# 1 Project: Rural Houses

We have been dealing with the problem of developing a reservation system for rural houses to be implemented via a three-tier software architecture. Some disciplines and iterations have been performed up to now by the old team. The following is a description of the project initial statement as the team *HoldOffDuty* received it.

## 1.1. Description of the information system

The town of San Pedro del Romeral wants to develop an application to manage rural house booking, whose users will be both the owners and the clients of the accommodation facilities. We are not charged with the web design (this is going to be outsourced) but we must build up a client-server application so that the functionality of the system needs only to be mapped into the skeleton of the website.

## 1.2. Functional requirements

A rural house owner can register his/her properties in the system providing the following data: district, number of bedrooms, bathrooms, kitchens and dining rooms, as well as the number of parking spaces. Additionally a general description of the house can be included. The Rural House National Regulations establish a minimum of 1 kitchen, 3 bedrooms and 2 bathrooms. Obviously an owner can also withdraw a property from the system.

Rural houses are rented for offers, and the owner is free to organize them as s/he wishes. For instance, there could be the case that in August there were only fortnights (from 1st to 15th and from 16th to 31st), that Labor Day's long weekend ought to be taken as a whole (from 28th April to 1st May), that in July renting had to be done by weeks but in winter by single days, etc... Each offer has its own rate, and must be taken as a whole. The customer has no option to take a subset of the days in an offer, but obviously can book more than one pack.

In order not to discourage potential customers, registration in the system is not required in order to consult or book rural houses. But the owners should have an account to register, modify or withdraw their properties.

The system shall allow searching houses by address. Once a list of houses is obtained, the customer can see its features. If the client knows in advance the code or name of the house the system gives option to display it directly to see the features.

To see the availability of a house the customer should select the house, the possible entry date and the number of nights. The system will show the available offers that meet the customer's requirements lying in the intended period. For instance, suppose that the available offers are [Apr 25th–May 2nd], [May 5th–May 7th], [May 10th– May 14th] and [May 17th–May 20th]. The user then says that s/he wants to see the availability of the house in 15 nights from May 3rd. The system then would return the following offers with their rates: [May 5th–May 7th] and [May 10th– May 14th].

To book a specific house a customer needs to choose a concrete available offer, and then provide the house code, the intended entry date and the number of nights. The system checks the existence and availability of the corresponding pack and confirms the booking. If everything is correct it will show the owner's bank account and a caution amount to be transferred. The customer is asked to provide a contact phone. Later, when the owner checks that the transfer has been produced then s/he will change the availability status of the involved offer (or offers, if more than one has been booked).

Additional functions will be discussed with the customer along the software development process.

# 2 Management

At present the project is thwarted. Urgent actions are being taken in order to recover it.

## 2.1. Responsible for the project

Bob Scene, Project Manager

## 2.2. Developing team

~~*HoldOffDuty*: Jim, Aurore & Doc~~ (Bob says fresh raw meet is coming)

## 2.3. Schedule

| According to | 1st prototye | 2nd prototype | Final product | Deployment |
|---|---|---|---|---|
| **Marketing** | ~~Nov, 08,14~~ | ~~Dec, 30, 15~~ | Feb, 28, 15 | Mar, 15, 15 |
| **Management** | ~~Nov, 10, 15~~ | ~~Jan, 12, 15~~ | Feb, 31, 15 WTF?? | Mar, 15, 15 |
| **Development** | ~~Jan, 23, 15~~ | Feb, 17, 15 | Mar, 30, 15 | Apr, 22, 15 |

## 2.4. Status

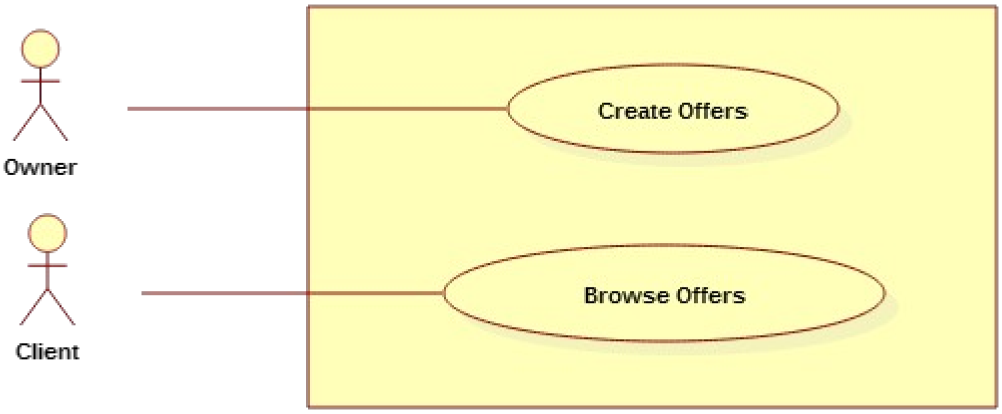
The project is thwarted due to unexpected labor claims from the development team. At present Bob is trying to recover it have been specified in confidential memo No  3245A.

The customer is unaware of all that shit but we cannot hide it for longer. This red-neck sucker expects the product fully operative before Easter campaign.

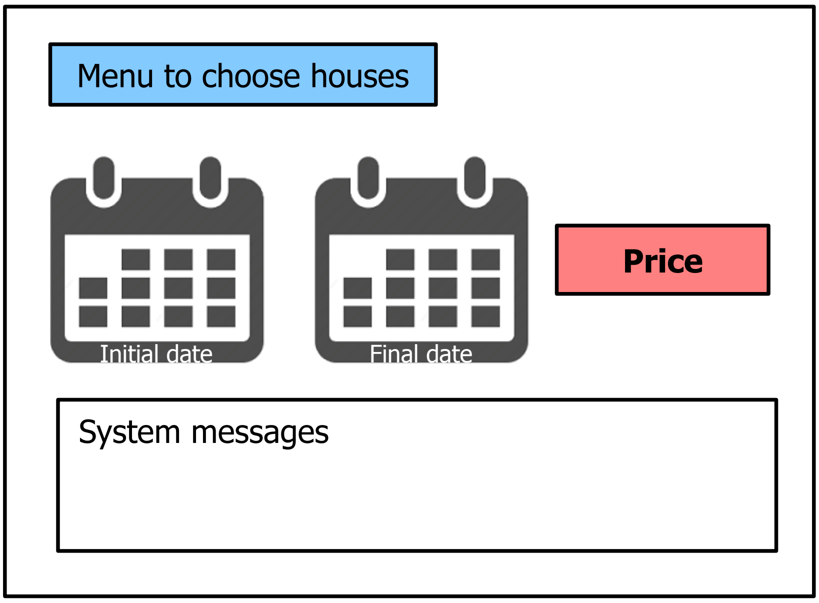
# 3 Recovered artifacts

Some moron in Human Resources forgot to remove permissions to one of the members of the team before we fired them and she had time to do some retaliation. We are investigating what failed in the process and consequences will be derived.

## 3.1. Use-Case diagrams



### 3.1.1. GUI Prototype for *Create Offers* UC



### 3.1.2. Event flow for *Create Offers* UC

**Normal flow**

1. **System** displays **Owner**'s rural houses.
2. **Owner** chooses House.
3. **Owner** chooses Starting day, Ending day and price for new Offer.
4. **System** checks dates
5. If dates are correct and new Offer doesn't overlap with existing ones, **System** creates new Offer with provided features.

**Alternate Flow**

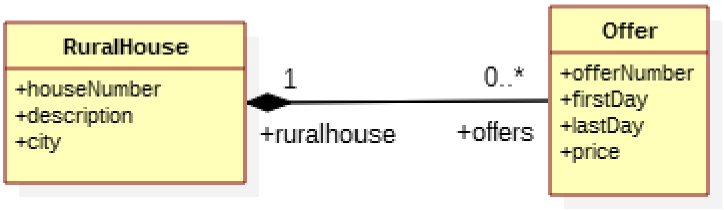
5A. If dates aren't correct or new Offer overlaps with existing one **System** writes error message. Back to step 1

### 3.1.3. Event Flow for *Browse Offers* UC

**Normal Flow**

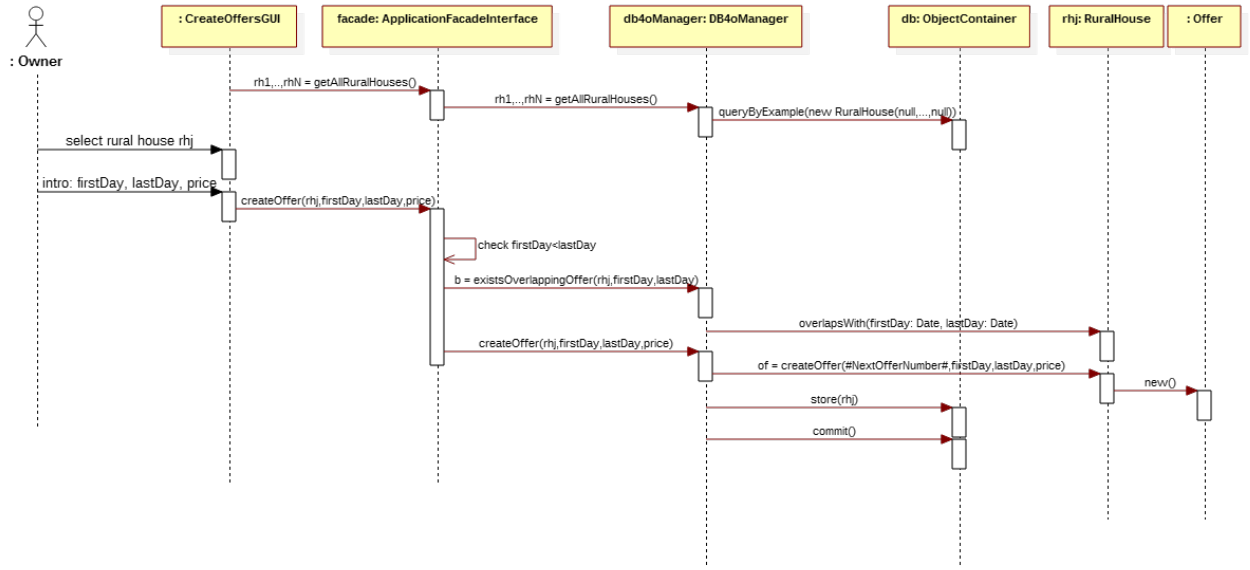
1. **System** presents houses to **Client**.
2. **Client** chooses House.
3. **Client** introduces Time-lapse to check.
4. **System** displays all existing offers whose periods lie in Time lapse, if any.

## 3.2. Domain model



## 3.3. Sequence diagrams

### 3.3.1. SD for *Create Offers* UC


: Owner
: CreateOffersGUI
facade: ApplicationFacadeInterface
db4oManager: DB4oManager
db: ObjectContainer
rhj: RuralHouse
: Offer
rh1,..,rhN = getAllRuralHouses()
rh1,..,rhN = getAllRuralHouses()
queryByExample(new RuralHouse(null,...,null))
select rural house rhj
intro: firstDay, lastDay, price
createOffer(rhj,firstDay,lastDay,price)
check firstDay<lastDay
b = existsOverlappingOffer(rhj,firstDay,lastDay)
overlapsWith(firstDay: Date, lastDay: Date)
createOffer(rhj,firstDay,lastDay,price)
of = createOffer(#NextOfferNumber#,firstDay,lastDay,price)
new()
store(rhj)
commit()