Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master's Thesis Nr. 0838-002

Concept and Implementation of Digital Beacons

Muhammad Bilal Chughtai

Course of Study:	INFOTECH
Examiner:	Prof. Dr. Frank Leymann
Supervisor:	Dr. Daniel Schleicher, Dipl.-Inf. Christoph Fehling
Commenced:	2015-02-02
Completed:	2015-08-11
CR-Classification:	H2.1, D2.11, E.2, J.4

Contents

List of Figures

List of Listings

1 Introduction

This chapter discusses the motivation and objectives of this thesis in sections 1.1 and 1.2, respectively. Some of the terminology used throughout this thesis is described in section 1.3. And an outline of the organization of this thesis is provided in section 1.4.

1.1 Motivation

A location-based service (LBS) is any information, entertainment, or social media service that is available on a mobile device and makes use of geographical position [Zah15]. People can use an LBS to find directions to a certain place, find restaurants nearby, locate friends, check local weather forecast and the list goes on. Their application areas range from business to public sector and consumer services.

Location-based services are growing rapidly due to various reasons including increase in smartphone adoption, improvement in positioning technologies and interest of commercial sector [Zic13, Mar14, Ber14]. Due to this growth, there is room for introducing new functionalities in location-based services or even introduce completely new forms of location-based services. One way to enable new functionalities and services is to make new forms of data available, which the developers can then use to implement new functionalities and services. Finding out what these data should be, is an important question.

The subject of a location-based service, the entities that people want to track, or navigate to, or find information about can be referred to *geographical entities*. Examples of geographical entities are hotel, restaurants, museums, parks, pedestrians or vehicles. Most of the location based services today only use location information to provide services. However, in addition to location, if further information is available about these entities, then location based services can extend the functionalities that they provide. This can also open door for new forms of location based services.

As an example of what this new form of data can be, consider a movie theater as a geographical entity. A person visiting this movie theater might want to go to eat something afterwards. So adding information about restaurants that are nearby can be a useful information for a location based service. Using this information, the location based service can suggest the restaurant to visit when the person leaves the movie theater. Another example of what information can be added, consider a convoy of vehicles that has to be monitored to stay together in a convoy management service. Suppose a threshold distance information can be added to this convoy which represents the region within which all vehicles in the convoy should remain. Using this threshold information a location based service can provide monitoring services to the owners of this convoy. With such a service, they would automatically be notified if a vehicle moves

out of the region that is specified. A similar requirement is also from a tour guide that wants to keep track of its followers and make sure that no one gets left behind.

This work describes a way of associating such new forms of data with geographical entities. A model that specifies these data is introduced in this work. Using this model, location based services can have more information about a geographical entity. They can add the required data with the entities and once this data is available, it can be used to provide various solutions.

1.2 Objectives

In this work, it is investigated that what types of data are useful to associate with a geographical entity. By considering requirements from various scenarios of real world, these data are determined and a model that represents these data is defined.

Furthermore, the scenarios in which this model is useful are identified and explained in detail. With the help of these scenarios, it is then explained how the defined model fits these scenarios and meets the requirement that are posed by these scenarios. The scenarios that are covered are taken from domains of consumer and enterprise services.

Lastly, an implementation of this model is described. The implementation covers a subset of the model. The goal of the implementation is to describe how location based services can use this model and implement various solutions.

The terminology that is used throughout this work is described next.

1.3 Terminology

To establish a common understanding of the terms that are used throughout this thesis, they are defined as follows.

Location-based service

A location-based service is any service that is enhanced with and depends on a mobile device's location, [VP02].

Location-based feature

A location-based feature is a specific functionality that depends on a mobile device's location.

Location-awareness

Location-awareness is the ability of a mobile device to determine its location.

Location determination technologies

Location determination technologies are various technologies that allow a mobile device to be aware of its location. These include Geographical Positioning System (GPS), Infrared and Ultrasonic proximity, Wi-Fi-based-, Cellular-based- and Radio Frequency ID-based positioning systems. Location determination technologies are also known as positioning technologies.

1.4 Outline

Following is an outline of how the rest of this thesis is structured.

Chapter 2 - Fundamentals

This chapter describes the necessary background required to follow the topics discussed in this thesis. The basics of location-based services and their different types will be discussed in detail.

Chapter 3 - Related Work

This chapter discusses the existing work related to this thesis and how it compares to the work done in this thesis.

Chapter 4 - Digital Beacon Concept

A model that satisfies the objectives of this thesis is described here in detail. Furthermore, use cases will be defined that demonstrate the suggested functionalities that are enabled by this model.

Chapter 5 - Implementation

This chapter describes an application architecture that uses the model presented in chapter 4. An implementation of this architecture in a smartphone application will be discussed. Lastly, the functionalities covered by this implementation described and discussed.

Chapter 6 - Summary

This chapter provides a summary of the contributions of this work.

2 Fundamentals

The work done is this thesis is related to location-based service. The basics of location-based services, their architecture and different types are discussed in this chapter.

2.1 Introduction to Location-Based Services

Location based services have been around for several years. The earliest consumer LBS were related to weather and traffic information and were introduced in 1999 [Bry14]. Today, there are a few areas that location based services are not a part of. Numerous enterprise, consumer and public safety services provide applications that are based on location. LBS can be considered to come under the research area of context-aware services that automatically adopt their behavior according to the physical environment of the user.

2.1.1 Definitions

To get an essence of how they are defined in the literature, some of their definitions are described below.

- "A wireless-IP service that uses geographical information to serve a mobile user or any application service that exploits the position of a mobile terminal." [Ppe15].
- "Location-based services (LBS) are the delivery of data and information services where the content of those services is tailored to the current or some projected location and context of a mobile user." [BL09]
- "Location-based services (LBS) are a general class of computer program-level services that use location data to control features." [Wik15c]

In our point of view, location based services are services that are dependent on location data to provide the core functionality as well as service that use location data to enhance the provided service.

2.1.2 Components

There are several components that contribute in providing a location-based service.

Positioning system: locates the mobile device either in an indoor or an outdoor environment. These include satellite-based (GPS), Cellular-based, RFID- based, Bluetooth and Wi-Fi based systems.

Communication Network: enables connectivity between the mobile device and the service provider.

Service provider: provider of the application or service itself.

Content Provider: stores and maintains information that may be used by the service provider. These include mapping, yellow pages and traffic information services.

Mobile Device: a portable device such as a smartphone, tablet PC, personal navigation device, laptop, or wearable devices (smartwatch or glasses).

2.1.3 Market Growth

The market of location-based services has grown rapidly in the past few years. This is due to the advances in positioning technologies, faster wireless networks and high availability of mobile devices. Figure 2.1 shows the forecasted growth in LBS.

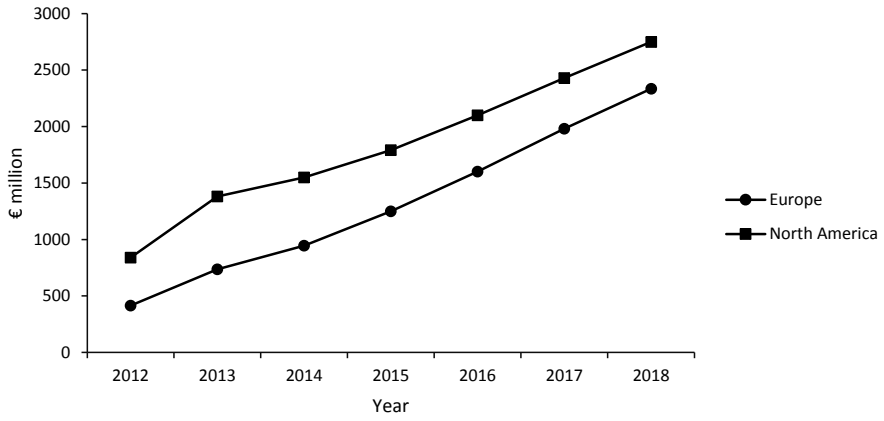


Figure 2.1: Growth in LBS revenues [Ber14]

Hence there is no doubt that location based services will continue to play an important role in our lives in future and therefore providing mechanism to enhance the capabilities of these service is equally important.

2.2 Categories

There are several ways to categorize location-based service. In terms of functionality, following eight categorizes can be described.

Navigation

Navigation is one of the most popular segment of location-based services. Navigation services allows users to find directions to a desired place. In addition to use by vehicle drivers, navigation services are also used by pedestrians.

Local search and information

Location data can be added to any type of digital media, or any information available on the Web. This allows services to provide content that is nearby to the current location of the user. Using such services user can get information about nearby places, weather forecast, real-time traffic information or browse through other location-tagged information available on the web. Some of the popular services in this sector are Foursquare, Facebook Places, Gas Buddy, Yelp and Google Maps.

Entertainment

There is a wide range of location based games including scavenger hunt, treasure hunts and role playing games that are growing popular among users. These applications make good use of location features and can be quite engaging for users.

Location-based social networking

This sector of location-based services has grown rapidly during the last few years and together with the entertainment sector, has the largest number of active users [Ber14]. Social networking platforms including Facebook, Twitter and Google Plus as well as new applications such as Swarm[1] provide features that allow users to share their location with their friends, get information about who has been to a certain place, or even who is currently nearby in real-time.

Fitness

Users always carry their smartphones with them. This allows fitness application to monitor the users' daily activities and suggest them diet plans and remind them about routinely doing exercises. These application are growing and some of the popular examples are Nokia Sports Tracker, Nike+, Run Keeper and Endomondo.

Family and people tracking

Family and people tracking services can be quite useful for certain scenarios. Such services, can be helpful to conveniently inform hosts about arrival of guests. Specially with restrictions on using mobile phones while driving, this can be a quite useful feature. Moreover, parents can keep track of their children and vice versa.

Location-based marketing

Location-based marketing is growing quite popular due to the ubiquitous nature of mobile devices. Based on user's preferences and his location history, these services analyze and provide relevant advertisements to the users when they are nearby.

[1]Swarm is an application by Foursquare which is dedicated for check-in feature

Other enterprise and Public services

There are several other enterprise and public services that take advantage of location based features. These include mobile fleet management, emergency services and billing.

2.3 Layers of a Location Based Service

The working of a location based service can be described using three layers. A positioning layer, a middleware layer and an application layer as shown in Figure 2.2.

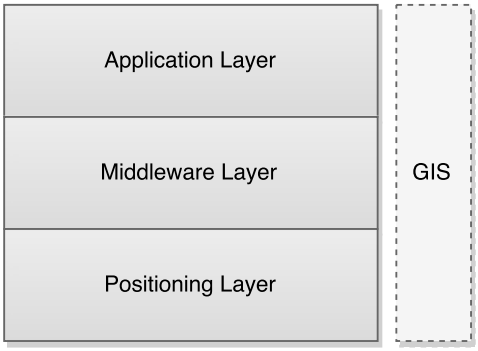


Figure 2.2: Layers of a Location Based Service

The positioning layer consists of several positioning technologies[2] which together with geospatial data held in geographical information system (GIS) provide location of the mobile device. Using the different positioning technologies in an efficient manner is not a trivial task. Usually it involves considering various aspects such as checking the availability of the technologies, monitoring their varying accuracy and performing a trade-off between the accuracy and power consumption. Hence to ease the process of application development mobile platform providers have developed location services as middleware. Location services make efficient use of different underlying positioning technologies and provide location information as output that is ready to be used by the application layer.

The work done in this thesis relates to the application layer of location based services.

2.4 Adoption Barriers

There are certain concerns related to location-based services. The most important ones are privacy and security, which can lead to hesitation in using location based services.

Mechanisms such as location obfuscation, cryptography and pseudonyms can be used to rectify such issues. The relation of work done in this thesis to these mechanism will be discussed briefly later on, however, privacy and security concerns are not a focus of this work.

[2]The positioning technologies include GPS, Cell ID, Wi-Fi, Bluetooth and inertial sensors of the device

3 Related Work

The work done in this thesis is related to defining what data should be associated with a geographical entity. A geographical entity is any geographical place or a moving object that is of interest to user of a location based service. A restaurant, hotel, lake, tourist attraction place can be a geographical entity that has a fixed location. Pedestrians and vehicles or other moving objects can also be geographical entity that have dynamic location. Geographical entities are digitally represented using some data structures in order to be used by location based services and provide functionality. For example, a place recommendation services may recommend nearby restaurant to a user. The restaurant is a geographical entity. In order for the LBS to find this restaurant, it had to identify it. Identification in case of fixed location geographical entity are usually done using latitude and longitude coordinates.

This section describes existing work related to representing location in a location based service and how application current use this information.

3.1 GeoLife

In this work a service called GeoLife is presented [ZXM13]. This service examines the trajectories of how users places that users visit. Using this information, the service suggest the people about other people that carry out similar activities. The goal here is to suggest friends. Furthermore, it gives people traveling recommendations to the users based on activities of other people.

This work contributes toward analysis on the user activities. However, it does not provide a way of representing this data for a location based service and how it can be used.

3.2 Geotags

A geotag is a geographical identification metadata that can be added to digital content such as images, videos, SMS and QR codes[WIK15b]. Geotags are also used to share locations in social networking platforms. The term used for adding a geotag in social networking is check-in.

Data Structure Geotag data usually consists of latitude and longitude coordinates, they can also contain altitude, bearing, distance, accuracy data, and place names.

Applications Geotagging has numerous applications. It is the core enabler of various LBS applications. Local search, location based gaming, location based social networking are all enabled by geotag.

Limitations Geotags form a basis of most of the location based services today. However, the data in a geotag is limited and it can only be used for fixed location geographical entities.

3.3 Geofence

"Geofence is a virtual perimeter for a real-world geographic area" [Wik15a]. Geofences are usually used for monitoring when someone enters or leaves a certain geographical region.

Data Structure Although the definition of a geofence uses term perimeter, which can have any shape. However, in most implementations, a geofence consists of latitude and longitude coordinates together with a range data.

Applications Geofences have applications in security, asset monitoring, people monitoring, employee attendance logging, and collecting consumer activity data for marketing analysis.

Geofences have particular importance in marketing services. Business can get data about the frequency of visits to a certain place and this way they can decide about importance of a specific place. For example, a business looking for opening his new branch at a new area can monitor the activity of people. Based on some additional information about interests of people who visit certain places, they can analyze potential areas where the business will have more advantage over other places. Or existing businesses can also use the data to analyze where it will be beneficial to notify users about advertisements.

Limitations Geofence is another important enabler in LBS. However defining some context data, such as relationship with another POI, is not accommodated in geofences.

3.4 Current state of Location Based Applications

This section describes the state of how applications today handle location. An analysis is shown in Fig. 3.1.

Hence it can be seen from the discussion that there is limited work that focuses on a general approach of representing data for a location based service.

Category	Service Name	Description	Comparison to Digital Beacon Concept
Marketing	Yelp		
Marketing	Foursquare	Rating of places and sharing experience of different places (recreational, business and whatnot)	-Not constrained to a range -Main usage is to get reviews on places
Hyperlocal news	Patch.com	-A service that informs locals about the latest news in their neighborhood.	-It's a local news service -Does not aim on a public usage of services
Social Media	Shout	An emerging service that allow sharing images in defined regions	-Limited to only regions -No further data can be associated with geographical entities
Social Media	Qork	-Allows users to post notes, images on a specific location -Upvoting system which is intended to promote positive use -Creators have intent to grow into hyperlocal ads business	-It is only focused on sharing multimedia content -Does not allow specifying categories or relationships

Figure 3.1: Current state of Location Based Applications

4 Digital Beacon Concept

This chapter introduces a concept of digital beacons. This concept is useful to describe information about geographical entities such as pedestrians, vehicles, hotels, shopping malls or other places of interest and enables location based services including navigation, tracking and social networking. The chapter is organized as follows:

First, *section 4.1* defines the geographical entities and describes what the digital beacon concept is. Afterwards, *section 4.2* introduces a model that is defined in this work to describe a geographical entity. The model consists of different types of data. The purpose and example uses of each type of data is provided.

After defining the digital beacon model, *section 4.3* discusses real-world scenarios where the digital beacon model can be useful. These scenarios can be, for example, a group of people wanting to travel together and keep track of each other while traveling; or a business person who wants to target a certain geographical region for advertisements. To show how the digital beacon model fits the requirement of a scenario, the functional requirements of these scenarios are first pointed out. And then, it is discussed how an application can use digital beacon model to implement the required functionalities.

4.1 Description and Goal

This section first defines the term geographical entities, then a description and goal of the digital beacon concept is provided.

Geographical Entities Within the scope of this work, a geographical entity is defined as any place of interest on earth; or any moving object on earth which is capable of communicating its location with a computer program over a network. Furthermore, a location based service should be interested in using information about the geographical entity to provide some functionality to a user.

Examples of geographical entities include public places such as shopping malls, restaurants, hotel and tourist area; or pedestrians and vehicle that carry a navigation device.

Digital Beacon Concept To use geographical entities in a location based service, information about them has to be known to the service. The interesting information about a geographical entity can be for example, a geographical region within which it holds relevance; or a relationship that it has with other geographical entities. For example, the *surrounding area* of a shopping mall holds relevance to the shopping mall[Sor14]. This means that advertisements can be sent about the products in the shopping mall to people that are within the surrounding area. Continuing further with this example, this

shopping mall can be related to another similar shopping mall that is situated within a walking distance of a buyer. An advertisement service, if it has this relationship information, can present advertisements to people in the relevance region of that shopping mall as well.

Describing what information should be known about a geographical entity, and how it can be stored, this is what the digital beacon concept describes. Hence the goal of digital beacon concept is to provide a model that can represent information about geographical entities so that location based services can use that information to provide various functionalities. The model which is developed in this work to meet this goal is described next.

4.2 Digital Beacon Model

Digital beacon model consists of a set of metadata that identifies a geographical entity, describes a context of the geographical entity and delivers some multimedia information about the geographical entity. In addition to this, it also specifies where, when and to whom the information about the geographical entity should be delivered. Taking all of this into account the metadata of digital beacon model, called its properties, can be classified into following four groups.

Identification

There are two properties that belong to this category namely, Location and Beacon ID. Location identifies the digital beacon with respect to its location and the beacon ID identifies it irrespective of the location. The later is useful for identifying moving geographical entities.

Visibility Control

These properties provide ways to control when, where and to whom the multimedia contents associated with digital beacon are accessible. The specific properties that fall into this category are Perimeter, Lifetime and VisibilityGroup.

Context

Tag and relationship properties provide a way to describe the context of a geographical entity that the digital beacon is associated with. Furthermore, Followers also belong to this category.

Contents

Contents are the properties that contain references to the multimedia information that the digital beacon should present.

Figure 4.1 shows a class diagram of the digital beacon model. In this figure, the properties that are essential to a digital beacon are mentioned in one class and the

properties that are optional are shown in additional classes. A composition relationship between these classes is shown to indicate that a digital beacon can be comprised of the additional classes.

A detailed explanation of each of the properties of the digital beacon and how they provide various features in location based services are explained. First, the beacon ID property belonging to the identification group is discussed.

4.2.1 Beacon ID

Purpose:

A beacon ID uniquely identifies a digital beacon and allows location based services to get data from it.

Description:

A unique identifier is essential for various purposes. It provides a way to refer to a digital beacon when it can not be located using location data.

Uses:

In location based services that allow tracking of a moving geographical entity, a beacon ID allows a way to get updated location of the geographical entity from the associated digital beacon. Other features that are enabled by beacon ID include subscribing to digital beacon and specifying relationships between various digital beacons.

4.2.2 Location

Purpose:

Location property provides information about the current location of the geographical entity that the digital beacon is representing.

Description:

Location can be a set a of latitude and longitude values. For static-location geographical entities, location may be set only once. However, for dynamic-location geographical entities, location data has to be updated continuously. The frequency with which it has to be updated depends on the requirements of the location based service under consideration. It can depend on either a time interval and updated each time the time interval expires; or on a threshold of change in the location and updated only if the geographical entity moves a certain distance.

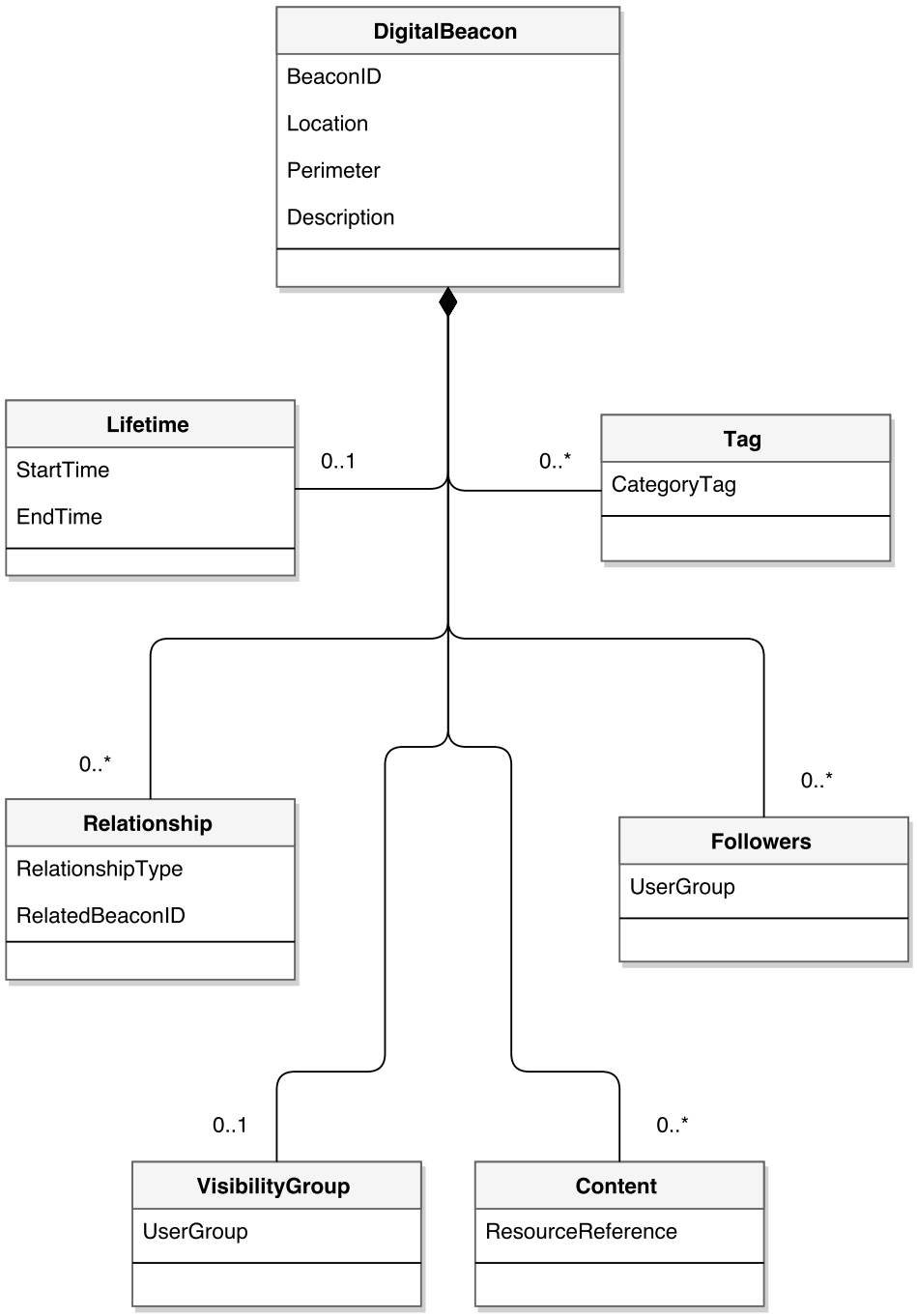


Figure 4.1: Digital Beacon Model

Uses:

By using location data, location based services can find out the digital beacons that are nearby to a user and hence present the multimedia information associated with the digital beacon.

Location is the main driver behind an LBS and hence it is also an essential part of the digital beacon model. However, a digital beacon does not take into account only the location data to present multimedia information. There are visibility aspects such as Perimeter, Lifetime and Visibility Group that are also taken into account.

4.2.3 Perimeter

Purpose:

Perimeter describes a geographical area that is covered by a digital beacon. It is the *where* part of the visibility control data.

Description:

To present the multimedia information that is contained in a digital beacon, perimeter plays an important role. The perimeter can be of any arbitrary shape. However, an arbitrary shape would make sense only for static location geographical entities such as parks, lakes, building, etc.

Uses:

Using the perimeter property, location based services can control where the information contained in digital beacon should be presented. Giving the control to the service about where the information should be presented has various advantages. For example in a location based advertisement service, by using a perimeter the advertiser has more control over the region that they want to target for advertisement. And hence they have a better option of controlling their reach to the customers.

It is important to note here the significance of using perimeter to specify boundary of a digital beacon instead of a simple circular boundary. This is useful to cover a geographical entity where the boundaries of its area are important. For example, consider a private fair held at an area that has a long shape. The participants want to share information about their offerings only with the attendees and not with the people that are nearby to the fair location. A small circular digital beacon would not cover all of the area. Or by expanding the size, people outside the fair would also be able to see the contents of the digital beacon. In such cases a digital beacon that can adopt to the shape of the area is useful to meet the requirement of the user.

Furthermore, the perimeter property of a digital beacon can also be used to collect consumer activity data in a certain region[Sha14]. This activity data can be useful for businesses to identify regions of interest for their business. Other examples where perimeter data of a digital beacon is useful include location-based social networking, traveler's group monitoring, asset tracking and fleet management services.

4.2.4 Lifetime

Purpose:

Lifetime consists of conditions that describes *when* the digital beacon is visible.

Description:

The data contained in this property can be used to specify a time duration within which the digital beacon and its contents can be accessed. Lifetime is an optional property of a digital beacon as shown in Fig. 4.1 using multiplicity of 0 or 1.

Uses:

This property can be used to specify a temporal association of digital beacon with a geographical entity. This is useful in various types of location-based services. For example consider a service which allows users to share a meeting location. Using a lifetime, this service can allow users to specify the time duration for which the meeting location is shared. Furthermore, this property also uses in location-based games, for example, a time-limited treasure hunt game. Or it can also be used in location-based marketing services to publish limited time offers.

4.2.5 VisibilityGroup

Purpose:

VisibilityGroup is to specify a groups of users of a location based service to *whom* the contents of the digital beacon are visible.

Description:

In location based services that have a concept of private sharing of information, this property of digital beacon can be used to specify users that can see a digital beacon. This is also an optional property of digital beacon.

Uses:

This property has uses in various services consumer and business applications. Examples of cases where this property can be useful are: sharing current location with friends, private sharing of multimedia information in a social networking service and publishing information in a trade fair with only the registered attendees.

4.2.6 Relationships

Purpose:

Relationship property provides a way to specify a relationship between two or more geographical entities.

Description:

This property consists of a beacon identifier that refers to a related beacon and also specifies what is the type of relationship as shown in Fig. 4.1. The type of relationship depends on the requirements of the location based service. The relationship type can convey information such as next places to visit, nearby related points of interest or list of places that belong to a city tour.

Uses:

Geographical entities are often related to each other. For example information about a popular restaurant nearby a tourist attraction place might be valuable for a place recommendation or an advertisement service. This is because the behavior of users visiting a tourist place can be that they want to look for a place to eat after their tour. Hence using relationship property, this information about a relation between various geographical entities can be captured and services can recommend to their users nearby places to visit.

Relationship data can also be used in tourism services to link together a group of places and recommend tours to users. Another use of relationship property is in location-based gaming and entertainment services. For example a treasure map having clues at different places that lead to discovery of the treasure can be described using a collection of digital beacons that are related to each other.

4.2.7 Tags

Purpose:

Tags are useful to specify a certain category or topic with which a geographical entity belongs.

Description:

A tag can be a simple string or a unique classification ID that specifies a category of geographical entity within a location based service.

Uses:

Advertisement services can tag different geographical entities to classify them into a certain category. This can be used to identify which category of places a user frequently visits. Furthermore, by having a way of classifying geographical entities, location based services can ask users to specify which categories of the geographical entities they are interested in. Once this is known, the advertisement service can send notification to a user to checkout a type of nearby place that he previously showed interest in.

4.2.8 Followers

Purpose:

Groups of people interested in a particular geographical entity can be specified as Followers.

Description:

Followers can be a list of user IDs that have explicitly specified interest in a geographical entity, or the interest is inferred from the user history of visits to a geographical entity.

Uses:

Location based services can not notify users about every geographical entity with a digital beacon that is nearby. In order to capture the interest of users and notify them only about the entities that they are interested in, followers property can be used. This can be inferred automatically based on a user history or it can be explicitly mentioned by a user[BNPZ]. For example users search or visit history can indicate a particular type of geographical entities that they are interested in. When a location based service detects such a behavior, they can save the user as follower of that category of geographical entity. After having this information, the location based service can then can notify the user whenever they are in a perimeter of any geographical entity of that category.

Further uses of followers property can be in social networking services to notify a user if a friend is nearby.

4.2.9 Content

Purpose:

References to the multimedia information or other resources related to a geographical entity can be stored in content property.

Description:

The resource references that could be added can be of any type depending on the requirements of the location based service. Examples include images, videos, or also webpage references.

Uses :

Content property of a digital beacon is used to associate multimedia information with a geographical entity. Advertisement companies can have their advertisements added to the digital beacon to present the content to a user whenever the user is nearby. Furthermore, Local community based services, can allow their users to share information with other locals using content property of digital beacons. An example of this would be a local discussion topic which can be seen by anyone who is nearby a particular region.

This can be useful in making local discussion topic. Or a digital version of community boards on which people share information with other people of the community.

Hence, by using these properties of a digital beacon, location based services can extend their functionalities or new location based services can be implemented. To demonstrate how these properties of a digital beacon can play a role in location based services, next section describes four application example scenarios that are covered by the digital beacon model.

4.3 Applications of Digital Beacon Model

A number of location based services can use the digital beacon concept introduced in the previous section. The applications range from business to consumer and public services. In addition to having geographical information about entities, a digital beacon combines information that enables a wide range of features. To demonstrate these features, four different scenarios are described in the following sections. These scenarios present various functional requirements for a location based service. The requirements are described using use case diagrams. Each of the use case diagram is discussed in detail and it is shown how the digital beacon meets the indicated requirements.

4.3.1 Meeting Organizer Service

A service provides a way to communicate meeting details with others nearby. These details can be only seen by people in a specified area and provides a way to keep them updated about the plans of the meeting event.

Driving Question:

How can a service allow users to share meeting location and details while also giving them an ability to control where, when and to whom the information is shared? Also how can it dynamically reflect any changes to details of the meeting?

Scenario:

To discuss the functional requirements, consider a crowded social festival. A group of people want to plan in advance to meet in the festival at a specific spot. They have shared the location with their friends but at time of the meet-up, the meeting spot and other shared detail might get changed. The organizers wants their friends to get together at the spot that they will be finally at. Furthermore, they also want to have the flexibility to change any detail of the meet up and update others automatically about the changes. One more option they want is to be flexible on who is allowed to join them. They may want to make the meet-up public and anyone who is nearby, can join. Or they may want to make it private and only invited friends can join and see the meeting and its details.

Functional Requirements:

The functional requirements from a service that wants to cover this scenario can be summarized as shown in Fig. 4.2.

Actors: This use case diagram has two types of actors: *Organizer* and *Attendee*. Organizer is the person who sets up all the details of the meeting. An attendee is a person who is allowed to join the meeting.

Description of Use Cases: The organizer *creates a meeting* and *enters details* such as name and description. As a required step, the organizer also *enters location and discovery region* of the meeting. Location indicates the exact spot where meeting is held and discovery region indicates the geographical area within which the details of the meeting can be seen. Another piece of information to be entered to complete the creation step is *meeting start and end time*. This conveys to the attendees that how long the meeting is planned and it also tells the system for how long the details of the meeting should be made available. As an optional step, organizer can also *make the meeting private* and *specify people* that they want to invite. The meeting creation process is completed once all of the required information is entered.

The main focus of this system is to provide a guide during the event itself and not ahead of time. Hence communicating the meeting details with attendees ahead of time is not a requirement of the system. The organizer can create the meeting in advance only as a convenience. The service does not communicate this information with attendees until the actual meeting time.

At the time of the meeting, the meeting details are made available. Using the service interested people can *view meetings* being held nearby their current location. They can *select* the one they are interested in joining and by doing this, they subscribe to receive updates about the meeting. If the organizer changes any detail of the meeting, the subscribers get a notification about the updated information. After selecting the meeting, the attendees can use the *navigation* option provided by the service to navigate to the exact location of the meeting.

Role of Digital Beacons:

Each meeting can be represented by a digital beacon. The location of the meeting can be seen as the geographical entity in this case. The properties of the digital beacon involved in this use case and their roles are described as follows:

- **Location:** The current location where the meeting is being held
- **Perimeter:** Region Boundaries where the meeting can be discovered
- **Lifetime:** Start and End time of the meeting
- **Visibility:** A list of reference of users who are allowed to see the meeting details, in case of a private meeting

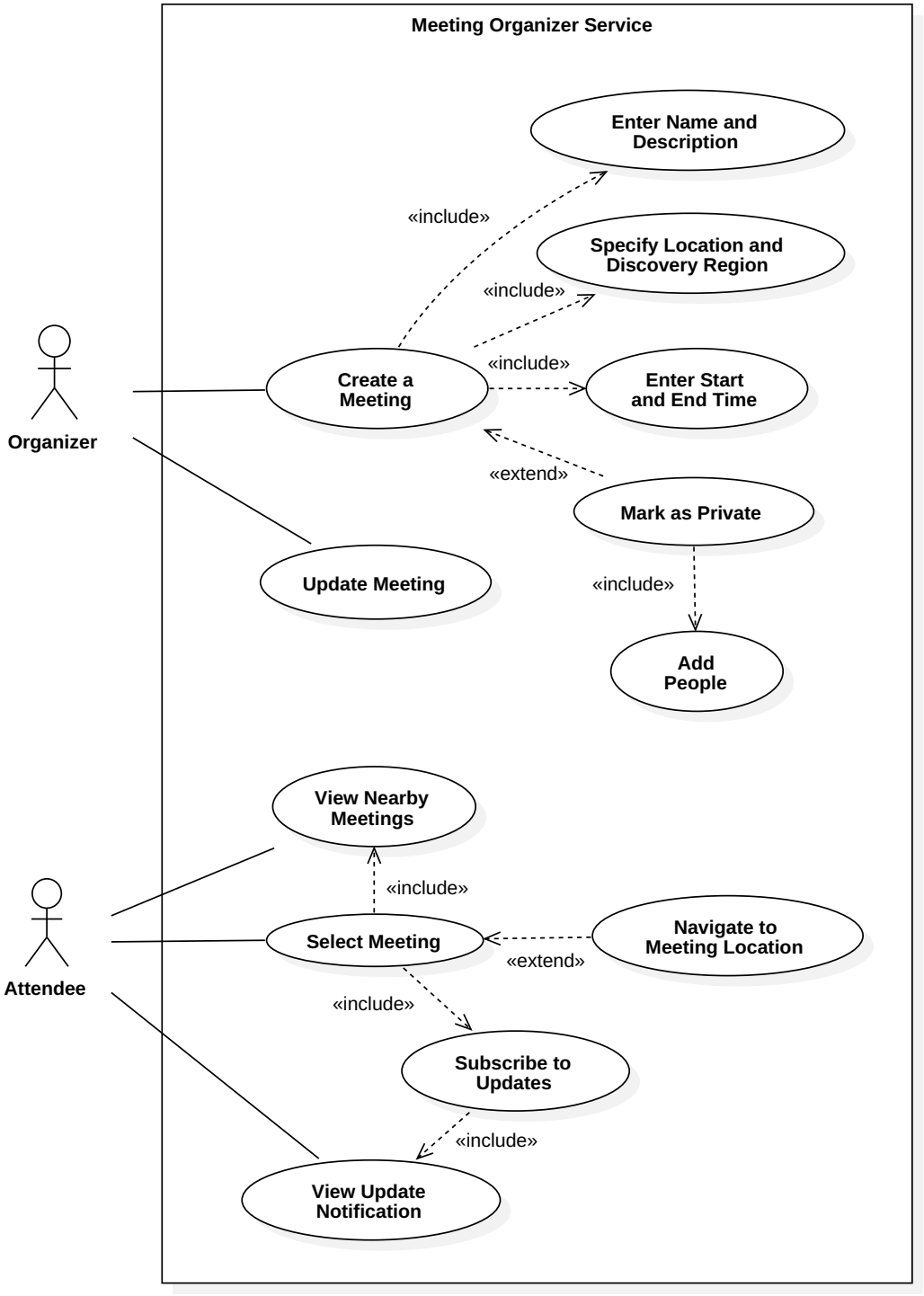


Figure 4.2: Meeting Organizer Service - Use Case Diagram

- **Followers:** A list of reference of users who have subscribed to the meeting and want to be notified about the updates
- **Description:** The name and description of the meeting

Using these set of properties, the digital beacon can be used to implement a service that provides the functionality that were laid out in Fig. 4.2

4.3.2 Group Tracking Service

A service lets users keep track of other while traveling in a group and notifies if a user gets left behind.

Driving Question:

How can a location based service let people track each other? How can a location based service assist in managing a group of travelers?

Scenario:

When people travel in a group they usually want to keep track of fellow group members. Such a group can be friends on an excursion trip, people taking tours in museums or children seeing sites in a city with a travel guide. These people may not always want to stay close to each other and may want to travel freely within a certain distance. For example a group of friends shopping together might want to go at different places and later get back together. Likewise, a group of school kids taking a city tour might want to stop by at a certain place for a longer time than the rest of the group. Despite getting separate, the people in the group usually do not want to be left far behind the group in order to avoid any problems. For example the schedule of a group might suffer (e.g. miss a train or bus, etc.) because of a fellow member that got left behind. Having a way to be alerted about someone getting left behind can allow the group members to react in time.

Functional Requirements:

The functional requirements from a service that wants to cover this scenario can be summarized as shown in Fig. 4.3

Actors: Two types of actors can interact with the service described in the Fig 4.3: A *Guide* who creates the group and specifies its details and a *Follower* who is interested in joining the group.

Description of Use Cases: In order to allow people to travel together and see each other's location, a guide *creates a group.* The process of creating a group involves *specifying a range of the group.* Furthermore, the location of the guide is used as a reference for the location of the group this is done automatically by the service and is shown in use case diagram as *set own location as reference* use case. Using this reference location and the group range, the service determines the distance within which followers

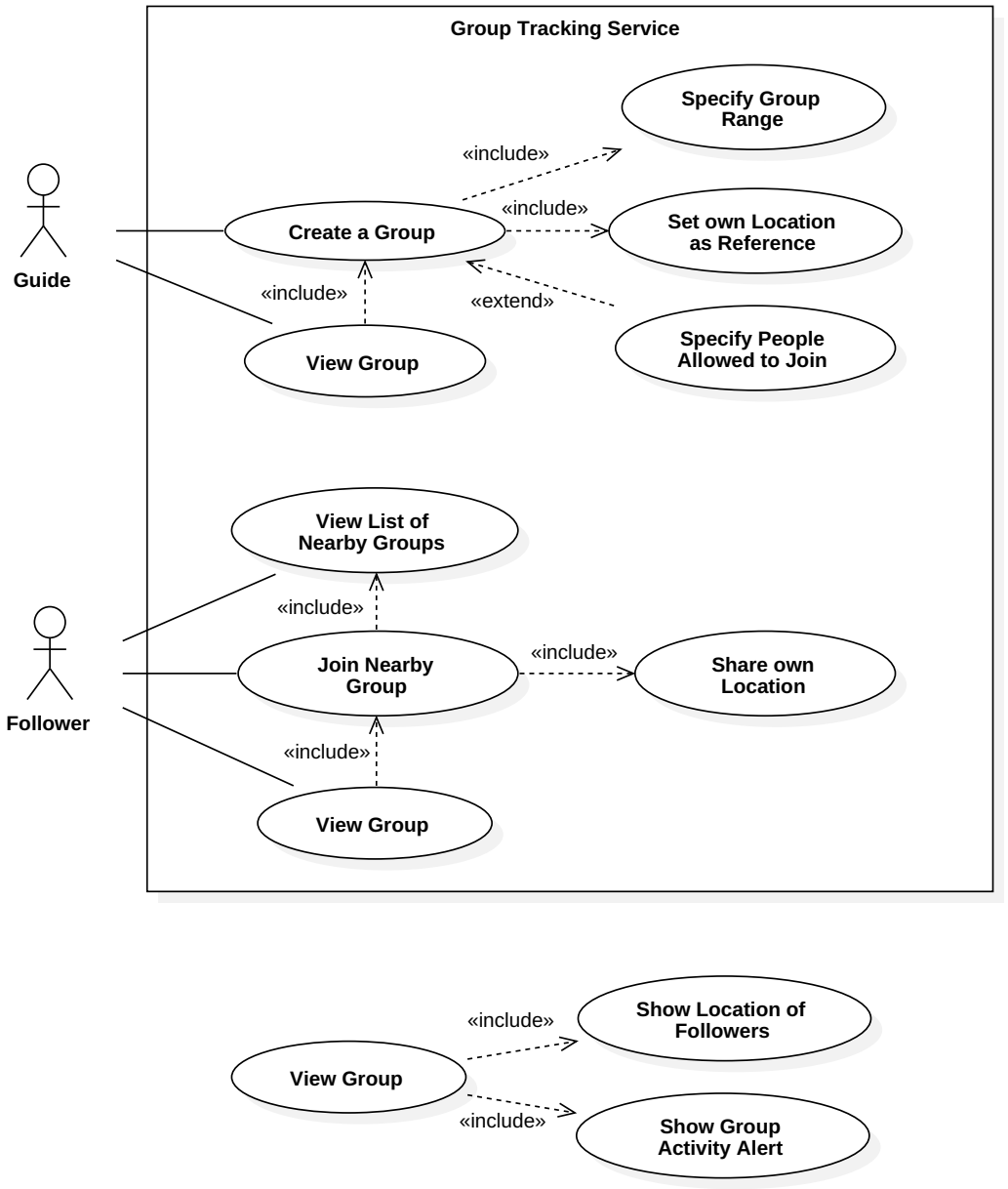


Figure 4.3: Group Tracking Service- Use Case Diagram

can travel freely. As an additional step, the guide can also *enter a list of people that are allowed to join the group.* If no one is specified, then the group is open and anyone can join.

Once a group is created, people within the range of the group are able to see using the service. This is shown in figure as *view list of nearby groups* use case. If the group has a specific list of people that are allowed to join then only those people are able to see the group. The people who are interested in joining the group are shown as Followers in the use case diagram. Upon *joining the group*, the location of the followers is also shared with the group (*Share own Location* use case). Followers and the guide can *view the group* which shows location of all other followers and the activities in the group. If one of the followers gets farther than the distance specified by the guide, an alert is sent to the group as an activity alert.

Role of Digital Beacons:

When the guide creates a group, a digital beacon is created which is set to mimic his/her location. The properties of the digital beacon involved in this use case and their roles are described as follows:

- **Location:** The current location of the guide which is dynamically updated as the guide moves
- **Perimeter:** The range of the group within which the followers can freely move without triggering an alert
- **Visibility:** A list of reference of users who are allowed to see and follow the group
- **Followers:** A list of reference of users that are following the group. These references are used to get current location of followers from their mobile devices and share it in the group
- **Description:** A name and description of the group to help people decide if they want to join or not

Using the properties described above, a digital beacon can represent a group and allow members to track each other's location and be alerted if someone leaves the group in order to react in time.

4.3.3 City Tour Sharing Service

A service lets users create a city tour that consists of different places and lets them share multimedia content at those places which others nearby are then able to see.

Driving Question:

How can a location based service allow users to create their own city tours? How can the users share multimedia content with each other at the places that they visit?

Scenario:

Tourists often require guidance about which places they should visit and what attractions it might present[CN13]. The perspective of other people who have been at a certain place gives an interesting insight into the offerings of that place. People want to see what other tourists have been doing in the nearby area, and this information can help them in deciding what they want to do next. For example, consider people who are new to a place and want to find out what next place they should visit. A service that provides them details on nearby places that other tourists visited, and also provides multimedia contents that others shared at those places can be useful to help tourist decide on places to visit. Furthermore, this can also be used as a medium to share personal tour experiences with people.

Functional Requirements:

In order to enable a service that lets users share city tours with each other, following functional requirements have to be considered. Fig. 4.4 shows a use case diagram that covers these functional requirements.

Actors: There are two types of actors involved in this service. *Tourist A* creates a tour and while visiting different places and shares it with others. *Tourist B* views nearby tours and follows it if interested.

Description of Use Cases: A tour is a group of places that tourist A visits and adds them to a tour. A relationship is specified between places that are part of a specific tour. This relationship allows the service to present to a user, next places in the tour and if interested the user can navigate to the location of these places.

For each of the place added to a tour, tourist A also sets a region within which this place can be discovered by other people. Furthermore, tourist A can also add multimedia contents to these places. This multimedia content will be visible to other people within the region of that place. And hence others can see this multimedia content before actually visiting the place to get an idea about the place.

Tourist B can see the tours that are nearby to him. The tours that are nearby is determined by the discovery region of places that are part of the tour. If the discovery region of a place is nearby to the user, the user can see this as a tour and one by one visit the next places in the tour. While being within the discovery range, he can see the multimedia information that is added to those places by others. Lastly, tourist B has the option to navigate to the exact location of the place with help of the navigation options provided by the service.

Roles of Digital Beacons:

Each of the place in the tour is represented by a digital beacon. Tours are the classification of all the digital beacons that are related to each other by a *Tour* relationship. As described in section 4.2.6, relationships can be of any type. Therefore, for this service, the type of relationship is *Tour*.

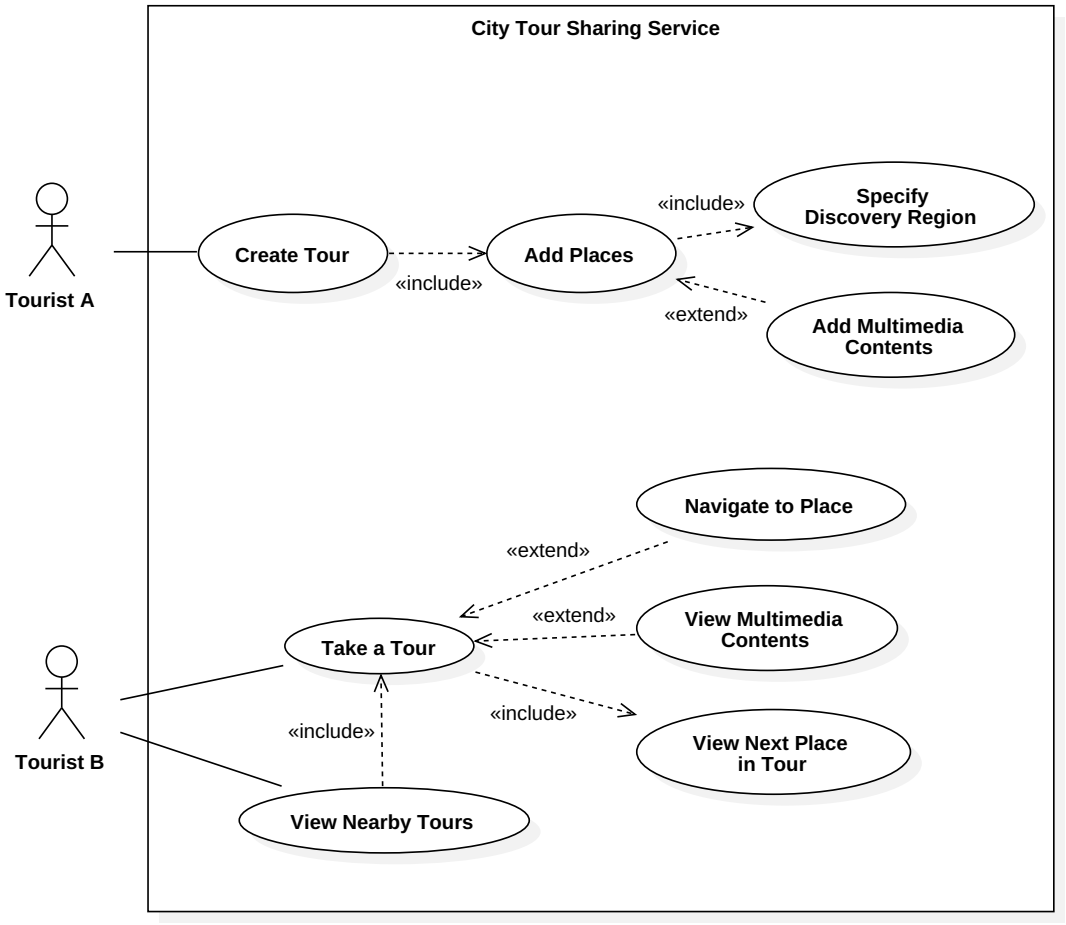


Figure 4.4: City Tour Sharing Service - Use Case Diagram

- **Location:** Location contains the coordinates of the place being represented by the digital beacon.

- **Perimeter:** Perimeter specifies the discovery region of the place. Users can see a tour place only if their current location is within this perimeter.

- **Multimedia Content:** A list of references of multimedia contents shared by the user are saved here.

- **Relationship:** For each digital beacon that represents a place or a site in a tour, a relationship of type Tour is specified. Along with the type of the relationship, beacon IDs of next places in the tour are also saved. These beacon IDs can be used to give a preview to the user about what the other places in the tour.

- **Description:** A name and description of the tour is saved here.

Hence, with the help of above mentioned properties of a digital beacon, a service that lets users share city tours with each other can be implemented.

4.3.4 Advertising Service

A service provides a way to advertise products and businesses, taking into account the context of the location of users.

Driving Question:

How can the location context of a user be represented? How can an advertiser use the location context to reach to potential consumers?

Scenario:

At several places the purchasing behavior of a consumer can be predicted to some extent. For example people shopping in an area are likely to look for nearby places to eat. Or people shopping for a certain category of products at one place will be interested in another business offering similar products. This information about the interest of a consumer at a certain place can be called location context. Advertising services can be improved by using this location context [ZWJH10]. This is because a service that is aware of location context, can customize the advertisements delivered to a consumer based on where he is at the moment. When a consumers receive an advertisement about something that they are already in a state of mind to buy, there are better chance of them actually responding to the advertisement and making a suggested purchase.

The way this location context is represented and how it is used to provide advertisements to the consumers is the objective here.

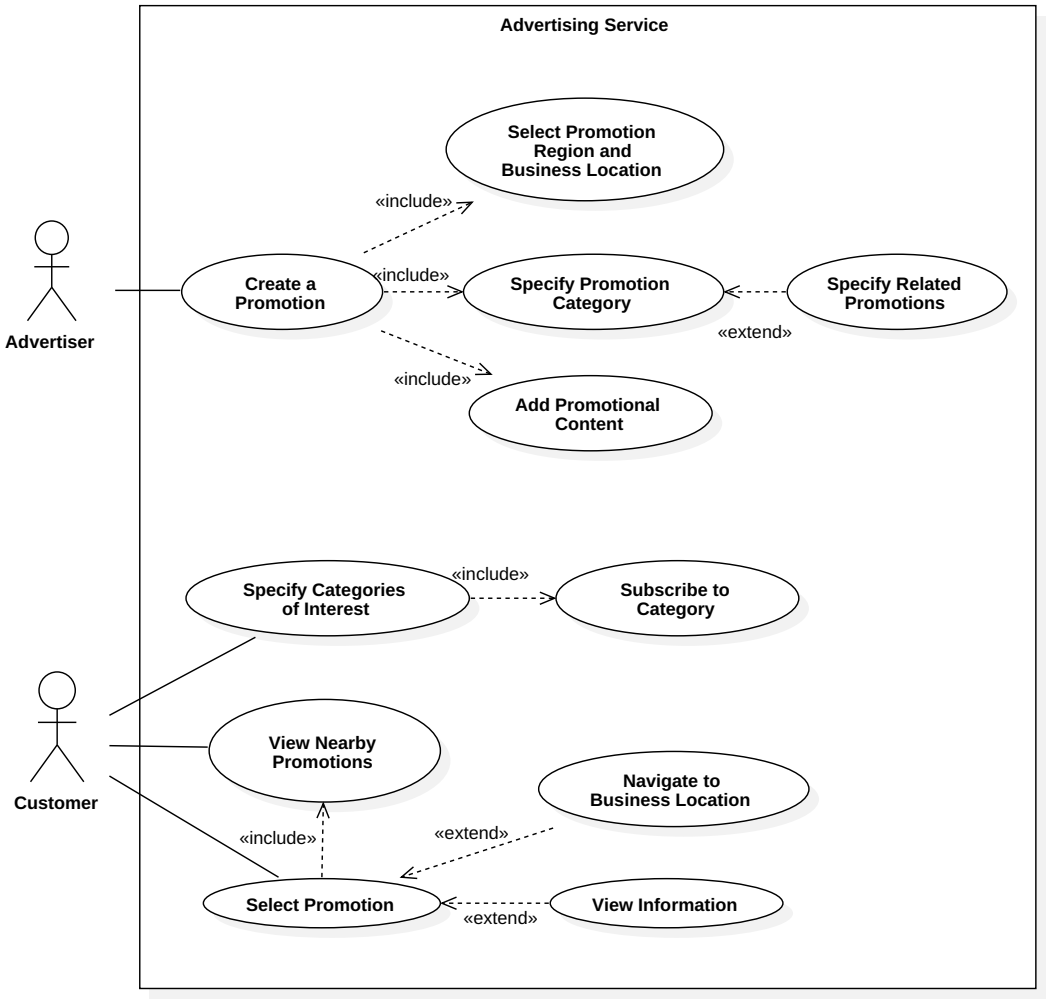


Figure 4.5: Advertising Service - Use Case Diagram

Functional Requirements:

The use case diagram described in Fig. 4.5 describes the functional requirements of an advertising service that uses location context of a consumer.

Actors: *Advertiser* creates the promotional content. *Consumer* is the target of the created promotional content.

Description of Use Cases: An advertiser who is interested in promoting his business creates a *Promotion.* A promotion is a business or product that the advertiser wants to promote. The process of creating this promotion requires to set a *Promotion Region* and *Business Location.* The promotion region is the region within which an advertisement about the promotion can be published to the consumer. This region can be of arbitrary shape depending on the area that the advertiser wants to target. The business location is used to help the consumer in navigating to the store where the advertised product/service is available.

Another information that the advertiser enters in order to create the promotion is a *Promotion Category.* Categorizing the promotion is useful to take into account interest of a consumer and further enhance the advertisements. Optionally, *Related Promotions* can also be specified. A related promotion can be any promotion that might have a geographical relevance or one the belongs to a similar business. As described in an earlier example, nearby restaurants can be described to be related to shopping areas. Using this information, an advertisement service can suggest next places to visit to a consumer even if they are not currently in the promotion region. Related promotion information can also be useful if it is known that the consumer is interested in its category and hence advertising about it will also make sense.

Lastly, the advertiser enters the actual *Promotional Content* and other information that will be presented to the consumer in the advertisement.

On the other hand, a consumers can specify categories that they are interested in. This helps the advertising service in tailoring the advertisements based on the interest of the consumer. The consumers can then see the promotions that are nearby to them. They can select the one that they are interested in, view information about it and navigate to its location.

Roles of Digital Beacons:

Each of the promotion is represented by a digital beacon. The use of each of the property of a digital beacon to enable this use case is as follows:

- **Location:** Location contains the coordinates the business location that has created the promotion.
- **Perimeter:** The promotion region is saved as the perimeter of the digital beacon. As described earlier, this perimeter can be boundaries of arbitrary shaped and is used to identify the region within which the promotion can be seen by a consumer.
- **Multimedia Content:** A reference to the actual promotional content can be saved in here.

- **Relationship:** A reference to all the promotions that are related to this one can be saved here.
- **Tag:** A unique tag can be used to classify promotions into specific categories.
- **Description:** In order to publish an advertisement, information about its details can be saved as name and description of the digital beacon.

Using these set of properties, the digital beacon can be used to implement an advertising service that takes into account the location context of the consumers.

This section described example scenarios that a digital beacon model covers. To demonstrate how a location based service can use this digital beacon model in their service, the next chapter provides details of an implementation of the digital beacon model that was done in this work.

5 Implementation

This chapter discusses in detail a smartphone application in which digital beacon model is implemented. This application is a pedestrian navigation application and a digital beacon service is added to it in this work. The chapter is organized as follows:

First, *Section 5.1* discusses the purpose and scope of the implementation that is presented in this chapter. This section also describes the functionalities provided by the implementation.

Secondly, *section 5.2* discusses the environment of the application and different components that play a role in its working the application under consideration and describes the functionalities that this application provides.

Lastly, *section 5.3* provides the implementation details of the application.

5.1 Scope

In chapter 4, digital beacon model and its usage scenarios were described. An implementation of a complete scenario is beyond the scope of this work. However, to discuss how the digital beacons can be used in a real world application, a part of the digital beacon model is implemented in this work.

From the provided implementation it should be clear how an application can handle digital beacons and how digital beacons can be used to provide the desired functionality. The functions related to a digital beacon that are implemented in this work are as follows:

1. **Creation of Digital Beacons:** A user can create a digital beacon and set its properties such as name, description and range. Furthermore, a user can also add images to a digital beacon.

2. **Specify category of a digital beacon:** While creating a digital beacon, its category can be specified. As an example, categories such as entertainment, shopping, landmarks, food and random are used in this implementation.

3. **View nearby digital beacons:** A user can view all the digital beacons in whose range they reside. Furthermore, they can also browse the nearby digital beacon by categories instead of viewing all of them.

4. **Navigate to a digital beacon:** Lastly, the user has the option to navigate to a digital beacon.

To provide these functionalities, the platform and external services that are used are described in the following section.

5.2 Implementation Environment and Components

The environment in which this implementation is done and various components that play a role in the working of this implementation are described in this section.

5.2.1 Platform

The platform used for this implementation is iOS. It is an operating system for Apple Inc. mobile devices. This platform is responsible for handling the application execution, providing objects that allow constructing user interfaces, delivering user interaction events, delivering location events and also providing communication mechanism between application objects.

A brief overview of the functionalities of iOS that are used in the implementation of digital beacons is described as follows:

User Interface To build user interface in iOS and respond to user interactions, standard classes are available that can be used. The two basic classes are used for this purpose are discussed below.

- *UIView:* UIView is the base class using which all user interface objects in iOS are constructed. A set of standard sub-classes of this class are available that can be used to build a user interfaces. Following are of interest for the implementation in this work:
  - *UILablel, UITableView and UIImageView:* Objects of these classes are used to present text, list of objects or images to the user.
  - *UIControl:* Objects of this class are used to construct elements with which a user can interact with an iOS application. Examples of these objects include buttons, gesture recognizes and sliders.
- *UIViewController:* The base class that provides the model for management of user interface in a iOS application is UIViewController. It is responsible for providing the contents that a UIView object should present and for responding to events that happen in a UIView object. Sub-classes of this classs that are important for description of this work are:
  - *UITableViewController:* Objects of this class are responsible for managing a table view. They provide the content that the table view should present and respond to events such as a user scrolling the table view or selecting a row in the table view.
  - *UISplitViewController:* It is a container view controller has two childs called master and detail. The *master* drives changes in the *detail.* Using this master-child interface, the content that the detail presents can be controlled. For example, a category view controller (master) can dictate the specific category of products that a list view controller (detail) should present.

Hence, using combinations of objects of classes and sub-classes of UIView and UIViewController views of an iOS application can be built and managed.

Segues In an iOS application, the transition between different views of an application is called a segue. A segues can be generated by a UIControl object or they can be triggered programmatically. The application programmer has to specify the action that should trigger the segue and to which view the transition should be performed. For example, if a user presses + button in a note application, a segue is performed to a new view that lets the user enter the information required to create a note.

Location Event Delivery The location of the mobile device on which an iOS application runs is delivered by an iOS framework. The framework of iOS that is responsible for delivering location is called Core Location framework.

This framework is responsible for using positioning technologies available on the mobile device to approximate location of the device. To use the Core Location framework, the application only has to configure the framework according to its requirement and by doing this configuration, the framework start delivering location events to the application. From a callback function of location event, location can be extracted and used in the application to implement desired logic.

Intra App Communication Mechanisms An iOS application consists of different objects that can communicate with each other. The communication mechanism used in this implementation are as follows:

- *Target-Action:* This communication mechanism is used to send messages between user interface (UI) control objects and other objects that are responsible for handling those messages. The messages are called action messages and the object that receives them is called target. The UI control objects know which message should be sent and to which object it should be sent. Within the scope of this implementation, the key word *action* will be used for messages that are generated by UI control objects.
- *Observers:* This communication mechanism is used to send messages called notifications to any object or objects called observers. The payload data that has to be sent is added to the notification and the notification is then sent on specific topics. Interested objects declare themselves as observer of the topic they are interested in and register a callback function. Whenever a notification is sent, this observer receives the notification with payload data and its callback function is called. Hence in the callback function, the observer can use the payload data and implement the desired functionality.

5.2.2 Parse Cloud

The database used in this implementation to store digital beacons is provided by a platform called Parse Core. It is a mobile backend as a service platform and allows

various backend functionalities for an application. For this implementation, cloud storage service of this platform is used and is referred to as Parse Cloud. The objects stored on this cloud and the methods available for these objects are described next.

ParseObject An object that is stored on the Parse Cloud is called a ParseObject. Different methods are available that can be called on a ParseObject. The methods that are relevant to the description of this implementation are described as follows:

- **saveInBackground** this method is used to save objects in a separate thread of the application using the Parse Cloud. However, this method does not provide a way to react when a save operation is complete for this purpose method described next is used.
- **saveInBackgroundWithBlock** using this method, once a save operation is completed in the background, the block that is passed to the method is then executed by the application on the main thread. This method will be used whenever the application has to respond to a complete save operation.
- **deleteInBackground** and **deleteInBackgroundWithBlock** these methods work in a similar way as described above but are used to delete an object from the Parse Cloud.

The above mentioned description is about how objects are created or deleted from Parse Cloud. However, in order to retrieve an object or a set of objects from Parse Cloud, a ParseQuery is used.

ParseQuery A ParseQuery allows querying objects using different methods. Each element in a ParseObject is a key that can be used to retrieve objects from Parse Cloud. In addition to using solely the keys, constraints can be added to a query to filter the objects according to the required criteria. The specific constraint on a ParseQuery that is used in this implementation is described as follows:

- **whereKey(keyname, nearGeoPoint)** This constraint consists of name of the key, a geographical point consisting of latitude and longitude values. Using this constraint, objects that are closest to the *nearGeoPoint* are returned.

5.2.3 Nifino Navigation Service

As described earlier, the digital beacons are implemented in an existing iOS application. This application is called nifino[Nif14] and it provides the navigation service that is used to navigate to a digital beacon.

The nifino navigation service can be used for digital beacons in following way:

1. A user can select a digital beacon that he/she want to navigate to.
2. The application switches to nifino navigation service.

To enable this functionality, a component called navigation handler can be used. A request for navigation is sent to the navigation handler and the task is then taken over by nifino navigation service.

By using the components and the platform is described above, the design of the implementation that is done in this work is discussed in the next section.

5.3 Implementation Design

In this section, the part of nifino application that implements the digital beacon functionalities is described. To refer to this part of nifino application we'll use the term DBeacon service, meaning digital beacon service.

First the implementation design of DBeacon service is described using a Model-View-Controller pattern. It is an architectural pattern introduced by Smalltalk inventors[Ree79] and is described in [C2.14] as "Model-View-Controller is the concept of encapsulating some data together with its processing (the model) and isolate it from the manipulation (the controller) and presentation (the view) part that has to be done on a UserInterface". Using this pattern the implementation done in this work is shown in Fig. 5.1.

This design provides to functionality described in section 5.1. A description of each component shown in the Fig. 5.1 is provided next.

5.3.1 Model

The model consists of a digital beacon manager which is described as follows:

Digital Beacon Manager:

The responsibilities of the digital beacon manager, or in short DBeacon Manger, are:

- *Managing DBeacons:* DBeacon manager creates, modifies or deletes DBeacon data from the Parse Cloud. It does this when it is instructed by the controller to do so. This is shown as "update" interaction between the controller and the model in Fig. 5.1. The DBeacon data is stored as ParseObject in the parse cloud as shown in Fig. 5.2.

- *Evaluation of current DBeacon list:* Another responsibility of DBeacon manager is to always keep a current list of DBeacons. The current list is a set of all DBeacons whose range covers the current location of the user. DBeacon manager carries out this evaluation each time the location of the user changes. For this purpose DBeacon manager always has to know the current location of the user. Hence to get the location, it has a location topic observer. This observer is based on observe-observable communication mechanism that is provided by the iOS as discussed in section 5.2.1. This observer gets a notification whenever the location of the

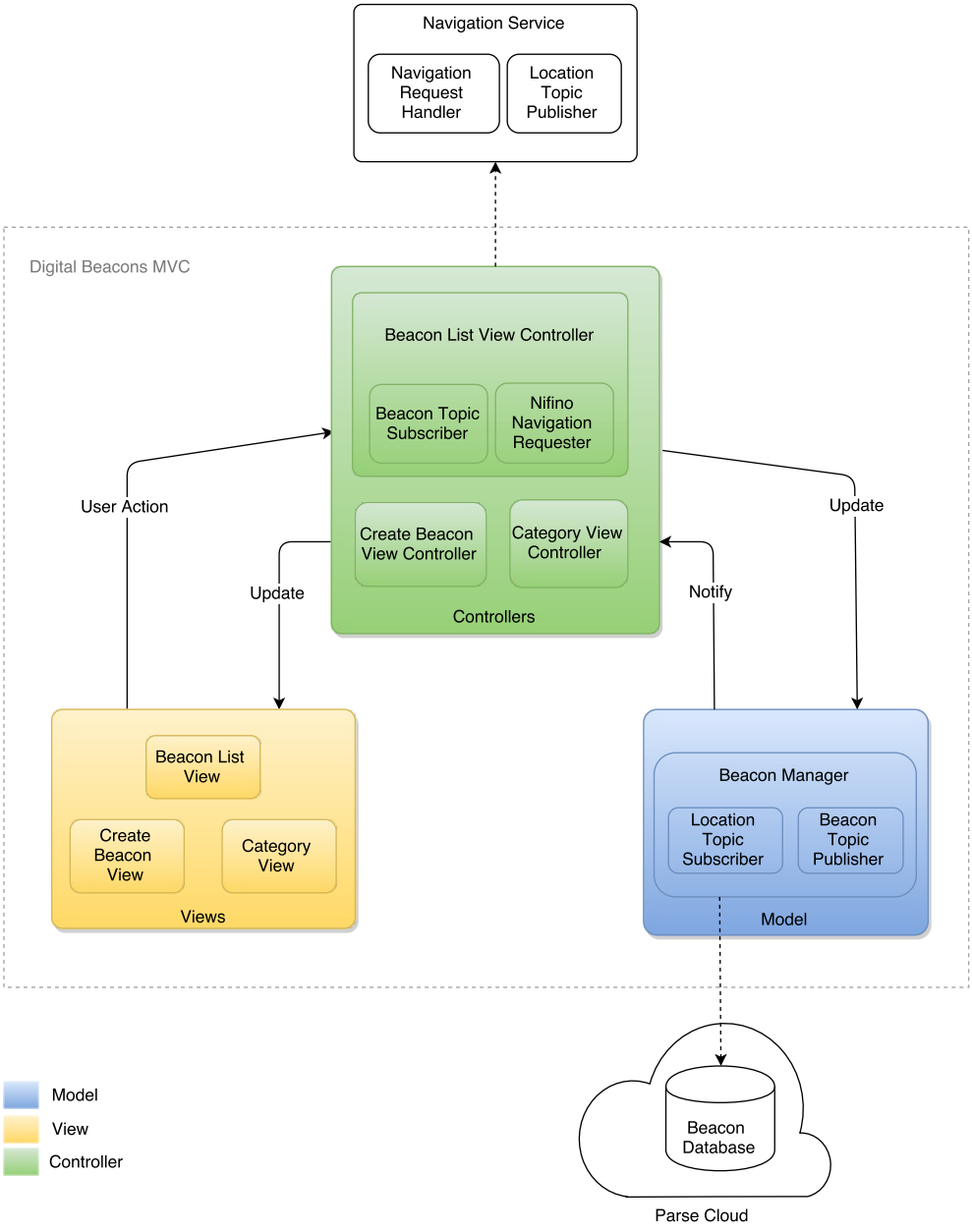


Figure 5.1: Implementation Design of Digital Beacons

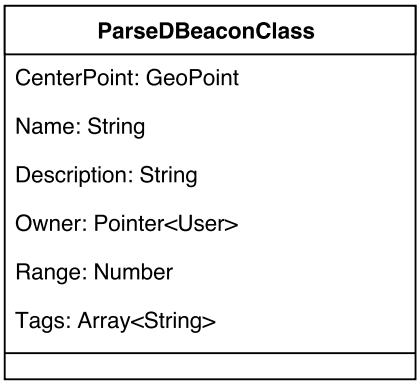


Figure 5.2: Structure of a Digital Beacon object stored on Parse Cloud

user changes. In response to this, the DBeacon manager sends a query to Parse Cloud and asks for DBeacons in an arbitrarily large area. Section 5.2.2 describes the structure of this query. Once a reply to this query is received, the DBeacon manager then checks one by one for range of each DBeacon that is returned in the query. It then discards all digital beacon for which the range does not cover the location of the user. The DBeacons that are left after this process are then stored locally as current beacon list as shown in Fig. 5.1.

- *Keep Track of current Category:* The digital beacon manager also holds information about the current category that is currently selected by the user. By keeping track of category, whenever DBeacon manager receives a Create message, it creates the DBeacon with a tag that belongs to the selected category.

- *Notifying changes in DBeacon List:* When DBeacon manager updates its current DBeacon list, it notifies the controller about this change. This is shown as "notify" interaction between model and controller in Fig. 5.1. The communication mechanism for this is also the observe-observable mechanism. DB manager send a notification on DBeacon topic with the current DBeacon list data attached to it. Any controller who observes this topic, gets this notification and can access the data.

5.3.2 Views

Views are responsible for presenting the model data to the user and delivering user interface events to the controller. The view however has no knowledge about the model data, the controller has this information and sends the data that the view then shows. As described in section 5.2.1, UI view and UI control objects are used to construct these views. The views in DBeacon service and the ways a user can interact with them are described as follows:

DBeacon List View:

This is the first view of the DBeacon service when it is first launched and is responsible for showing the list of DBeacons in whose range the user currently is. The design of this view can be seen in Fig. 5.3. Each prototype cell is a row that represents a DBeacon. These rows construct a list and display the information associated with their associated DBeacon to the user. This view is dynamically created based and is loaded as the user scrolls through the list of beacons. Since it is dynamic, the view continuously requests data from the controller and when the controller responds to these requests, the new data is played on the view.

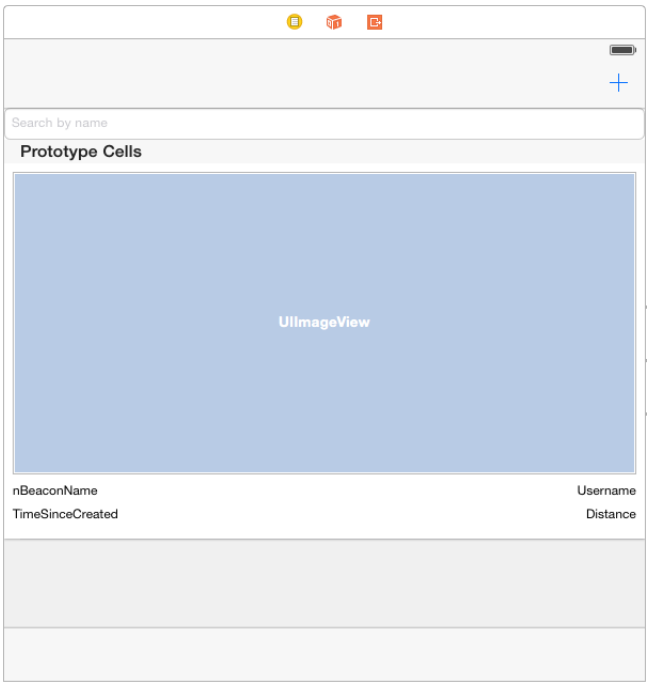


Figure 5.3: Design of DBeacon List View

Categories View:

The categories view is a static view that shows a list of categories to which DBeacon are divided. The design of this view is shown in Fig. 5.4. This is a master view to the DBeacon List View. This means that it is able to control the configuration in which DBeacon List View is loaded. The configuration here means selection of a category from this view and with this category, the DBeacon List View is loaded.

Create-Modify DBeacon View: Similar to Categories View, this is also a static view ans is shown in Fig. 5.4. This view is responsible for requesting input data from the user

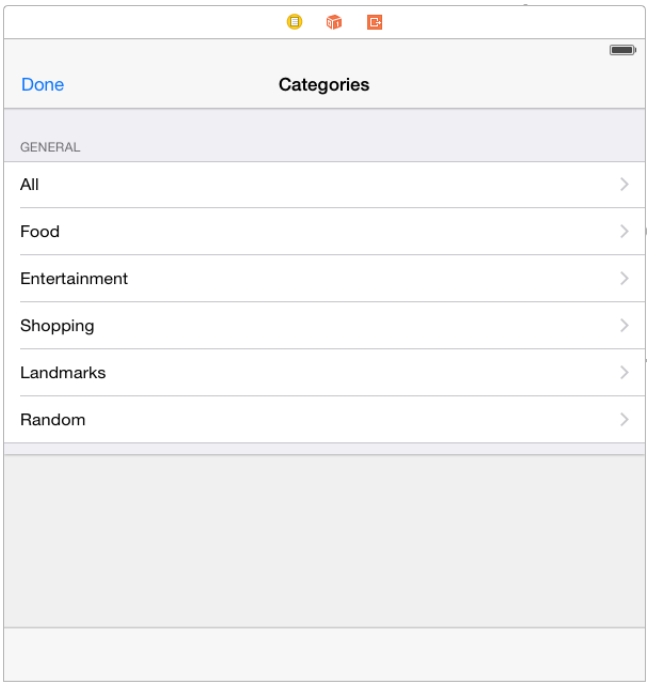


Figure 5.4: Design of Categories View

for creating or modifying a DBeacon. The data that is entered in this view, is sent to DBeacon List View controller. This controller in return asks the DBeacon manager to create or modify the DBeacon accordingly. The information that can be entered using this view is Name, Range, Description and an Image as can be seen in the Fig. 5.5

5.3.3 Controllers

The DBeacon service consists of three controllers that are responsible for their respective views. From each view, user actions can be generated and these actions are received by the corresponding controller which then handles these actions. This interaction is shown as "User Action" in Fig. 5.1 and follows the Target-Action communication mechanism as described in section 5.2.1. Details of responsibilities of each of the controller is as follows:

DBeacon List View Controller:

The DBeacon List View controller has following responsibilities:

- *Handle User Actions:* This controller handles actions that are generated by the DBeacon List View. These actions are tapping + button to create beacon as shown in Fig. 5.5 and selection of a row (not visible in the figure).

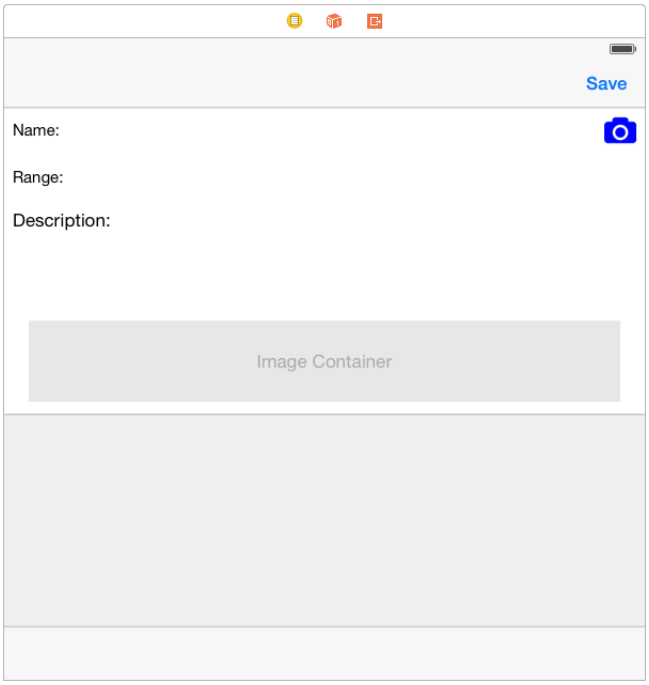


Figure 5.5: Design of Create-Modify DBeacon View

  - The response to + button is performing a segue to Create-Modify DBeacon View. The responsibly of handling further events are then transferred to Create-Modify DBeacon View controller.
  - The other action that this controller handles i.e. selection of a row is responded with presenting a list of further actions to the user. The row, as described earlier in section 5.3.2, represents a DBeacon object. Hence, actions related to a DBeacon that can be performed are presented. These actions are View, Edit and Start Navigation actions. The View action, does not show any special information and simply shows one DBeacon that is selected. However, the action of Edit and Start buttons are as follows:
    * The Edit button action similar to + button action performs segue to Create-Modify DBeacon View. However, the difference here is this the segue contains a payload which is an existing DBeacon object that is to be modified.
    * The Start Navigation button action is responded with terminating the DBeacon service and requesting the Nifino navigation service to perform navigation to the selected DBeacon.

- *Provide Data To DBeacon List View:* The other main responsibility of this controller is to provide data to the list view. As the user scrolls through the list, the DBeacon List view requests this controller to provide more data. The controller then asks model for data.

- *Hold Category information:* This controller holds the information about the current category that is selected. The current category is selected by the master of this controller which is the Categories View Controller.

- *Update Model State:* This controller also informs the model (i.e. DBeacon manager) about the category when it changes. The model has to know this information in order to create a DBeacon with appropriate tag when it is requested as described in section 4.2.

- *Interpreting Model Data:* This controller is also responsible for interpreting the model data. Interpreting model data means the following here:

  - First, this controller knows which data of the DBeacon should go in which UI views to show it to the user.

  - Secondly, this controller performs any desired operations on the data such as converting it to a form that is meaningful to the user before sending it to the DBeacon List View. For example, calculating an approximate distance to the DBeacon from current location of the user to show it in "Distance" field as shown in Fig. 5.3

  - Lastly, depending on the category that is currently selected, this controller only shows the DBeacons of that category in the DBeacon List View.

Create-Modify DBeacon View Controller:

The DBeacon List View controller segues to this controller when the user presses + button as can be seen in Fig. 5.3 or when user selects "Edit" action that is available for each row in DBeacon List View. This controller is responsible for following:

- *Collecting user input data:* This view as shown in Fig. 5.5 consists of fields where the user can input information. These fields are types of UI control objects. When the user finishes entering data and presses Save button (see Fig. 5.5, the data that is set by the user in these fields is read by this controller and it send this data to the model using a Create message or a Edit message depending on the segue that caused this view to be shown. The model then uses this data to create a beacon on Parse Cloud or edit an existing one.

Categories View Controller:

This controller is responsible for responding to category selection done by the user in Categories View as shown in Fig. 5.4. Each row in the view shown in figure triggers a segue. The payload of the segue is the category selection and the destination is the DBeacon List View. The payload information in used to create the DBeacon List view of selected category.

5.3.4 Implementation Detail and Working

The actual DBeacon service implemented in this work is described in this section. An overview of the user interface that is implemented in this work is shown in Fig. 5.6.

The description of this implementation is organized into the tasks that the user can perform using the service. These tasks are described next.

5.3.4.1 Launch DBeacon service

The service is implemented in nifino application, it can be launched from the main view of the application. This view is shown in Fig. 5.7. The button  is pressed to launch the DBeacon service. Upon pressing this button, following execution happens:

Listing 5.1 Launching DBeacon service

```
function MainViewController::onDBeaconButtonPressed()
  instantiate DBeaconInitialViewController
  segue to DBeaconInitialViewController
endfunction
```

As shown in the Listing 5.1, at first, the DBeaconInitialViewController is instantiated. This controller is a split view controller as shown in Fig. 5.6. This split view controller as described in section 5.2.1, maintains a master-detail interface. The master here, is the CategoriesViewController and detail is DBeaconListViewController. These childs are embedded in a special controllers called navigation controller whose purpose is only to manage navigation between controllers. During the instantiation process the child of DBeaconInitialViewController are initialized.

After initialization is complete, segue to DBeaconListViewController is triggered. The execution that happens next is described in following section.

5.3.4.2 View Nearby Digital Beacons

When a segue is triggered to the DBeaconListViewController, it prepares itself before doing the actual segue. This preparation process is described in Listing 5.2

Listing 5.2 Preparation Process of DBeacon List View Controller

```
function DBeaconListViewController::prepare()
  if a category is not set yet by CategoriesViewController then
    set category to "ALL"
  endif
  instantiate DBeaconManager
  start observing on DBeacon topic and register callback function-
   with name dBeaconNotificationCallBack
endfunction
```

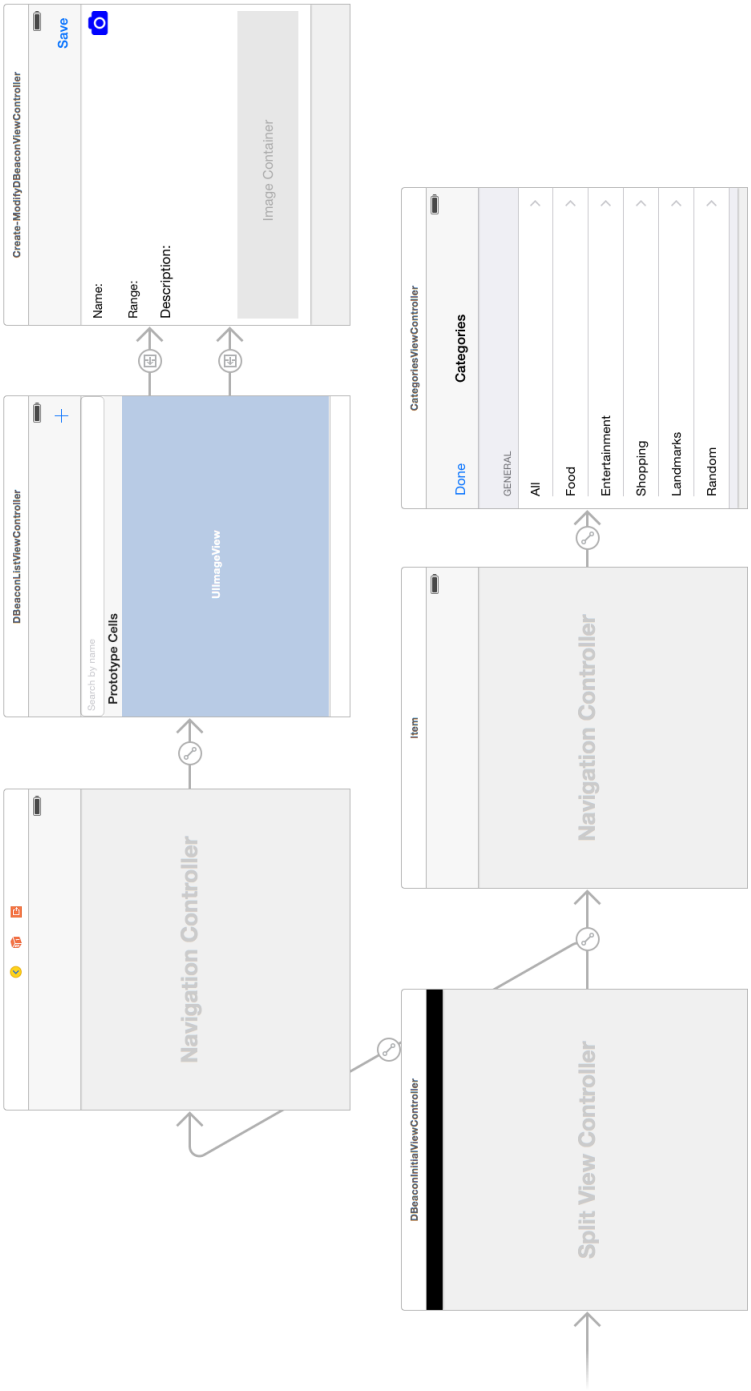


Figure 5.6: Overview of DBeacon Service User Interface

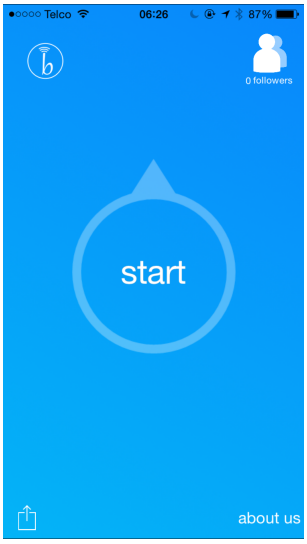


Figure 5.7: Start-up view of Nifino Application

In prepare() function, first it is checked if a *category*[1] was set by the CategoriesViewController. On first launch of DBeacon service, no category is selected. Hence, the category variable of DBeaconListViewController is set as "All". Secondly, DBeaconListViewController creates a DBeaconManager. Lastly, DBeaconListViewController starts observing on the topic on which DBeacon data is expected and also registers a callback function that is called each time a notification on DBeacon topic is received.

After the execution of the prepare() function is complete, DBeaconListViewController perform the segue to DBeaconListView however, it does not display any data yet. It displays data only when it receives a notification on DBeacon topic. Hence the following execution happens next:

Listing 5.3 Initialization of DBeaconManager

```
function DBeaconManager::init()
  start observing on Location topic and register callback function-
   with name locationNotificationCallBack
endfunction
```

The initialization of DBeaconManager happen as part Listing 5.2, when DBeaconManager is created. In the initialization function as described in Listing 5.3, the DBeaconManager start to observe on location topic. It is a topic on which the Nifino application sends notifications whenever a new location is received. The process of Nifino sending a notification is shown in Listing 5.4.

The function onNewLocationEvent() is executed each time the Core Location framework of iOS deliver a location event. When a notification on location topic is sent, since

[1]Within the scope of section 5.3.4, an italic styled word means it is a variable that is owned by the object whose function is being described.

Listing 5.4 Sending a notification on Location Topic

```
function LocationTopicObservable::onNewLocationEvent(locationdata)
  create a notification object with Location topic name
  add locationdata to the notification
  send notification
endfunction
```

DBeaconManager is observing this topic, it receives this notification. The callback that happens is described in Listing 5.5.

Listing 5.5 DBeaconManager Response to a Notification on Location Topic

```
function DBeaconManager::locationNotificationCallBack(location)
  save location received in notification as currentdevicelocation
  create parse query for DBeacon data
  add a constraint that first 100 DBeacons near currentdevicelocati-
   on be returned
  send query asynchronously with callback function name parseLocati-
   onQueryCallback
endfunction
```

DBeaconManager sends query to Parse and requests for first 100 DBeacon near the current location of the device. The number 100 is an arbitrary number. If the list of returned DBeacons were to exceed this number, then the DBeaconManager can ask for more DBeacons from Parse.

Once response to the query is received, the callback function in Listing 5.6 gets executed.

Listing 5.6 DBeaconManager Response to Parse Query

```
function DBeaconManager::parseLocationQueryCallback(DBeaconlist)
  foreach DBeacon in DBeaconlist
    get centerpoint of beacon
    get range of DBeacon
    calculate distance between centerpoint and currentdevicelocaiton
    if distance > range then
      delete DBeacon from DBeaconlist
    endif
  endfor
  save remaining DBeaconlist as filteredDBeacons
  create a notification object with DBeacon topic name
  add filteredDBeacons
  send notification
endfunction
```

This function filters the received DBeacons. The filtration process looks for range of each DBeacon. If the distance between center location of DBeacon and the current location of the device exceeds the range, this means that the DBeacon is outside the current location of the device. Such DBeacons are discarded and the remaining ones are saved in filteredDBeacon variable. These filtered DBeacon are the "Current DBeacon List" data base of the model that was previously shown in Fig. 5.1..

Listing 5.7 DBeacon List View Response to a Notification on DBeacon Topic

```
function DBeaconListViewController::
 DBeaconNotificationCallBack(filteredDBeacons)
  foreach DBeacon in filteredDBeacons
    get tag of DBeacon
    if value of tag does not match the categroy then
     delete DBeacon from filteredDBeacons
    endif
  endfor
  load rows with remaining DBeacons in filteredDBeacons
  show the loaded rows on display DBeaconListView
endfunction
```

When the executions described in Listing 5.5 - 5.7 are completed the DBeacon-ListViewController updates the DBeaconListView with DBeacons whose tag match the selected category. As, for an initial start-up of the DBeacon service, this is "All" hence all received DBeacon are displayed on the screen. The list of DBeacons that are displayed can be seen in Fig. 5.8.

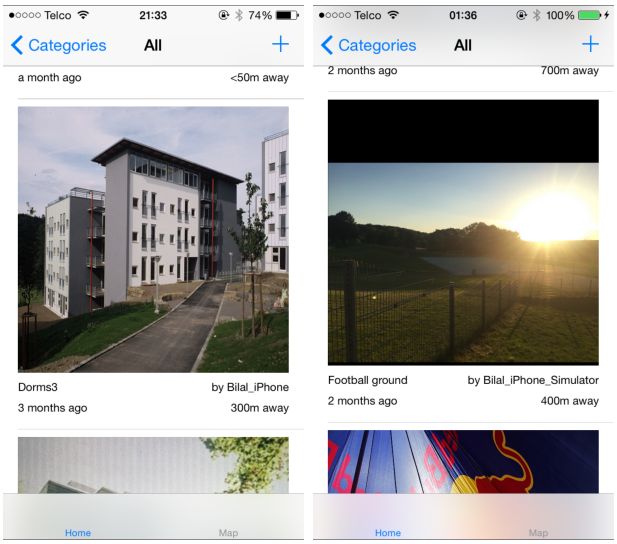
5.3.4.3 Create or Modify a Digital Beacon

Within the DBeaconListViewController, the user can press + button as shown in Fig. 5.8 to create a DBeacon. The execution that happens when this button is pressed, is as follows:

Listing 5.8 Create a DBeacon Action

```
function DBeaconListViewController::createButtonAction()
  segue to Create-ModifyDBeaconViewController
endfunction
```

On the other hand, the user can also select a row and by doing this, a list of further actions as shown as shown in Fig. 5.9. From this list of actions, they can select Edit action. This action is implemented as described in Listing 5.9.



(a) Test DBeacon called Dorms3

(b) Test DBeacon called Football ground

Figure 5.8: DBeacon List View Showing Test DBeacons

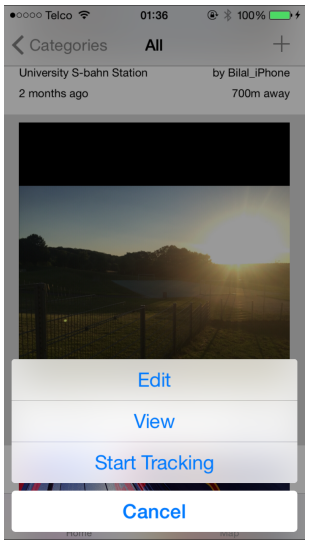


Figure 5.9: Selecting a Row in DBeaconListView

Listing 5.9 Edit a DBeacon Action

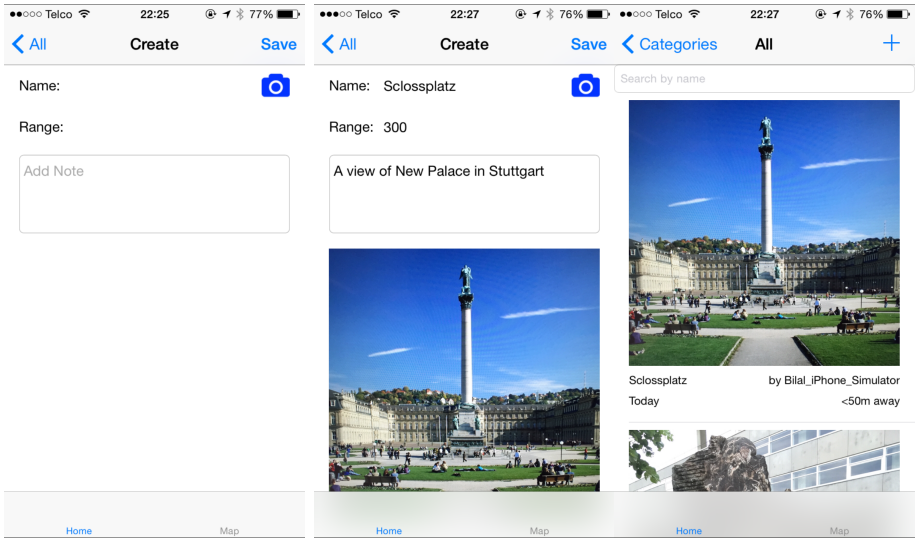
```
function DBeaconListViewController::editButtonAction()
  prepare a segue with selected beacon as payload and-
   Create-ModifyDBeaconViewController as destination
  execute the created segue
endfunction
```

When either of this segue happen, Create-ModifyDBeaconViewController is initialized and it prepares itself for segue. The prepare() function is described in Listing 5.10.

Listing 5.10 Preparation of Create-ModifyDBeaconViewController before Performing Segue

```
function Create-ModifyDBeaconViewController::prepare()
  if segue contains a DBeacon then
    load this DBeacon data into fields of the view
  endif
endfunction
```

After the execution of prepare() function, the segue is carried out and Create-Modify DBeacon View Controller show the user a display where they can enter data for new DBeacon or modify the data of existing DBeacon. This view can be seen in Fig. 5.10. After entering the data, if the user wants to save the DBeacon, Save button can be pressed. When this button is pressed, the execution that happens can be described as in Listing 5.11.



(a) Blank View to Enter DBeacon Data

(b) View Showing Entered DBeacon Data

(c) View Showing Created DBeacon

Figure 5.10: Views showing steps of Creating or Modifying a DBeacon

The create DBeacon function is shown in Listing 5.12. A similar function that modifies the existing beacon is described in Listing 5.13. The only difference is that instead of creating a new ParseObject of type ParseDBeaconClass (see Fig. 5.2), the existing DBeacon ParseObject is loaded with the data that user entered and saved.

Listing 5.11 Response of Save Button Action

```
function Create-ModifyDBeaconViewController::saveButtonAction()
  get name, description, range, image, tag from the view
  if no DBeacon passed to the controller then
    call createDBeacon function of DBeaconManager with the-
     data gathered from view as arguments
  else
    get the existing DBeacon
    call modifyDBeacon function of DBeaconManager with the-
     data gathered from view and DBeacon as arguments
  endif
endfunction
```

Listing 5.12 Function to Create a DBeacon

```
function DBeaconManager::
   createDBeacon(name, description, range, image, tag)
  if range is not within constraints then
    clamp range to the to allowed range
  endif
  get currentuser
  get currentdevicelocation
  create a Parse object named DBeaconObject of type ParseDBeaconClass
  load DBeaconObject with name, description, range, image, tag,
    currentuser,currentdevicelocation
  save DBeaconObject on Parse
  segue back to DBeacon List View
endfunction
```

Listing 5.13 Function to Modify a DBeacon

```
function DBeaconManager::modifyDBeacon(DBeacon, name, description,
   range, image, tag)
  if range is not within constraints then
    clamp range to the to allowed range
  endif
  get currentuser
  get currentdevicelocation
  load Dbeacon with name, description, range, image, tag, currentus-
   er, currentdevicelocation
  save DBeacon on Parse
  segue back to DBeaconListView
endfunction
```

Once the DBeacon is saved on Parse, the view is switched back to the DBeacon-ListView as can be seen in Fig 5.10c.

5.3.4.4 Navigate to a beacon

When the user selects a row in the DBeaconListView, they are shown actions as shown earlier in Fig. 5.9. From the list of available action, the use can also choose to navigate to a DBeacon. For navigation, Nifino navigation service is used. This is shown in Listing 5.14.

Listing 5.14 Requesting Navigation from Nifino Application

```
function DBeaconListViewController::
  startNavigationAction(selectedDBeacon)
  call trackingRequest method of navigation wtih selectedDBeacon-
   as argument
  dismiss DBeaconListView
endfunction
```

After sending the tracking request, the DBeaconListViewController dismisses itself. Further process of navigation is handled by the navigation service and its description is beyond the scope of this implementation. A view of the application during the navigation process is shown in Fig. 5.11.

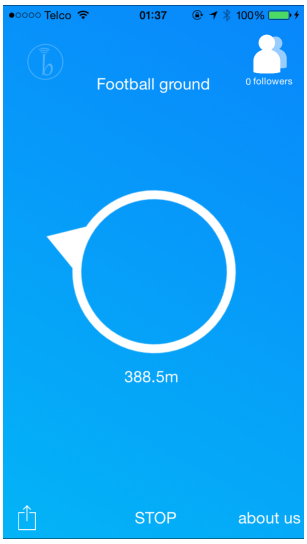


Figure 5.11: A View of Application While Nifino Navigation Process is Running

5.3.4.5 Selecting a Category

From the DBeaconListView, the user can press 

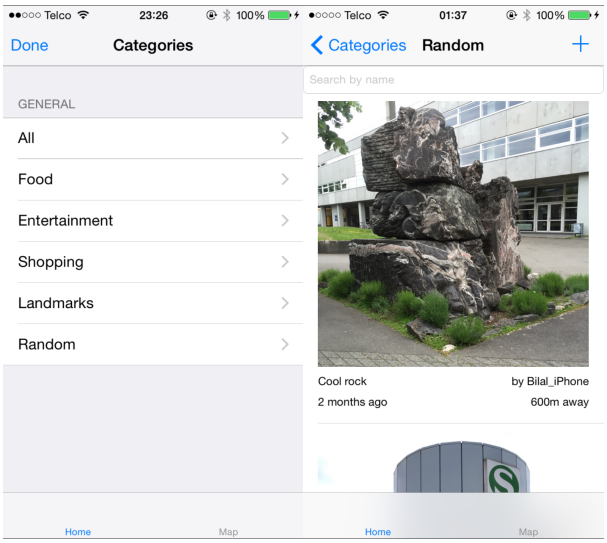
button to switch the view to CategoriesView. Once this button is pressed, a segue is performed to the CategoriesView-Controller. This segue is handled by the split view controller which is the parent of

DBeaconListViewController and CategoriesViewController. The CategoriesView that is presented to the user once the segue is complete is shown in Fig. 5.12a. From this view the user can then select the desired category.

When a selection is performed, the execution can be described as in Listing 5.15.

Listing 5.15 Requesting Navigation from Nifino Application

```
function CategoriesViewController::prepareForSegue()
  get the category that the user selected
  get a reference to destinationController
  set category variable of destination controller
endfunction
```



(a) View Showing a List of Available Categories

(b) Newly loaded DBeaconListView after Category "Random" is Selected

Figure 5.12: Categories View and an Example Selection Result

After prepareForSegue function is executed, the segue is carried out to the destination view controller. The destination view controller here, is DBeaconListViewController because of the master-detail between these two controllers. Since a category is set for DBeaconListViewController, therefore, when this controller prepares for displaying data, this time it loads the DBeacons of selected category (see Listing 5.2). The complete process of loading the DBeaconListView as described in section 5.3.4.2 happens again. After this process is complete, the final view is as shown in Fig. 5.12b. This figure shows the result of an example selection of "Random" category.

6 Conclusion

This chapter provides a summary of the work that is done in this thesis and reviews the contribution that this work makes to the area of location based services.

6.1 Summary

A model for representing geographical entities of the real world is introduced in this work. Geographical entities can be static places like restaurants, hotels, park or other points of interest. Geographical entities can also be moving objects such as pedestrian or a vehicle. Using the model that is introduced in this work, identification, context, and visibility control characteristics can be associated with these entities. These characteristics are captured by a digital beacon model that is introduced in this work.

The model consists for various types of metadata and references to the multimedia content. Using this information, this model data allows location based services to provide various functionalities such as, tracking the geographical entity, sharing information about geographical entity, defining time based events about the geographical entities, specifying a relationship between geographical entities, categorizing geographical entities and limiting their visibility to a certain user group.

Furthermore, the real world scenarios in which this model can be used are also described. With help of use case diagrams, the functionality requirements of these scenarios are pointed out. It is the discussed how the digital beacon model fits to provide solutions to cover these scenarios.

Hence the contributions of this work can be summarized as follows:

1. A mechanism of representing real world geographical entities in the digital world to implement a wide range of location based services.

2. Extends existing mechanism such as geotags and geogencing used for geographical identification to include dynamic location and allows tracking of moving objects.

3. Provides a way to add contextual information to the geographical entities with help of relationships and tags.

4. Provides a way to restrict the availability of the information associated with geographical entities to specific user groups and to a specific geographical region.

6.2 Limitation and Outlook

Privacy Concerns

Location based services are often challenges with location privacy concerns by the users. Since the digital beacon model that is introduced, requires continuous location updates from a user, users can be hesitant in using a service that does not tend to these privacy concerns. There are various methods that are available that can be used to cater these concern. These methods include cryptography, location obfuscation, dummy queries and pseudonyms [ZGR14].

Extensions The digital beacon model introduced in this work can be extended as follow:

- The perimeter information which is a part of the digital beacon model can be used to secure the data that is shared in the perimeter.
- The context data that is described in this work can be derived based on analysis of the user visits in a geographical region. This includes the relationship and tag information.

Bibliography

[Ber14] BergInsight: Mobile Location-Based Services. (2014)
(Cited on pages 5, 7, 12, and 13.)

[BL09] Brimicombe, A. ; Li, C.: *Location-Based Services and Geo-Information Engineering*. John Wiley and Sons, 2009
(Cited on page 11.)

[BNPZ] Belz, B. S. ; Nick, A. ; Poslad, S. ; Zipf, A.: Personalized and Location-based Mobile Tourism Services.
(Cited on page 26.)

[Bry14] Bryant, G.: Public Access Mobile Geography: The first Location Based Service. (2014)
(Cited on page 11.)

[C2.14] C2.com: Model View Controller. (2014)
(Cited on page 43.)

[CN13] Chong, A. Y. ; Ngai, E. T. W.: What Influences Travellers Adoption of a Location-based Social Media Service for Their Travel Planning. In: *Pacific Asia Conference on Information Systems* (2013)
(Cited on page 33.)

[Mar14] Marketsandmarkets: Location Based Services Market. (2014)
(Cited on page 7.)

[Nif14] Nifino.com: Why are Navigation Systems for Pedestrian Navigation Different? (2014). `http://www.nifino.com/why-are-navigation-systems-for-pedestrian-navigation-different/`
(Cited on page 42.)

[Ppe15] Ppengeospatial.org: Glossary of Terms - L. (2015). `http://www.opengeospatial.org/ogc/glossary/l`
(Cited on page 11.)

[Ree79] Reenskaug, T.: MVC Xerox PARC 1978-79. (1979)
(Cited on page 43.)

[Sha14] Shamah, D.: How big data means big changes for location-based services. (2014)
(Cited on page 23.)

[Sor14] Sorrell, S.: Mobile Context and Location Services. (2014). http://www.juniperresearch.com/researchstore/enabling-technologies/mobile-context-location-services/navigation-tracking-social-local-search
(Cited on page 19.)

[VP02] Ververidis, C. ; Polyzos, G. C.: *Proc. 1st International Conference on Mobile Business (M-Business 2002)*. 2002
(Cited on page 8.)

[Wik15a] Wikipedia.com: Geofence. (2015). https://en.wikipedia.org/wiki/Geo-fence
(Cited on page 16.)

[WIK15b] WIKIPEDIA.COM: Geotagging. (2015). https://en.wikipedia.org/wiki/Geotagging
(Cited on page 15.)

[Wik15c] Wikipedia.com: Location-based service. (2015). https://en.wikipedia.org/wiki/Location-based_service
(Cited on page 11.)

[Zah15] Zahradnik, F.: Location Based Service. (2015). http://gps.about.com/od/glossary/g/location_based_service.htm
(Cited on page 7.)

[ZGR14] Zurbaran, M. ; Gonzalez, L. ; Rojas, P. W.: A Survey on Privacy in Location-Based Services. (2014)
(Cited on page 62.)

[Zic13] Zickuhr, K.: Location-Based Services. (2013)
(Cited on page 7.)

[ZWJH10] Zhu, T. ; Wang, C. ; Ja, G. ; Huang, J.: Toward Context-Aware Location Based Services. In: *International Conference on Electronics and Information Engineering* (2010)
(Cited on page 35.)

[ZXM13] Zheng, Y ; Xie, X. ; Ma, W. Y.: GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory. (2013)
(Cited on page 15.)

All links were last followed on August 11, 2015.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature