REFACTOR

- **Egileak:** Asier Aldai eta Aimar Villegas
- **GitHub helbidea:** https://github.com/Aimarville/Rides25

Bad Smell Errefaktorizatu

Asier:

- ### Write short units of code:
  - #### Hasierako kodea:

```
241    public boolean login(String email, String password, String type) {
242        List<User> res = null;
243
244        TypedQuery<User> query = db.createQuery("SELECT u FROM User u WHERE u.email=?1 AND u.password=?2 AND u.type=?3",User.class);
245
246        query.setParameter(1, email);
247        query.setParameter(2, password);
248        query.setParameter(3, type);
249        res = query.getResultList();
250
251        if(res.size()!=0){
252            return true;
253        }else {
254            return false;
255        }
256    }
257
```

  - #### Errefaktorizatutako kodea:

```
241    public boolean login(String email, String password, String type) {
242        List<User> res = null;
243        TypedQuery<User> query = db.createQuery("SELECT u FROM User u WHERE u.email=?1 AND u.password=?2 AND u.type=?3",User.class);
244        query.setParameter(1, email);
245        query.setParameter(2, password);
246        query.setParameter(3, type);
247        res = query.getResultList();
248        if(res.size()!=0){
249            return true;
250        }else {
251            return false;}
252    }
253
```

  - #### Egindako errefaktorizazioaren deskribapena:
    - Beharrezkoak ez ziren hutsuneak kendu eta if/else-ren kortxeteak aurreko lerrora igo.

- **Write simple units of code:**
  - **Hasierako kodea:**

```
794    public List<Ride> getAlertRides(String pGmail){
795
796        List<Ride> resu = new ArrayList<Ride>();
797        Passenger pass = null;
798        pass = db.find(Passenger.class, pGmail);
799
800        List<Alerta> a = pass.getAlerts();
801
802        Date today = new Date();
803
804        for(Alerta al : a) {
805            TypedQuery<Ride> query = db.createQuery("SELECT r FROM Ride r WHERE r.from=?1 AND r.to=?2",Ride.class);
806            query.setParameter(1, al.getOrigin());
807            query.setParameter(2, al.getDestination());
808            List<Ride> rides = query.getResultList();
809            for(int i = 0; i<rides.size();i++) {
810                if(rides.get(i).getDate().getDay()==al.getDate().getDay()) {
811                    if(rides.get(i).getDate().after(today)) {
812                        resu.add(rides.get(i));
813                    }
814                }
815            }
816        }
817        return resu;
818    }
```

  - **Errefaktorizatutako kodea:**

```
794    public List<Ride> getAlertRides(String pGmail){
795
796        List<Ride> resu = new ArrayList<Ride>();
797        Passenger pass = null;
798        pass = db.find(Passenger.class, pGmail);
799
800        List<Alerta> a = pass.getAlerts();
801
802        Date today = new Date();
803
804        for(Alerta al : a) {
805            TypedQuery<Ride> query = db.createQuery("SELECT r FROM Ride r WHERE r.from=?1 AND r.to=?2",Ride.class);
806            query.setParameter(1, al.getOrigin());
807            query.setParameter(2, al.getDestination());
808            List<Ride> rides = query.getResultList();
809            for(int i = 0; i<rides.size();i++) {
810                if(rides.get(i).getDate()==al.getDate() && rides.get(i).getDate().after(today)) {
811                    resu.add(rides.get(i));
812                }
813            }
814        }
815        return resu;
816    }
```

  - **Egindako errefaktorizazioaren deskribapena:**
    - Azken 2 if-ak batean batu ditut, horrela konplexutasuna 5-etik 4-ra jaisten da.

- **<u>Duplicate code:</u>** (DataAccess-en ez dago duplikaziorik, beraz FindAndBookGUI klasean aurkitutako kode duplikatu zati bat zuzenduko dut).
    - **Hasierako kodea:**

```
private JComboBox<String> jComboBoxOrigin = new JComboBox<String>();
DefaultComboBoxModel<String> originLocations = new DefaultComboBoxModel<String>();

private JComboBox<String> jComboBoxDestination = new JComboBox<String>();
DefaultComboBoxModel<String> destinationCities = new DefaultComboBoxModel<String>();
```

    - **Errefaktorizatutako kodea:**

```
private JComboBox<String> jComboBoxOrigin = createJComboBox();
DefaultComboBoxModel<String> originLocations = createDefComboBoxModel();

private JComboBox<String> jComboBoxDestination = createJComboBox();
DefaultComboBoxModel<String> destinationCities = createDefComboBoxModel();


  public DefaultComboBoxModel<String> createDefComboBoxModel(){
      return new DefaultComboBoxModel<String>();
  }

  public JComboBox<String> createJComboBox(){
      return new JComboBox<String>();
  }
```

    - **Egindako errefaktorizazioaren deskribapena:**
        - Errepikapena kendu eta hauen ordez berdina egiten duen metodoak definitu.

- **Keep unit interfaces small:**
  - **Hasierako kodea:**

```
    */
    public Ride createRide(String from, String to, Date date, int nPlaces, float price, Car car, String driverEmail) throws  RideAlread
        System.out.println(">> DataAccess: createRide=> from= "+from+" to= "+to+" driver="+driverEmail+" date "+date);
        try {
            if(new Date().compareTo(date)>0) {
                throw new RideMustBeLaterThanTodayException(ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.ErrorRideMus
            }
            db.getTransaction().begin();

            Driver driver = db.find(Driver.class, driverEmail);
            if (driver.doesRideExists(from, to, date)) {
                db.getTransaction().commit();
                throw new RideAlreadyExistException(ResourceBundle.getBundle("Etiquetas").getString("DataAccess.RideAlreadyExist"));
            }
            Ride ride = driver.addRide(from, to, date, nPlaces, price, car);
            //next instruction can be obviated
            db.persist(driver);
            db.getTransaction().commit();

            return ride;
        } catch (NullPointerException e) {
            // TODO Auto-generated catch block
            db.getTransaction().commit();
            return null;
        }
```

  - **Errefaktorizatutako kodea:**

```
    public Ride createRide(String rideAndDriverInfo[], Date date, float rideValues[], Car car) throws  RideAlreadyExistException, RideM
        System.out.println(">> DataAccess: createRide=> from= "+rideAndDriverInfo[0]+" to= "+rideAndDriverInfo[1]+" driver="+rideAndDri
        try {
            if(new Date().compareTo(date)>0) {
                throw new RideMustBeLaterThanTodayException(ResourceBundle.getBundle("Etiquetas").getString("CreateRideGUI.ErrorRideMus
            }
            db.getTransaction().begin();

            Driver driver = db.find(Driver.class, rideAndDriverInfo[2]);
            if (driver.doesRideExists(rideAndDriverInfo[0], rideAndDriverInfo[1], date)) {
                db.getTransaction().commit();
                throw new RideAlreadyExistException(ResourceBundle.getBundle("Etiquetas").getString("DataAccess.RideAlreadyExist"));
            }
            Ride ride = driver.addRide(rideAndDriverInfo[0], rideAndDriverInfo[1], date, (int) rideValues[0], rideValues[1], car);
            //next instruction can be obviated
            db.persist(driver);
            db.getTransaction().commit();

            return ride;
        } catch (NullPointerException e) {
            // TODO Auto-generated catch block
            db.getTransaction().commit();
            return null;
        }
    }
```

```
    @WebMethod
    public Ride createRide( String from, String to, Date date, int nPlaces, float price, Car car, String driverEmail ) throws RideMustBe

        dbManager.open();
        String rideAndDriverInfo[] = {from, to, driverEmail};
        float rideValues[] = {(float) nPlaces, price};
        Ride ride=dbManager.createRide(rideAndDriverInfo, date, rideValues, car);
        dbManager.close();
        return ride;
    };
```

  - **Egindako errefaktorizazioaren deskribapena:**
    - Sartzen diren 3 string parametroak String[] zerrenda batean sartu eta float eta int parametroak float[] zerrenda batean sartu (int parametroa float batean casting eginez). Zerrendak sartu parametro bezala metodoan eta sarrerako parametro kopurua 4ra jaisten da. Guzti hau egiteko BLFacadeImplementation klasean createRide metodoari pare bat lerro gehitu dizkiot.

Aimar:

- ## Write short units of code:
    - ### Hasierako kodea:

```java
/**
 * This method retrieves from the database the dates a month for which there are events
 * @param from the origin location of a ride
 * @param to the destination location of a ride
 * @param date of the month for which days with rides want to be retrieved
 * @return collection of rides
 */
public List<Date> getThisMonthDatesWithRides(String from, String to, Date date) {
    System.out.println(">> DataAccess: getEventsMonth");
    List<Date> res = new ArrayList<>();

    Date firstDayMonthDate= UtilDate.firstDayMonth(date);
    Date lastDayMonthDate= UtilDate.lastDayMonth(date);


    TypedQuery<Date> query = db.createQuery("SELECT DISTINCT r.date FROM Ride r WHERE r.from=?1 AND r.to=?2 AND r.date BETWEEN ?3 and ?4",Date.class);

    query.setParameter(1, from);
    query.setParameter(2, to);
    query.setParameter(3, firstDayMonthDate);
    query.setParameter(4, lastDayMonthDate);
    List<Date> dates = query.getResultList();
     for (Date d:dates){
       res.add(d);
      }
    return res;
}
```

    - ### Errefaktorizatutako kodea:

```java
/**
 * This method retrieves from the database the dates a month for which there are events
 * @param from the origin location of a ride
 * @param to the destination location of a ride
 * @param date of the month for which days with rides want to be retrieved
 * @return collection of rides
 */
public List<Date> getThisMonthDatesWithRides(String from, String to, Date date) {
    System.out.println(">> DataAccess: getEventsMonth");
    List<Date> res = new ArrayList<>();
    Date firstDayMonthDate= UtilDate.firstDayMonth(date);
    Date lastDayMonthDate= UtilDate.lastDayMonth(date);
    TypedQuery<Date> query = db.createQuery("SELECT DISTINCT r.date FROM Ride r WHERE r.from=?1 AND r.to=?2 AND r.date BETWEEN ?3 and ?4",Date.class);
    query.setParameter(1, from);
    query.setParameter(2, to);
    query.setParameter(3, firstDayMonthDate);
    query.setParameter(4, lastDayMonthDate);
    List<Date> dates = query.getResultList();
     for (Date d:dates)
       res.add(d);
    return res;
}
```

    - ### Egindako errefaktorizazioaren deskribapena:
        - Metodoaren lerro kopurua maximo 15 izateko tarteko zuriuneak ezabatu ditut eta kode lerro bakarreko for-ari kortxeteak kendu dizkiot.

- **Write simple units of code:**
    - **Hasierako kodea:**

```
public boolean deleteUser(String uGmail) {
    boolean res = false;
    db.getTransaction().begin();
    User u = db.find(User.class, uGmail);
    db.getTransaction().commit();
    if (u.getType().equals("Driver")) {
        Driver d = (Driver) u;
        if (d.getRides() != null) {
            for (Ride r : d.getRides()) {
                deleteRide(r.getRideNumber(), d.getEmail());
            }
        }
        for (Car ca : d.getCars()) {
            deleteCar(ca.getlicensePlate());
        }
        for (Mugimenduak mu : d.getMugimenduak()) {
            deleteMugimendua(mu.getMugiZenb());
        }
        db.getTransaction().begin();
        d = db.find(Driver.class, u.getEmail());
        d.getRides().clear();
        d.getCars().clear();
        d.getMugimenduak().clear();
        db.remove(d);
        db.getTransaction().commit();
        res = true;
    }else {
        Passenger p = (Passenger) u;
        for (RideBooked rb : p.getBookedrides()) {
            deleteRideBooked(rb.getBookNumber());
        }
        for (Mugimenduak mu : p.getMugimenduak()) {
            deleteMugimendua(mu.getMugiZenb());
        }
        for (Alerta al : p.getAlerts()) {
            deleteAlerta(al.getZenb());
        }
        db.getTransaction().begin();
        p = db.find(Passenger.class, u.getEmail());
        p.getBookedrides().clear();
        p.getMugimenduak().clear();
        p.getAlerts().clear();
        db.remove(p);
        db.getTransaction().commit();
        res = true;
    }
    return res;
}
```

- **Errefaktorizatutako kodea:**

```
public boolean deleteUser(String uGmail) {
    boolean res = false;
    db.getTransaction().begin();
    User u = db.find(User.class, uGmail);
    db.getTransaction().commit();
    if (u.getType().equals("Driver")) {
        Driver d = (Driver) u;
        deleteDriverUser(u, d);
        res = true;
    }else {
        Passenger p = (Passenger) u;
        deletePassengerUser(u, p);
        res = true;
    }
    return res;
}
```

```
private void deletePassengerUser(User u, Passenger p) {
    for (RideBooked rb : p.getBookedrides()) {
        deleteRideBooked(rb.getBookNumber());
    }
    for (Mugimenduak mu : p.getMugimenduak()) {
        deleteMugimendua(mu.getMugiZenb());
    }
    for (Alerta al : p.getAlerts()) {
        deleteAlerta(al.getZenb());
    }
    db.getTransaction().begin();
    p = db.find(Passenger.class, u.getEmail());
    p.getBookedrides().clear();
    p.getMugimenduak().clear();
    p.getAlerts().clear();
    db.remove(p);
    db.getTransaction().commit();
}
```

```
private void deleteDriverUser(User u, Driver d) {
    if (d.getRides() != null) {
        for (Ride r : d.getRides()) {
            deleteRide(r.getRideNumber(), d.getEmail());
        }
    }
    for (Car ca : d.getCars()) {
        deleteCar(ca.getlicensePlate());
    }
    for (Mugimenduak mu : d.getMugimenduak()) {
        deleteMugimendua(mu.getMugiZenb());
    }
    db.getTransaction().begin();
    d = db.find(Driver.class, u.getEmail());
    d.getRides().clear();
    d.getCars().clear();
    d.getMugimenduak().clear();
    db.remove(d);
    db.getTransaction().commit();
}
```

```
private void deleteDriverRides(Driver d) {
    for (Ride r : d.getRides()) {
        deleteRide(r.getRideNumber(), d.getEmail());
    }
}
```

- **Egindako errefaktorizazioaren deskribapena:**
  - deleteUser metodoak konplexutasun handia duenez sinplifikatu behar izan dut beste hiru metodo berri sortuz lana banatzeko. Batean gidariak ezabatzen dira, bigarrenean bidaiariak eta . Honela, metodoaren konplexutasuna 9tik 2ra, 3ra, 3ra eta 1era jaisten da.

- **<u>Duplicate code</u>** (Ez dut DataAccess-en honelakorik aurkitu, horren ordez FindRidesGUI-tik hartuko dut):
  - **Hasierako kodea:**

```
private JComboBox<String> jComboBoxOrigin = new JComboBox<String>();

private JComboBox<String> jComboBoxDestination = new JComboBox<String>();
```

- **Errefaktorizatutako kodea:**

```
private JComboBox<String> jComboBoxOrigin = createJComboBox();

private JComboBox<String> jComboBoxDestination = createJComboBox();

private JComboBox<String> createJComboBox() {
    return new JComboBox<>();
}
```

- **Egindako errefaktorizazioaren deskribapena:**
  - Zuzenean JComboBox objektuak sortu beharrean eta horren ondorioz kode errepikapenak izan metodo bat sortu dut objektu hauek sortzeko.

- **<u>Keep unit interfaces small:</u>**
  - **Hasierako kodea:**

```
public boolean addCar(String license, String brand, String color, int seats,  String dGmail) {
    db.getTransaction().begin();
    Driver dr = null;
    dr = db.find(Driver.class, dGmail);

    Car ca = null;
    ca = dr.addCar(license, brand, color,seats, dr);
    db.persist(dr);
    db.getTransaction().commit();
    if(ca!=null) {
        return true;
    }else {
        return false;
    }
}

@WebMethod
public boolean addCar(String license, String brand, String color, int seats,  String dGmail) {
    dbManager.open();
     boolean res = dbManager.addCar(license, brand, color, seats, dGmail);
     dbManager.close();
     return res;
}
```

- **Errefaktorizatutako kodea:**

```
public boolean addCar(String[] carAttributes, int seats,  String dGmail) {
    db.getTransaction().begin();
    Driver dr = null;
    dr = db.find(Driver.class, dGmail);

    Car ca = null;
    ca = dr.addCar(carAttributes[0], carAttributes[1], carAttributes[2],seats, dr);
    db.persist(dr);
    db.getTransaction().commit();
    if(ca!=null) {
        return true;
    }else {
        return false;
    }
}

@WebMethod
public boolean addCar(String license, String brand, String color, int seats,  String dGmail) {
    dbManager.open();
    String carAttributes[] = {license, brand, color};
     boolean res = dbManager.addCar(carAttributes, seats, dGmail);
     dbManager.close();
     return res;
}
```

- **Egindako errefaktorizazioaren deskribapena:**
  - Jasotako elementu kopurua murrizteko String zerrenda batean gorde ditut kotxe batek behar dituen elementu batzuk. Honetarako, BLFacadeImplementation klasean dagoen addCar aldatu behar izan dut.