UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

Functional and Architectural Requirements

Client: Gavin Potgieter

Team: CodeBlox

Tshepo Malesela (Bsc: Computer Science)

Lorenzo Spazzoli (Bsc: Computer Science)

Bilal Muhammad (BIS: Multimedia)

Dirk de Klerk (BIS: Multimedia)

July 29, 2016

Contents

1 Introduction

The purpose of this document is to provide a detailed overview of the functional and architectural requirements with regard to the DropOff Project that was assigned to team CodeBlox. The document will be under constant revision, and will change as the project progresses.

It will provide the development team with a reference that can be used to develope from, thus ensuring that all functional and architectural requirements are met.

The document will also serve as means of communication between the client and development team. Thus any additional requirements from the clients' side can be appended to the document, and has to be adhered to from the development side.

2 Project Objectives

The **primary objective** of the project is to create an automated service system that will provide customers with the ability to grant delivery personnel access to a drop safe and/or demarcated area of their home. This will ensure that deliveries to be made can be completed without the need of someone being physically present on the premises.

Initially this will consist of the generation of a One Time Pin (OTP), that will be sent to the delivery personnel, granting them single access to the demarcated area. Once the delivery has been made. The user needs to be able to ensure that the residence is secure.

If time allows it. The system will be expanded to use audio and video communication to validate the identity of the delivery person.

In addition to the functionality of the system, it needs to be as **cost effective** as possible. This will allow a larger audience to gain access to the system.

The **secondary objective** (if time permits) is to make the system scalable. This will enable services to expand to other areas of the home. Thus allowing users an affordable solution to home automation.

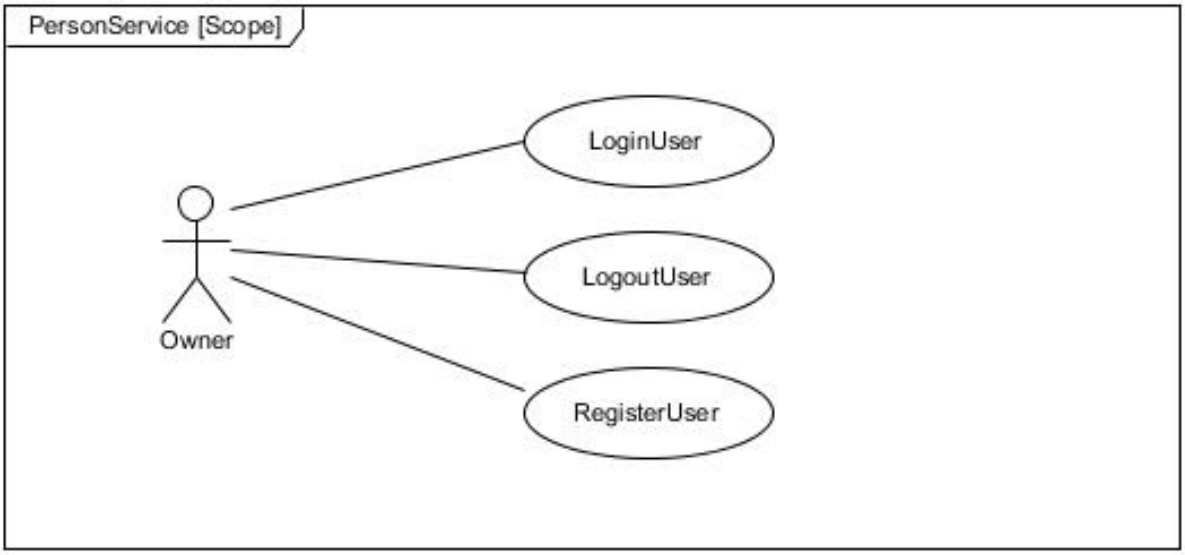
3 Functional Requirements

3.1 PersonService

The PersonService module will be responsible for maintaining user information. The module will also have the responsibility of storing demographic information about the users, store information about the users' next of kin, and finally keep track of the users account status.

3.1.1 Scope

The scope of the Users module is shown below

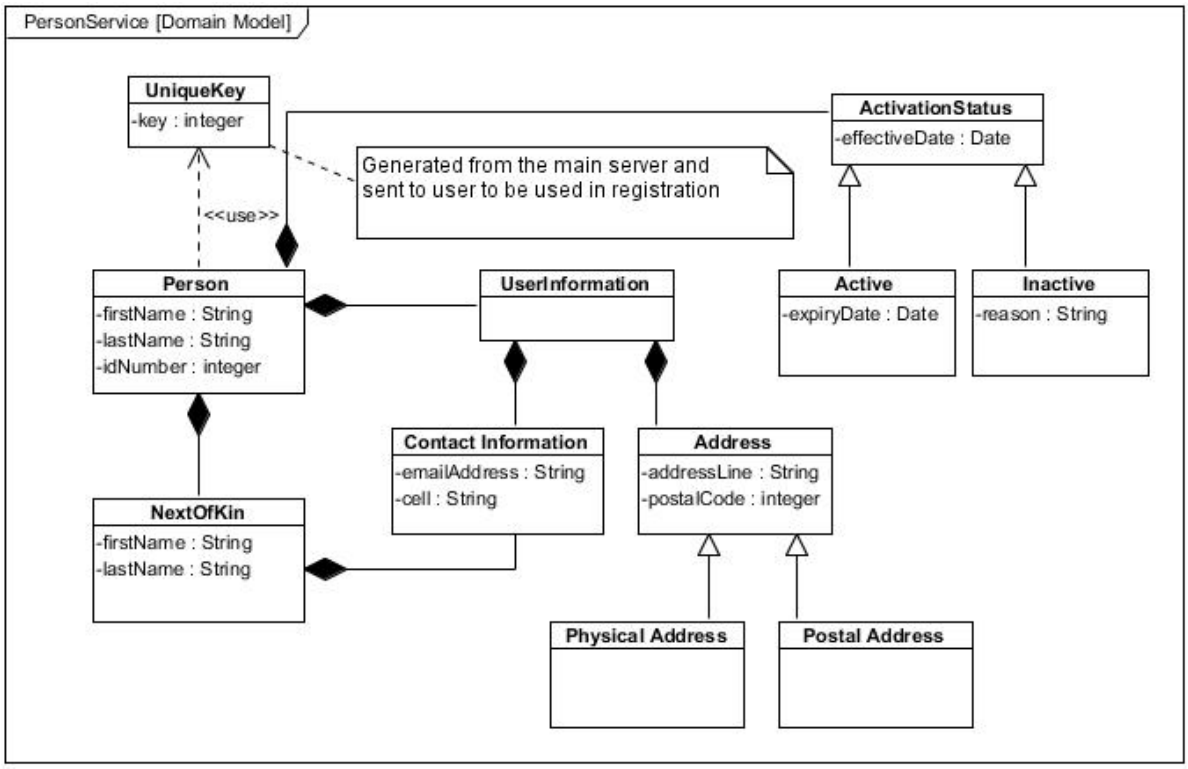


The scope of the Users module includes

- adding, removing, and modifying users on the system
- changing user information

3.1.2 Domain Model

The domain model for the Users module is shown below



Each user has a firstname and a lastname, as well as an id number. Each person will also have associated with them UserInformation, which will include email, cellphone, as well as Address. A next of kin will also be available in case the user cannot be reached. Lastly the Person will have associated with it an ActivationStatus, that will determine if the user is still able to receive services.

3.1.3 Use Cases

- **loginUser** - Allows a user to log into the system.

  preconditions:

  - The user has already been registered with the system.
  - The user is still considered an active user.
  - The user has provided the correct login credentials.

  postconditions:

  - The user is logged into the system.
  - User gains access to all the services available to him/her.

- **logoutUser** - Allows a user to logout of the system.

  preconditions:

  - The user is already logged into the system.

  postconditions:

  - The user is logged out of the system.
  - The user no longer has access to the services provided by the system.

- **registerUser** - Allows a user to be registered on the system and provide them access to the various services.

  preconditions:

  - The user does not already exist on the system.
  - The user has been provided with a unique registration key

  postconditions:

  - The user has been added to the system and may continue.

- **updateDetails** - Allows a user to update their information.

  preconditions:

  - The user exists on the system
  - The user has to correct credentials

postconditions:

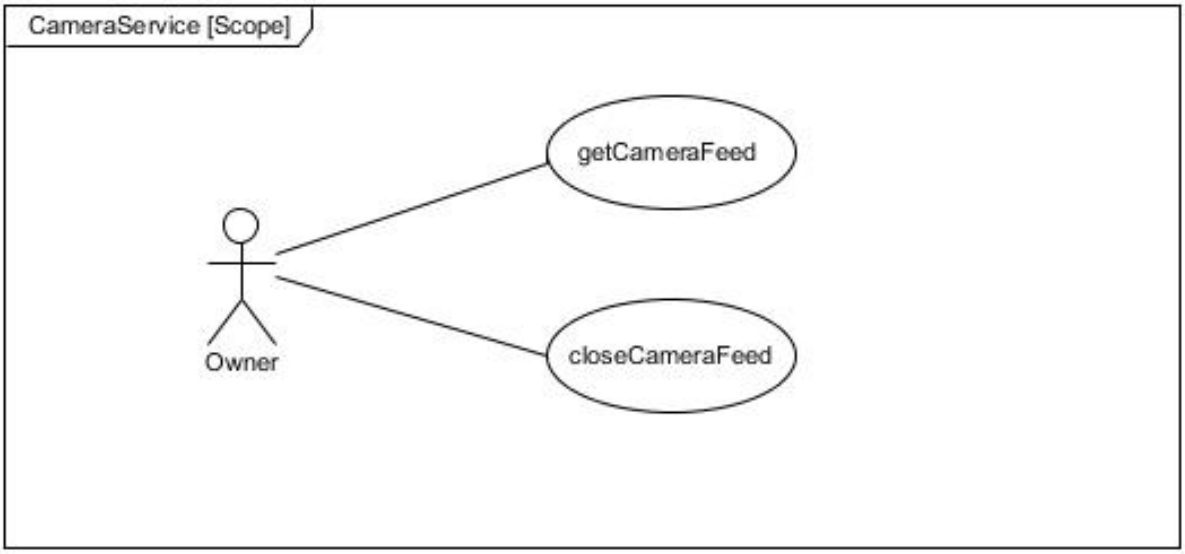
- The users' information is updated.

3.2 CameraService

The CameraService is responsible for providing the user with access to the live camera feed. The user needs to be able to access a feed, and also needs to be able to close a camera feed. This will ultimately allow them to verify the identity of the delivery person.

3.2.1 Scope

The scope of the CameraService module is shown below

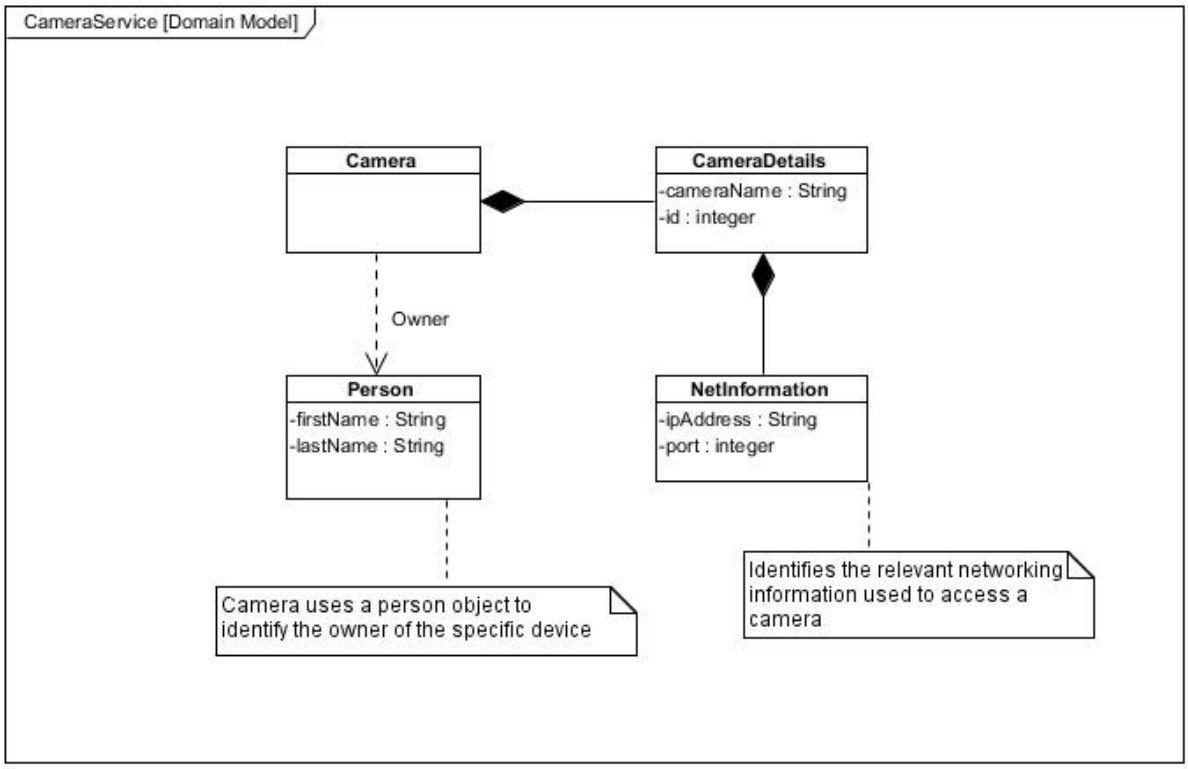


The scope of the CameraService module includes

- Gaining access to the live camera feed.
- Close the camera feed as a means to save on bandwidth.

3.2.2 Domain Model

The domain model for the CameraService module is shown below



Each Camera will have associated with it CameraDetails that will provide information to how the camera may be accessed. Also the Camera Class will be associated with a Person class to identify them as the owner of the device.

3.2.3 Use Cases

- **getCameraFeed** - Enables the user to gain access to the live camera feed.

  preconditions:

  - The user is still active in the system.
  - The camera is available to be accessed.

  postconditions:

  - The user receives a live stream.

- **closeCameraFeed** - Allows a user to close a camera feed.

  preconditions:

  - The camera feed is currently streaming.

  postconditions:

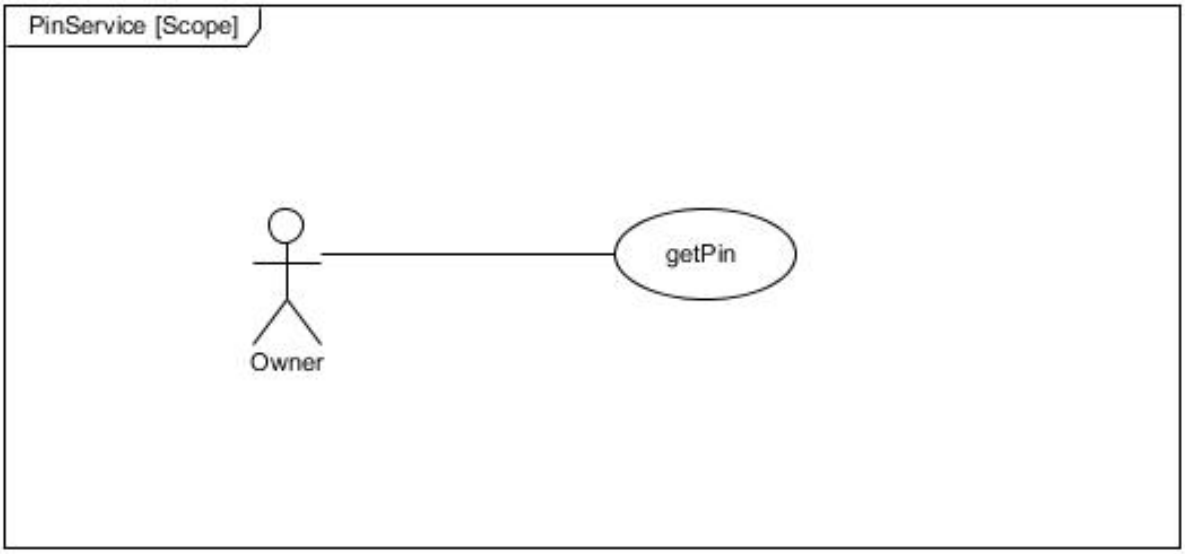
  - The camera stops streaming a live feed.

3.3 PinService

The PinService Module is responsible for generating new pins as a backup, should the camera system fail. The module will also need to keep track of the status of the pin i.e. whether or not it has been used or not, as well as its life time. If a pin has been used or it has reached its life time maximum, a new pin has to be generated.

3.3.1 Scope

The scope of the PinService module is shown below

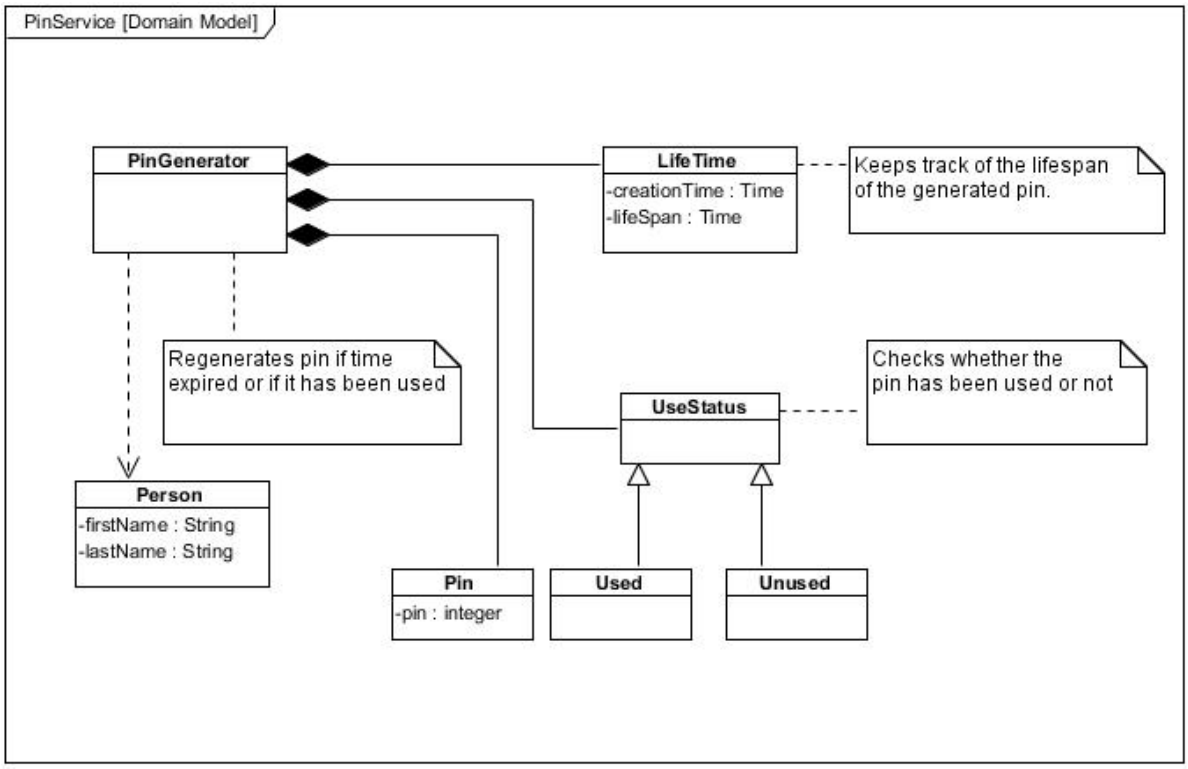


The scope of the PinService module includes

- Generating a new pin.
- Keeping track of a pins' validity through life time and status.

3.3.2 Domain Model

The domain model for the PinService module is shown below



The main class will be the PinGenerator. Associated with each pin generator will be a Pin, which will store the relevant pin that may be used as a fail safe. Also associated with the PinGenerator are the LifeTime and UseStatus classes, which are responsible for monitoring the Pins' validity. The PinGenerator is also associated with a Person so that each generated pin is sent to the correct person.

3.3.3 Use Cases

- **generatePin** - The System will generate a new pin to be used as a fail safe.

  preconditions:

  - A pin has expired.
  - A pin has already been used.

  postconditions:

  - A new pin is generated.

- **displayKeypad** - The keypad to enter the pin will be displayed.

  preconditions:

  - The system is making use of the failsafe.

  postconditions:

  - The keypad is displayed on screen.

- **updateServer** - The pin is updated server side for later use.

  preconditions:

  - A new pin has been generated.

  postconditions:

  - The new pin is reflected on the server.

- **getPin** - Allows the user to receive the generated pin.

  preconditions:

  - The user is logged into the system.
  - The pin is available.

  postconditions:

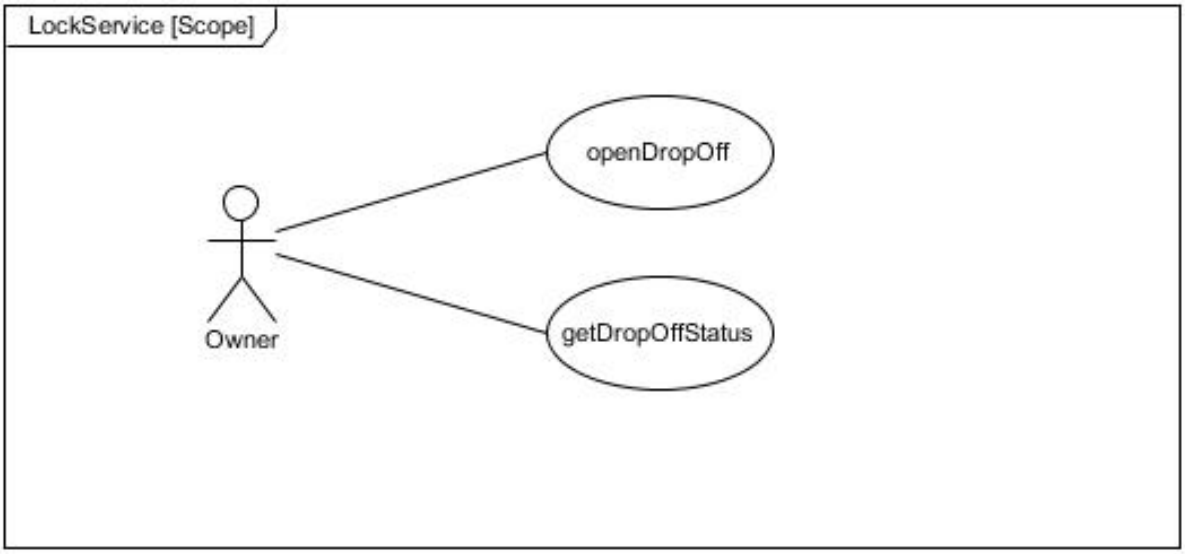
  - The pin is sent to the user.

3.4 LockService

The LockService Module will have the responsibility of controlling access to the dropbox. The lock mechanism can be comprised of any suitable technology for example a step motor or magnetic lock. The service should be able to unlock and lock the device, as well as keep track of the device status.

3.4.1 Scope

The scope of the LockService module is shown below

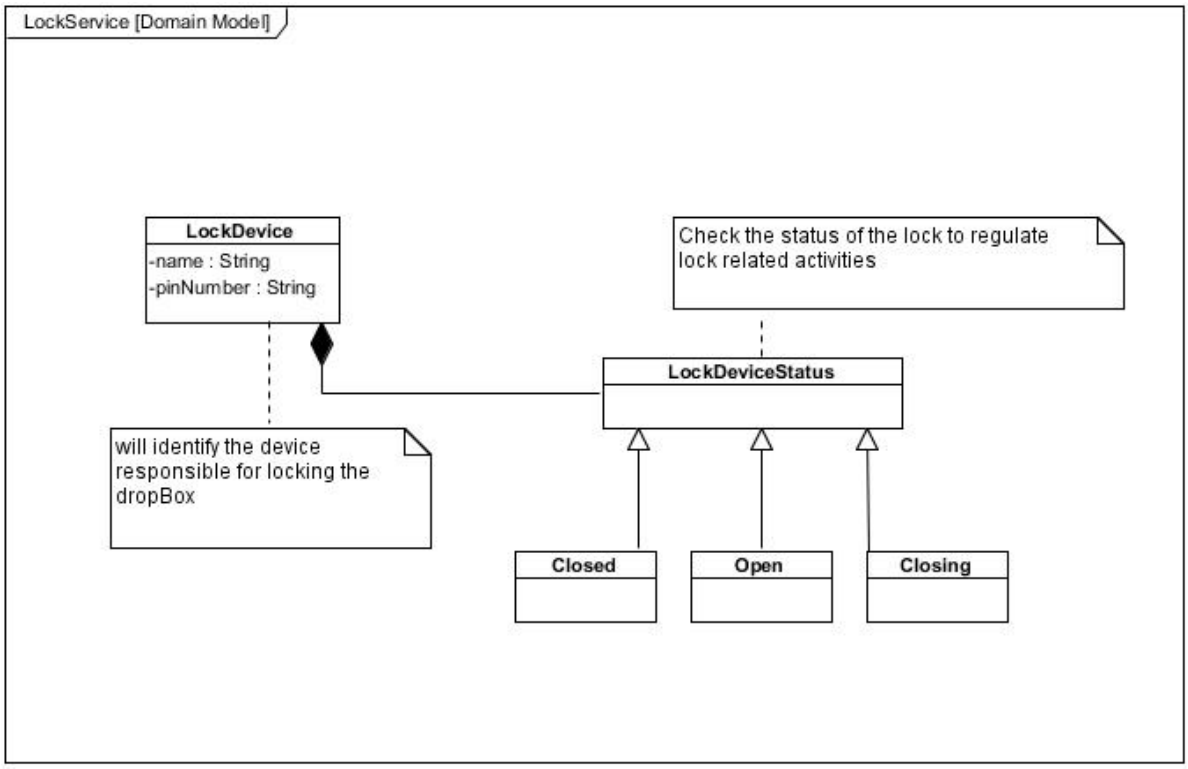


The scope of the LockService module includes

- Open up the dropOff box (via unlock or rotate of motor)
- Lock the dropOff box
- keep track of the devices' status.

3.4.2 Domain Model

The domain model for the LockService module is shown below



The LockDevice Class consists of a name and a pinNumber. The name is used to identify the device and pinNumber will be used to identify the pin used to interface with the raspberry pi. Associated with the LockDevice class will be a LockDeviceStatus, that will be used to keep track of the devices' current status i.e. Closed, Open, and Closing.

3.4.3 Use Cases

- **openDropOff** - Will allow the user to open up the box once identity has been verified.

  preconditions:

  - The box is closed.
  - The user is logged in.
  - The camera feed is accessible (System is functioning normally).

  postconditions:

  - The box is opened up.

- **lockDropOff** - This function will allow the dropOff box to be locked and secured again.

  preconditions:

  - The device is currently open/unlocked.

  postconditions:

  - The device is locked.

- **getDropOffStatus** - Provides the user/system with state information about the system.

  preconditions:

  - The system is in some state, either Open, Closed, Closing.

  postconditions:

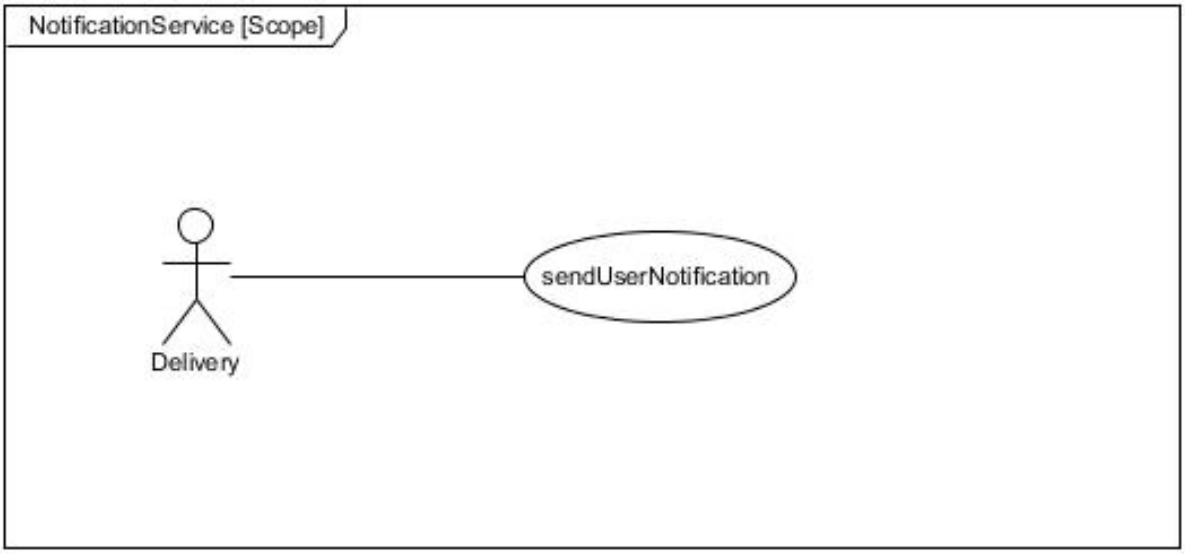
  - The appropriate state is returned to the system/user.

3.5 NotificationService

The NotificationService Module is responsible for informing the user that a delivery person is at his/her residence. If the home device is unable to communicate with the server, a message has to be displayed locally informing the delivery person to contact the owner to obtain a pin.

3.5.1 Scope

The scope of the NotificationService module is shown below

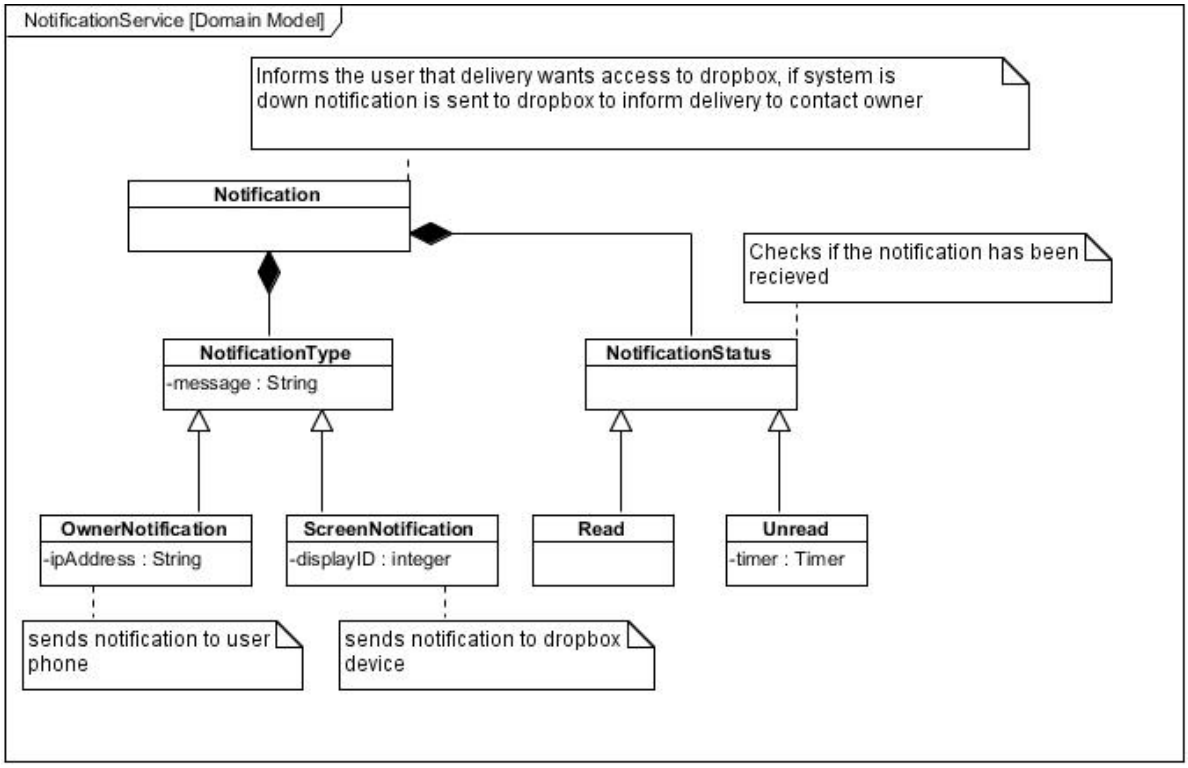


The scope of the NotificationService module includes

- Send a notification to the user to inform him/her of a potential delivery.
- Send a message to the local device to instruct delivery person.
- Set and get notification status.

3.5.2 Domain Model

The domain model for the NotificationService module is shown below


NotificationService [Domain Model]
Informs the user that delivery wants access to dropbox, if system is down notification is sent to dropbox to inform delivery to contact owner
Notification
Checks if the notification has been recieved
NotificationType
-message : String
NotificationStatus
OwnerNotification
-ipAddress : String
ScreenNotification
-displayID : integer
Read
Unread
-timer : Timer
sends notification to user phone
sends notification to dropbox device

3.5.3 Use Cases

- **sendUserNotification** - Will send a notification to the user, informing him/her of a delivery.

  preconditions:

  - The alert button has been pressed.
  - The home device is able to communicate with the server.

  postconditions:

  - A notification is sent to the user.

- **sendLocalNotification** - Will generate a message that will be displayed on the local device.

  preconditions:

  - The alert button has been pressed.
  - The local device cannot communicate with the server.

  postconditions:

  - A message is displayed on the local device interface.

- **setNotificationStatus** - Allows the status of the sent notification to be set as Read, or Unread.

  preconditions:

  - A notification has been sent.

  postconditions:

  - The status of the notification is set to Unread. Once read it is set to red.

- **getNotificationStatus** - Provides the user/system with the current status of the notification.

  preconditions:

  - A notification status has been set.

  postconditions:

– Returns the status of the notification.