bilbo-Donostia	false
Naia Nuñez	1	Vigo-Gasteiz	true

Pay Cancel Confirm Complaint Rate

Close

- Check transaction

## Check Transactions Flow of events

### Basic Flow

1. *System* displays Check Transactions **menu**.

### Alternative flow

1. There are no transactions, so *System* displays a message saying that.

your transactions

From	To	Amount
Bank	Naia Nuñez	1000000.0
Naia Nuñez	Oier Lacalle	5.0
Oier Lacalle	Naia Nuñez	5.0

Close

- Check own reservations

## Check Own Reservations Flow of events

### Basic Flow

1. *System* displays Check Own Reservations **menu**
2. *Traveller* selects a **reservation**
3. *System* displays the **reservation's information**.

### Alternative flow

1. There are no reservations, so *System* displays a message saying that.

Rides

Traveler	Seat amount	Ride	Accepted
Naia Nuñez	1	Donostia-Araia	true
Naia Nuñez	1	Bilbo-Donostia	false
Naia Nuñez	1	Vigo-Gasteiz	true

Pay Cancel Confirm Complaint Rate

Close



- Pay

## Pay Flow of events

### Basic Flow

1. Traveller selects the **reservation** it wants to pay for.
2. System takes the **money** from the **traveller's money** and puts it in the **driver's money** if the driver has previously accepted the **reservation**.

### Alternative Flow

1. The **traveller** does not have enough money.
2. The **reservation** is not accepted yet.

### Rides

Traveler	Seat amount	Ride	Accepted
Naia Nuñez	1	Donostia-Araia	true
Naia Nuñez	1	Bilbo-Donostia	false
Naia Nuñez	1	Vigo-Gasteiz	true

Pay

Cancel

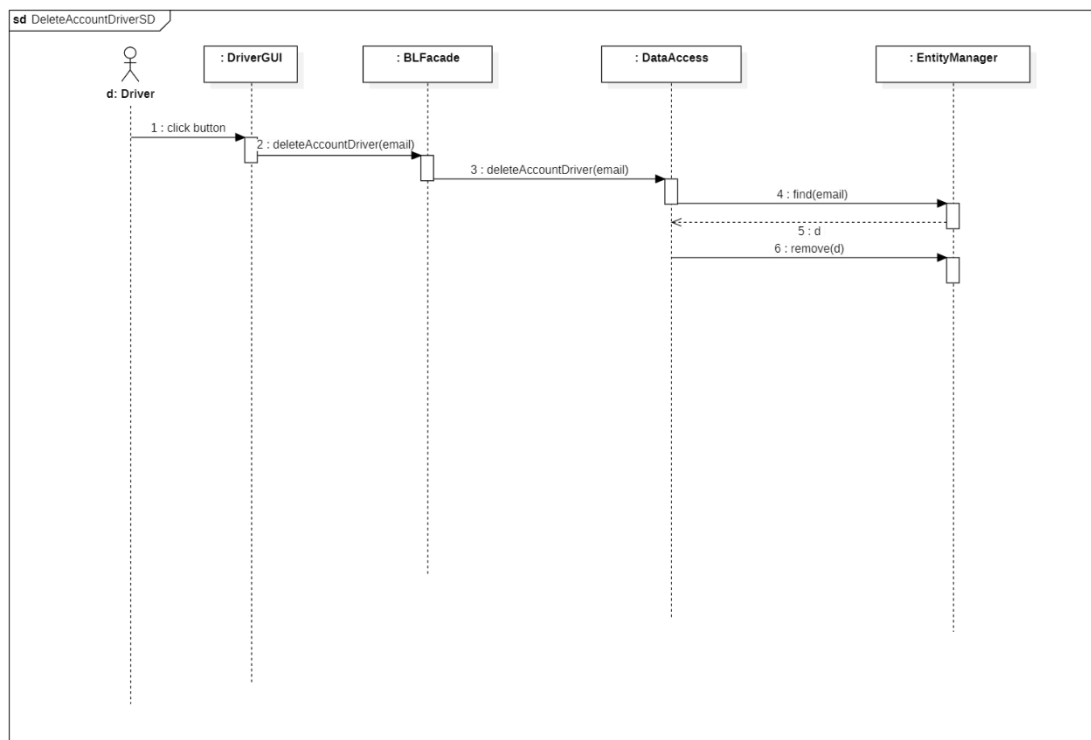
Confirm

Complaint

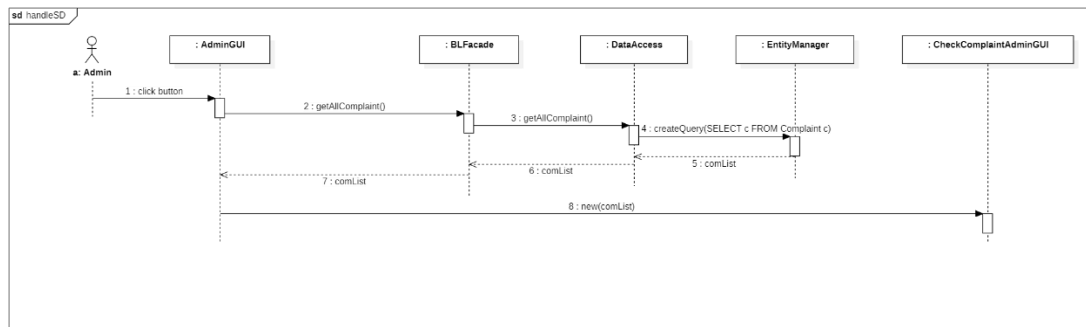
Rate

Close

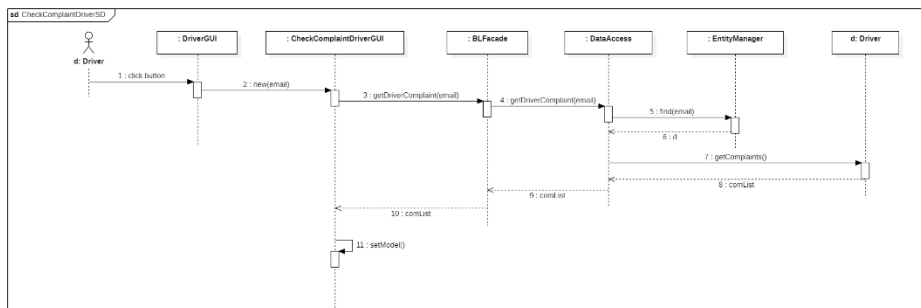
- Diseinua
  - Sekuentzia diagramak
    - **DeleteAccountDriverSD**: Driverrak erabiltzen ari den kontua ezabatzeko erabilpen kasua.



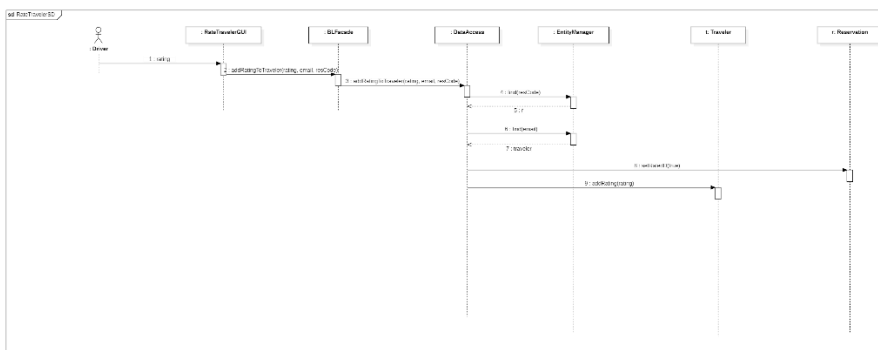
- **HandleSD:** Administratzaileak heltzen zaizkion kexak kudeatzeko erabilpen kasua.



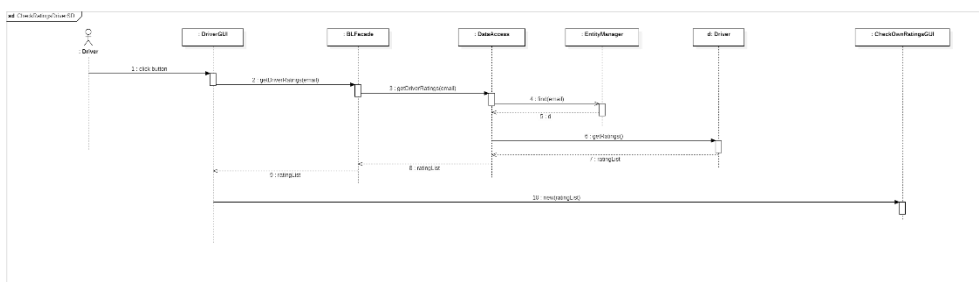
- **CheckComplaintDriverSD:** Driverrek, haiei jarri dizkieten kexak ikusteko erabilpen kasua.



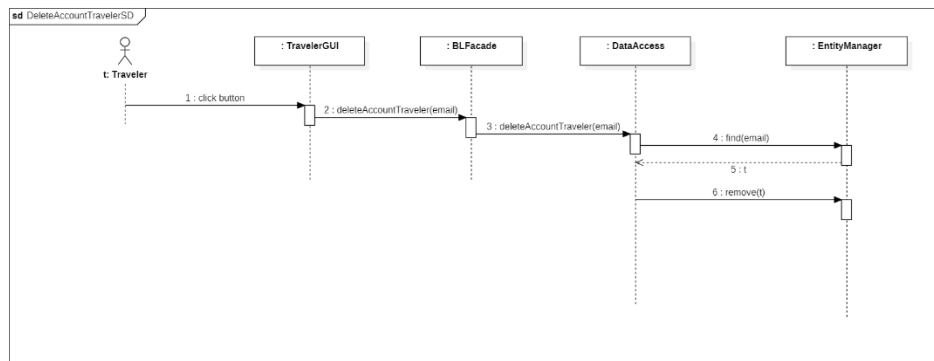
- **RateTravelerSD:** Driverrek beraien bidaietan joan diren bidaiariak baloratzeko erabilpen kasua.



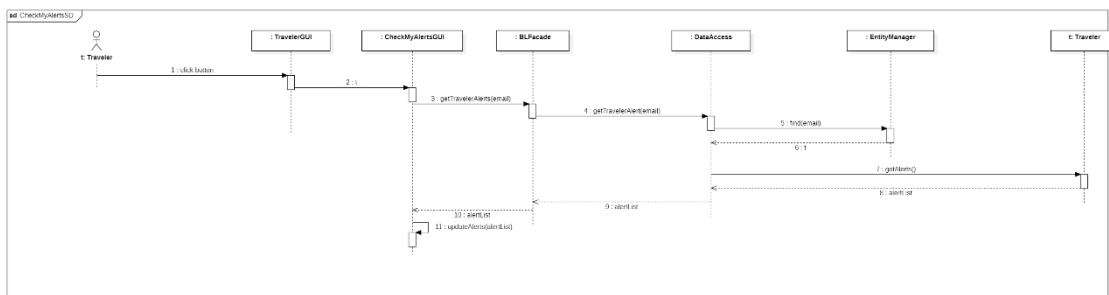
- **CheckRatingsDriverSD:** Driverrek beraien balorazioak ikusteko erabilpen kasua



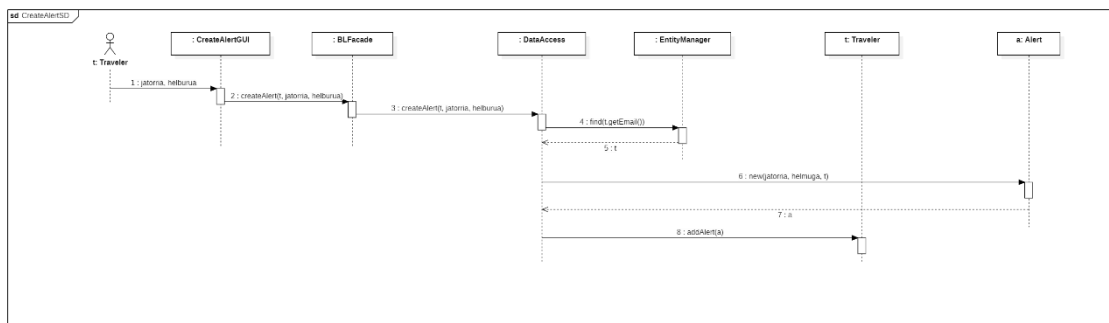
- **DeleteAccountTravelerSD:** Travelerek beraien kontua ezabatzeko erabilpen kasua.



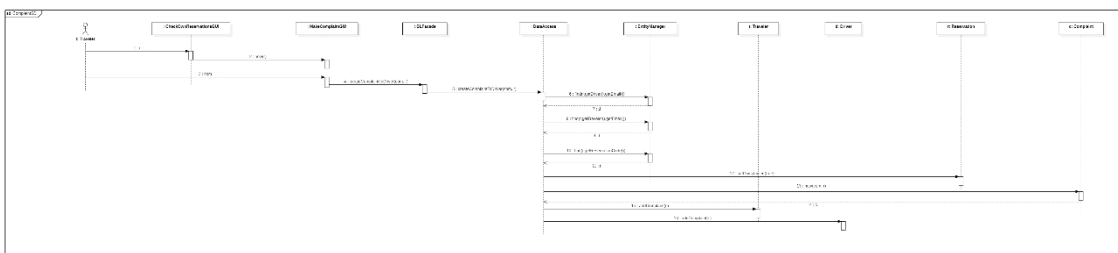
- **CheckMyAlertsSD:** Travelerrek beraien balorazioak ikusteko erabilpen kasua



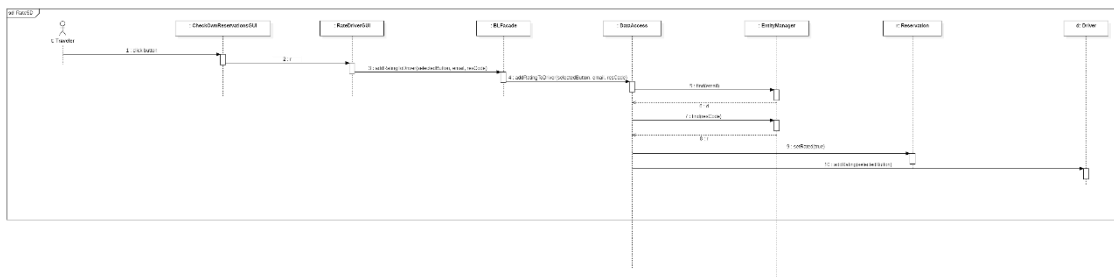
- **CreateAlertSD:** Traveler batek alerta bat sortzeko erabilpen kasua



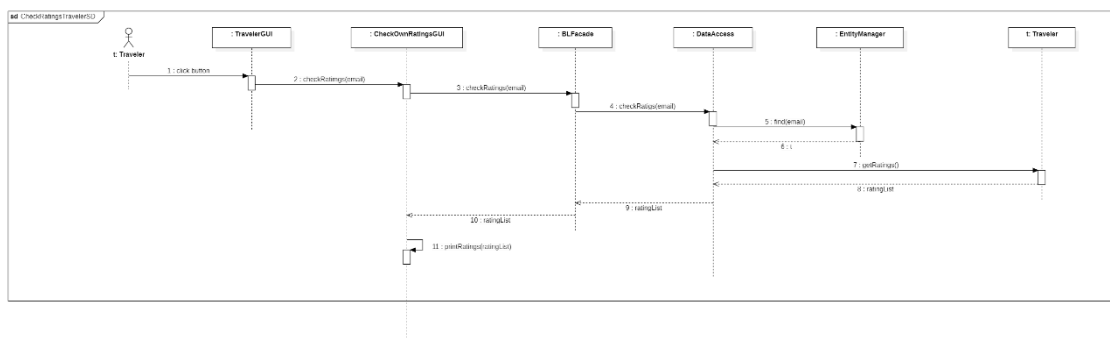
- **ComplaintSD:** Traveler batek driver bati buruz kexa bat sortzeko erabilpen kasua



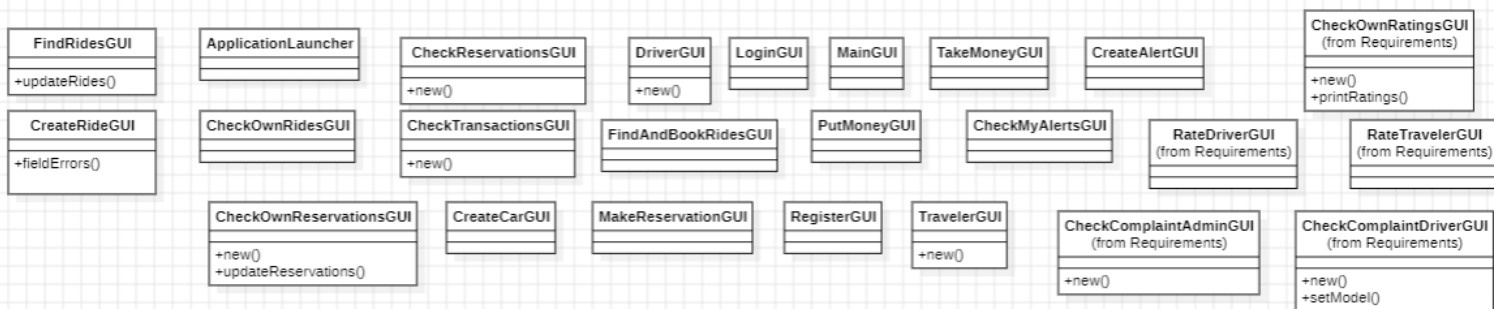
- **RateSD: Traveler batek driver bat baloratzeko erabilpen kasua**



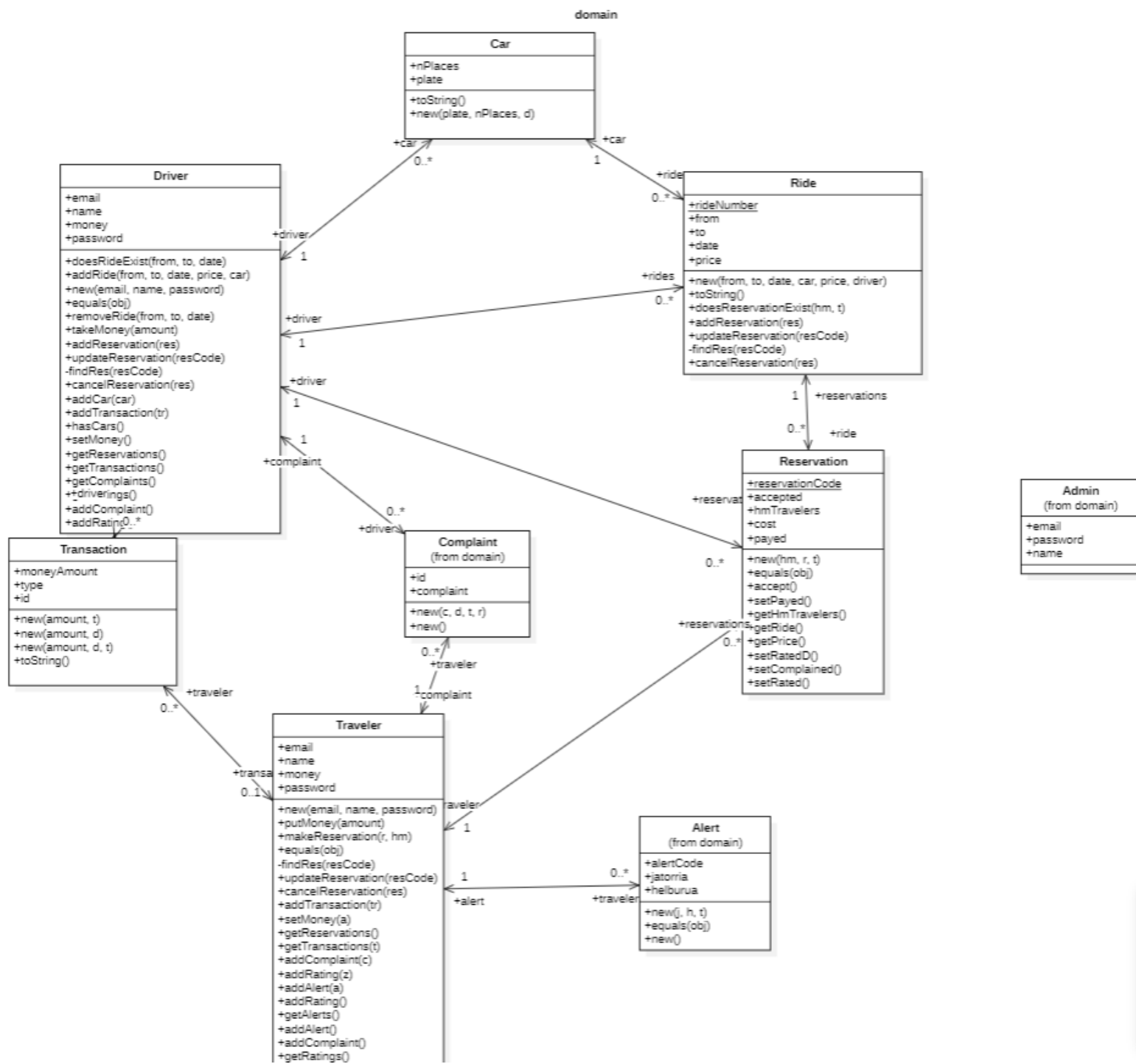
- **CheckRatingsTravelerSD: Traveler batek bere balorazioak ikusteko erabilpen kasua**



- **Klase diagrama**



BLFacade	DataAccess
+getDepartCities() +getArrivalCities(city) +getThisMonthDaysWithRides(from, to, today) +getRides(from, to, date) +createRide(from, to, date, price, driverEmail, carPlate) +initializeDB() +createDriver(email, name, password) +createTraveler(email, name, password) +createReservation(hm, rideNumber, travelerEmail) +getDriverByEmail(email, password) +getTravelerByEmail(email, password) +pay(res) +close() +getDriverReservations(email) +getTravelerreservations(email) +takeMoneyDriver(email, hm) +putMoneyTraveler(email, hm) +updateReservation(res) +cancelReservation(res) +addCarToDriver(driverEmail, carPlate, nPlaces) +getTravelerTransactions(email) +getDriverTransactions(email) +getDriverCars(email) +removeRideDriver(rideNumber, email) +getDriverRides(email) +pay() +deleteAccountDriver(email) +deleteAccountTraveler(email) +getDriverComplaint(email) +createComplaintToDriver(complaint, reservation) +addRating(email, zenb) +createAlert(traveler, from, to) +isRideBeenCreated(alert) +getTravelerAlert(email) +deleteAccountDriver() +getAllComplaint() +getDriverComplaint() +addRatingToTraveler() +getDriverRatings() +deleteAccountTraveler() +getTravelerAlerts() +createAlert() +createComplaintToDriver() +addRatingToDriver() +checkRatings()	+getDepartCities() +getArrivalCities(city) +getThisMonthDaysWithRides(from, to, today) +getRides(from, to, date) +createRide(from, to, date, price, driverEmail, carPlate) +initializeDB() +createDriver(email, name, password) +createTraveler(email, name, password) +createReservation(hm, rideNumber, travelerEmail) +getDriverByEmail(email, password) +getTravelerByEmail(email, password) +pay(res) +close() +getDriverReservations(email) +getTravelerreservations(email) +takeMoneyDriver(email, hm) +putMoneyTraveler(email, hm) +updateReservation(res) +cancelReservation(res) +addCarToDriver(driverEmail, carPlate, nPlaces) +getTravelerTransactions(email) +getDriverTransactions(email) +getDriverCars(email) +removeRideDriver(rideNumber, email) +getDriverRides(email) +deleteAccountDriver(email) +deleteAccountTraveler(email) +getDriverComplaint(email) +createComplaintToDriver(complaint, reservation) +addRating(email, zenb) +createAlert(traveler, from, to) +doesAlertExist(traveler, from, to) +isRideBeenCreated(alert) +getTravelerAlert(email) +deleteAccountDriver() +getAllComplaint() +getDriverComplaint() +addRatingToTraveler() +getDriverRatings() +deleteAccountTraveler() +getTravelerAlert() +createAlert() +createComplaintToDriver() +addRatingToDriver() +checkRatigs()



- **Implementazioa:** (Bakarrik guk egindakoak)

- **createDriver:** Sarrera bezala, driverraren emaila, izena eta pasahitza jasota; emailarekin datu basean begiratzen du ea erabilia dagoen eta driverrik existitzekotan errorea ematen du, bestela erabiltzaile berria sortzen du datu basean.
- **createTraveler:** Sarrera bezala, travelerraren emaila, izena eta pasahitza jasota; emailarekin datu basean begiratzen du ea erabilia dagoen eta travelerrik existitzekotan errorea ematen du, bestela erabiltzaile berria sortzen du datu basean.
- **createReservation:** Sarrera bezala zenbat bidaiarientzat egingo den erreserba, bidaiaren identifikadorea eta erreserba egiten ari den erabiltzailearen emaila jasota, identifikadorearekin datu basean bidaia bilatzen du eta leku nahikoa geratzen den begiratzen du. Ez badago leku nahikorik, errorea bidaltzen du, bestela erreserba egin nahi duen bidaiaria eta bidaiaren gidaria datu basetik eskuratu eta erreserba sortzen du.
- **getDriverByEmail:** Email eta pasahitza emanda, datu basean begiratzen du ea email hori existitzen den eta horrela izatekotan, pasahitza zuzena bada, datu basetik gidaria eskuratzen du.
- **getTravelerByEmail:** Email eta pasahitza emanda, datu basean begiratzen du ea email hori existitzen den eta horrela izatekotan, pasahitza zuzena bada, datu basetik bidaiaria eskuratzen du.
- **Pay:** reservation bat sarrera bezala jasota, erreserbaren informazioarekin gidaria eta bidaiaria eskuratzen ditu datu basetik eta erreserbaren prezioa kalkulatzeko du (bidaiaren kostua \* erreserban doazen pertsona kopurua). Bidaiariak ordainketa egiteko diru nahikoa ez badauka, errore bat jaurtitzen du. Bestela, erreserba datu basetik eskuratu, ordaindu dela adierazten du eta transakzio bat sortzen du bidaiariaren eta gidariaren artean.
- **getDriverReservations:** Email bat jasota, gidaria datu basetik eskuratzen du eta bere erreserba guztien zerrenda itzultzen du.
- **getTravelerReservations:** Email bat jasota, bidaiaria datu basetik eskuratzen du eta bere erreserba guztien zerrenda itzultzen du.
- **takeMoneyDriver:** Sarrera bezala email eta diru kopurua jasota, datu basetik gidaria eskuratu eta bere saldoa eguneratzen du eta transakzioa sortzen du bankua eta gidariaren artean. Ateratzeko diru nahikorik ez badauka, errore bat jaurtitzen du.
- **putMoneyTraveler:** Sarrera bezala email eta diru kopurua jasota, datu basetik bidaiaria eskuratu eta bere saldoa eguneratzen du eta transakzioa sortzen du bankua eta bidaiariaren artean.
- **updateReservation:** Erreserba bat sarrera bezala jasota, erreserbaren informazioari esker, gidaria, bidaiaria eta bidaia datu basetik eskuratzen ditu eta bakoitzak atributu bezala duen erreserba eguneratzen du.
- **cancelReservation:** Erreserba bat sarrera bezala jasota, erreserbaren informazioari esker, gidaria, bidaiaria eta bidaia datu basetik eskuratzen ditu eta bakoitzak atributu bezala duen erreserba bertan behera uzten du eta datu basetik ezabatzen du.

- **addCarToDriver:** Gidariaren emaila, kotxearen eserleku kopurua, ezgaitasundun pertsonentzako prest dagoen edo ez eta kotxearen matrikula jasota; gidaria datu basetik eskuratzen du eta kotxea ez badago sortuta (matrikula ez da oraindik erabili), kotxea sortzen du eta gidariari esleitzen dio, bestela errorea jaurtitzen du.
- **getTravelerTransactions:** Email bat jasota, bidaiaria datu basetik eskuratzen du eta bere transakzio guztien zerrenda itzultzen du.
- **getDriverTransactions:** Email bat jasota, gidaria datu basetik eskuratzen du eta bere transakzio guztien zerrenda itzultzen du.
- **getDriverCars:** Email bat jasota, gidaria datu basetik eskuratzen du eta bere kotxe guztien zerrenda itzultzen du.
- **removeRideDriver**
- **getDriverRides:** Email bat jasota, gidaria datu basetik eskuratzen du eta bere bidaia guztien zerrenda itzultzen du.
- **deleteAccountDriver:** Email bat jasota, gidaria datu basetik eskuratzen du eta ezabatzen du.
- **deleteAccountTraveler:** Email bat jasota, bidaiaria datu basetik eskuratzen du eta ezabatzen du.
- **getDriverComplaint:** Email bat jasota, gidaria datu basetik eskuratzen du eta bere kexa guztien zerrenda itzultzen du.
- **createComplaintToDriver:** Kexaren deskribapena eta erreserba jasota, erreserbaren gidaria, bidaiaria eta erreserba datu basetik eskuratzen ditu. Erreserbari bere kexatuta atributua eguneratzen dio , kexa sortzen du eta bidaiariari eta gidariari esleitzen dizkie eta kexa datu basean gordetzen du.
- **addRatingToTraveler:** Email bat, balorazio bat eta erreserbaren kodea parametro bezala jasota, datu basetik erreserba eta bidaiaria eskuratzen ditu. Erreserbaren baloratuta atributua eguneratzen du eta bidaiariari balorazioa ematen dio.
- **addRatingToDriver:** Email bat, balorazio bat eta erreserbaren kodea parametro bezala jasota, datu basetik erreserba eta gidaria eskuratzen ditu. Erreserbaren baloratuta atributua eguneratzen du eta gidariari balorazioa ematen dio.
- **createAlert:** Bidaiari bat, jatorri bat eta helmuga bat sarrera bezala jasota, alerta dagoeneko sortuta dagoen begiratzen du. Horrela bada, abisua jaurtitzen du. Bestela, bidaiaria datu basetik eskuratzen du, alerta berria sortzen du eta bidaiariari esleitzen dio.
- **isRideBeenCreated:** Alerta bat sarrera bezala jasota, datu basean bilatzen ditu alerta horren jatorria eta helburua duten bidaia guztiak. Zerbait aurkitzen badu, aurkitutako bidaien zerrenda itzultzen du.
- **getTravelerAlert:** Email bat jasota, bidaiaria datu basetik eskuratzen du eta bere alerta guztien zerrenda itzultzen du.
- **RemoveAlert:** Alerta baten identifikadorea sarrera bezala jasota, alerta datu basetik eskuratzen du eta ezabatzen du.
- **getDriverRatings:** Email bat jasota, gidaria datu basetik eskuratzen du eta bere balorazio guztien zerrenda itzultzen du.
- **getTravelerRatings:** Email bat jasota, bidaiaria datu basetik eskuratzen du eta bere balorazio guztien zerrenda itzultzen du.



- **getAdminByEmail:** Email eta pasahitza emanda, datu basean begiratzen du ea email hori existitzen den eta horrela izatekotan, pasahitza zuzena bada, datu basetik administratzailea eskuratzen du.
- **getAllComplaint:** Datu basetik kexa guztiak eskuratzen ditu eta lista batean itzultzen ditu.
- **denyComplaint:** Kexa bat jasota, kexa datu basetik eskuratzen du eta ezeztatuta bezala jartzen du.
- **acceptComplaint:** Kexa bat jasota, datu basetik kexa, bidaiaria, gidaria eta erreserba eskuratzen ditu. Erreserbaren prezioa kalkulatu du eta bidaiariari bere dirua bueltatzen dio. Gero, gidaritik eta bidaiaritik kexa ezabatzen du eta kexa datu basetik ezabatzen du. Azkenik, transakzio bat sortzen du gidariaren eta bidaiariaren artean.
- **denyComplaintAdmin:** Kexa bat jasota; kexa, bidaiaria eta gidaria datu basetik eskuratzen ditu. Gero, gidaritik eta bidaiaritik kexa ezabatzen du eta kexa datu basetik ezabatzen du.
- **updateAlerts:** Jatorri bat eta helburu bat jasota, datu basetik eskuratzen ditu jatorri eta helburu hori dituzten alerta guztiak eta bidaia sortu dela adierazten duen atributua eguneratzen die guztiei.
- **Ondorioak**
  - Proiektua orokorren gustatu zaigu eta aplikazioaren helburua interesgarria iruditu zaigu. Oso ondo etorri zaigu eginda heldu zaigun lan guztia, beraz ez dugu proiektuan hobekuntza handirik ikusten.
  - Proiektuarekin ez dugu arazo handirik izan, hasieran kostatu zitzaigun window builder ulertzea, interfaze grafikoak egiten ditugun lehenengo aldia zelako baina hortik aurrera nahiko ondo eraman dugu proiektua.
  - Taldearen dinamika espero zenaren zeharo desberdina izan da, oso hasieran 4-tik, 2 taldekide bakarrik geratu ginelako. Beraz, lan guztia elkarrekin egin dugu eta Scrum master rola ez du pisu handirik izan.
  - Lehenengo aldia izan da konplexutasun honetako proiektu bat egiten dugula, datu basearen kudeaketa, interfaze grafiko eta web zerbitzuarekin beste gauza askoren artean. Baina gai izan gara azkar ikasteko eta ez dugu ikusi proiektuak gainetik pasatzen gintuela, maila egokia iruditzen zaigu bigarren mailan suposatzen zaigun mailarako.
  - Proiektuetan oinarritutako irakaskuntza oso interesgarria iruditu zaigu, guk biok behintzat askoz gehiago ikasten dugulako ariketak landuz eta gauzak bisualko ikusiz teoria hutsa ematen baino; beraz guri askoz hobeto datorkigu metodologia hau.
- **Bideoaren URL-a: <https://youtu.be/YbCrAKboujU>**
- **Kodearen zip entregatuta**