UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Architectural Requirements Specification and Design

Eiffel Team - March 13, 2017

Members:
Dilapisho Monkeli 15074260
Eksteen Ernst 28398603
Haasbroek Henri-Dawid 15046657
Mahoko Dimpho 15175091
van Schalkwyk John 14307317
Veldsman Dian 12081095
Mdluli Paul 13049756

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to outline the design of the system to be developed. This document can be analyzed from a more technical view point and is meant to be an technical outline for developers. Similar to the requirements specification and the architectural specification, the design specification also forms part of a formal contract between developers and clients.

1.2 Document conventions

- Documentation formulation: LaTeX
- Architectural schemas: JGraph's Draw.io

1.3 Related documents

- System Requirement Specification (SRS) for the NavUP System

2 System description

The proposed system is an integration of four subsystems that in essence encompass the entire product. The system is aimed to manage users (the three types), and give each an enjoyable experience of the Notifications, GIS, Fitness and Points of Interest modules.

3 Overall Architecture

The overall architectural structure that has been chosen is the Model-View-Controller(MVC) architecture. MVC encapsulates the intractions between the system and its users. A multi-layered approach will be used in-conjunction with the MVC architecture.

3.1 Architectural patterns

The most notable architectural pattern is that of the MVC which is used to describe the notifications module best. This pattern however is not limited to the notifications section as the GIS module (map interface) uses it as well. The overall system works using the n-tier architecture, where the three tiers are Presentation Layer (What the user sees and experiences), Application Layer (The functions and functionality the system provides) and the Data Access Layer (Where all the server side functionality resides). This architectural pattern was selected in order to reduce the coupling of systems however increase their cohesion. The User Modules describe the interactions with the system at the Application layer as well as some functionality which resides within the Data

Access Layer. The Notifications, Fitness and Points of Interest Modules each deal mainly with the Application and Data Access Layers where their information is translated and brought forward to the user through the Application Layer.

3.2 Software system attributes

The system is required to be reliable, available, secure and maintainable. In terms of reliability the software should accurately provide navigational information to the user (With the only constraint being the current accuracy of geo-location services). The user should be able to access the system from anywhere within the campus (This is limited to wifi access points) and from any smart device. The user should also be afforded the assurance that whatever information is shared with the system is kept private as far as possible. This means that database security and network transfer protocols are required to be of the best nature. Finally for the system to be maintainable it needs to be well designed. To this end the above mentioned architectural patterns are employed.

3.3 Design constraints

The design and implementation of the system is limited to the capabilities of the software and hardware in use. The system is required to run smoothly on all devices and as such cannot be RAM expensive, meaning that the computations should mainly be handled server-side. For the sake of ubiquity, the high-level programming language chosen for Android and IOS devices will limit the type of functionality. The intention being to release a singular product as opposed to multiple versions of a product. Therefore one has to be conscious of the capabilities of either language.

3.4 Technology and framework selection

The system requires a device which has internet connectivity, and a wifi network card. This hardware comes standard with most current mobile technologies and any other technology that supports a mapping system. On the server side, a database management tool is required, such as a MySQl server, to house user data.

The system is intended to be released on Android and IOS devices.

4 External Interface Requirements

4.1 Software Interfaces

Internet Client (Most Browsers i.e. Chrome, FireFox, Safari)
App Client (Android, IOS)

4.2 Communications Interfaces

The http/https protocol shall be used for both software interfaces.

5 Details of subsystem

5.1 Subsystem Modules

5.1.1 User Module

Scope and Structure
The user module describes how the user will engage with other systems at an overall level. Due to the nature of the system and the dependence on other modules, aspects in certain figures will be vague only to be described later on. The user module uses inheritance to distinguish the three kinds of users with

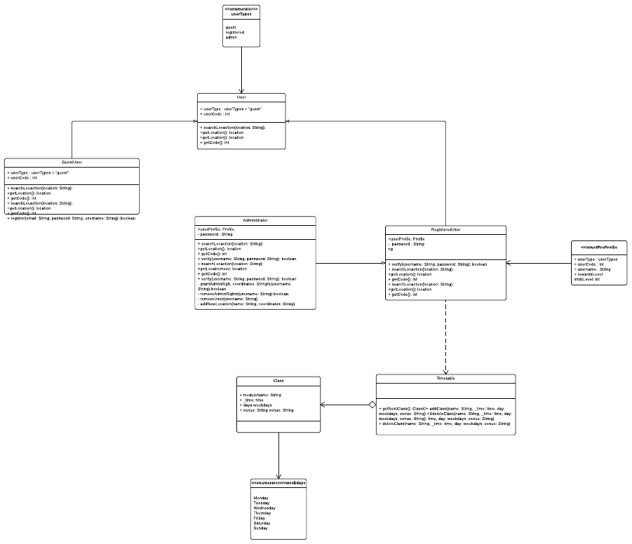

Figure 1: User Module Class Diagram

the primary/parent type being "User". The Registered user acts as the link differential point between both administrator and registered user accounts and

guest users. The Registered users are allowed to interact with the timetable and Class classes. These are included within the user module as they are exclusive to the user modules functionality

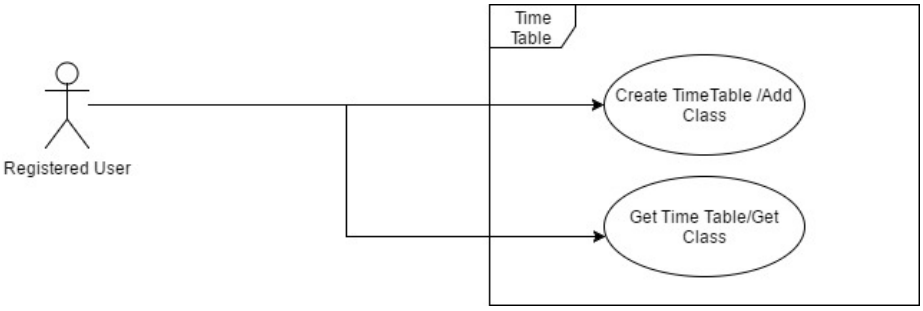

Figure 2: Use Case Diagram for User module with Timetable

The Timetable Class describes how the actions of adding/removing a class as well as retrieving ones timetable are achieved. This system forms part of the core functionality of the entire system as a whole

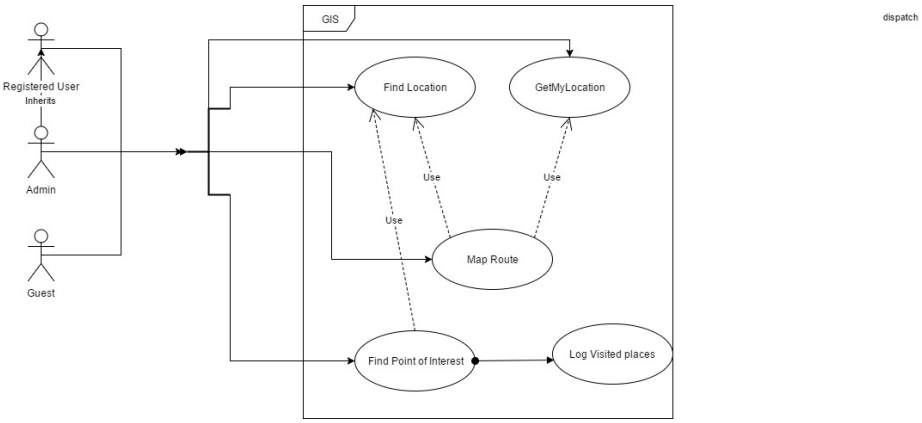

Figure 3: Use Case Diagram for User Module with GIS services

All three users will require navigational services. The GIS service module handles the relay of geographic data between users and the server. This service, for the extended function of internal building mapping, would require blueprints/maps of the buildings to be uploaded and mapped. The "Find point of interest" use case is described fully by the "Point of Interest" use case diagram.

The user interacts simply with the fitness module which corresponds in turn with the Notifications Module. Guest users are only allowed to view the leader board but are however not able to track their own progress.

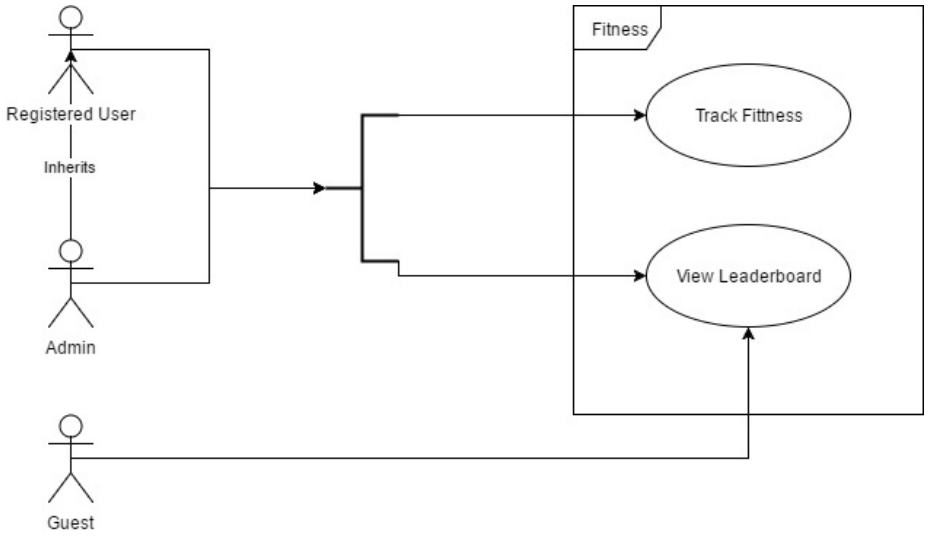

Figure 4: Use Case Diagram for the Fitness Module

5.1.2 Notifications Module

Scope

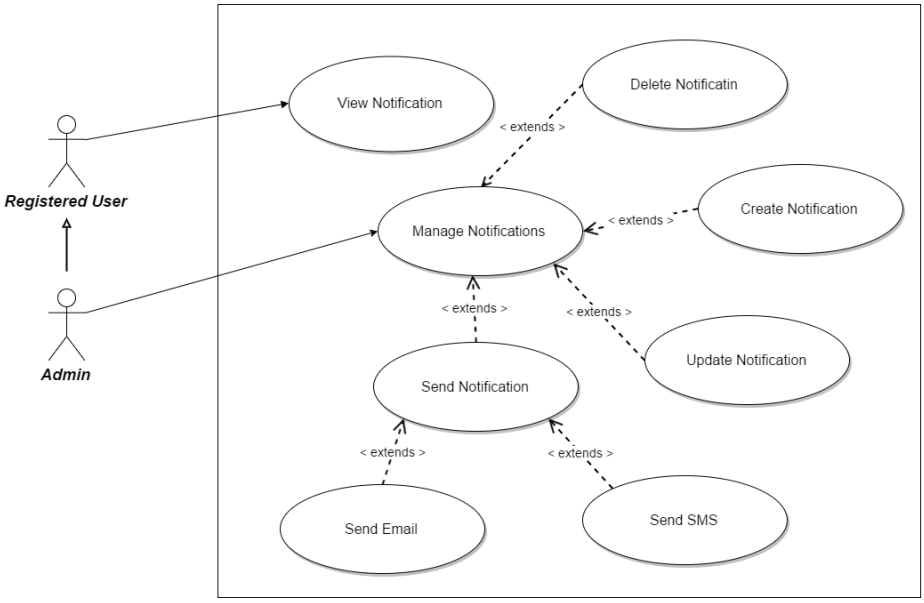

Figure 5: Notification Module Use-Case Diagram

The notification Module will provide end-users notifications with regards to the different aspects of the system, such as system updates or events taking place. Both subsystems as well as administrative users will use the notification system. The notification subsystem will send notifications either via email or SMS, depending on the end-user's noted preference.

Requirements, Interfaces and Constraints

The system is required to be light weight and must be easily interfaced with to allow any subsystem that needs the ability to send a user a notification the ability to do so. It thus needs to be extremely modular

The Notification Module will need to interface with the user module, as it will need access to different users and their contact details as well as their preference methods of notification. The module will also need to provide a simple interface, as mentioned above, for simple messaging through the system. The Notification subsystem will also need to interface with an external browser or email client in order to send email to the UP email server so that those emails may in turn be forwarded to the end-user and/or an email to SMS gateway that will SMS the end-user.

The Notification Module will need to be able to built message all users, certain groups of users and specific users. The Notification Module will also need to be able to send time scheduled messages and as such be able to update said notifications or update them. Thus notifications must be retrievable.

Since the Notification Module does not host its own in application messaging system, the subsystem is constrained to interfacing with web based methods of communication through the use of email and gateways. This means that their is no way for the notification subsystem to know if the end-user has indeed reviewed their notification and thus can not create or send reminder notifications. There is also lot less control over the presentation and user experience of the notifications as they will have to conform to a standardized means of test presentation. However this means the system is easier to implement as a lot of leg work has already been done with regards to sending emails from software packages and frees developers up from creating a in application based notification system from scratch.

Notification Module Structure

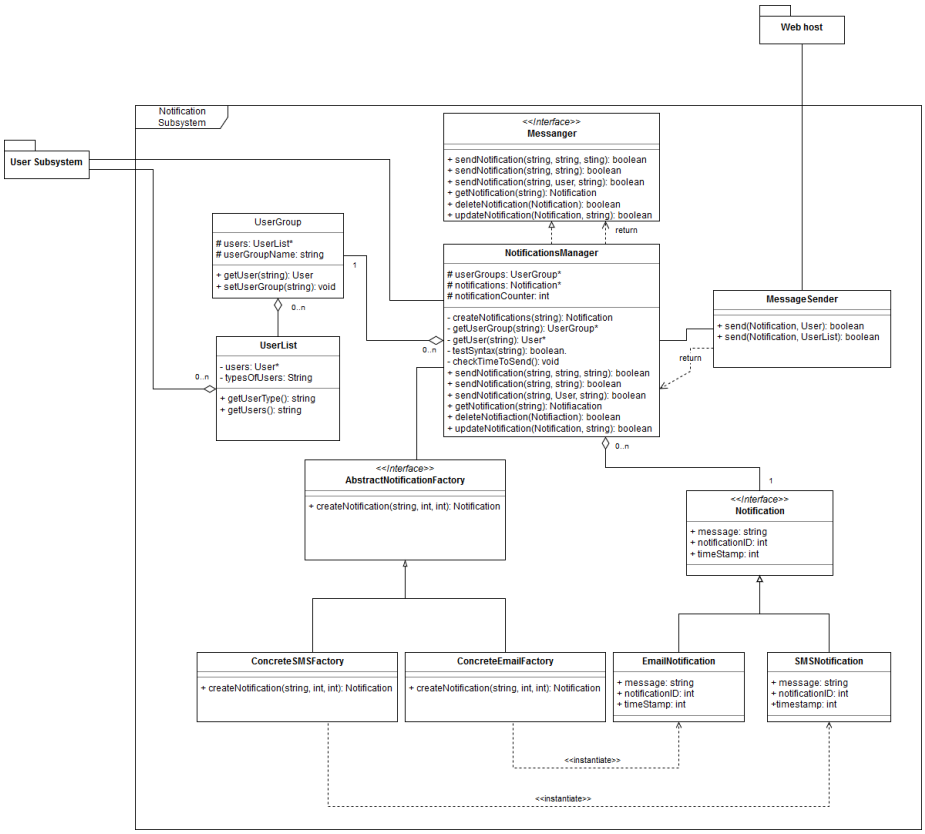

Figure 6: Notification Module Class Diagram

The Notification Module will provide a simple interface through which most communication and interaction with the subsystem will be done. This ensures a highly modular design. This interface allows for easy message creation by creating notification with various overloaded functions. It also allows for notification retrieval, updating and deletion.

The NotificationsManager class implements the interface and provides the administrative functionality required to send notifications. It interfaces with the User Module to get users for their details by either getting users or lists of users, called user groups. Updates notifications and retrieves them from a stored list. It also ensures that notification are sent when they need to go out. Furthermore it tests the syntax structure of the message it is required to send before handing it off to the abstract factory class AbstractNotifcationFactory.

The AbstractNotificationFactory class is an interface that allows access to two concrete factories that implement a different concrete type of the Notification class. one being a email factory and the other being a SMS factory. Once these different kinds of concrete Notification types have been created they are handed back to the NotificationsManger, which stores them for when they need to be sent.

Using dependency injection, the NotificationsManager hands off the required notifications and users to the MessageSender class who interfaces with the required web technologies and handles the sending of notifications to each or groups of end-user.

5.1.3 Fitness Module

Scope

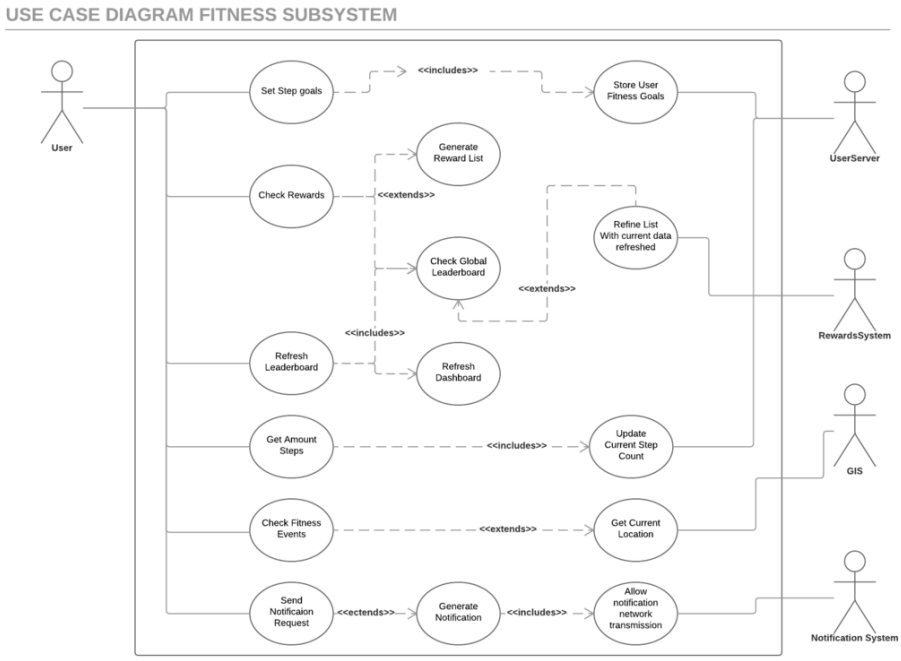

Figure 7: Fitness Module Use-Case Diagram

This system will be responsible for the fitness information and tracking set out by the user. Their movement will be tracked and the app will help the user live a fit lifestyle on campus by allowing them to set fitness goals.

Fitness Module Structure

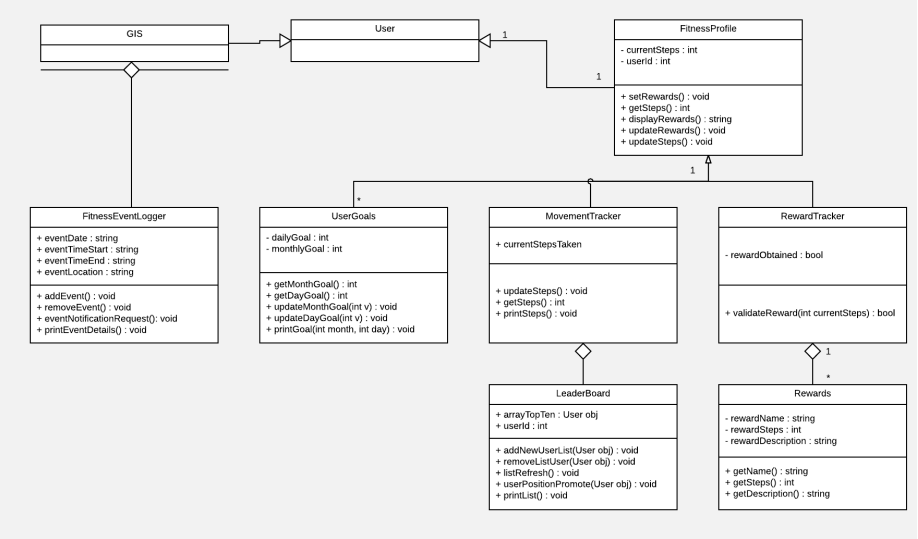

Figure 8: Fitness Module Class Diagram

The fitness subsystem is responsible for the handling of all fitness aspects of the application. It makes use of the GIS, Notification and User systems to form the fitness system.

The system gives the user the ability to set fitness goals and monitor the progress that the user is making in regards to this. It will also link into the Points of Interest System in order to give the user fitness events that will take them to special locations on campus!

The users of the app will also be exposed to a top ten global list (leaderboard) to show the top ten participants with the most steps taken. This will act as a method to encourage users to reach the fitness goals that they set as well as urge on competitive behaviour

The technologies used is the GIS to track the location of the user as well as the User profile in order to track the progress of the fitness goals that have been set for each user.

5.1.4 Point Of Interest Module

Scope

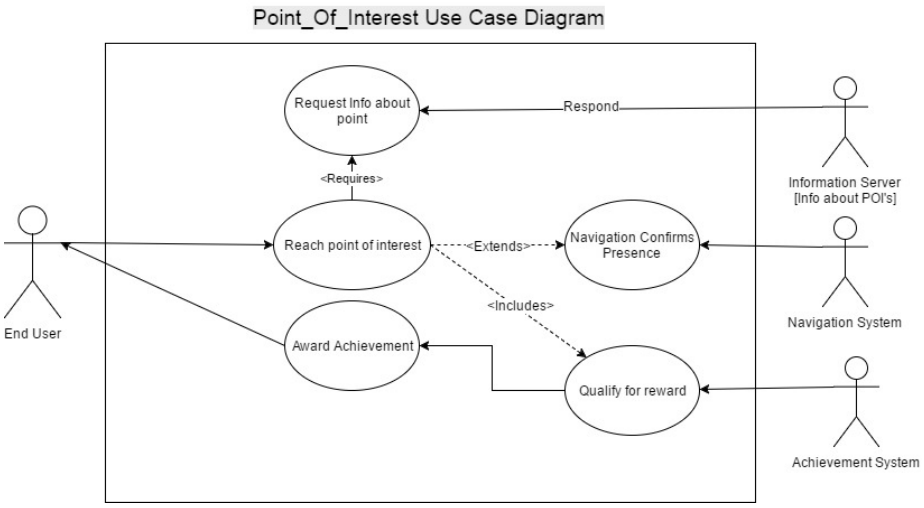

Figure 9: Point of interest module use case diagram

Point of interest Module Structure

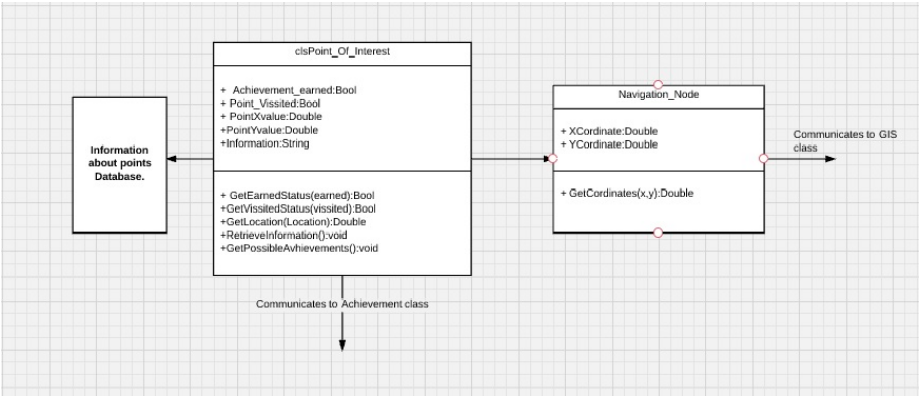

Figure 10: Point of interest module class diagram

The point of interest module will be used to give an interface that will allow users to visit points of interest on campus and view information about the var-

ious points of interest. There are only 2 classes in this module, and those are the clsPoint_of_interest and the navigation_Node. The clsPoint_of_Interest will communicate with 3 sections, the database that contains all of the information about the particular point, and also with the Navigation node and lastly with the Achievements classes.

The Navigation_Node is used to retrieve the user's current location and match it with the point of interest's location to determine if the user is currently at the point. The Navigation_Node retrieves the user's location by communicating with the GIS and receiving the location. When the location is retrieved, it is requested by the clsPoint_of_Interest and then the information about the current point can be retrieved and displayed. The Point_Of_Interest needs to communicate with the achievement classes in order to determine if a user qualifies for an achievement or not

5.2 Physical view

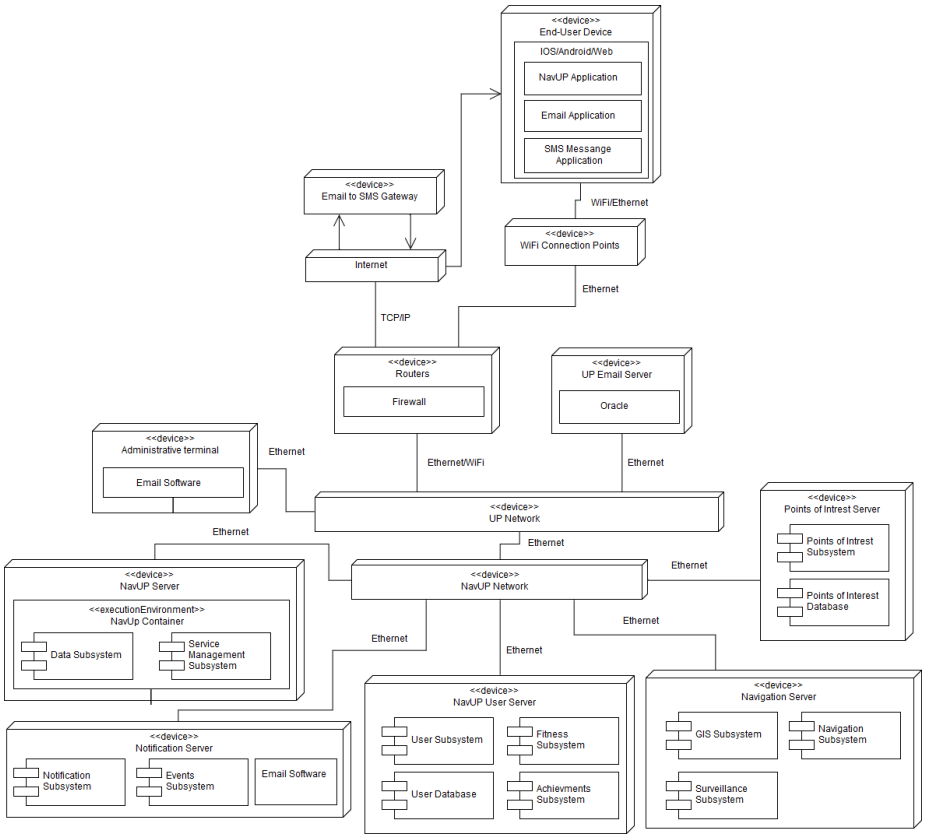

Figure 11: Notification Module Class Diagram

The NavUp system will primarily exist on its own network, but interfaces with the University of Pretoria's network at large to send and receive its data. Each subsystem will exist on its own server (virtual or physical) that will deal with incoming requests.

This splitting of the system into several servers will ensure that no server is bogged down by a lack of resources or access time when dealing with the masses of concurrent end users. Thus the GIS server will be able to service users while the fitness server or navigation server periodically retrieve information from them on request, rather then having the server slow down under huge loads a end users try access GIS services and others their Fitness data, and others still update their user accounts.

Subsystems that are closely linked will run on the same server and will only re-

quire each server accessing another server when certain information is required, else each server will be able to directly service the end-user without putting further strain on the other servers servicing other end-users.

As the NavUp sub-network is connected to the UP network, administrative duties will be possible from an administrative terminal. The access to the UP network will also provide the NavUp network access to the UP Email Server to provide notification functionality which will then travel back through the network, out through the router and fire wall and to a gateway for emails and emails to SMS's and to the end-users device.

The end users device will primarily have access to the NavUp system on campus through it's connection to WiFi Connection Points, that are found all over campus. These points will be utilised through the NavUp application and sent to the NavUp Server through the UP router, firewall and network to calculate GIS information. Similar methods and functions will be used to give the user real time access to the navigational subsystem and points of interest subsystem.////The NavUp User Server will maintain and deal with requests regarding the various systems requests for user data, and thus require its own server as multiple sub servers will require the updating of this server constantly. An achievements subsystem will also need to be implemented to maintain a higher degree of modularity when implementing a rewards program for end-users based on their fitness tracking as well as their exploration of campus.