eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

informatika fakultatea
facultad de informática

PROIEKTUAREN MEMORIA

Urtzi Diaz, Itziar Altuna eta Iñigo Sanz

2015/05/12

AURKIBIDEA

SARRERA

Hurrengo dokumentuan,Informatika ingenieritzako graduko 2. mailako Software Ingenieritza
Ikastarorako Itziar Altuna, Urtzi Diaz eta Iñigo Sanz-en artean egindako proiektuaren nondik norakoak azalduko dira.

Esan bezala, proiektu hau 3 pertsonaz osatutako taldeetan egin da. Lana era egokian egiteko
SCRUM izena duen garapen prozesua erabili dugu. Prozesu honen ezaugarri nagusia, iteratiboa eta inkrementala dela esan dezakegu. Iteratiboa dela esaten dugunean, bere izenak dioen moduan, lana iterazio ezberdinetan egiten dela esan nahi dugu. Praktika hau 3 iterazio ezberdinetan banatua izan da(0. kontatu gabe).

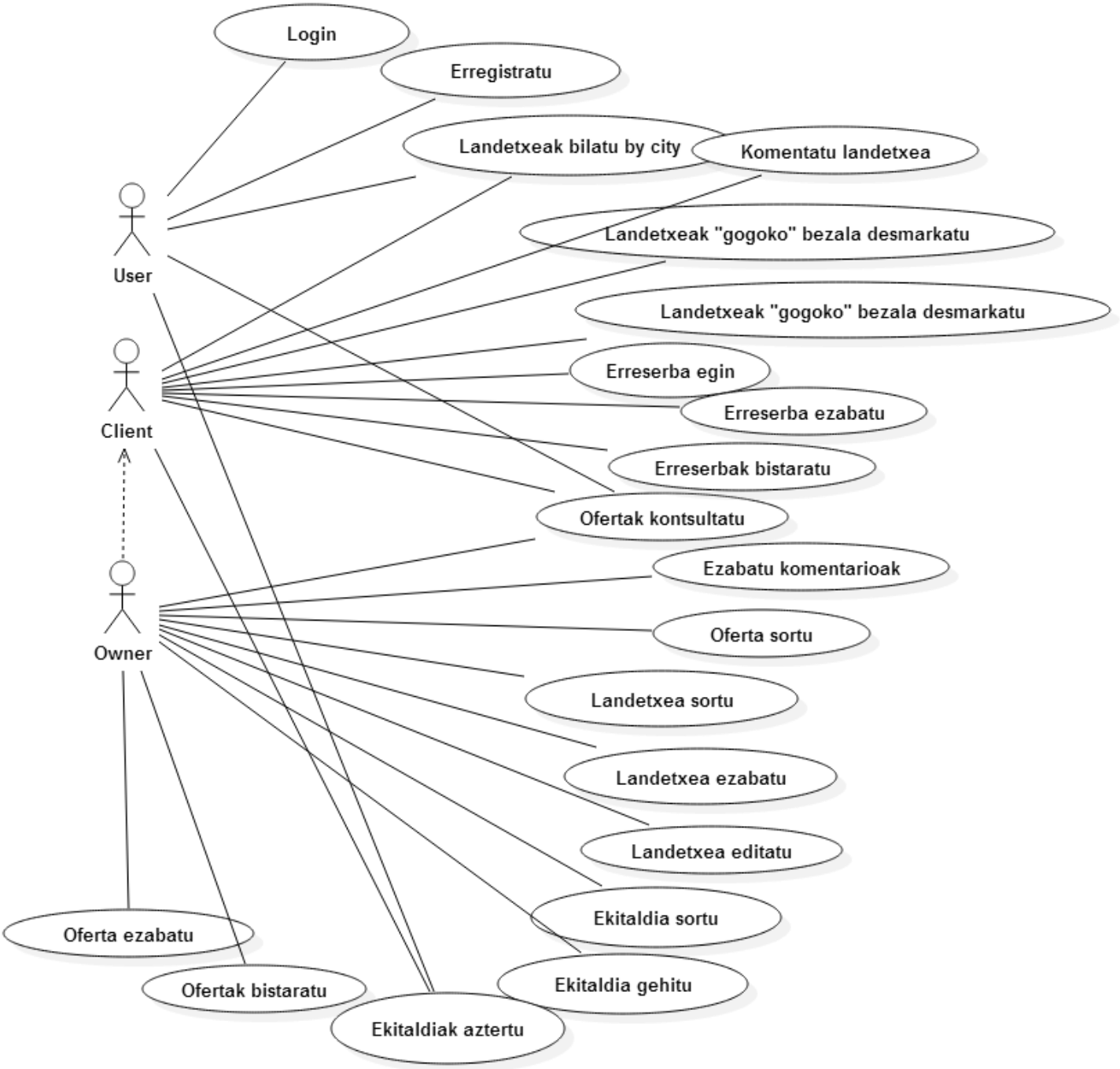
Iterazio bakoitzaren hasieran taldekide guztien artean iterazio horretan garatuko genituen erabilpen kasuak aukeratu ditugu. SCRUM metodologiaren beste ezaugarri bat, taldekideen rol desberdinak izatea izango litzateke gure kasuan iterazio bakoitzean zuzendaria aldatu dugu.
Iterazio bat amaitzerakoan anti-doping motako azterketa batzuk egin ditugu. Horrez gain iterazio bakoitzaren amaieran egindako aurrerapenen aurkezpen publikoa egin dugu.

Erabilitako metodoa azaldu ondoren, lanaren laburpen txiki bat egingo dugu. Proiektuaren
helburu nagusia, landetxeak alokatzeko aplikazio bat garatzea izan da. Lehen prototipoa ikusi ondoren, 0 iterazioa, interfaze grafikoaren hobekuntza batean pentsatu genuen horrez gain, gehitzea interesgarri litzatekeen funtzionalitateen bila hasi ginen internet-eko hainbat orrialdeetan. Lan hori egin ondoren hainbat funtzionalitate berri jartzea erabaki genuen, hala nola:
-Landetxeak komentatu
-Landetxeak baloratu
-Landetxeen alokairuarekin jarduera ezberdinak egiteko aukera izatea hala nola, surf.

Egindako lanaren egunerokoa modu ulergarri eta sinplean eramateko orduan "github" Tresna erabiltzen saiatu ginen baina azkenean Dropbox erabili dugu gehienbat.

ESKAKIZUN BILKETA

Erabilpen kasuak



1. Iterazioan:

- Login egin.
- Landetxeak sortu.
- Erregistratu.

2. Iterazioan:

- Landetxeak bilatu (hiriaren arabera).
- Landetxeak gogoko bezela markatu.
- Landetxeak editatu.
- Landetxeak ezabatu.
- Ofertak sortu
- Ofertak kontsultatu

3. Iterazioan:

- Komentatu landetxeak. (*domeinu berria)
- Ezabatu komentarioak.
- Ekitaldiak sortu. (*domeinu berria)
- Ekitaldiak gehitu landetxeei.
- Ekitaldiak aztertu (ikusi).
- Erreserba egin
- Egindako ofertak ikusi
- Oferta ezabatu
- Client ezabatu
- Owner ezabatu

Proiektuaren diseinua:

1.- ITERAZIOA:

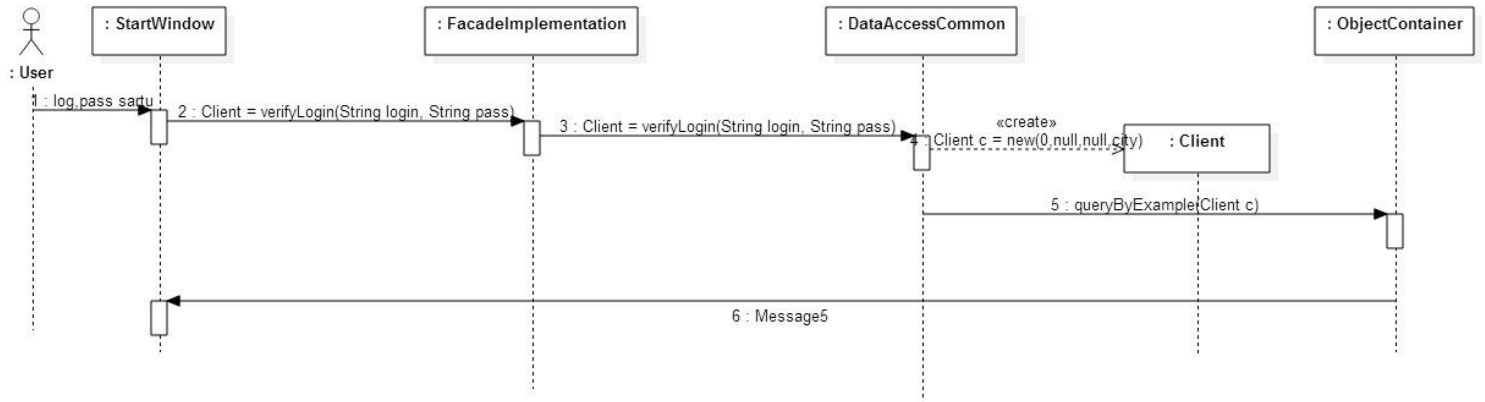
LOGIN:

- *GERTAERA FLUXUA:*

#Basic flow
1. Hasierako panelean Login egiteko laukitxoa agertuko da. Edonork logeatzeko aukera izango du.
2. Erabiltzaileak bere login (erabiltzaile izena) eta bere password-a sartuko du sisteman.
3. Botoia sakatu ondoren, sistemak erabiltzaile horren panelera eramango dio.

##Alternative flow
1. **Login** egitean, erabiltzailea edo pasahitza datu basean daukagunarekin ez badatoz bat **Sistemak** jakinaraziko digu mezu baten bitartez.

- *SEKUENTZIA DIAGRAMA:*



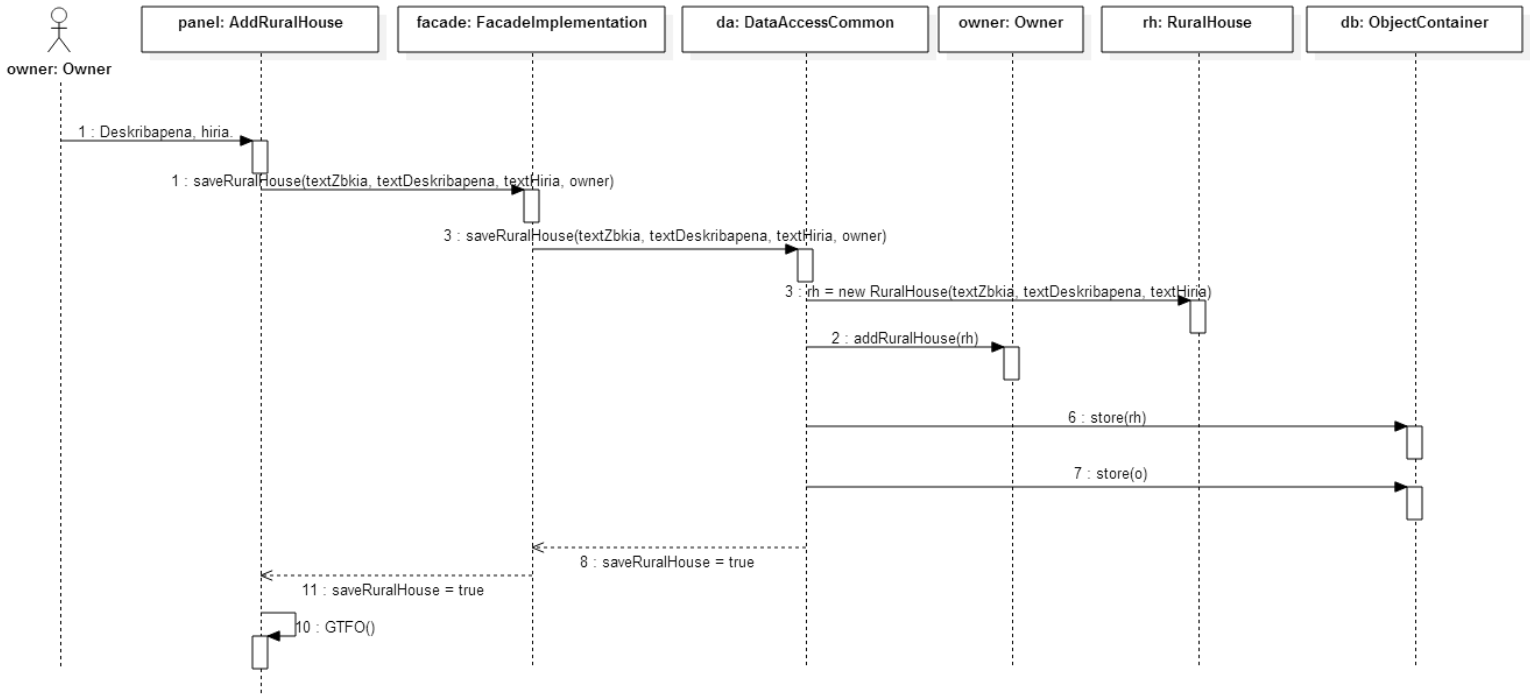
LANDETXEA SORTU:

❖ *GERTAERA FLUXUA:*

##Basic flow
1. **Owner** jabeak "Landetxea gehitu" botoia sakatu beharko du behin logeatuta.
2. **Sistemak** datuak landetxearen datuak betetzeko eremuak pantailaratuko ditu (zenbakia, deskribapena eta hiria).
3. **Owner** eremuak bete eta "SORTU" botoia sakatuko du.
4. **Sistemak** landetxea gordeko du datubasean logeatutako jabearekin (Owner) erlazionatuta.

##Alternative flow
1. Eremu guztiak ez badira bete, **Sistemak** jakinaraziko dio erabiltzaileari.
2. Zenbakia dagoeneko hartuta badago, **Sistemak** arazoa azalduko dio erabiltzaileari.

❖ *SEKUENTZIA DIAGRAMA:*



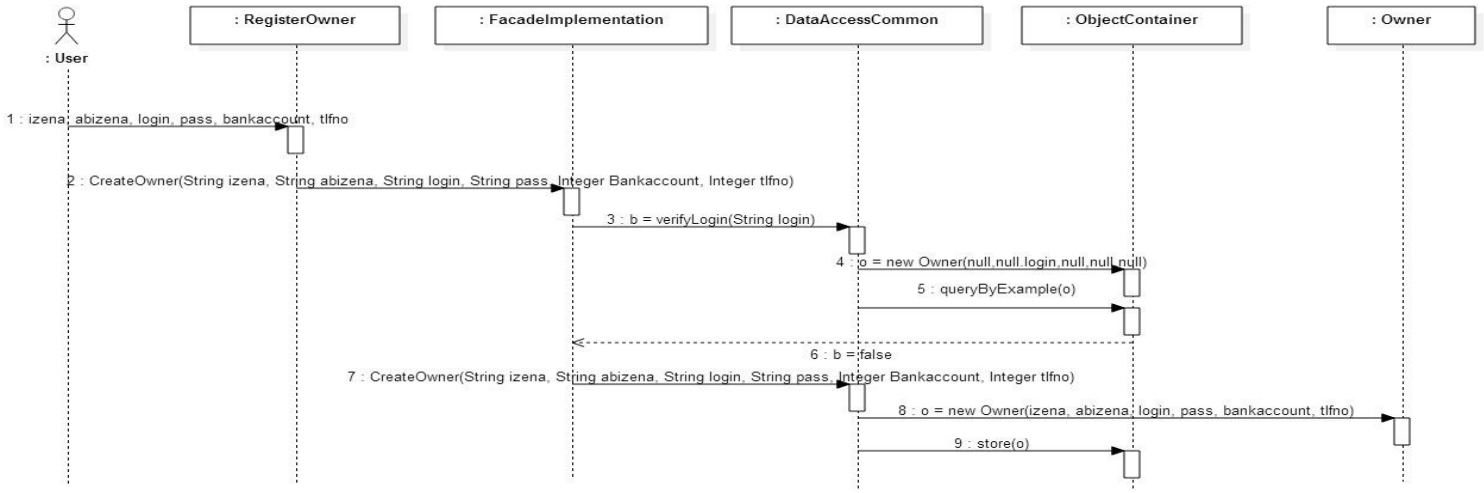
REGISTER:

❖ *GERTAERA FLUXUA:*

#Basic flow
1. User ez erregistratuak erregistratzeko aukera izango dute.
2. 2 aukera izango ditugu: Client bezala erregistratzea ala Owner bezala .
3. Owner bezala erregistratzeko Izena, Abizena, Nickname-a, Password, Telefono zenbakia etaKontu korronteko zenbakia eremuak bete beharko ditugu.
4. Client bezala, ordez, Telefono zenbakia eta Kontu korronteko zenbakia ez dira beharrezkoak.

##Alternative flow
1. **Sign in** egitean, Nickname-a hartuta baldin badago **Sistemak** jakinaraziko digu mezu baten bitartez.
2. Eremu guztiak bete behar dira.

❖ *SEKUENTZIA DIAGRAMA:*



2.- ITERAZIOA:

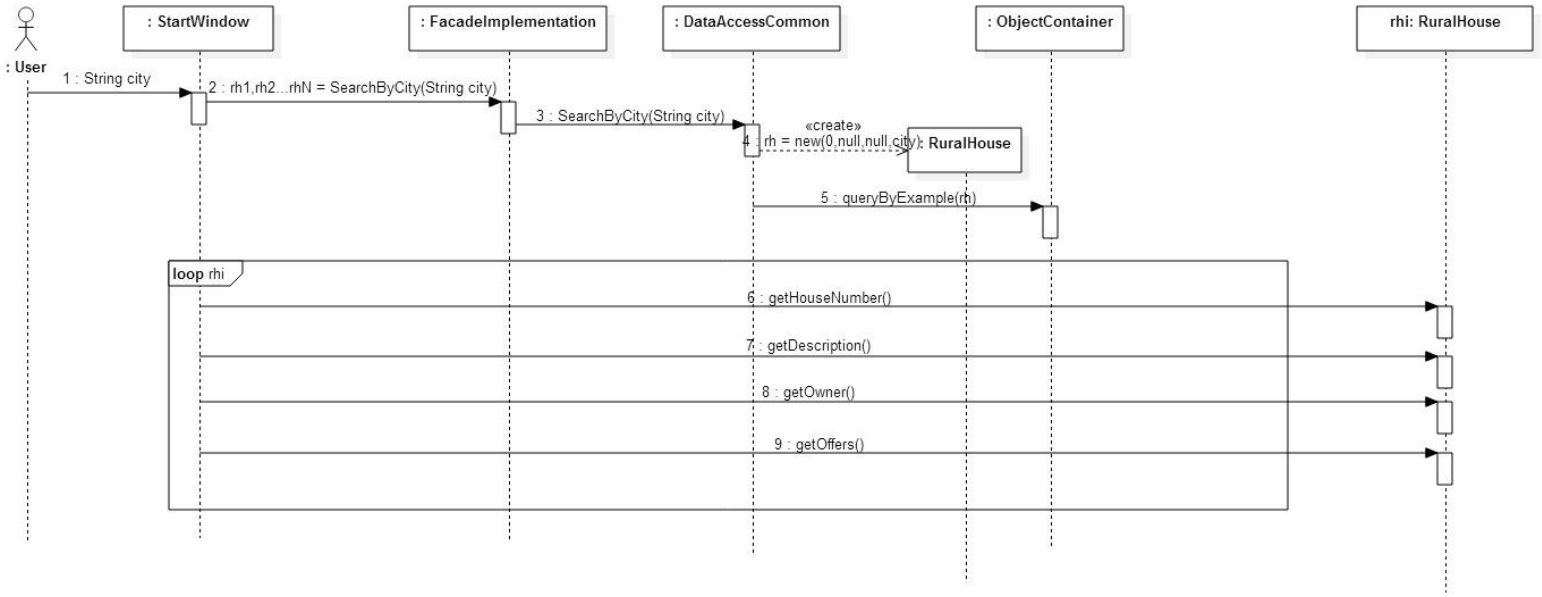
LANDETXEAK BILATU (by City):

- *GERTAERA FLUXUA:*

##Basic flow
1. Hasierako paneletik bilatzaile bat agertuko zaigu. Edonork landetxeak bilatzeko aukera izateko.
2. Erabiltzaileak **Landetxeak** zein hiritan bilatu nahi dituen idatziko du.
3. Botoia sakatu ondoren, sistemak hiri horretan dauden **landetxeak** erakutsiko dizkio.

##Alternative flow
1. **Landetxea** bilatzean, sartu dugun hirian **landetxerik** ez badaude **Sistemak** jakinaraziko digu.

- *SEKUENTZIA DIAGRAMA:*



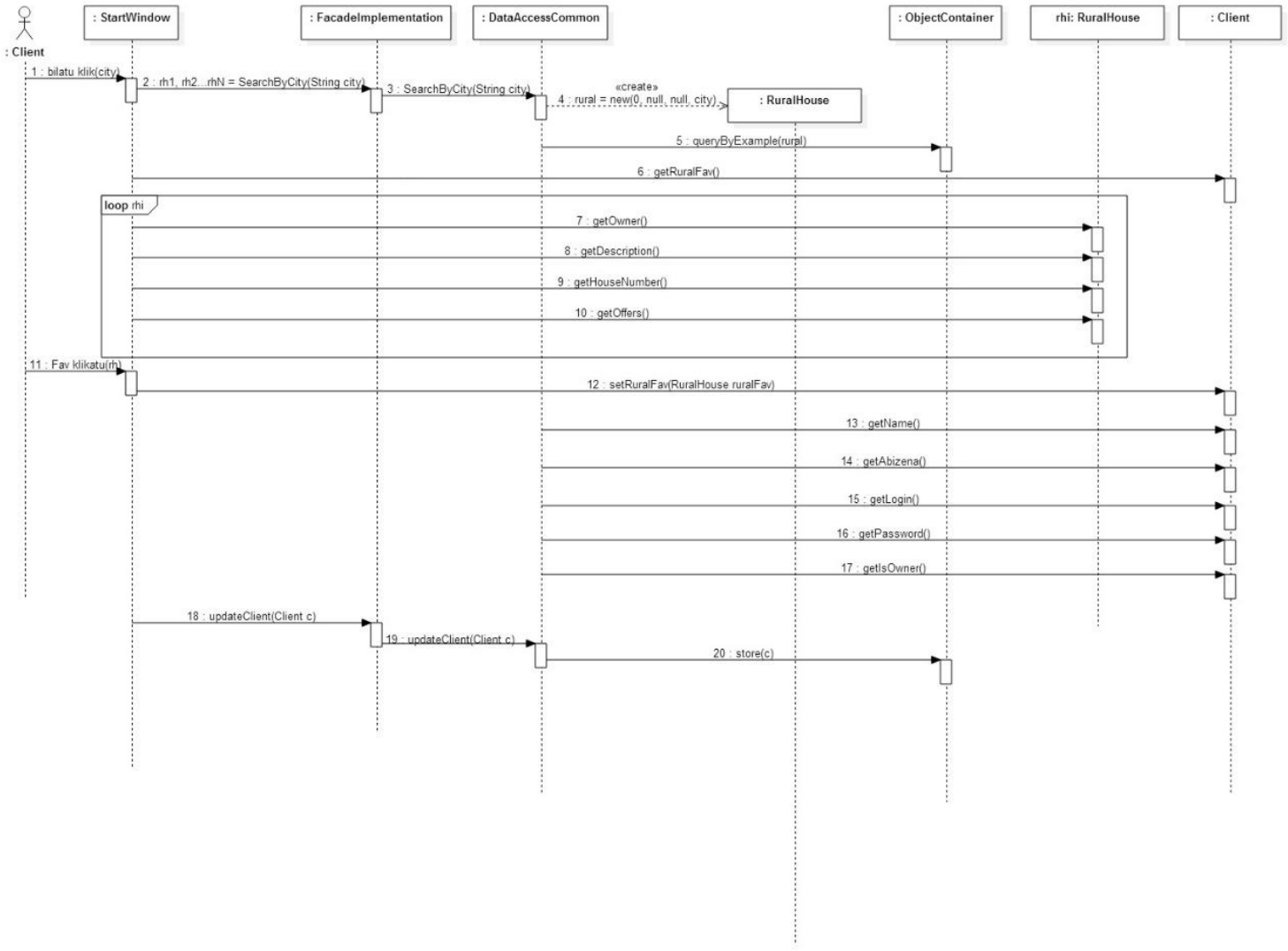
LANDETXEAK GOGOKO BEZALA MARKATU:

- *GERTAERA FLUXUA:*

##Basic flow
1. **Erabiltzaile erregistratuak** soilik landetxeak gogoko bezala markatzeko aukera izango du.
2. **Erabiltzaileak**-ak landetxe bat hautatuko du.
3. **Landetxearen** favorite botoia sakatu ondoren, Landetxea hori client-aren Favorite List-ean gordeta geratuko da.

##Alternative flow
1. **Landetxe** hori jadanik client-aren Favorite List-ean gordeta baldin badago mezu bat agertuko da errepikapenaz ohartaraziz.

- *SEKUENTZIA DIAGRAMA:*



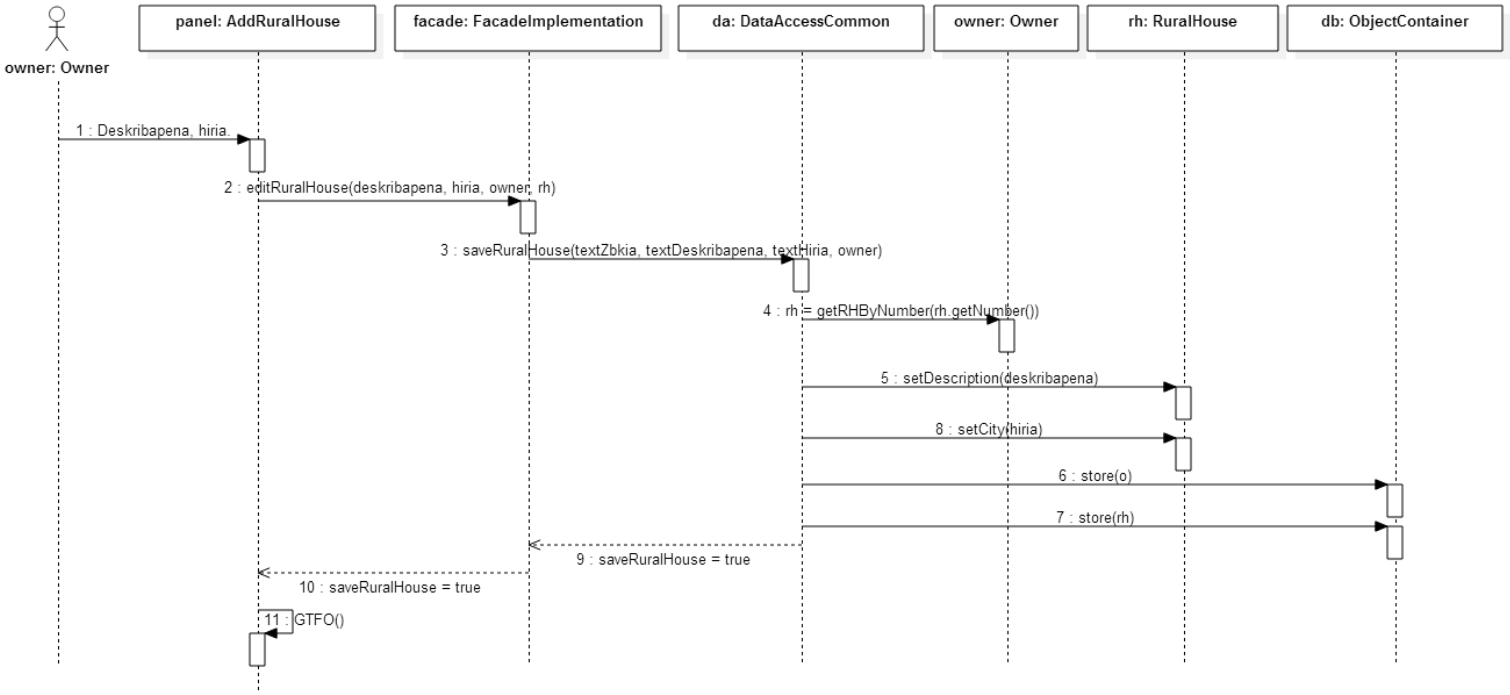
LANDETXEA EDITATU:

- *GERTAERA FLUXUA:*

##Basic flow
1. **Owner**jabeak landetxea aukeratzen du eta "editatu" botoia sakatu beharko du landetxearen atributuak editatzeko.
2. **Sistemak** datuak eguneratzeko eremuak pantailaratuko ditu.
3. **Owner** eremuak beteko ditu eta "gorde" botoia sakatu beharko du.
4. **Sistemak** landetxea eguneratuko du datubasean datu berriekin.

##Alternative flow
1. Datu bat ez bada egokia, **Sistemak** jakinaraziko dio erabiltzaileari non dagoen arazoa.

- *SEKUENTZIA DIAGRAMA:*

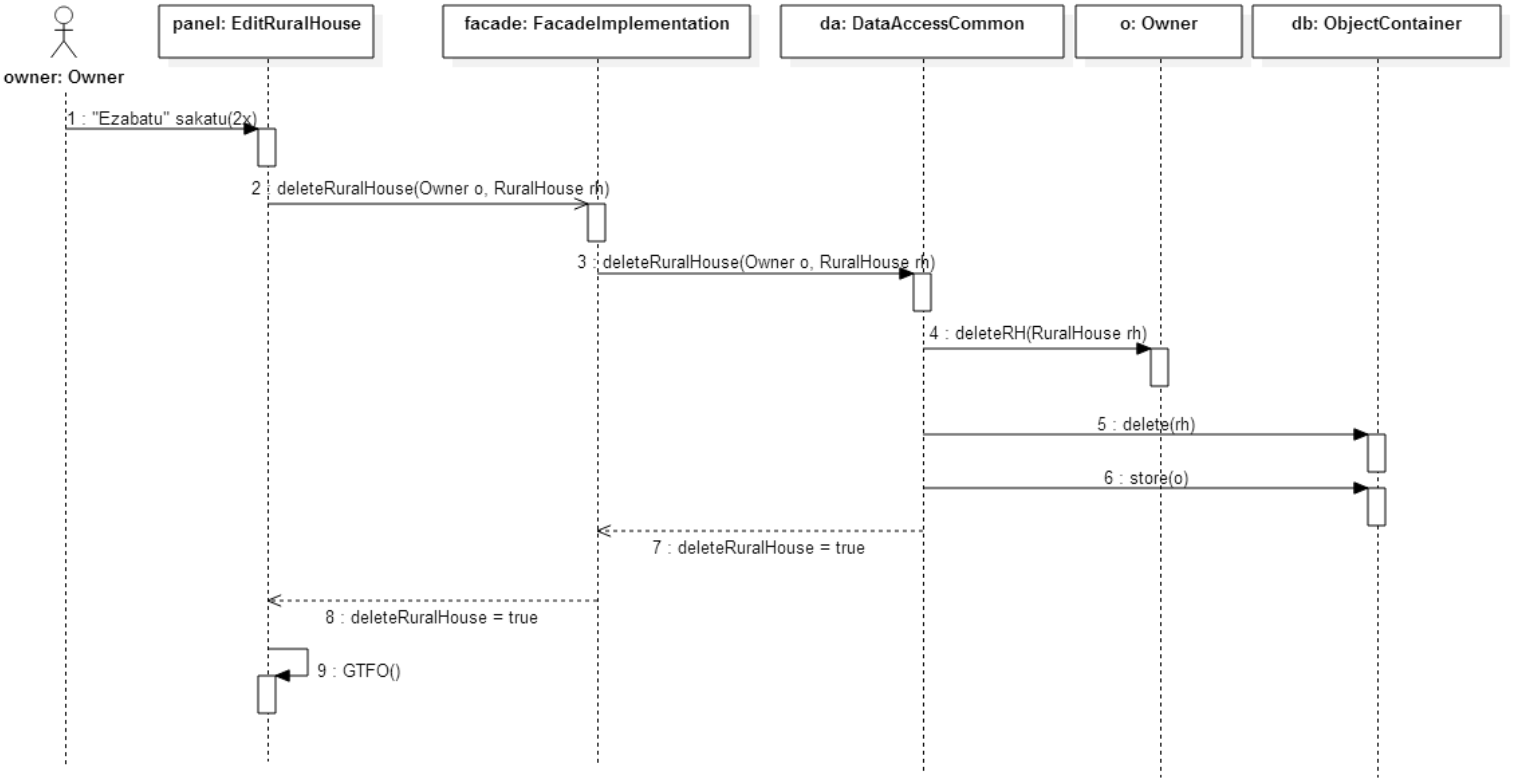


LANDETXEA EZABATU:

❖ *GERTAERA FLUXUA:*

##Basic flow
1. **Owner** landetxea aukeratuko du, eta "ezabatu" botoia sakatu beharko du landetxea ezabatzeko datu baseetik.
2. **Sistemak** landetxea datubasetik ezabatuko du.

❖ *SEKUENTZIA DIAGRAMA:*



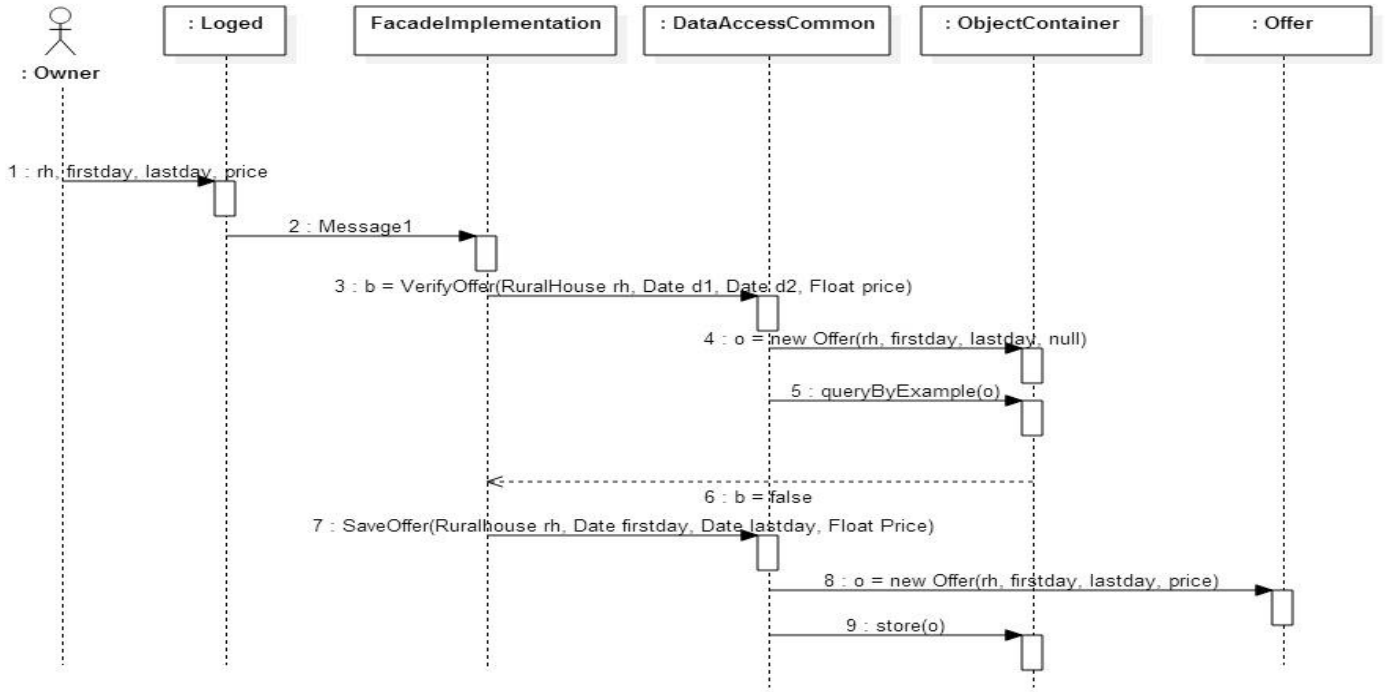
OFERTA SORTU:

- *GERTAERA FLUXUA:*

##Basic flow
1. **Owner** soilik ofertak sortzeko aukera izango du.
2. Oferta berri bat sortzeko **Owner**-ak bere profilera sartu behar da eta **Loged** panelean**Oferta gehitu** aukera aurkituko du.
3. Oferta eratzeko etxea, noiztik eta noiz arte egongo den libre etxe konkretu hori eta prezioa espezifikatu behar da.
4. Botoia sakatu ondoren, Oferta datu basean gordeta geratuko da.

##Alternative flow
1. Oferta eratzeko momentuan, datu bat gaizki sartzen bada **Sistemak** jakinaraziko dio Owner-ari nun dagoen arazoa.
2. Botoia sakatzerakoan, oferta hori jadanik datu basean gordeta baldin badago mezu bat agertuko da errepikapenaz ohartaraziz.

- *SEKUENTZIA DIAGRAMA:*



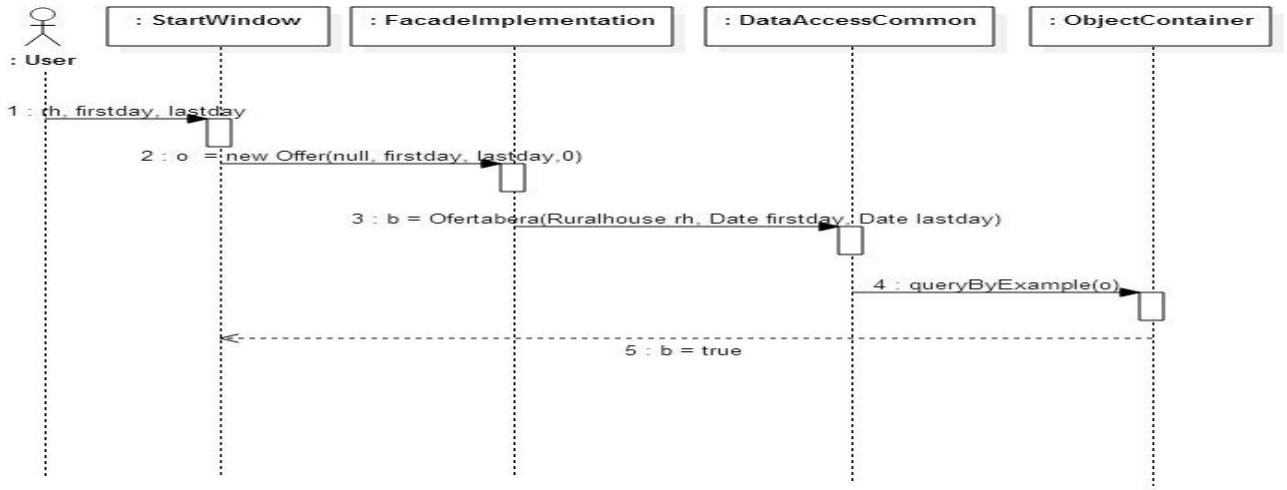
OFERTAK KONTSULTATU:

❖ *GERTAERA FLUXUA:*

##Basic flow
1. **Owner** eta **Client** ofertak kontsultatzeko aukera izango dute.
2. Ofertak kontsultatzeko etxea, data tartea eta zenbat egun pasatzeko espezifikatu behar da .
3. ComboBox batean agertuko dira **Available** dauden etxeak baldin badaude, bestela mezu bat agertuko da **Not available offers** esanez.

##Alternative flow
1. Ofertak bilatzeko momentuan, datu bat gaizki sartzen bada **Sistemak** jakinaraziko dio erabiltzaileari nun dagoen arazoa.

❖ *SEKUENTZIA DIAGRAMA:*



3.- ITERAZIOA:

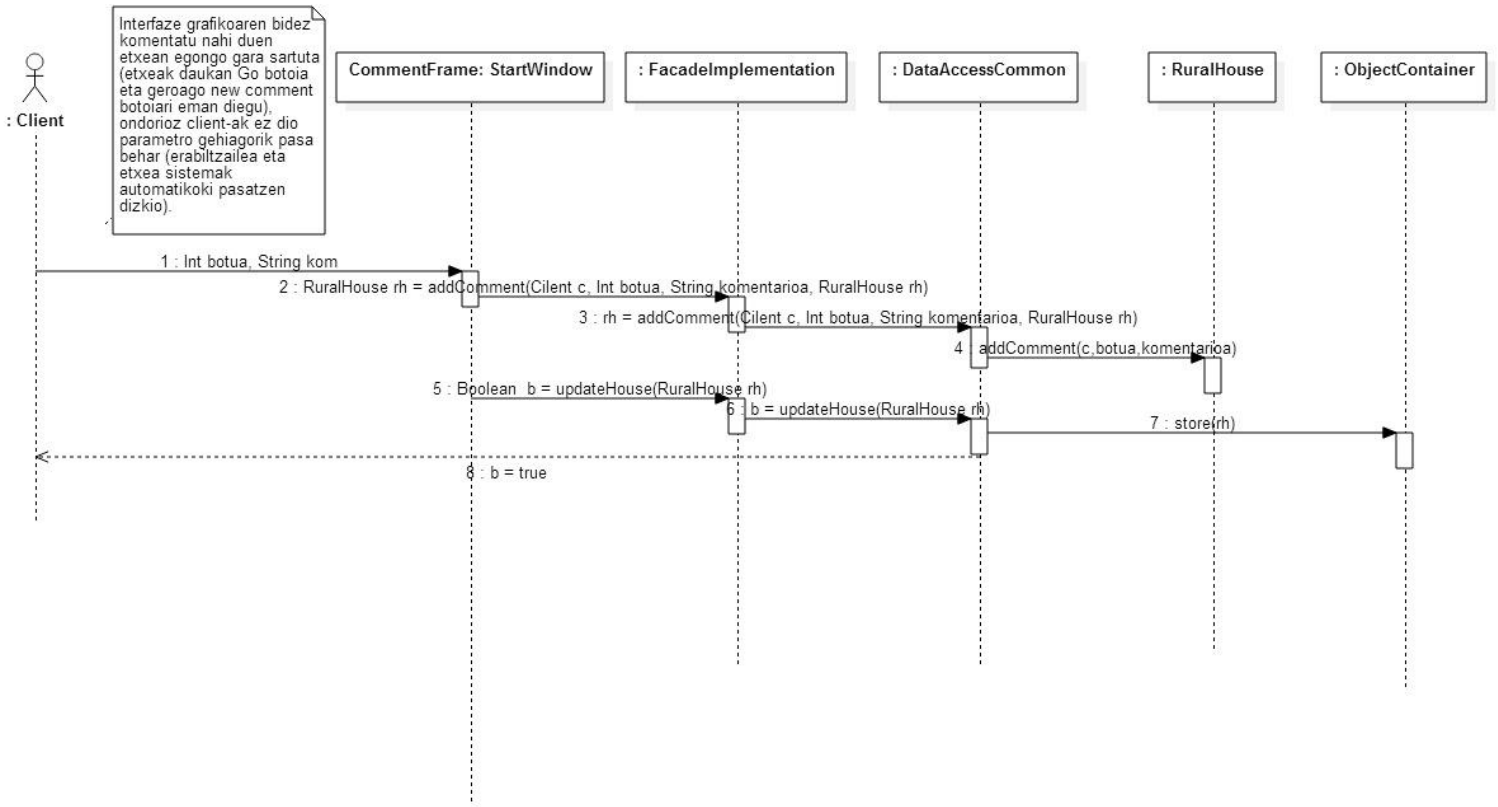
KOMENTATU LANDETXEA:

- *GERTAERA FLUXUA:*

##Basic flow

1. **Client-ak** soilik landetxeak komentatzeko aukera izango du.
2. **Erabiltzaileak**-ak landetxe bat hautatuko du.
3. Erabiltzaileak **Landetxearen** komentarioaren hutsunean idatziko du.
4. Komentarioa idatzi ondoren komentatu botoia sakatu..
5. Botoia sakatzean sistemak komentarioa landetxean gordeko du.

- *SEKUENTZIA DIAGRAMA:*



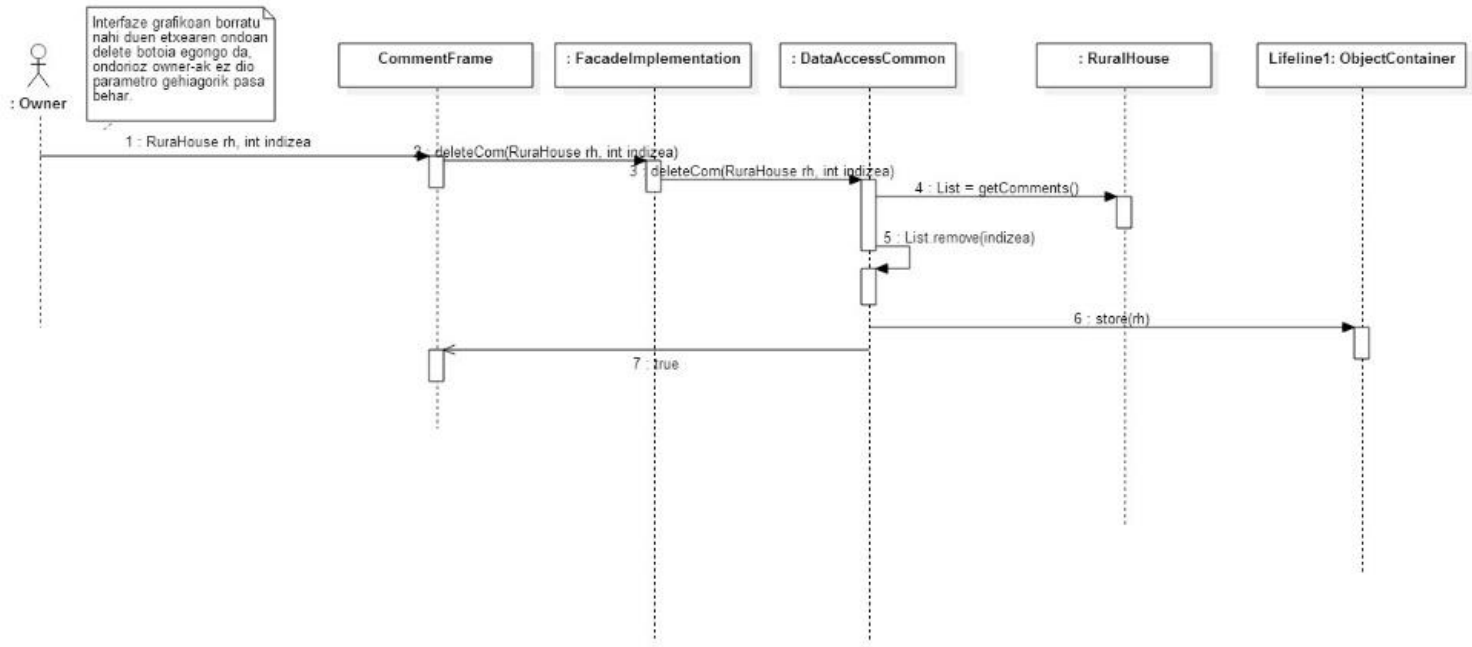
EZABATU KOMENTARIOAK:

- *GERTAERA FLUXUA:*

##Basic flow

1. **Owner-ak** soilik landetxearen komentarioak ezabatzeko aukera izango du.
2. **Owner**-ak berea den landetxe bat hautatuko du.
3. Owner-ak **Landetxearen** komentarioen hartean zein ezabatu nahi duen aukeratuko du.
4. Aukeratu ondoren ezabatu botoiari emango dio.
5. Botoia sakatzean sistemak landetxearen komentarioa ezabatuko du.

- *SEKUENTZIA DIAGRAMA:*



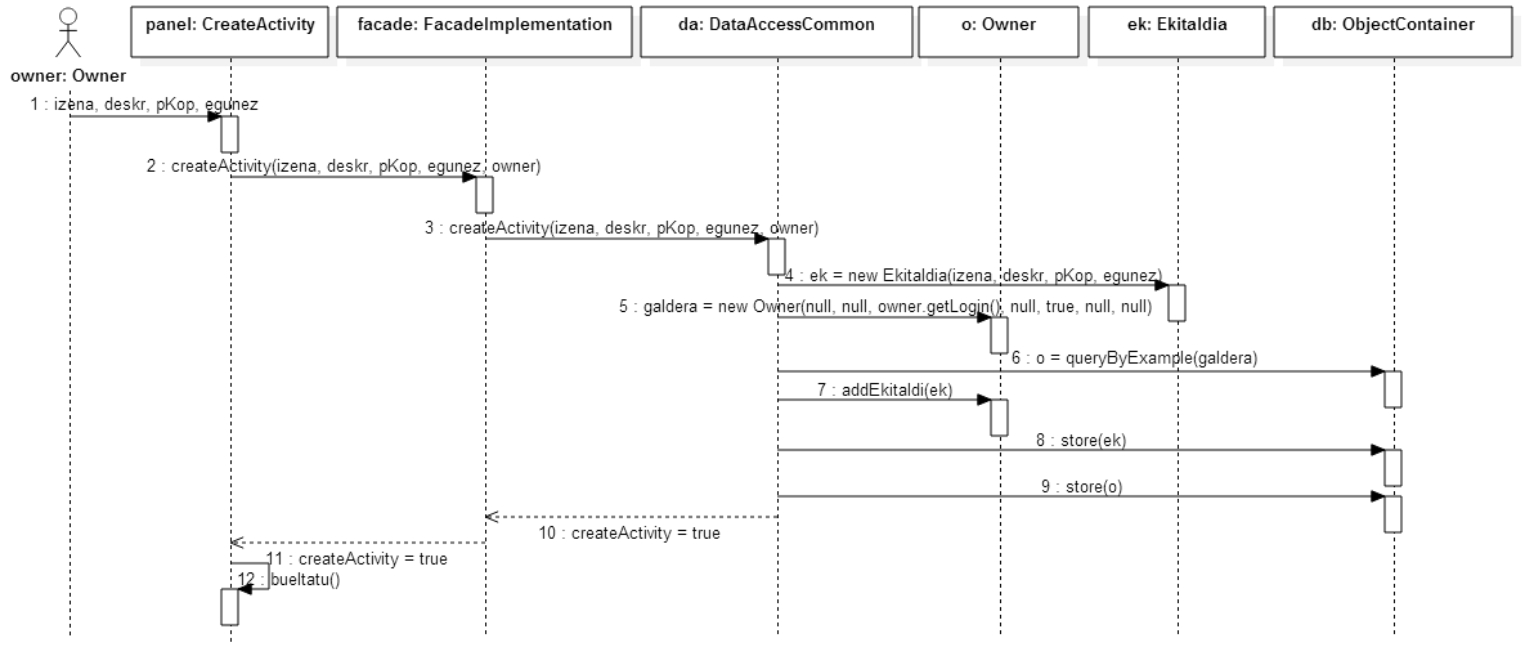
EKITALDIA SORTU:

❖ *GERTAERA FLUXUA:*

##Basic flow
1. **Owner** soilik izango du ekitaldiak sortzeko aukera.
2. Ekitaldi berri bat sortzeko **Owner**-ak bere profilera sartu behar du eta **Loged** panelean **Ekitaldia sortu** aukera izango du.
3. Ekitaldia eratzeko honen izena, deskribapena, partaide kopuru gehienezkoa eta egunez edo gauez den aukeratu beharko du.
4. Botoia sakatu ondoren, Ekitaldia datu basean gordeko da.

##Alternative flow
1. Ekitaldia eratzeko momentuan, datu bat gaizki sartzen bada edo denak ez badira betetzen **Sistemak** jakinaraziko dio Owner-ari non dagoen arazoa.
2. Botoia sakatzerakoan, izen berarekin beste Ekitaldi bat badago **Sistemak** hau adieraziko du eta ez du berriro gordeko.

❖ *SEKUENTZIA DIAGRAMA:*

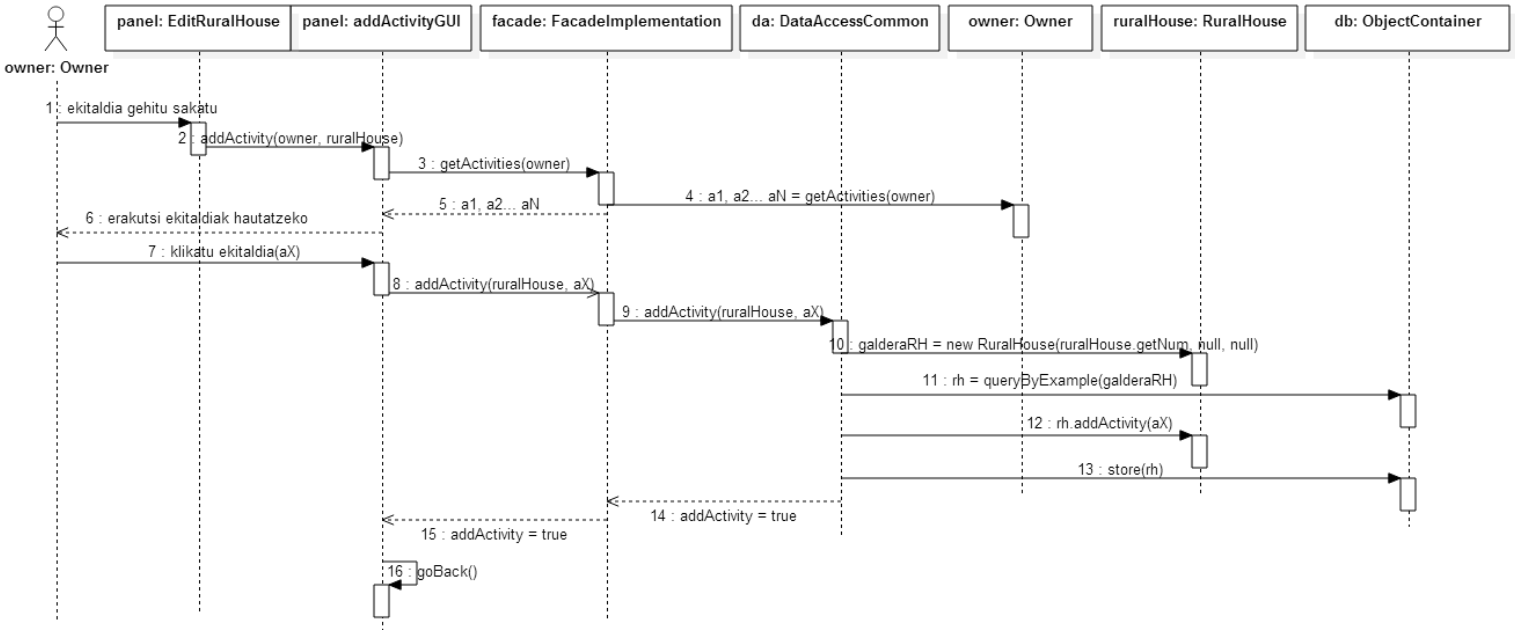


EKITALDIA GEHITU (landetxeari):

❖ *GERTAERA FLUXUA:*

##Basic flow
1. **Owner** soilik izango du ekitaldiak gehitzeko aukera.
2. Landetxe bati ekitaldi bat gehitzeko **Owner**-ak bere Landetxe bat editatu beharko du. Bertan, **Ekitaldia gehitu** aukera sakatuko du.
3. Ekitaldia aukeratzeko zerrenda bat izango du. Nahi dena markatu eta "Gehitu" botoia sakatu behar da.
4. Gehitu botoia sakatzerakoan, Landetxeari Ekitaldia gehituko zaio.

❖ *SEKUENTZIA DIAGRAMA:*

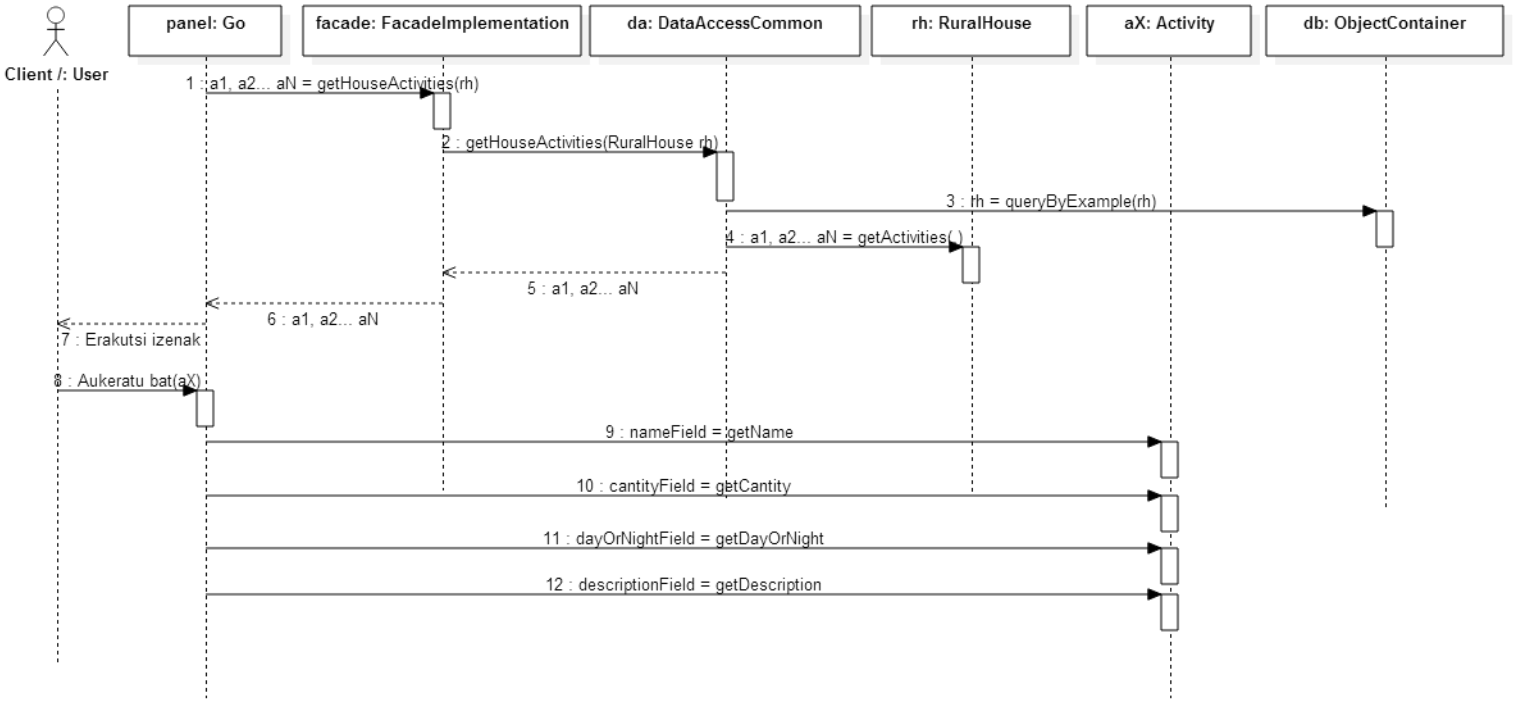


EKITALDIAK AZTERTU:

❖ *GERTAERA FLUXUA:*

##Basic flow
1. **Client** eta bisitariek bakarrik izango dute ekitaldi hauek aztertzeko aukera.
2. Landetxe baten informazioa bistaratzeko panelean (GO) sartu.
3. Sistemak landetxe horren ekitaldien zerrenda lortuko du eta izenak lista destolesgarri batean ipiniko ditu.
4. Erabiltzaileak nahi duena aukeratu eta "Ekitaldia aztertu" sakatuko du.
5. Sistemak ekitaldiaren informazio gehigarria erakutsiko du.

❖ *<SEKUENTZIA DIAGRAMA:*



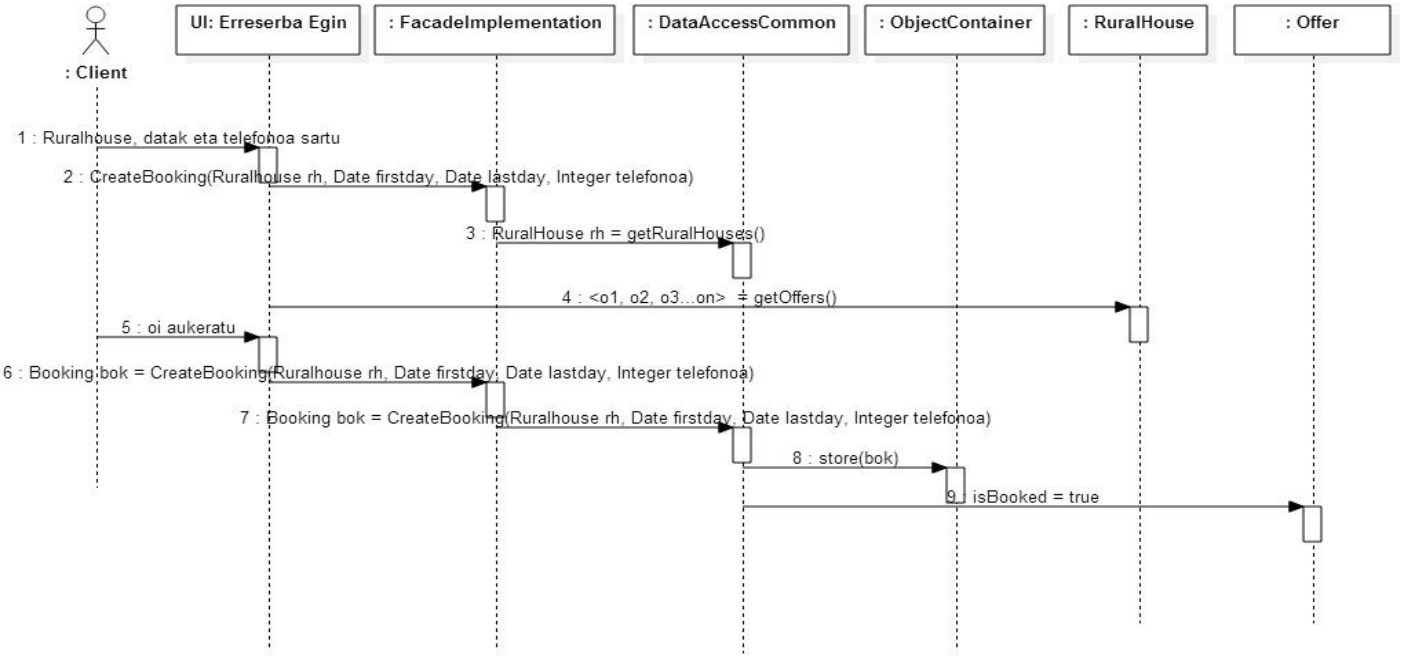
ERRESERBA EGIN:

❖ *GERTAERA FLUXUA:*

#Basic flow
1. Client erregistratu batek, etxe bat aukeratuko du.
2. Hainbat eremu bete beharko ditu hala nola: Sarrerako eguna, azkeneko eguna eta telefonoa.
3. Ofertaren bat baldin badago **Sistemak** jakinaraziko digu mezu baten bidez.
4. Ofertarik ez badaude **Sistemak** eskatuko digu beste dataren bat sartzea.

##Alternative flow
1. Eremu guztiak bete behar dira.

❖ *SEKUENTZIA DIAGRAMA:*



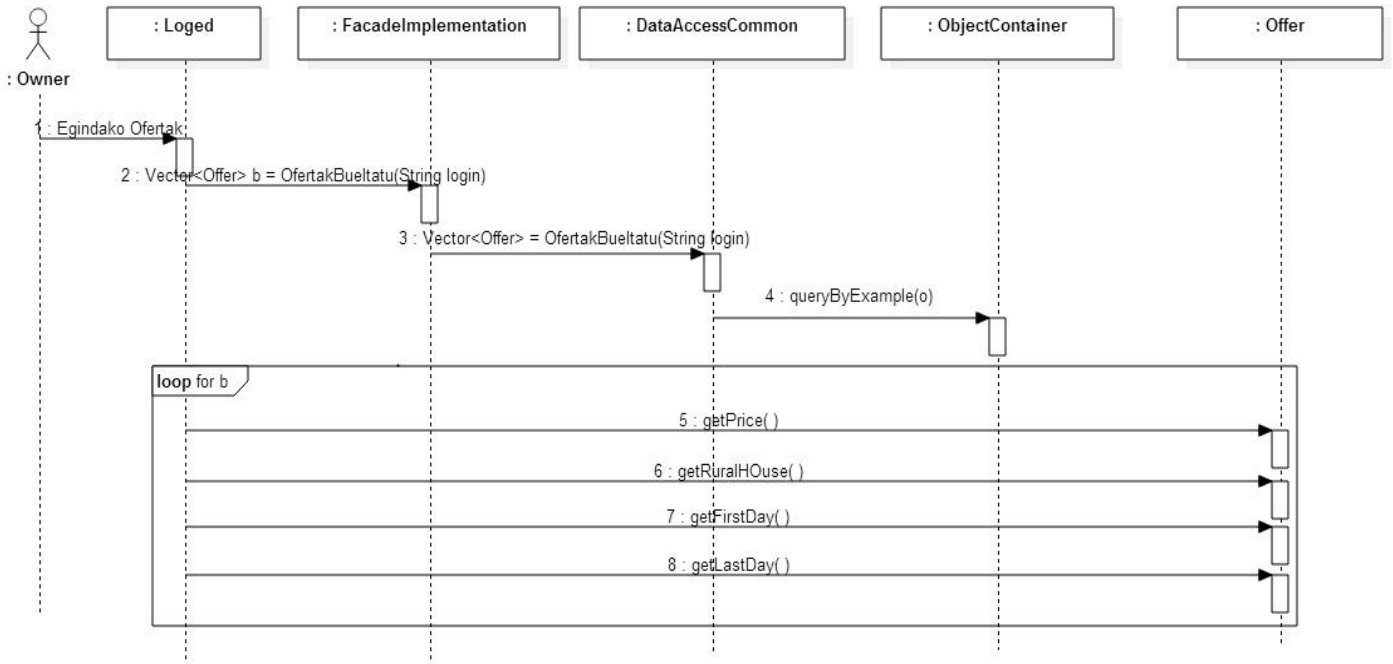
EGINDAKO OFERTAK:

❖ *GERTAERA FLUXUA:*

##Basic flow
1. **Owner** baten etxe bakoitzean dauden oferta guztiak pantailaratuko du.

##ALternative flow
1. Owner horrek ez baditu etxerik, **Sistemak** pantailaratuko du mezu bat etxerik ez daudela jakinaraziz.
2. Etxeak egon arren ofertaz ez badituzte **Sistemak** jakinaraziko digu.

❖ *SEKUENTZIA DIAGRAMA:*



OFERTA EZABATU:

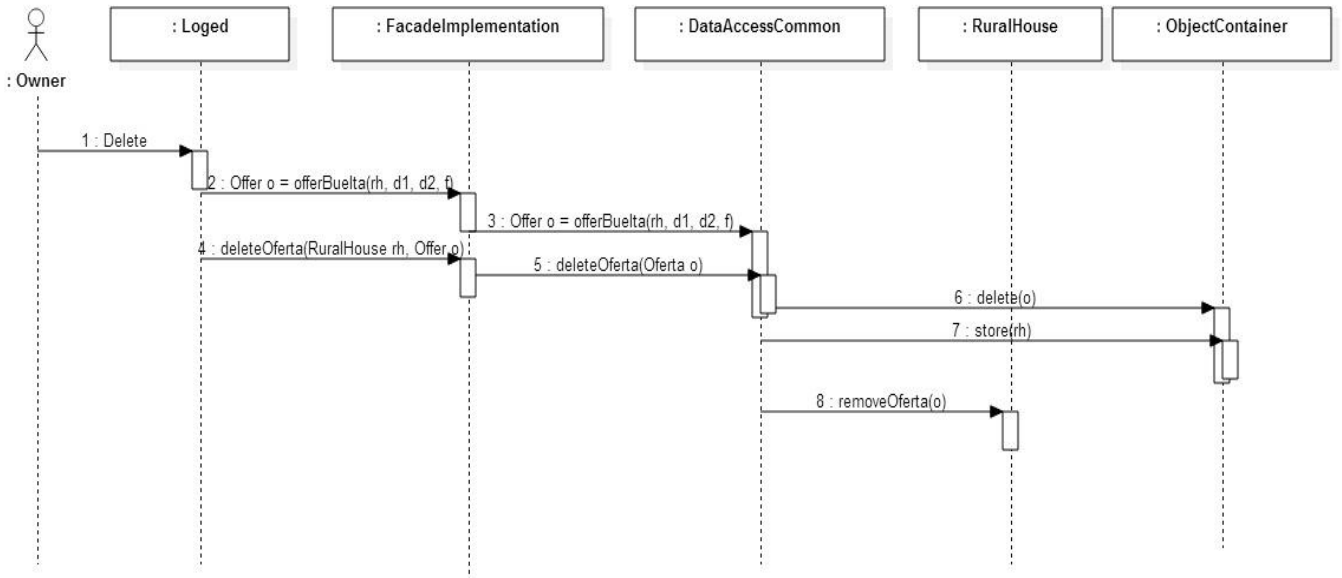
❖ *GERTAERA FLUXUA:*

#Flow of events: OFERTA EZABATU

##Basic flow

1. Lehendabizi ofertak pantailaratu beharko ditugu.
2. Behin ezabatzea nahi dugun oferta aukeratuta egon, "Ezabatu" botoia sakatu beharko dugu.
2. **Sistemak** oferta datubasetik ezabatuko du.

❖ *SEKUENTZIA DIAGRAMA:*



CLIENT EZABATU:

- *GERTAERA FLUXUA:*

#Flow of events : CLIENT EZABATU

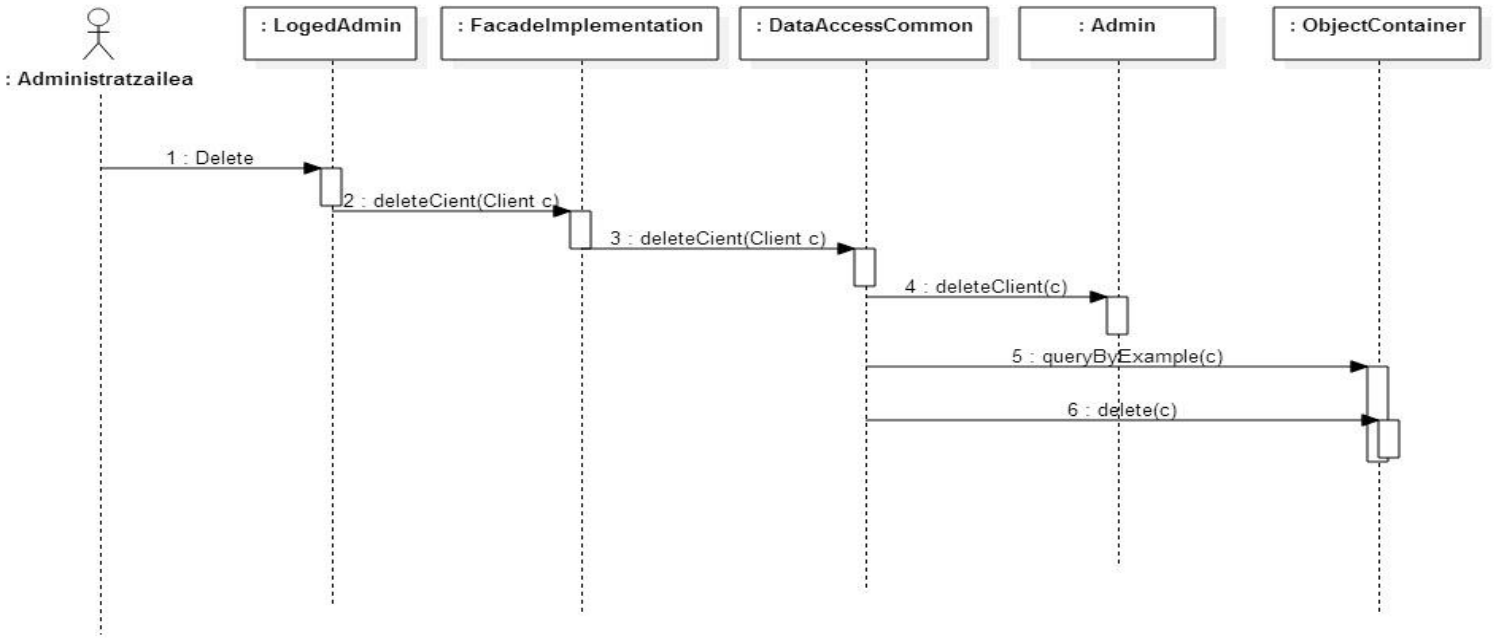
##Basic flow

1. **Administratzaileak** soilik Clienteak ezabatzeko aukera izango ditu.
2. **Administratzaileak** Client bat hautatuko du.
3. **Client** aukeratu ondoren, bi aldiz klikatu beharko ditu Client hori datu-basetik kentzeko.

##Alternative flow

1.Ezabatzerakoan arazoren bat gertatzen baldin bada, amaiera.

- *SEKUENTZIA DIAGRAMA:*



OWNER EZABATU:

- *GERTAERA FLUXUA:*

#Flow of events : OWNER EZABATU

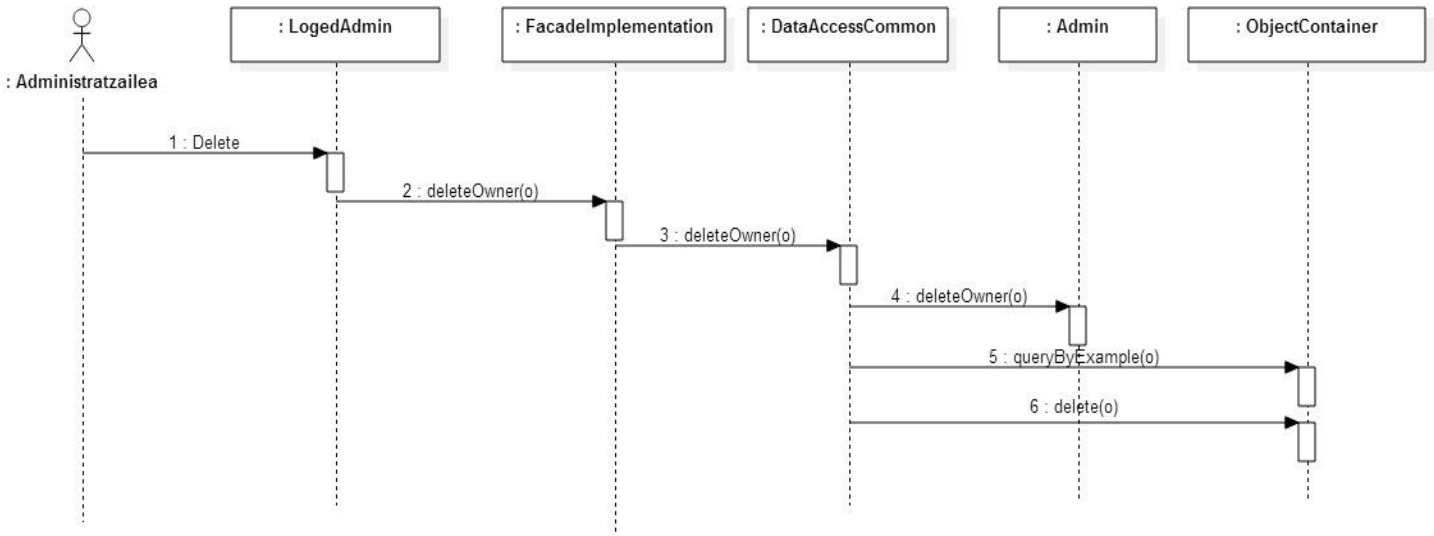
##Basic flow

1. **Administratzaileak** soilik Ownerrak ezabatzeko aukera izango ditu.
2. **Administratzaileak** Owner bat hautatuko du.
3. **Owner** aukeratu ondoren, bi aldiz klikatu beharko ditu Owner hori datu-basetik kentzeko.

##Alternative flow

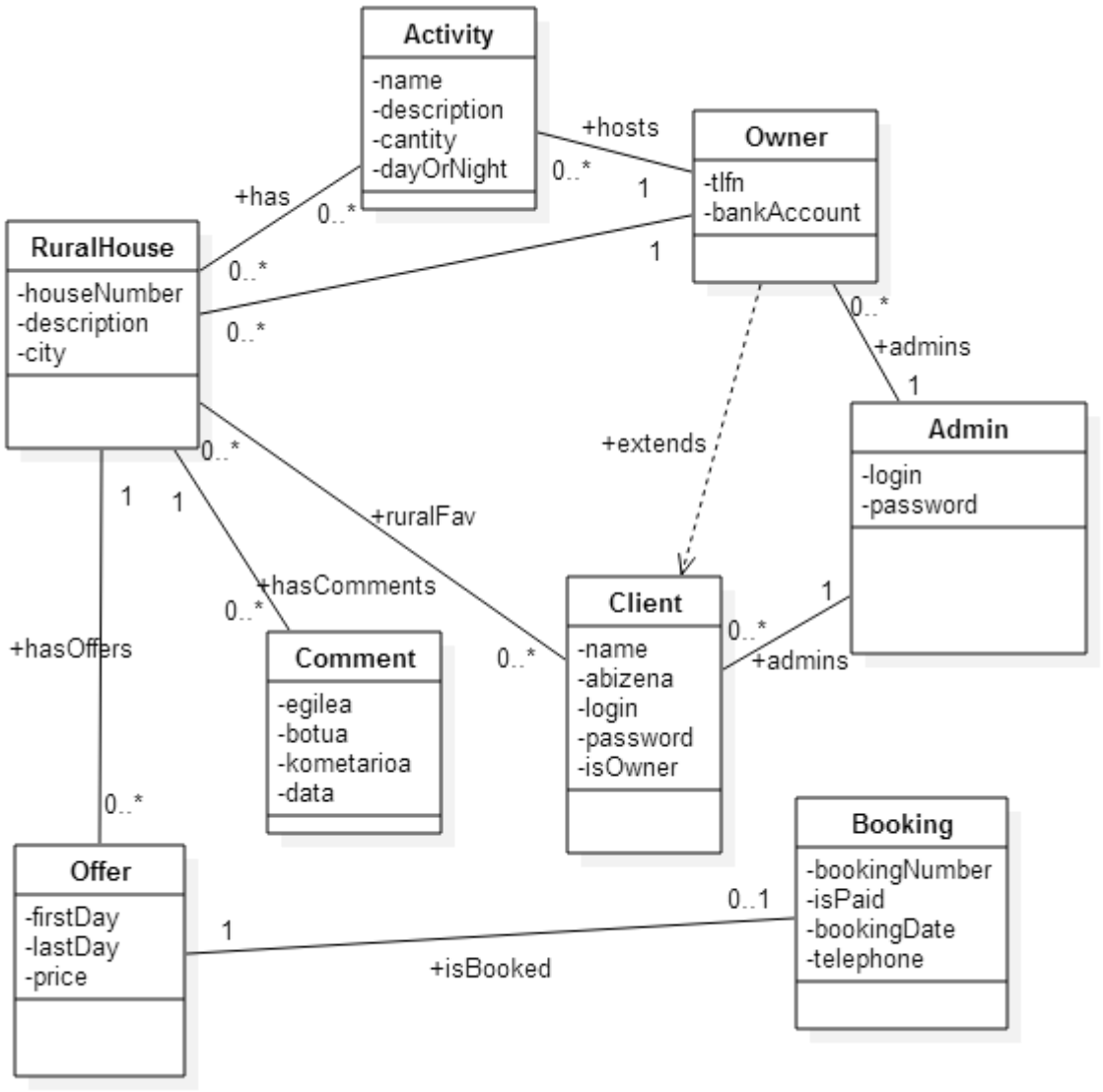
1.Ezabatzerakoan arazoren bat gertatzen baldin bada, amaiera.

- *SEKUENTZIA DIAGRAMA:*



INPLEMENTATUTAKO KLASEEN AZALPENA

Domeinuaren eredua:



Aplikazioan erabiliko diren objektuen inplementazioak daude hemen.

Comment:

Klase honek erabiltzaileei landetxe bakoitzaren feedback-a egiteko aukera ematen die.

```
private int botua;
private Client egilea;
private String coment;
private Date eguna;

public Comment(Client c, int bot, String com){
	this.egilea=c;
	this.botua = bot;
	this.coment=com;
	Date dat = new Date();
	this.eguna = dat;
}
```

Activity:

Klase honek landetxeen jabeek haien landetxeetan egingo dituzten jarduerak izango dira.

```
private String name;
	private String description;
	private int cantity;
	private Boolean dayOrNight;
	private Owner owner;

	public Activity(String n, String d, int c, Boolean dn, Owner o){
		name = n;
		description = d;
		cantity = c;
		dayOrNight = dn;
		owner = o;
	 }
```

Client:

Klase hau erregistratutako bezero bat da, landetxerik izango ez dituena.

```
private String name="";
private String abizena = "";
private String login="";
private String password="";
private Boolean isOwner=false;
private Vector<RuralHouse> ruralFav = new Vector();

public Client(String name, String abizena, String login, String
password,Boolean isOwner, Vector<RuralHouse> ruralFav){
	this.name=name;
	this.abizena=abizena;
	this.login=login;
	this.password=password;
	this.isOwner=isOwner;
}
```

Owner:

Owner-ak Client-en extends batekin erlazionatzen dira. Hauek izango dira landetxeen jabeak.

```
private Integer tlfn = null;
private String bankAccount = "";
private Vector<RuralHouse> ruralHouses=new Vector<RuralHouse>();
private Vector<Activity> activities = new Vector<Activity>();

public Owner(Integer tlfn, String bankAccount,Vector<RuralHouse>
ruralHousesBektorea,String name, String abizena, String login, String
password,Boolean isOwner, Vector<RuralHouse> ruralFav){
	super(name,abizena,login,password,true, ruralFav);
	this.bankAccount=bankAccount;
	this.tlfn=tlfn;
}
```

RuralHouse:

Klase honek erabiltzaileei landetxe bakoitzaren informazioa ikusteko eta honen feedback-a egiteko aukera ematen die.

```
private int houseNumber;
private String description;
private Owner owner;
private String city;
public Vector<Offer> offers = new Vector<Offer>();
public LinkedList<Comment> comments = new LinkedList<Comment>();
private Vector<Activity> activities = new Vector<Activity>();

public RuralHouse(int houseNumber, Owner owner, String description,
                  String city) {
	this.houseNumber = houseNumber;
	this.description = description;
	this.owner = owner;
	this.city = city;
	offers = new Vector<Offer>();
	comments = new LinkedList<Comment>();
}
```

Offer:

Klase honek landetxea noiz dagoen libre jakiteko da.

```
private Date firstDay;
private Date lastDay
private float price;
private Booking booking;
private RuralHouse ruralHouse;

public Offer(RuralHouse ruralHouse, Date firstDay, Date lastDay, float
price){
	this.firstDay=firstDay;
	this.lastDay=lastDay;
	this.price=price;
	this.ruralHouse=ruralHouse;
}
```

Booking:

Klase hau ofertak erreserbatzeko izango da.

```
private int bookingNumber;
	private boolean isPaid;
	private Date bookingDate;
	private String telephone;
	private Offer offer;



	public Booking(int bookingNumber,String telephone, Offer offer) {

		this.bookingNumber = bookingNumber;
		this.telephone=telephone;
		this.offer = offer;
		this.bookingDate= new java.sql.Date(System.currentTimeMillis());
		this.isPaid=false;

	}
```

Admin:

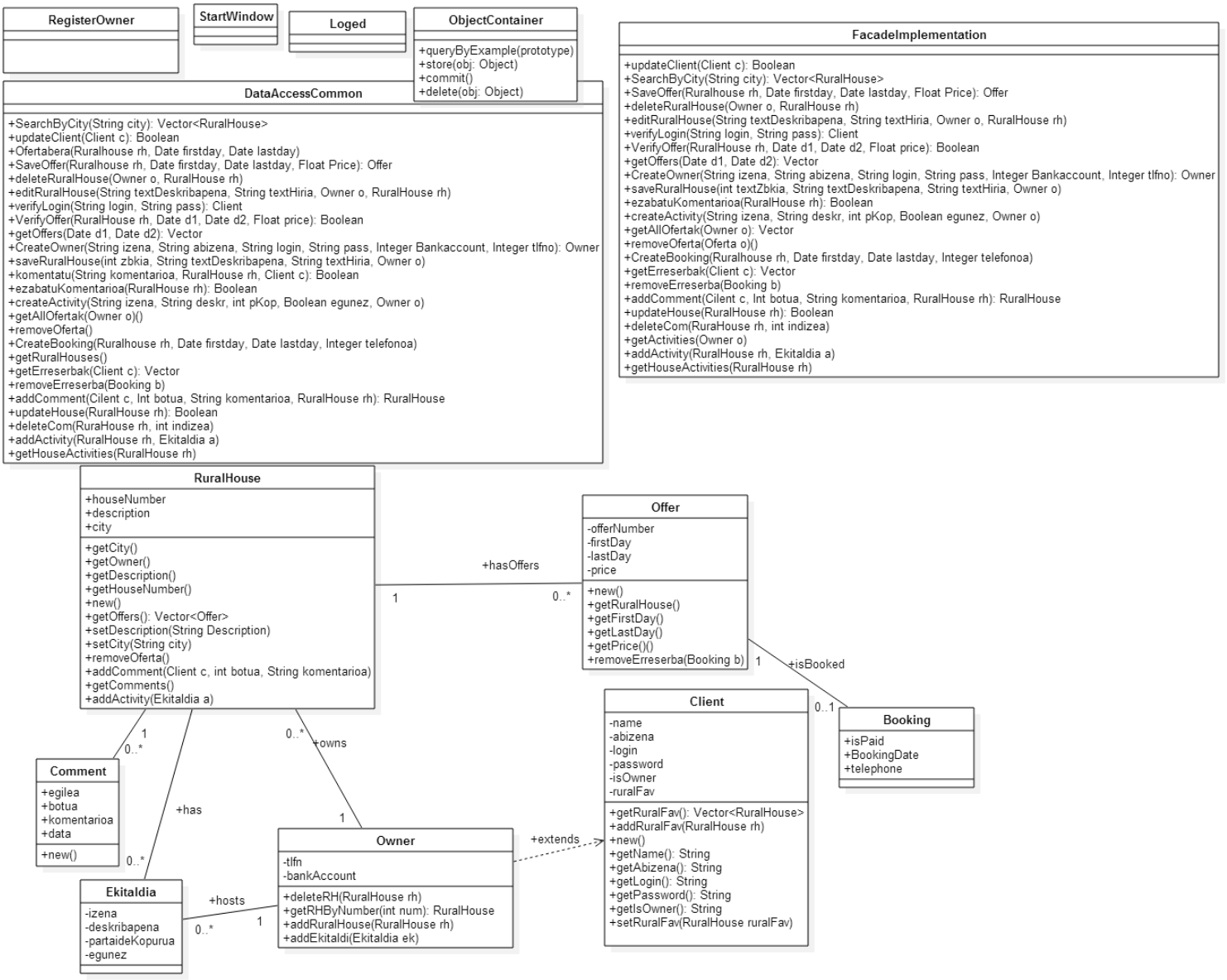
```
public class Admin implements Serializable {
	private String login="admin";
	private String password="1234";
	private Vector<Owner> owners = new Vector();
	private Vector<Client> clients = new Vector();



	public Admin(String login, String password, Vector<Owner> owners,
Vector<Client> clients) {
		// TODO Auto-generated constructor stub
		this.login=login;
		this.password=password;

	 }
```

Negozio logika:



ApplicationFacadeInterface:

```
public boolean createClient(String name, String surname, String login,
String password, boolean isOwner)
```

Client bat eratzeko behar diren parametro guztiak sartu ondoren, metodo honek true bueltatuko digu Client berria sortzerakoan arazoren bat gertatu ez bada, bestela false.

```
public boolean createOwner(Integer tlfn, String bankAccount, String
name, String abizena, String login, String password)
```

Owner bat eratzeko behar diren parametro guztiak sartu ondoren, metodo honek true bueltatuko digu Owner berria sortzerakoan arazoren bat gertatu ez bada, bestela false.

```
public Boolean saveRuralHouse(Integer ze, String hi, String de, Owner
o)
```

Landetxe bat sortuko du sartutako datuekin, logeatutako Owner-ari esleituko diona. Metodo honek true bueltatuko digu RuralHouse berria sortzerakoan arazoren bat gertatu ez bada, bestela false.

```
public Boolean editRuralHouse(String hi, String de, Owner o,
RuralHouse rh)
```

Landetxe baten informazioa aldatzeko erabiltzen den metodoa. Landetxe-zenbakia errepikaezina eta aldaezina denez (automatikoki sortua) ez da eskatzen. Metodo honek true bueltatuko digu RuralHouse-a arazorik gabe aldatzen bada, bestela false.

```
public Boolean deleteRuralHouse(Owner o, RuralHouse rh)
```

Datubasetik landetxea ezabatuko du, eta bere jabearen zerrendatik ere. Metodo honek true bueltatuko digu arazorik gabe exekutatu bada, false bestela.

```
public boolean saveOffer(RuralHouse rh, Date d1, Date d2, Float
prezioa)
Sortuko dugu Offer berri bat pasatutako parametroekin eta store(Offer
berria) egingo du datu basean gordetzeko.
```

```
public void inprimatuEtxeakOwner(Owner ow)
```

Eskatutako jabearen landetxe guztiak inprimatuko ditu kontsola, probak egiteko metodoa. Ez du ezer bueltatzen.

```
public Boolean verifyLogin(String a, String b)
```

Metodo honekin erabiltzaileak login egiterako orduan datu basean dagoeneko existitzen direla konprobatzen da. Horretarako erabiltzailearen izena eta pasahitzaz baliatuz, jabe berri bat sortzen da. Ondoren datu basearen kontrako query bat egiten da. Jasotako emaitza hutsa baldin bada false bueltatuko du (erabiltzailea ez dela existitzen) eta ez zaio sistemara sartzen uzten.

Jasotako emaitza hutsa ez baldin bada orduan erabiltzaile hori existituko da (dagoeneko datu basean sartua dagoela), sisteman sartzeko aukera ematen zaio.

```
public Boolean verifyLoginName(String log)
```

Metodo honekin jakingo dugu erabiltzaile izena hartuta dagoela ala ez. Horretarako metodoan sartzen dugu konprobatu nahi dugun login-a eta libre baldin badago true bueltatuko digu.

```
public boolean VerifyOffer(RuralHouse ruralHouse, Date firstDay, Date lastDay)
```

Metodo honen bitartez jakingo dugu sartzen dugun oferta datu basean gordeta dagoen ala ez. Horretarako sartuko dugu oferta gordeko den rural house, eta datak. False itzultzen badu oferta berri bat sortu dezakegu data horretan eta rural house horretarako.

```
public Vector<Offer> findOffer(Date firstDay, Date lastDay
```

Buletatuko digu data horretan dauden Oferta guztiak .

```
public Vector<RuralHouse> SarchByCity(String city)
```

Erabiltzaileak hiri baten izen bat pasako dio metodo honi, izen horrekin datu basearen etxeetan query bat egiten da. Jasotako emaitza zuzen guztiak bektore batean gordeko dira gero interfaze grafikoan ikusteko.

```
public Boolean updateClient(Client c)
```

Eragiketa honek, parametro moduan jasotzen duen `Client-a` datu basean eguneratzen du.

```
public void ImpgetAllRuralHouses(Vector<RuralHouse> bek)
```

Datubaseko landetxe guztiak inprimatuko ditu kontsolan, probak egiteko metodoa. Ez du ezer bueltatzen.

```
public String toString();
```

Klase horren String balioa itzultzen du.

```
public Owner getOwner(Client c)
```

Client bat sartu ondoren, falta diren eremuak Owner bat izateko beteko ditu Owner bat bueltatuz.

```
public Vector<RuralHouse> bektoreaLortu(String login)
```

Owner baten logina sartu ondoren bueltatuko digu Owner horren RuralHOuse guztiak.

```
public Owner ownerBuelta(String login)
```

Login bat sartuko dugu Owner guztia eskuratzeko

```
public Vector<RuralHouse> SarchssByOwner(Owner o)
```

Eragiketa honek, parametro moduan jasotzen duen `Owner-ak datu basean dituen etxeak bueltatzen ditu`.

```
public Boolean RuralHouseRefactorComment(RuralHouse rh)
```

Landetxearen komentarioak “ordenatuko” ditu bistaratzerakoan errorerik ez hutsunerik ez egoteko.

```
public RuralHouse addComment(Client c, int botua, String kom,
RuralHouse rh)
```

Pasa zaion etxeari pasatako parametroekin komentario bat gehitzen zaio.

```
public Boolean updateHouse(RuralHouse rh)
```

Eragiketa honek, parametro moduan jasotzen duen `RuralHouse-a` datu basean eguneratzen du.

```
public Boolean deleteAllCom(RuralHouse rh)
```

Pasa zaion etxearen komentario guztiak borratzen ditu eta etxea komentariorik gabe datu basean berriro eguneratzen da.

```
public Boolean deleteCom(RuralHouse rh, int i)
```

Pasa zaion etxearen komentarioa borratzen du eta etxea iruzkin hori gabe datu basean berriro eguneratzen da.

```
public int getFreeNumber()
```

Landetxe bat sortzean lehen zenbaki librea itzuliko du, landetxe-zenbaki moduan esleitzeko. Hauek hautomatikoki sortu ahal izateko.

```
public Boolean createActivity(String izena, String deskribapena, int
kop, Boolean egunez, Owner owner)
```

Ekitaldi bat sortuko du datubasean pasatako informazioan, eta jabeari esleituko dio beranduago honek bere landetxeetan ipini ahal izateko.

```
public Vector<Activity> getOwnerActivities(Owner o)
```

Pasatako jabeak aurrera eramaten dituen ekitaldien zerrenda itzultzen du. Lista hutsik bat ere ez badu.

```
public Vector<Activity> getHouseActivities(RuralHouse rh)
```

Pasatako landetxean egiten diren ekitaldien zerrenda itzultzen du. Lista hutsik bat ere ez badu.

```
public Boolean addActivity(Activity a, RuralHouse rh)
```

Pasatako ekitaldia landetxe horri gehitzen dio eta datubasean gordetzen du.

```
public Vector<Offer> OfertakBueltatu(String Login)
```

Owner baten nickname-a sartu ondoren, Owner hori egindako Oferta guztiak landetxe guztietan bueltatuko digu metodo honek.

```
public Boolean deleteOferta(RuralHouse rh, Offer o)
```

Sartuko dugu zein landetxetik ze oferta kendu nahi dugun eta ezabaketa ondo egiten bada true bueltatuko du. Arazoren bat gertatzen bada aldiz, false.

Datu atzipena:

DataAccessCommon:

```
public boolean createClient(String name, String surname, String login,
String password, boolean isOwner)
```

Client bat eratzeko behar diren parametro guztiak sartu ondoren, metodo honek true bueltatuko digu Client berria sortzerakoan arazoren bat gertatu ez bada, bestela false.

```
public boolean createOwner(Integer tlfn, String bankAccount, String
name, String abizena, String login, String password)
```

Owner bat eratzeko behar diren parametro guztiak sartu ondoren, metodo honek true bueltatuko digu Owner berria sortzerakoan arazoren bat gertatu ez bada, bestela false.

```
public Boolean saveRuralHouse(Integer ze, String hi, String de, Owner
o)
```

Landetxe bat sortuko du sartutako datuekin, logeatutako Owner-ari esleituko diona. Metodo honek true bueltatuko digu RuralHouse berria sortzerakoan arazoren bat gertatu ez bada, bestela false.

```
public Boolean editRuralHouse(String hi, String de, Owner o,
RuralHouse rh)
```

Landetxe baten informazioa aldatzeko erabiltzen den metodoa. Landetxe-zenbakia errepikaezina eta aldaezina denez (automatikoki sortua) ez da eskatzen. Metodo honek true bueltatuko digu RuralHouse-a arazorik gabe aldatzen bada, bestela false.

```
public Boolean deleteRuralHouse(Owner o, RuralHouse rh)
```

Datubasetik landetxea ezabatuko du, eta bere jabearen zerrendatik ere. Metodo honek true bueltatuko digu arazorik gabe exekutatu bada, false bestela.

```
public boolean saveOffer(RuralHouse rh, Date d1, Date d2, Float
prezioa)
Sortuko dugu Offer berri bat pasatutako parametroekin eta store(Offer
berria) egingo du datu basean gordetzeko.
```

```
public void inprimatuEtxeakOwner(Owner ow)
```

Eskatutako jabearen landetxe guztiak inprimatuko ditu kontsola, probak egiteko metodoa. Ez du ezer bueltatzen.

```
public Boolean verifyLogin(String a, String b)
```

Metodo honekin erabiltzaileak login egiterako orduan datu basean dagoeneko existitzen direla konprobatzen da. Horretarako erabiltzailearen izena eta pasahitzaz baliatuz, jabe berri bat sortzen da. Ondoren datu basearen kontrako query bat egiten da. Jasotako emaitza hutsa baldin bada false bueltatuko du (erabiltzailea ez dela existitzen) eta ez zaio sistemara sartzen uzten.

Jasotako emaitza hutsa ez baldin bada orduan erabiltzaile hori existituko da (dagoeneko datu basean sartua dagoela), sisteman sartzeko aukera ematen zaio.

```
public Boolean verifyLoginName(String log)
```

Metodo honekin jakingo dugu erabiltzaile izena hartuta dagoela ala ez. Horretarako metodoan sartzen dugu konprobatu nahi dugun login-a eta libre baldin badago true bueltatuko digu.

```
public boolean VerifyOffer(RuralHouse ruralHouse, Date firstDay, Date
lastDay)
```

Metodo honen bitartez jakingo dugu sartzen dugun oferta datu basean gordeta dagoen ala ez. Horretarako sartuko dugu oferta gordeko den rural house, eta datak. False itzultzen badu oferta berri bat sortu dezakegu data horretan eta rural house horretarako.

```
public Vector<Offer> findOffer(Date firstDay, Date lastDay
```

Buletatuko digu data horretan dauden Oferta guztiak .

```
public Vector<RuralHouse> SarchByCity(String city)
```

Erabiltzaileak hiri baten izen bat pasako dio metodo honi, izen horrekin datu basearen etxeetan query bat egiten da. Jasotako emaitza zuzen guztiak bektore batean gordeko dira gero interfaze grafikoan ikusteko.

```
public Boolean updateClient(Client c)
```

Eragiketa honek, parametro moduan jasotzen duen `Client-a` datu basean eguneratzen du.

```
public void ImpgetAllRuralHouses(Vector<RuralHouse> bek)
```

Datubaseko landetxe guztiak inprimatuko ditu kontsolan, probak egiteko metodoa. Ez du ezer bueltatzen.

```
public String toString();
```

Klase horren String balioa itzultzen du.

```
public Owner getOwner(Client c)
```

Client bat sartu ondoren, falta diren eremuak Owner bat izateko beteko ditu Owner bat bueltatuz.

```
public Vector<RuralHouse> bektoreaLortu(String login)
```

Owner baten logina sartu ondoren bueltatuko digu Owner horren RuralHOuse guztiak.

```
public Owner ownerBuelta(String login)
```

Login bat sartuko dugu Owner guztia eskuratzeko

```
public Vector<RuralHouse> SarchssByOwner(Owner o)
```

Eragiketa honek, parametro moduan jasotzen duen `Owner-ak datu basean dituen etxeak bueltatzen ditu`.

```
public Boolean RuralHouseRefactorComment(RuralHouse rh)
```

Landetxearen komentarioak “ordenatuko” ditu bistaratzerakoan errorerik ez hutsunerik ez egoteko.

```
public RuralHouse addComment(Client c, int botua, String kom,
RuralHouse rh)
```

Pasa zaion etxeari pasatako parametroekin komentario bat gehitzen zaio.

```
public Boolean updateHouse(RuralHouse rh)
```

Eragiketa honek, parametro moduan jasotzen duen `RuralHouse-a` datu basean eguneratzen du.

```
public Boolean deleteAllCom(RuralHouse rh)
```

Pasa zaion etxearen komentario guztiak borratzen ditu eta etxea komentariorik gabe datu basean berriro eguneratzen da.

```
public Boolean deleteCom(RuralHouse rh, int i)
```

Pasa zaion etxearen komentarioa borratzen du eta etxea iruzkin hori gabe datu basean berriro eguneratzen da.

```
public int getFreeNumber()
```

Landetxe bat sortzean lehen zenbaki librea itzuliko du, landetxe-zenbaki moduan esleitzeko. Hauek hautomatikoki sortu ahal izateko.

```
public Boolean createActivity(String izena, String deskribapena, int
kop, Boolean egunez, Owner owner)
```

Ekitaldi bat sortuko du datubasean pasatako informazioan, eta jabeari esleituko dio beranduago honek bere landetxeetan ipini ahal izateko.

```
public Vector<Activity> getOwnerActivities(Owner o)
```

Pasatako jabeak aurrera eramaten dituen ekitaldien zerrenda itzultzen du. Lista hutsik bat ere ez badu.

```
public Vector<Activity> getHouseActivities(RuralHouse rh)
```

Pasatako landetxean egiten diren ekitaldien zerrenda itzultzen du. Lista hutsik bat ere ez badu.

```
public Boolean addActivity(Activity a, RuralHouse rh)
```

Pasatako ekitaldia landetxe horri gehitzen dio eta datubasean gordetzen du.

```
public Vector<Offer> OfertakBueltatu(String Login)
```

Owner baten nickname-a sartu ondoren, Owner hori egindako Oferta guztiak landetxe guztietan bueltatuko digu metodo honek.

```
public Boolean deleteOferta(RuralHouse rh, Offer o)
```

Sartuko dugu zein landetxetik ze oferta kendu nahi dugun eta ezabaketa ondo egiten bada true bueltatuko du. Arazoren bat gertatzen bada aldiz, false.

ONDORIOAK

Proiektuaren atalik zailena hastea da. 3 geruzako programazioa nola funtzionatzen duen ulertzea zaila izan daiteke azalpen egokirik gabe. Horregatik, uste dugu agian hobe izango zela hasieratik gehiago zentratzea hau ondo azaltzen, diagrametan baino.

Bestalde, datu-base hau (db4o) nahiko anbiguoa eta korapilatsua iruditu zaigu hainbat gauzetan, eta agian ondo egon liteke 2. urteko datu baseak irakasgaiarekin bat eginez eredu erlazionala jarraitzen duen datu base bat erabiltzea. Azalpen bikoitzak izango genituzke hau ulertzeko, eta lan berarekin 2 irakasgaietan lortuko genituzke aurrerakuntzak.

RMIaren gaia oso astuna izan daiteke ez bada hasieratik ondo egiten. Proiektuak 3 geruzako eredua modu egokian inplementatzen badu, geure kasuan bezala, arazorik gabe exekutatuko da, baina hala ez bada proiektuan aldaketa oso sakonak eta astunak egin behar litezke. Hasieran ez dela oso ondo ulertzen kontuan hartuz, hau oso litekeena da. Horregatik, lehen puntuan aipatu dudan bezala, aproposagoa iruditzen zait denbora gehiago dedikatzea 3 geruzako eredu hau ondo ulertzeari programa handiarekin hasi baino lehen.