Technical University of Denmark

Master Thesis in Computer Science and Engineering

Presence determination and prediction in the context of an intelligent home

Student

Andreas Rask Jensen – s083165

Supervisor

Christian D. Jensen

October 2014

Abstract

Determining presence and expected presence using smartphones can potentially save energy. In most homes people need to manually set the temperature and switch off stand-by power consuming devices. People living in those homes are required to make active decisions and remember to perform those actions when it is efficient to do for instance when leaving for work or going on vacation. Some homes have automation systems that can have a daily or even weekly schedule, but this is fixed schedule that cannot be changed easily. By using carried devices such as smart phones will it be possible to find a dynamic weekly schedule that would allow a home automation system to automatically make decisions based on expected presence?

Smartphones are carried by a person most of the time which enables an app to periodically report their location to a server. This information would allow a prediction framework to calculate an expected schedule showing when presence in a house is expected. Disclosing the location is however a concern for a lot of people and insurance companies, therefore the goal is to make a system that does not use information traceable to any persons or locations while still being able to make qualified predictions and determinations on presence in their home.

In the project data was collected from 13 anonymous test subjects in a period of three months. This location data showed that clear weekly patterns applied for some of the test subjects making it possible to predict periods of presence and absence. The data also showed that for test subjects that have Wi-Fi enabled the accuracy of the measurements are better and that the fact that the test subject's smartphone is connected to home Wi-Fi can be used as an indicator that the person is at home.

Table of content

Figure index

Table index

1. Preface

Green initiatives are needed more than ever. Political ambitions are set in Denmark for 2020 where a reduction of greenhouse should reach 40% compared to 1990. Some of the focus areas are transportation, agriculture and buildings. This will require an effort by the government to make legislations that require new buildings to be low-energy and renovation of existing building to be affordable and feasible. Private persons also need to be involved in lowering their emissions. They can do it by making "green choices" in everything from grocery shopping to selecting windows for their new house. The problem calls on intelligent solutions that are able to do save energy without lowering the comfort. Intelligent and affordable solutions have to make their way into private homes to save energy without being expensive and decrease the living standard significantly.

In Denmark the stand-by consumption makes up approx. 10% of the combined energy consumption in private homes[1], and an American study has shown that between 14% to 25% energy can be saved when lowering temperatures during the night[2]. Both of these energy saving initiatives require humans to perform actions, but what if these settings could be done automatically based on the presence of the people living in those houses?

1.1. Problem statement

More and more people install more and more controllable systems[3] in their homes and the Internet of things, as it is often referred to, is becoming reality. All these systems that has sensors and actuators that are not tightly coupled with a static cause and effect relation, but instead has the possibility to be reprogrammed and to listen to virtual sensors such as a software program that monitors behaviour over time and uses this behaviour to predict what temperature the inhabitant prefers and controls the thermostats dynamically.

The above example is reality and on of the applications of this is from NEST[4] that is a thermostat that learns from the inhabitants' behaviour and it is a perfect example of the direction in which controllable systems are going. These applications are trying to bring context into their knowledge base and use this context to provide an enhanced experience to the user. This enhancement is provided by using new kinds of sensors that get their input from software calculations instead of a voltage. This kind of sensors or knowledge requires calculation and that entails some form of power consumption, which is not necessarily a desired property. In a smart phone this consumption leads to lower battery life and in households it leads to a bigger electricity bill. In that context it is relevant to consider that the energy used to make these calculation should be more than saved with the initiative over a period of time. In

[1] http://www.dongenergy.dk/privat/energitips/sparetips/tvogelektronik/Pages/standby.aspx
[2] Szydlowski, R.F et al http://www.osti.gov/scitech/biblio/10124579
[3] In this context a controllable system is defined as a system with 1 or more sensors or actuators that is able to receive commands from 3rd party software application via an API.
[4] https://nest.com/

the case of NEST it means that the energy saved during uninhabited hours should make up for the electricity used to power the controller and any aiding services used for calculation for instance in the cloud.

1.2. Delimitation

The goal of the project is to make a system that can record the movement of persons using measurements from smart phones. These measurements must on no account disclose personal information about locations or persons.

The project uses a home automation context and it is therefore important to consider the how the gathering of data relate to actions in the home automation systems, and how the information gathered in this project can be used in home automation. It will not be possible and it is not the goal that a home automation system will be integrated, but this project will serve as a proof of concept that smartphone location data can be used to determine and predict presence.

Heating and cooling systems usually require a significant amount of time to reach its set point and therefore presence information will be usable to preheat a building. In this project this will not be covered because it requires knowledge about building constructions and dimensioning of such systems.

No actual integrations to hardware systems will be made, but they will be considered as they are eventually going to be on the receiving end of decisions made on the data gathered. In this report home automation systems are used as an abstract system with generic capabilities.

Due to the low number of test subjects the data gathered in this report is not statistically evident and does only serve the purpose of showing that data can be used to predict presence.

1.3. Term clarification

This section is a collection of terms and definitions that can potentially be ambiguous and thus misinterpreted or cause confusion. To clarify the meaning and to avoid confusion these terms have been defined below.

Smartphone

A smartphone is defined in the Oxford dictionary as:

> *"A mobile phone that performs many of the functions of a computer, typically having a touchscreen interface, Internet access, and an operating system capable of running downloaded apps."*[5]

[5] http://www.oxforddictionaries.com/definition/english/smartphone

This definition (and any other dictionary or encyclopaedia definition) is not as rich as desired for the purpose of this project. Therefore the following definition will be used:

> *"A mobile device with Internet access via WIFI and the following sensor capabilities: geographic location and rotation running an operating system capable of running 3rd party apps. A device usually carried by the owner as this person's primary personal digital assistant."*

The main differences between those two definitions are the sensor requirements that are able to capture some physical context and the sense of usage where it is implied that the location of the device is the same as the owner most of the time. This term is central to the use of smart phones in this project.

Presence

In this report presence is defined as a confirmation that a given location has presence of at least one person. Presence of any other object or being of such, as furniture, domestic robots, and pets are not considered.

The location can be any area that has clear boundaries such as a room or a plot. The term "is in" or "is on" should not be clearly defined for the respective premises. In the example of a room the definition could be that if a person is in the room then the room has presence. The problem with that definition is that it is vague and ambiguous. In the case where a person is in the door between 2 rooms, is the person then present in either rooms or none of the rooms? This question is interesting if the information is used for something. For instance if there was a function that counted the number of persons in a house it would be a problem if a person was counted twice. Similarly the lack of presence in any rooms might indicate a false uninhabited situation. In the case of this report it will not be a problem if a person is counted twice and therefore presence is defined when any part of a person is within that room.

For a plot it can be difficult to define the presence term as there are no ceiling and there isn't necessarily an enclosure. To be able to decide whether a person is on the lot there needs to be a strict definition of the height and the borders of the plot.

Modelling the dependencies between locations is also important. For instance if a house h has 4 rooms r{1-4} it should hold that is there is presence in r3 then there is also presence in h if r3 is a part of h.

2. Analysis

This section will define the problem are and divide it into smaller solvable problems. The chapter will describe how the decisions made in the analysis phase were taken and explain the consequences.

2.1. Domain

As described in the preface we are dealing with domestic life and heating and cooling systems. The domain depicts the user in his everyday life. The main element is the premises of the inhabitant. Premises are in most cases an apartment or a house, but it could be anything - in reality just the location that the user considers home.

A home has a Wi-Fi network that *can have* multiple access points. For simplicity multiple APs will be treated as one Wi-Fi network. A premise has several persons living there called inhabitants and a premise can have intelligent systems. In a real scenario at least one intelligent system is required for the project to make sense, but in this project a home does not necessarily have to have any intelligent systems as no actual integrations are made. The intelligent systems have sensors that provide information about the state of the house and actuators that can perform operations such as turning on light or changing the set point on a thermostat. For simplicity the actuators are left out as no actual decisions are made.

The persons could have a smartphone, which can connect to a Wi-Fi hotspot. The smartphone also have various sensors that provide information about the context of the phone. The sensors are the source of knowledge and they have different possibilities to report information about presence in a house.

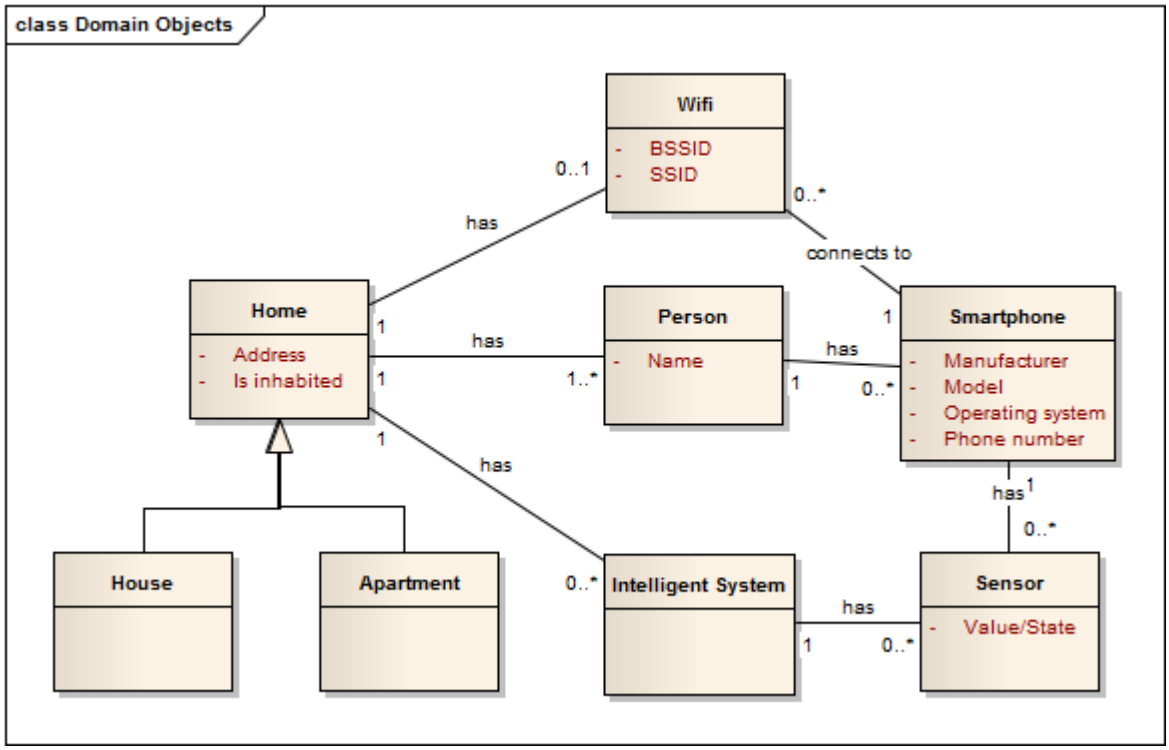


Figure 1 - class Domain Objects

2.1.1. Homes and inhabitants

In real life a smartphone usually belongs to a single person and used only by that person. It is possible to share a phone this is not possible in the context of this project due to this project's definition of a smartphone: being carried along a single person most of the time.

It would be a possibility to remove the notion of persons and instead treat smartphones and person as one entity. By doing this, the complexity of dealing with individual persons disappears and instead only devices need to be considered. If a person receives a new device this will effectively count as a new person. The complexity in the data is lower as there are fewer objects and thus relations between those. The result will however also be that patterns for a given person can not be inferred between two devices as there is no way of identifying that they belonged to the same person and there is expected to have the same pattern even though the device is new. An example of this is of Bob and Alice lives in a house and one day Bob breaks his phone and gets a new one. In the scenario where there are no persons this is just a 3rd device that is connected to the house whereas the second scenario allows for the system to know that this new device belongs to Bob and is expected to have the same patterns of Bob.

2.1.2. Intelligent systems

Home automation systems in this project are domestic systems that have sensors that they receive data from and actuators that they can perform actions on. These actions have to be programmable so that they are able to receive input from other systems and perform actions based on these. Interaction with

home automation systems can happen in numerous ways and are defined by the physical possibilities and the capabilities to use a certain protocol. Two examples of this is the Modicon M340 PLC that has an ethernet port and communicates via Modbus and the IHC from Schneider Electric that also has an ethernet interface but communicates via SOAP over HTTP. In this project the integration to actual home automation systems will not be considered however the integration to the systems still needs to be considered. This integration is thought as a HTTP capable software application accessible from the Internet. HTTP capable means that it is able to send and receive HTTP requests in a predefined format. By having this abstraction the actual integration can be omitted and the focus can be transferred to knowledge and how it is distributed along the system rather than what this knowledge is used for.

In this project intelligent systems could play an important role because information from these systems could serve as a verification that a person was in fact at home at a given point in time. Due to several complexities[6] data from these systems will not be collected, however in the analysis and design sections of this report the modelling of such systems will be considered due to the fact that it could provide vital information if implemented.
This section will examine home automation sensor types and analyse how the respective sensors can provide information about presence.
Motion sensors
This kind of sensor senses motion from moving objects or from heat emitted from persons in the house. Such a sensor is a really good indicator for presence but can report false positives with the presence of pets or domestic robots such as robot vacuum cleaners. The latter threat is however not a problem for the so-called PIR (passive infrared) sensors that measures infrared heat emission.

Latent presence detection
Sensors that do not directly report presence can still provide latent presence information simply by changing a value. For indirect sensors a manual decision about how to infer presence from a given sensor value, has to be made.

Magnet sensors
Provide information about whether doors and windows are open. If a window is opened or closed and there is no motor attached to perform this operation it can be inferred that someone is at home. Likewise it is possible to infer a change in state when the door is opened. If the house is uninhabited and the door is opened it might impose a change from an uninhabited state to an inhabited one.

Push buttons
The use of push buttons indicates that someone is in the house. If these push buttons can be emulated from an app this can however be a false positive. If there is no way to distinguish the actual push of a button from the same action being performed programmatically no inference can be made.

[6] For elaborate explanation see appendix 8.2.

House alarm

Events from an alarm system can also be extremely useful for presence detection. An alarm system usually has states such as activated, deactivated and partially activated. Partially activated meaning that certain rooms can have movement without the alarm going off. Alarm systems typically use motion sensors and magnet sensors, so the additional information that can be obtained from this "sensor" or system is the events where the state changes. When activating the alarm it could mean that the house is going to be uninhabited for a significant amount of time, and when it is partially activated it could mean that some rooms will not inhabited. This information can aid the optimization of heat profiles.

2.1.2.1. Smartphone sensors

As explained earlier smartphones have a lot of capabilities. In this section smart phones will be analysed to provide a clear picture of what they are and how they can be used.

One of the biggest differences between a laptop and a smartphone is the sensors available. Sensors that can provide information about the physical context. This section will look at the different possibilities and decide whether the information provided is useful for this application.
The sensors are divided into 2 categories: actual sensors and state. The items in state are not actual sensors, but the state of the smartphone in a given context. When reading values from the state the information can be treated as sensor values in the system.

State

- Wi-Fi state: Is the phone connected to a wireless access point and is that access point the one at the person's home? Information about whether Wi-Fi is activated or connected is not important unless it is connected to the home Wi-Fi. In that case it will be possible to determine that the person is home.
- Phone id can be used for making sure that the device is always identified, as the same device is e.g. the app is reinstalled. This id can be the phones serial number or the MAC address of the Wi-Fi NIC (network interface card).
- Charging: It might be interesting to know if the phone is charging or not. When the phone is charging it cannot be in a pocket, which may mean that it is less probable to be carried.
- Battery level: The battery level may affect the behaviour of a person or the device. An example of the first is that if the battery is low then the person might go home to charge it. An example of the latter is that if battery is low a 3rd party app might turn off data to save power. In that case further logging will not be possible. Due to high uncertainty this feature will not be used in this application.

Sensors

- Location: Location is the most important sensor of all and actually more than one sensor can provide location information. The location sensor delivers a location fix. Such a fix can consist of a lot of properties with latitude and longitude and the most important ones. Other features are:

    - Bearing: In which direction is the person travelling (not to confuse with magnetometer - see below)
    - Speed: the current speed of travel. If this value is high it must be expected that a later location fix is different. Not available for all kinds of location providers.
- Motion of a device can be measured in different ways. If a phone is in motion it has to be due to some kind of force moving it. It is expected that motion can provide information about whether the phone is carried or not and is therefore very useful.
- Magnetometer: The device's orientation relative to magnetic north can be used for navigation when looking at the device held in a specific way, but when the device is in an unknown direction this information is not useful. The bearing from a location fix along with motion of the device makes this feature unimportant.
- Proximity: The proximity sensor can determine how far an object is from the phone and is often used to determine if the phone is held to the ear while talking. Another situation can be when the device is in a pocket. Both examples provide meaningful information, but in both cases the motion sensor will also be able to determine that the device is in possession of a user.
- Lux: As for the proximity sensor this sensor could hint that the phone is in a pocket or indicate whether the person is in a lit room, which might open the possibility to infer if the person is sleeping. This feature is however not interesting for this application.
- Sound: Presence of sound or at least voices shows that persons are present, but this kind of sensor also infringes privacy with the possibility that a recorded conversation is shared. Therefore this sensor is not interesting.

2.1.2.2. Smartphone operating systems

Having mentioned the operating system capable of running apps, let's examine the domain of available smartphone operating systems and their respective market share.

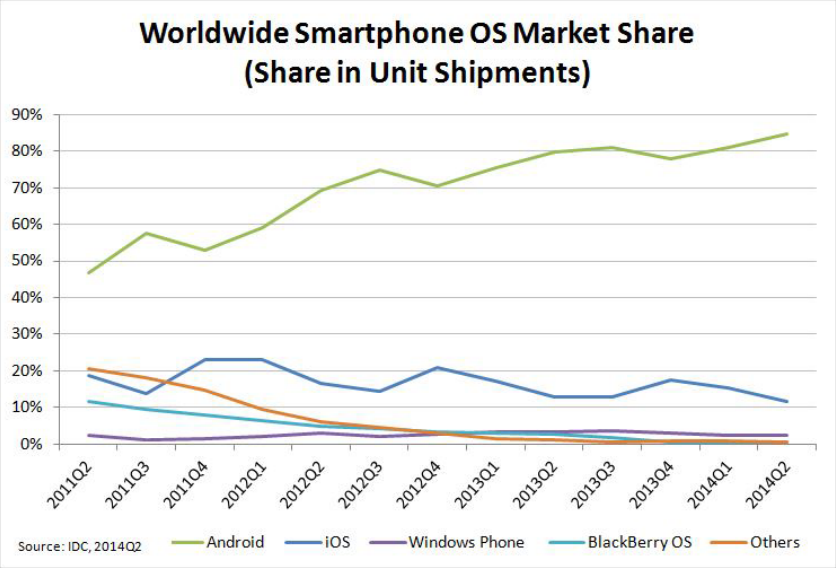


Figure 2 - Smartphone market share 2011-2014[7]

Although the graph does not take the definition into the account some devices in the chart does not necessarily meet the criteria from the definition. With that in mind there is a clear market leader in Android with iOS as the closest competitor. When designing and choosing platform for the operating system, this information will be taken into consideration.

As the sensors are central to this project this section will analyse the sensor availability, capabilities and limitations of the respective operating systems.

Sensor availability

Sensor	Android	iOS
Location	GPS, Cell-ID, and Wi-Fi[8] The location of Wi-Fi access points are collected and used by Google to determine a location at a later stage[9]. Android treats location in 2 levels: coarse and fine which corresponds to data coming from either GPS or network where network is either cell-id or Wi-Fi. These permissions have to be asked for in the configuration in the app[10].	Cellular, Wi-Fi, GPS and iBeacons[11]. iBeacons are physical devices with a predetermined location that can be used for locating the user when an iBeacon is within range of an iOS device[12]. Apple uses crowdsourcing to collect locations of Wi-Fi hotspots[13] Apple does not distinguish the accuracy in the permissions and thus if location is allowed by the user then location fixes with any precision can be obtained. The accuracy is specified in meters and the operating system will then pick the appropriate provider[14].

[7] http://www.idc.com/prodserv/smartphone-os-market-share.jsp
[8] http://developer.android.com/guide/topics/location/strategies.html
[9] http://www.google.com/googleblogs/pdfs/google_submission_dpas_wifi_collection.pdf
[10] http://developer.android.com/guide/topics/location/strategies.html
[11] http://support.apple.com/kb/HT5594
[12] https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf
[13] http://support.apple.com/kb/HT5594
[14] https://developer.apple.com/videos/wwdc/2013/#307

Motion	Android primarily relies on the hardware sensors accelerometer, which measures acceleration applied to the device (including gravity) and gyroscope that measures the changes in orientation along the phones 3 axis. The accelerometer is more widely available on devices and uses about ten times less power.[15]	In iOS both accelerometer and gyroscope is available using the Core Location framework.[16] This framework also allows for detection of the users activity such as walking, running or stationary.[17]
Phone id	The mac address of the network interface can serve as a phone id. It can be obtained from the WifiManager provided that Wi-Fi is enabled.[18]	Apple does not allow the app to read the mac address. Instead an id is generated per vendor and can therefore uniquely identify the device to the vendor.[19]
Charging state	The charging state can be obtained through the system-wide broadcast that needs to be subscribed to. The states are: charging, and not charging and if charging it can be AC or USB.[20]	The charging state can be determined through the UIDevice and is either :Unknown, Unplugged, Charging or BatteryStateFull[21]
Wi-Fi info	Information about the currently connected Wi-Fi access point can be obtained through the WifiManager[22] The id of the access point is the BSSID, which is the mac address of the access point.	The BSSID can be obtained through the CaptiveNetwork class in the System Configuration framework.[23]
Device information	The device model and manufacturer is available through the Build class, but they are optional and may be empty.[24]	The specific version cannot be obtained but information about whether it is an iPod, iPhone or iPad is available in UIDevice.model.[25]

Table 1 - Sensor availability

(B)SSID

Wireless networks identify themselves using an SSID (service set ID) and a BSSID (basic SSID). The SSID identifies the network (potentially multiple access points (APs)) whereas the BSSID identifies a single AP.

---

15 http://developer.android.com/guide/topics/sensors/sensors_motion.html

16 https://developer.apple.com/library/ios/documentation/coremotion/reference/cmmotionmanager_class/Reference/Reference.html

17 https://developer.apple.com/videos/wwdc/2014/#612

18 http://developer.android.com/reference/android/net/wifi/WifiManager.html

19 https://developer.apple.com/library/ios/documentation/uikit/reference/UIDevice_Class/Reference/UIDevice.html

20 http://developer.android.com/training/monitoring-device-state/battery-monitoring.html

21 https://developer.apple.com/library/ios/documentation/uikit/reference/UIDevice_Class/Reference/UIDevice.html

22 http://developer.android.com/reference/android/net/wifi/WifiManager.html

23 https://developer.apple.com/Library/ios/documentation/SystemConfiguration/Reference/CaptiveNetworkRef/Reference/reference.html

24 http://developer.android.com/reference/android/os/Build.html

25 https://developer.apple.com/library/ios/documentation/uikit/reference/UIDevice_Class/Reference/UIDevice.html

As this application needs the location of a single AP the BSSID is used. The BSSID is the MAC address of the AP and the SSID can be chosen (and changed) by the network administrator[26].

Android

Android is an open source operating system owned by Google and developed in collaboration with the organization Open Handset Alliance The Android platform lets developers make apps that utilize the different APIs related to the OS and the sensors that the device has to offer. Unlike iOS, Android is ported to many devices from different vendors. The Android platform and different frameworks provided in the operating system handles the hardware differences and presents them uniformly. To do this Android puts forward a specific requirement that has to be met when implementing these frameworks. The sensor framework is a good example of this. Android is designed to be able to contain a variety of predefined sensors and to have none or multiple of those. To cope with this Google has made a document that defines the requirements of a valid Android device implementation[27]. This guide operates with the verbs "MUST", "SHOULD" and "MAY" that are used to indicate how important a given feature is. Examples of this are the sensors where the accelerometer, magnetometer, GPS and gyroscope "SHOULD" be included where barometer, thermometer, photometer and proximity sensors only receive the importance of "MAY" be included. This means that it is not guaranteed that all Android devices have the sensors this application wants to use.

OS Versions

Having released versions 2.2 through 4.4 in less than 4 years Google releases the different Android versions quite often. These releases however are not guaranteed to be installed on the devices immediately. Following the Android version graph below it can be seen that for any new version it takes time before it becomes the dominant one and the phase-out seems to take even longer. There are 2 reasons that can explain this: there is a natural sense of new versions having higher hardware requirements and thus some models reach a point where they can simply not upgrade to a newer version of Android any more. The other possible reason is the fact that some device manufacturers make their own Android distributions which means that they are in control of when and if a new version of the operating systems becomes available. HTC, Samsung and Sony each have their own distributions and manage their own updates. This means that these manufacturers decide how long the devices will receive new versions of Android.

[26] http://www.juniper.net/techpubs/en_US/junos-space-apps12.3/network-director/topics/concept/wireless-ssid-bssid-essid.html
[27] http://static.googleusercontent.com/media/source.android.com/en//compatibility/android-cdd.pdf

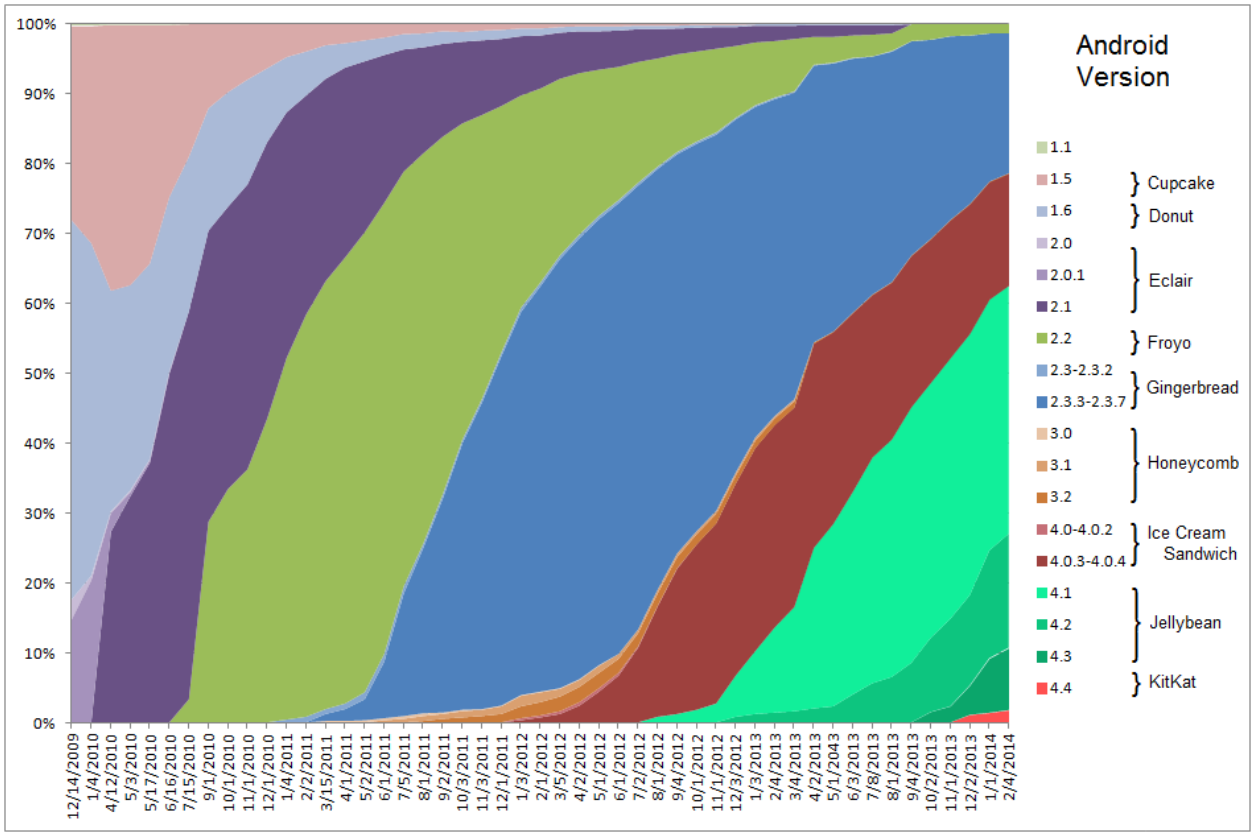


Figure 3 - Visualization by Fjmstak using data from Android Developer Dashboard[28]

Choosing a version for this project has 2 conflicting goals: supporting the newest features of the operating system and allow as many devices to be compatible with the app. Version 2.1 seems to be 0 on the graph since September 2013 but the explanation for this is that Google released a new version of the Google Play store[29] which is not available for version 2.1 and backwards. Therefore 2.2 will be the choice for the app developed in this project. Another reason for picking 2.2 rather than any older version is that API level 8[30] that comes with Android 2.2 has a "New bug reporting feature for Google Play apps enables developers to receive crash and freeze reports from their users. The reports will be available when they log into their publisher account." This feature could save time in an experimental project like this.

Threats

This section will try to describe any threats in the operating system or system environment that might counteract on the requirements of the project. In the Android operating system, apps are allowed to control the environment, which eventually makes certain features unreliable because they are not predictable.

The threats are hard to describe in full but 2 concrete examples can illustrate the problem:

[28] https://commons.wikimedia.org/wiki/File%3aAndroid_historical_version_distribution.png
[29] http://developer.android.com/about/dashboards/index.html
[30] http://developer.android.com/guide/topics/manifest/uses-sdk-element.html

Example 1: Sony Experia Stamina:
The stamina mode is something Sony has developed to aid power saving on a device. It works by turning off Wi-Fi and 3g when the screen is off. When there is no Internet connection it will not be possible to log data.

Example 2: Juice Defender
Juice Defender is an app by Latedroid that allows the user to turn off for example Wi-Fi and 3g based on user defined rules such as a schedule, battery charge status or battery level. Just as the prior example this makes the logging unreliable because it can be shut off at any time.

These examples will degrade the quality of data and may even make it impossible to infer anything from the sparse data that can be expected. The users participating in the project will be asked not to use this kind of service to remove these threats. In the future it might be possible to find ways that these services can coexist with this project, but for time being this is considered out of scope.

In the operating system itself there are also possibilities that the user might counteract on AtHome. As an example a user can turn off location services, which makes logging impossible.

iOS
iOS is an operating system developed and maintained by Apple unlike Android there is only one manufacturer of devices namely Apple itself. This gives Apple full control of the device portfolio and their capabilities with regards to sensors and connectivity. Updates of the operating system can be rolled out directly to the devices and Apple can make a decision of which devices to support for each update. This effectively means that a device can be updated to the newest version of the operating system as soon as it is released from Apple and provided that the device is still supported. When choosing the version to support for AtHome it is therefore only a matter of supporting devices. The changes that relate to AtHome between version 6 and 7 respectively is the introduction of Region Monitoring[31] which allows the app to perform actions when entering or exiting a predefined geographical region. More about this feature in the design section. This decision allows iPhone 4 and onwards where iPhone 3(GS) will not be supported.

[31] https://developer.apple.com/library/ios/releasenotes/General/WhatsNewIniOS/Articles/iOS7.html

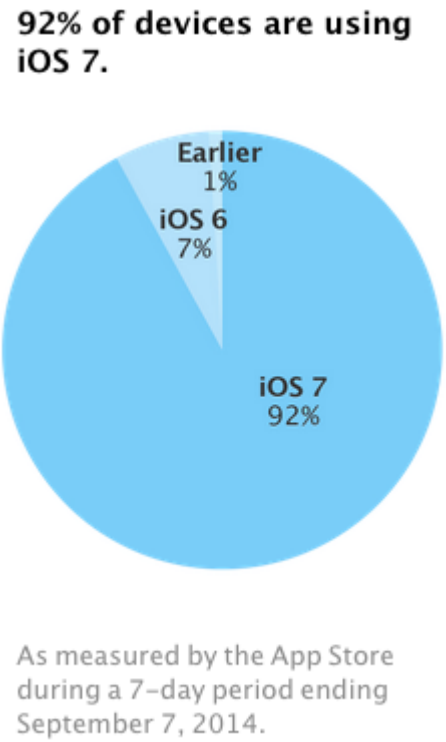


Figure 4 - Screenshot from App Store distribution[32]

As a platform iOS platform is more closed than the Android platform. This postulate will be accounted for later in this chapter but as a quick example let's examine location services. Location data cannot be requested from a specific sensor, but the OS will decide what sensor is needed based on an accuracy specified by the developer.[33] The approach of abstracting decisions from the developer is general and limits the ways developers can make new kinds of applications such as AtHome. The permission to have a process that starts with the operating system is limited to VoIP apps[34]. If however the app does not contain this functionality it will not get approved in the app store. The approval in the Apple App Store has several steps and requires validation to "avoid applications that degrade the core experience of iPad, iPhone, and iPod touch … and should conform to the iOS Human Interface Guidelines". This process ensures that Apple has approved the quality of the apps but it also indicates a strict and possibly time-consuming procedure[35].

[32] https://developer.apple.com/support/appstore/

[33] https://developer.apple.com/library/ios/documentation/userexperience/Conceptual/LocationAwarenessPG/CoreLocation/CoreLocation.html

[34] https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/AdvancedAppTricks/AdvancedAppTricks.html

[35] https://developer.apple.com/appstore/resources/approval/index.html

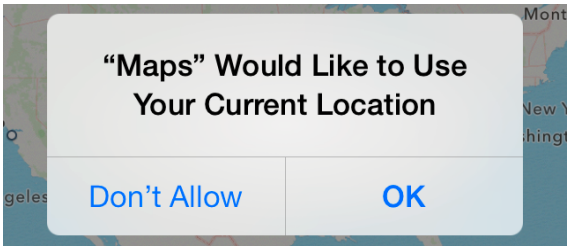


Figure 5 - Location prompt from iOS[36]

iOS does not allow apps to control the Internet connection or in other ways limit AtHome in logging. The user has the possibility to deny an app of location data, or turn off location services for the entire device. Further the phone can also be put into airplane mode, which turns off Internet connectivity.

Logging

The central part of this project is logging sensor data and the end goal of this is of course to have all the data gathered in one place for data processing. The different devices that perform the logging are smart phones and home automation systems. As previously described the home automation systems are treated in an abstract way such that it can be guaranteed that they can send data in a predefined format and protocol.

To fully understand the platforms and to be able to make a decision on how to perform the logging the properties of the platforms needs to be clarified.

Home automation

In the home automation abstraction made for this project the home automation system always notifies when a change in a sensor value happens. This is of course achieved differently in the actual systems by polling and comparing or by native functionality in the API of the system. In this project a fictive middleware platform will make sure that the differences in the systems can be treated abstractly. The processing power of the middleware uses power, and requires a server to run on. It is possible to have the middleware run on a cloud platform to minimize the hardware requirements so instead of having a server in all homes, a central server will run the software and thereby save power. This poses some privacy concerns, but this is out of scope of this report.

Data connectivity

In this project the home automation systems are connected to a wired Internet connection and the connection is assumed to be reliable. Some Internet subscriptions have a monthly usage limit and although these limits are relatively high the traffic has to be kept to a minimum.

How and when to log

Considering that the Internet connection is stable and the processing power is less important, logging can be performed when new values are available instead of repeating the entire state periodically. The

[36] http://support.apple.com/kb/HT5594

result of this logging will be event based and require data storage that can handle events of different types and values.

Smart phone

Choosing how to monitor and transmit sensor readings to the Orchestration Service is central to the application. There are a multitude of constraints and considerations to be made in order to choose an appropriate strategy. These constraints and considerations have to be dealt with in respect to their impact on the data quality and the user perception and willingness to use the AtHome system. The correctness of measurements is paramount to the perception of AtHome, but it also has to be considered that if the data quality is high but the app drains the battery within a few hours it must be expected that people will be resistant to using it.

Monitoring sensors and sending and receiving data uses power and smartphones are known to have limited battery life and users of smartphones concerned about the battery consumption of an app. Regardless of operating system there is a trade-off between how often data is sampled and sent and the battery life. Therefore every step towards prolonging battery life that can be taken is important, however it must never compromise data in such a way that it becomes unusable. In the end it is a trade-off between a lots of things:

	Frequent logging	**Sparse logging**
+	High data quality	Battery life
-	Short battery life Users reluctant to use app High data usage	Low data quality

Table 2 - Battery trade-offs

A smartphone uses wireless communication for receiving and transmitting data to the Internet, and since it is expected to move around a lot the connectivity might be lost from time to time. There are usually 4 ways smartphones can access the Internet. Wi-Fi, Bluetooth, USB and cellular data. Bluetooth and USB are rarely used, so for this analysis they are put in the same category as Wi-Fi because the properties of these networks are, in context of this project, roughly the same as Wi-Fi.

Cellular data

The details of how a cellular network is out of scope of this project, but therefore the concept of a cellular network is still relevant. Cellular data is obtained through a mesh of radio towers. These towers or base stations have limited range but are usually placed close enough that a call is not dropped when switching from one base station to another. In less populated areas where the base stations are longer apart it is possible to find places without possibility to connect to a base station. In such a case there is

no Internet connection and logging is impossible. When connecting to the provider's network is not possible, for instance when abroad, some operators provide a roaming option. Although this kind of service makes it possible to log from abroad these services are usually charged at high rates. Actually network traffic is also charged on the home network, but at lower rates and usually as a part of a monthly traffic limit deal. In this project these properties mean that logging is not always possible via cellular data and when it is it has to be kept to a minimum so the monthly data usage is as small as possible. If it gets too big the users might be reluctant to use the app.

Wi-Fi

Wi-Fi hot spots are readily available almost everywhere but in contrast to cellular data the networks are not run by carriers but by private people and businesses. Each network requires you to actively connect and in most cases input a passphrase to gain access. This means that the Wi-Fi connectivity happens when a user is visiting a location for a while. There is a possibility to connect automatically when a known network is within range so sporadic connections might happen although most Wi-Fi traffic will indicate that a person is staying at a location for a period of time. Wi-Fi traffic is also subject to charge but the limits are relatively high and do not need consideration for this project.

When the user is abroad the logging is as explained expected to be sparse which might be a threat to the prediction. If the research participant lives less than an hour away from a roaming border and the person is driving home the first data point in a series will appear: *60 minutes - logging interval* before the person is at home which might not be enough to create the desired comfort conditions. A way to accommodate that problem will be to use data from other sources to aid the prediction. An example of this could be a calendar integration as explained in the preface. For most cases the prediction will be hard to make as the user has probably not been to that location a lot of times and therefore prediction was already hard. Another argument is that the user is far away for a long period of time and therefore a manual signal would be better, and the user would remember because it is special: looking forward to coming home.

When should logging happen?

When deciding on when and how to log, there are more possible strategies. Having examined the transport possibilities it might be tempting to choose to log all changes locally and transmit them when a reliable Wi-Fi connection is available. Although that possibility guarantees perfect data it consumes power to always listen for changes and data will not arrive before a user is connected to Wi-Fi which is not guaranteed to happen. For this research project it would be possible to aggregate several measurements and send them in larger packages when a Wi-Fi connection is available. If the prediction should work in real time however, the prediction framework needs the data when it is available in order to make timely decisions. To meet the requirements of the prediction framework and to preserve energy on the device is would make sense to make periodic samples and send the full state of the phone. By doing that it is not required to have the smartphones monitoring state constantly thereby saving power.

The result of this kind of logging will be tabular instead of event based as in the case of home automation. Both formats can be processed and used, but in different ways as they give different guarantees. Event based logging has the correct time stamps of the events that happened whereas the precision in the interval decides the precision of the data.

Feature extraction

The data collected can hold latent information that can be inferred using the data itself or other sources. This section will look at the features that might be extracted from the data.

Weekday

As explained above the weekday can be extracted from TimePoint.

IsHoliday

Using the country of the person or a calendar integration it would be possible to add this feature. At the moment unfortunately none of this information is available.

Speed

Speed will only be provided from the location framework then using GPS, therefore speed might be inferred from the distance travelled and the time since last log.

Data treatment and transfer of knowledge

As the system gets more and more data it will be possible to determine if the user is at home and make a qualified guess about if the home is inhabited at a given point in time. To save energy this information is sometimes needed in advance for instance with HVAC systems. Therefore it would not be enough only to tell the current state but also provide prophecies about future presence.

Three different scenarios have been identified and will serve as examples when deciding how to provide information:

Inhabited (active)	When at least one person is in the home and the person is awake e.g. moving around. In practise this is done with a time threshold.
Inhabited (hibernate)	When at least one person is known to be at home but no activity has been detected. Can be activated for individual rooms.
Away	This state is activated automatically when there is no presence in the home. This should be inferred from daily routines combined with a time threshold since last (latent) sensor event. When switching to this state an estimate of next expected inhabitation has to be calculated. If the state is set manually the user provides this time.

Table 3 - Three different scenarios

The table above shows the different states that a house and rooms can be in. This state is either manually set or determined from historical behaviour and observed events. In the state diagram below the states and transitions can be seen.

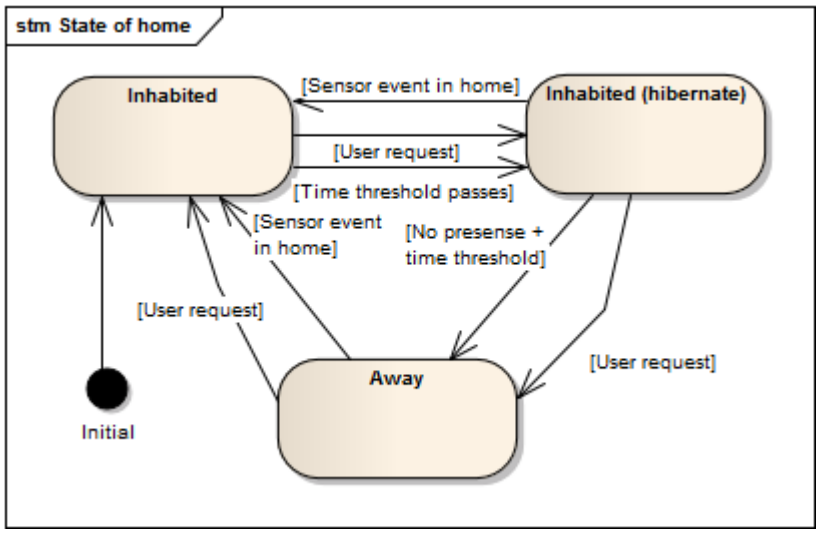


Figure 6 - stm State of home

The problem with this diagram is that for HVAC systems the sensor event happens at the moment the home becomes inhabited and the user request can be forgotten. The intent of AtHome is to free the user of thinking about when the radiators needs to be turned on.

The actual decisions have to be made in the home automation system and therefore AtHome is not responsible for calculating the time it takes to heat a room to a certain temperature. It is solely responsible for notifying when a user is on his/her way home and to create a schedule of when the home is expected to be inhabited. The home automation system will then have to use this information to make decisions.

There are 2 ways of transferring knowledge from AtHome to the respective homes:

1. The home automation system requests information from AtHome
2. AtHome notifies the home automation system when interesting things happen

Although the logging in itself maintains integrity of the inhabitants' locations the knowledge has to be transferred to a physical installation at some point in time to be able to perform an action. Due to the anonymous nature of the system a single measurement does not reveal any information that is traceable to a physical location. However when the information about the ending of a period with presence or information about the probabilities of the home being inhabited within the next couple of

hours has to be transferred to the home at some point.

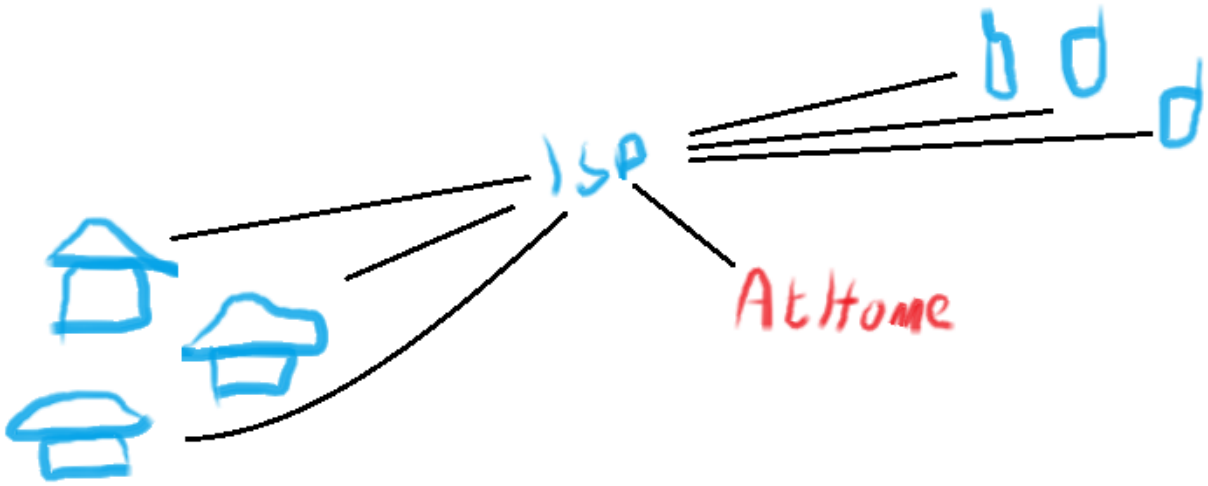


Figure 7 - Drawing of architecture with cloud or real integration

In the drawing above the entire home automation system reside inside the house meaning that the software responsible for communicating with the sensors and actuators is either a part of the home automation system or located on a computer inside the home LAN. In this scenario AtHome needs to know the IP of the home and the home needs to be able to receive data from AtHome. This requires that AtHome and the home automation system agree on a data format, protocol and that the port that is open in the firewall.

A practical example of a cloud deployment could be the one below. The diagram is from a project at DTU where two home automation systems reside inside a house behind a router with a firewall. The router has a built in VPN server using PPTP which allows for a cloud software system to interact with the components in a secure way. The cloud software system is composed by all the components inside the cloud in the diagram. In the context of this project the cloud is the thing being referred to as the home automation system. This means that the cloud becomes an abstraction of the actual hardware systems and presents itself as one home automation system even though there are actually two controllers in the installation. This example fulfils the requirements of the home automation systems for AtHome since the cloud software system is HTTP capable. The so-called “App-Service” is a web service that could request data from AtHome and receive data in a predefined format.

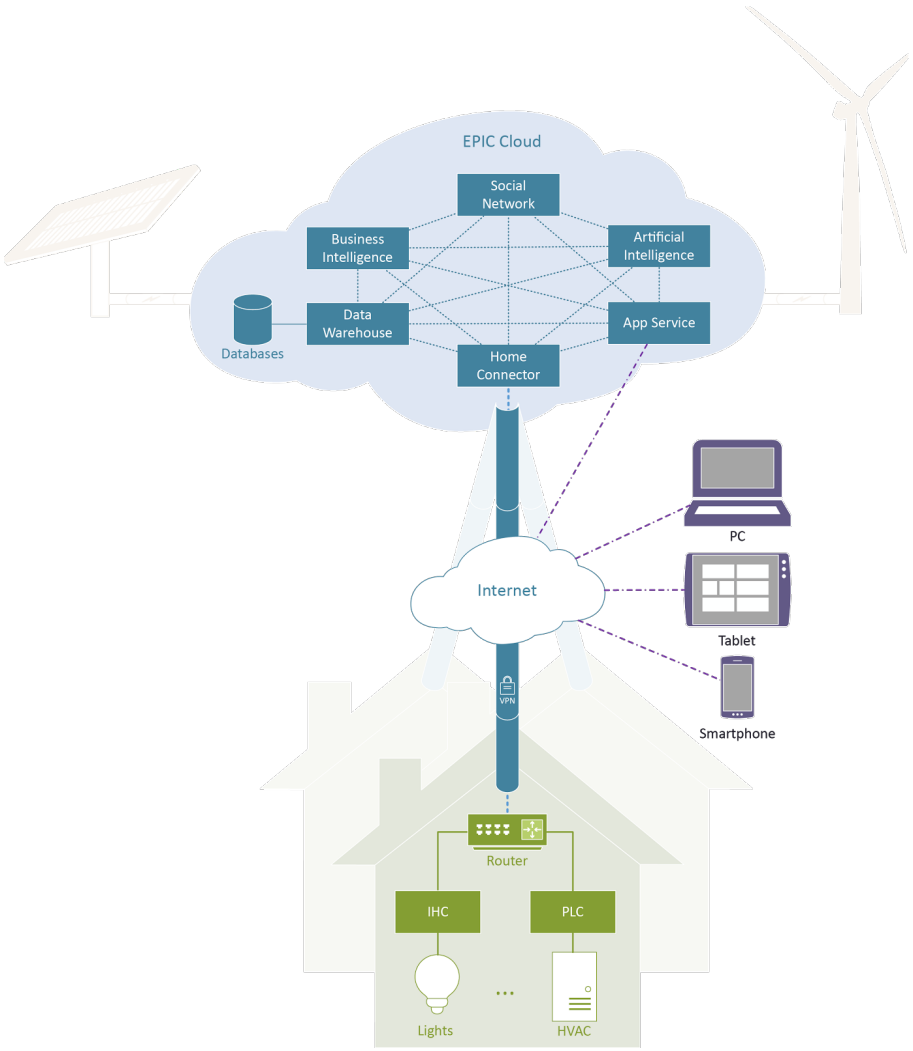


Figure 8 - Practical example of cloud deployment (C. Nilson)

2.1.2.3. Requirements

Having analysed the domain and the possibilities and limitations of the different areas it is possible to make a requirement specification for the system. The specification will cover the functional and non-functional requirements as well as define use cases and roles that perform those use cases. This section will also extract the platforms or physical interfaces needed to support the requirements. The requirements will not include the home automation area due to the reasons described in the preface.

Functional requirements

It should be possible to add and remove homes to the system.

It should be possible to add and assign persons living in those homes to the homes.

It should be possible to add and assign devices to the persons such that no device is registered for multiple persons.

The devices should be able to report or log data from the relevant sensors periodically.

It should be possible to predict how probable it is for a home to be inhabited for a given time in the future.

It should be possible to predict when a change in inhabitance occurs either from inhabited to uninhabited or the other way depending on the current state.

Non-functional requirements
No physical locations or any other information that can identify a person or location must be stored in relation to the logging data.

Use cases
In the project there are a research conductor and several research participants. Based on this knowledge and based on the section above the following roles exist for the system:

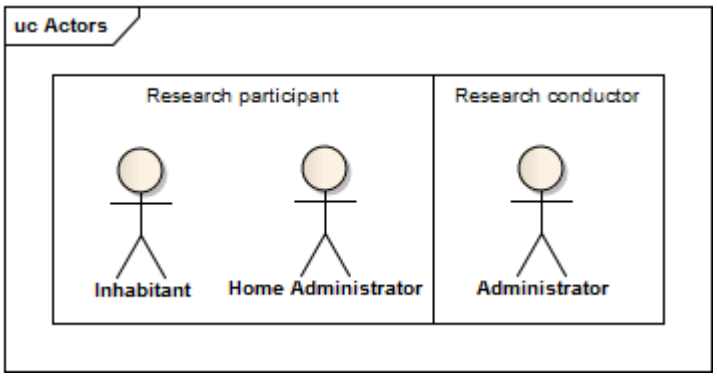


Figure 9 - uc Actors

Inhabitant: The inhabitants in a home are users in the role inhabitant of home X and can supply information related to their devices.

Home administrator: A person in a home that is responsible for maintaining the entities related to the home. A person can be inhabitant and home administrator at the same time.

Administrator: The research conductor is overall administrator. This role is the only one who can remove data. This privilege is not trusted to the research participants as data must not be removed without the accept of the research conductor.

Use cases derived from homes and inhabitants

Having introduced a need for maintaining inhabitants and their association to a home and their devices brings forth a need for a way of utilizing this. These tasks can be cumbersome to perform on a smartphone and developing platform specific ways of administering something generic takes time to develop and maintain. Therefore a separate system is introduced to handle these tasks.

In the premise administration interface the following use cases should be performed:

Figure 10 - uc Primary Use Cases

At this point explaining the individual use cases is secondary, but they will be explained in detail in the design section of this report.

The home administrator is allowed to add and update entities related to the home he/she is responsible for. Deletion of entities should only be allowed by the administrator to avoid loss of critical data. As no data contains information that could lead to a physical identification this is not considered to be a problem.

The smartphone app interface

In the smartphone app only two use cases are possible: adding the device and logging values. The use case "Log values" is also trivial at this point and will be explained in the design section. The use case add device however is different from the smartphone than on the website.

Add device

As stated in the requirements no device can be added twice for a user. Therefore it is necessary to ensure that the id of the device is always registered with the same id when added by the same person. Another desired property is that this id remain the same across installations on the same device. To embrace this, a hash value is generated from the concatenation of the person's id and a constant id of the phone. As described earlier Android will use the MAC of the Wi-Fi NIC and iOS uses the "id for vendor". By doing this the device will always appear to be the same for a given person, but can still be inherited to another family member and appear as another device.

When adding a device the device id will be calculated on the client side (in the app) and enrolled using the identifier of the person that the device should belong to.

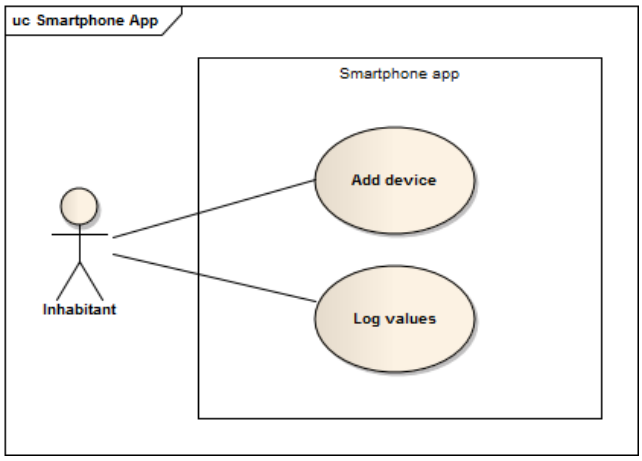


Figure 11 - uc Smartphone App

Having presented the use cases that needs to be implemented the analysis chapter is concluded. The design chapter will elaborate on how the use cases and concepts defined in the analysis will be designed and realized.

3. Design

The design chapter will seek to explain the design of the solution to the problems stated in the analysis. The chapter will be divided into logical sections each describing a component of the solution. First of all the different components will be declared and it will be explained how they interact.

As described in the analysis section the devices log values using an app to a database. This data is stored through a service that also handles registration and relating the entities in the system. This registrations is done through a website where changes to members of the household and new devices also is handled. Finally data from the database is processed in a prediction framework. These components can be seen in the diagram below:

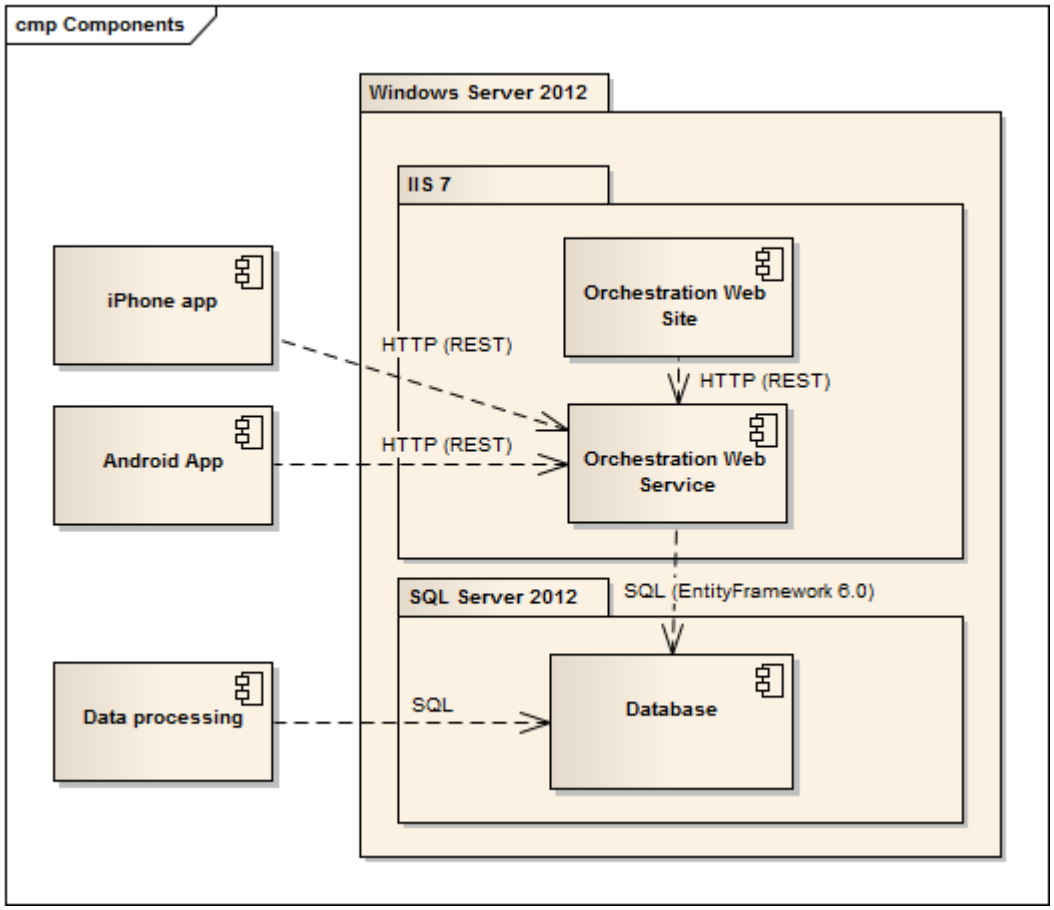


Figure 12 - cmp Components

The diagram above shows the components of the system. This chapter will explain all of the components and how they communicate with each other. The diagram above shows the state of the project at this phase where the prediction consists only of statistical analysis and evaluations. The diagram in a real scenario where the prediction is automated and shared with a home automation system can be seen in appendix 8.1.

Before describing the components of the project an important issue needs to be addressed. As mentioned in the analysis no person-traceable information can be stored. For location data that poses a big problem because a location can potentially lead to the identification of persons and this is a huge threat to the system. The following section will describe how anonymity is reached by designing privacy preserving location scheme that does not loose critical information.

3.1. Privacy

Disclosing precise location information compromises privacy and it must be expected that people will be reluctant to use services that obtain and store such data. To embrace this assumption the locations used in this project has to be anonymised. Anonymized in the sense that the location data should not be able to be traced to a concrete physical person or location.

Names can be avoided by using aliases but addresses and location requires a different approach. As location data is given as coordinates it would be an approach to add an offset like a Caesar cipher but having just a few measurements would eventually disclose the offset and all coordinates would be exposed. If however instead of using a global offset one used an individual offset the exposed set of coordinates will be limited if the offset were to be disclosed.

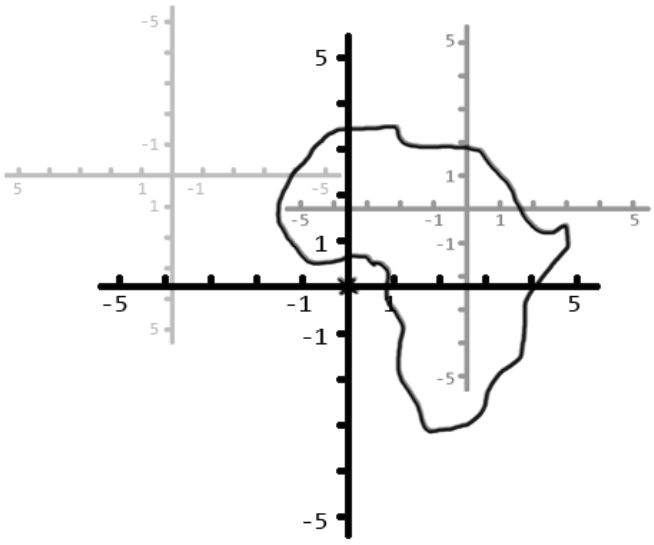


Figure 13 - Coordinate example

This individual offset could be the coordinate known as home for the individual inhabitant. Having the possibility of decrypting all coordinates to absolute locations for a single user at the time is considered acceptable, but fortunately there is a smarter solution. Given that the project only cares about time and distance it would be an idea to not store the actual locations but simply the distance to home, however that approach removes a lot of information that might be useful to make better predictions. For instance the approach does not include whether the person is static or driving or in which direction the person is driving. Consider the scenario below where two location recordings X_1 and X_2 have the same distance to home but is not the same place.

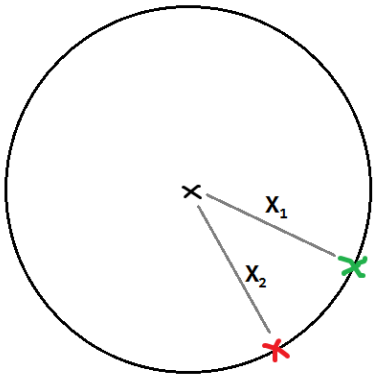


Figure 14 - Coordinate example 2

So how is it possible to enrich the distance-based location model with as much information as possible without disclosing actual locations? Consider the situation where the device logging not only knows the current locations but also stores the most recent location. By having both the current and previous location is possible to maintain some of the information that is lost by only using distances. By comparing the current location to the most recent location it is possible to determine the distance travelled and to get a sense of the speed of which the distance has been travelled however there is no direct sense of direction. When comparing the previous distance (to home) to the current ditto taking in mind the distance between those individual points it will be possible to determine an angle. This angle can also be obtained from comparing compass bearings and subtracting them.

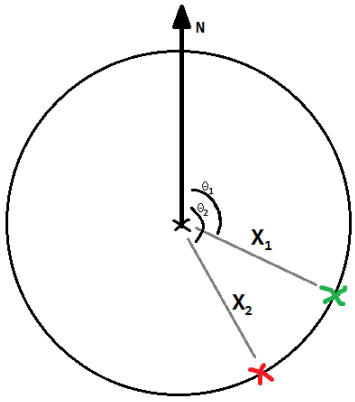


Figure 15 - Coordinate example 3

Given the angle θ as the angle between 2 coordinates and magnetic north and given the angle as either the interval [0-180] when the angle is eastbound where 180° equals due south and the interval]-180;0] when the angle is westbound it will be possible to determine the relative direction to home using the formula:

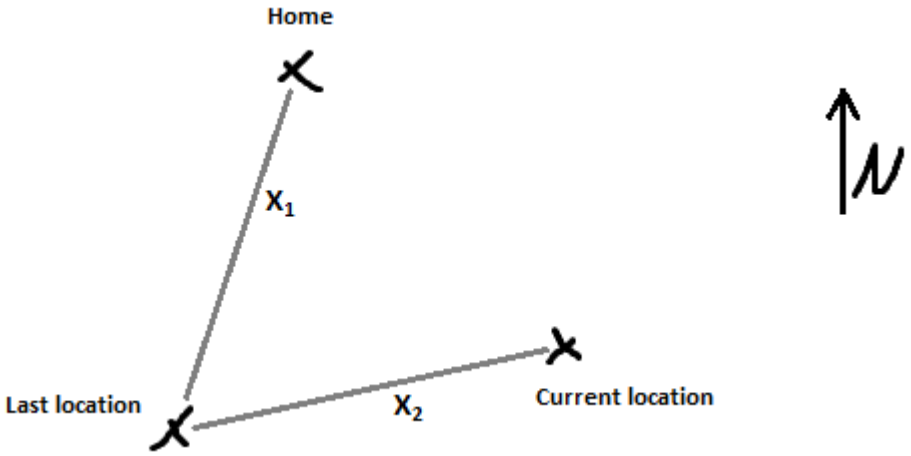


Figure 16 - Coordinate example 4

θ(last,home) = 15°
θ(last,current) = 40°

angle = θ(last,current) - θ(last,home) = 25°
if angle > 180:
  angle = 360 - angle
else if angle < -180
  angle = -360 - angle

The above formula will return the angle relative to the home. The formula returns the angles in the same interval as the original angles, but instead of north as the base, the base is shifted to be x1.

The result of the data being logged is that it is possible to make a drawing of a route without knowing where the person has not been nor in which direction he/she went. Looking at the example below the data for the green and red route respectively will be the same as the angles and distances are the same.

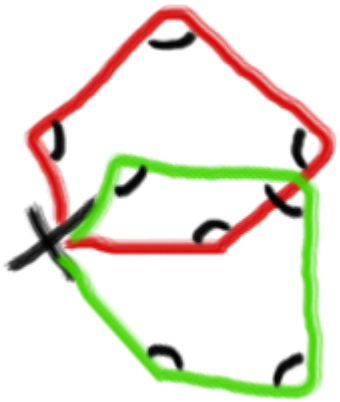


Figure 17 - Example of route drawing

Consequences
The consequences of this decision are that it is not possible to distinct different locations with the same distance to home. However for recognizing if a person is staying for a longer time at some location it can be detected as a “preferred location” these preferred locations can later be used for predicting when the person will come home.

Potential threats
Having this anonymised, coordinate-free location data does not completely rule out the risk of identifying locations for a given person. Over time data points leading to the same destination will form a road stretch. Although this road stretch is in an arbitrary direction, it is a theoretical possibility that a multitude of such road stretches can be constructed and by having access to data of real road stretches and a significant amount of processing power over a long time it can not be guaranteed that it won’t be possible to come up with a match. Lets call this scenario the “fingerprint attack”. This scenario requires the data points to be quite accurate and the processing of such a task must be considered complex because all identified road stretches has to be rotated 360° and fit on every stretch in the data set.

Another risk is to guess locations based on distances. Consider the situation where the home location has been compromised and the relative location data is available. After having identified locations where a person has been for a longer period of time these distances can be explored by drawing a circle on a map with the distance as the radius. Although the possible locations are many given that the circumference is (2*pi*distance) it will potentially be possible to make qualified guesses on where the person has been. Lets denote this attack the “locus attack”.

3.2. Data model
As this project revolves around data collection and processing the data model is paramount. Therefore the data model was the first thing to be developed and the other parts of the project are based on it. The data model contains two main parts: Home automation and smart phones. The home automation part will not be populated in this iteration of the project, but for completeness it is included. The two models represent data in two different ways: tabular and event based as explained in the analysis chapter. The consequences of these representations can be seen in the diagrams below. The premises entities in the two diagrams are the same and together the two diagrams depict the entire data model for the project.

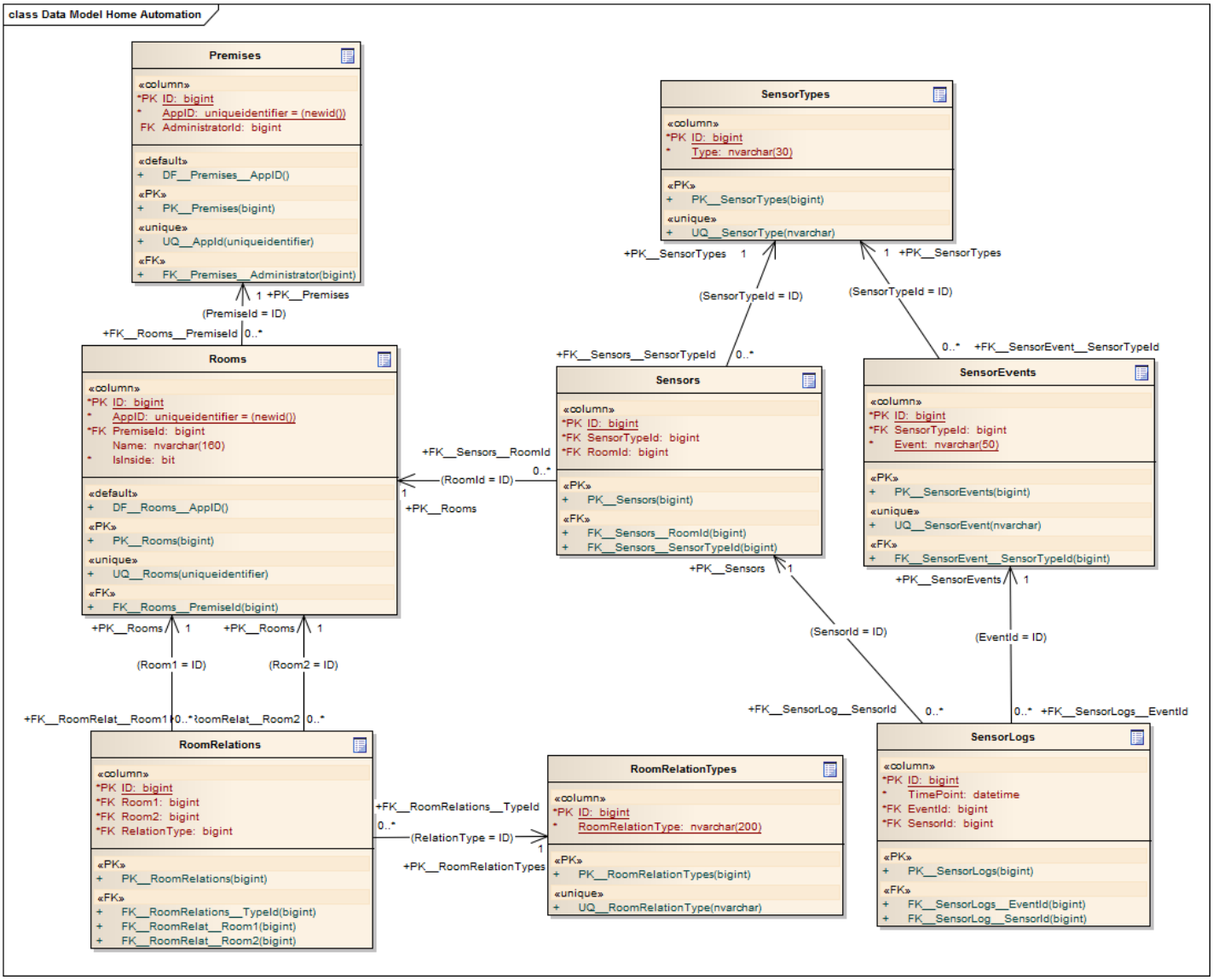


Figure 18 - class Data Model Home Automation

The home automation part of the data model represents the event based way of storing data. This means that every event from a system is stored as a record in the SensorLogs table. The remaining tables are created to add semantics to the measurement that can be used for finding patterns in rooms, across rooms and across homes. Before explaining these possibilities let's have a look at the structure.

Each home or premise is divided into different areas also called Rooms. These rooms are not necessarily physical rooms but an indication of a defined enclosed space. The property IsInside is used to indicate if the Room is situated inside or outside to be able to distinguish these two.

Room relations are used to identify if two rooms are adjacent and if they have a door between them. The room relation types are globally defined and the intend is that they should be chosen from a list of

predefined relation types to add semantics. These relations can be used for multiple things. If there is a door or an opening between the rooms it can be used for expecting presence in the adjacent rooms. If there is no door between the 2, the rooms might still impact each other for instance in heat distribution. Room is therefore a many-to-many relation with a semantic layer built in from the notion of relation types.

The sensor part of the diagram is also made with semantics in mind. To be able to compare the states from different vendors of sensors they have to conform to a predefined set of states. The Sensor table holds the sensors that are physically installed in the rooms and so there is a reference to a room and a type. The sensor type is a predefined type such as the ones mentioned in the analysis chapter: alarm, push button and motion sensor. The idea is that no matter which technology or manufacturer the sensor uses or comes from it should be comparable with another sensor that provides the same information on a higher level. This approach is the same as the one of interfaces in programming languages.

3.2.1. Smartphone data model

The smartphone side of the data model represents the tabular way of storing data. The Sensor-part could also be described as tabular since data is stored in tables, but it makes more sense in the case of the smartphone since the entire state of the device is stored in one row in the table LocationLogs.

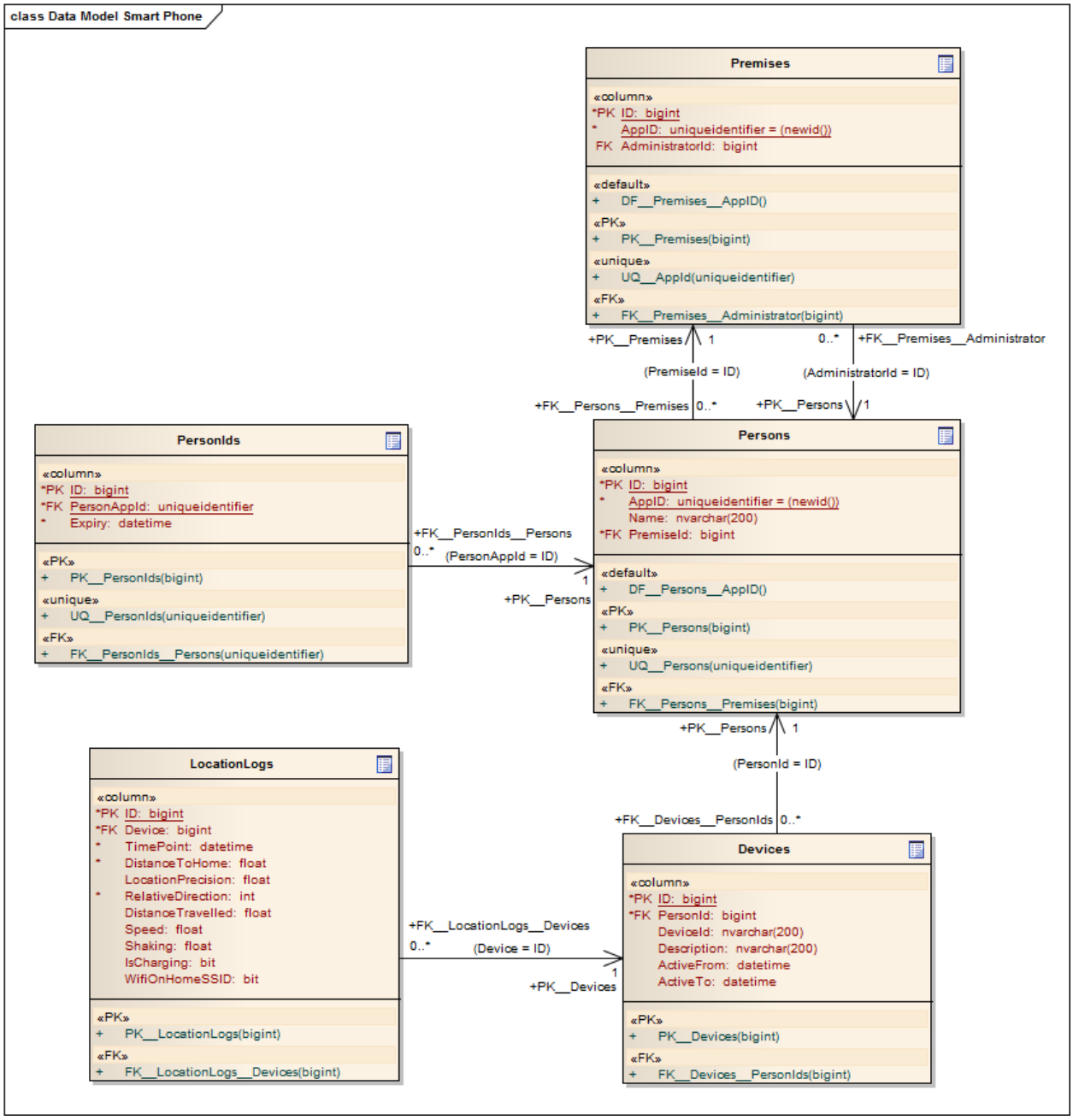


Figure 19 - class Data Model Smart Phone

As mentioned the data model is not two different ones but a part of the same large data model, however the only link between the two is the Premises table the holds the homes.

Before explaining the parts of the data model, some general decisions have to be explained to understand the model. The primary keys in the data model are of type long (64bit integer) with auto-incrementation. This was chosen because it has good performance during both INSERT, UPDATE and

JOIN[37] while still allowing a very large number of entries (2^63-1 using only positive numbers). The large number of entries is actually only expected to happen in the two tables containing Logs but to keep consistency this data type was chosen as the primary key in all tables. In order to avoid disclosing the number of rows in a table and as a way of avoiding URL hacking a second key was added. This so-called "App Id" is used in the orchestration service to identify resources. Read more about API authentication in the orchestration service section.

The structure is as mentioned in the analysis divided in persons and devices. Persons can be associated with the home and devices to the persons. The PersonIds table was not initially part of the project, but was created to ease installation for the users. On smart phones it can be cumbersome to input a full length GUID of 36 characters including the hyphens. On smart phones it is possible to define an input field as an integer which will tell the phone to bring up a numeric keyboard when active. This keyboard is as it can be seen below really easy to use and no confusions between '0' and 'O' or '1' and 'I' has to be considered. Therefore a digit-only registration id was made. This decision raises some security concerns as the range of valid entries decreases from 128 bit to 64 bit and as it is explained in the orchestration service chapter the valid range is even smaller.

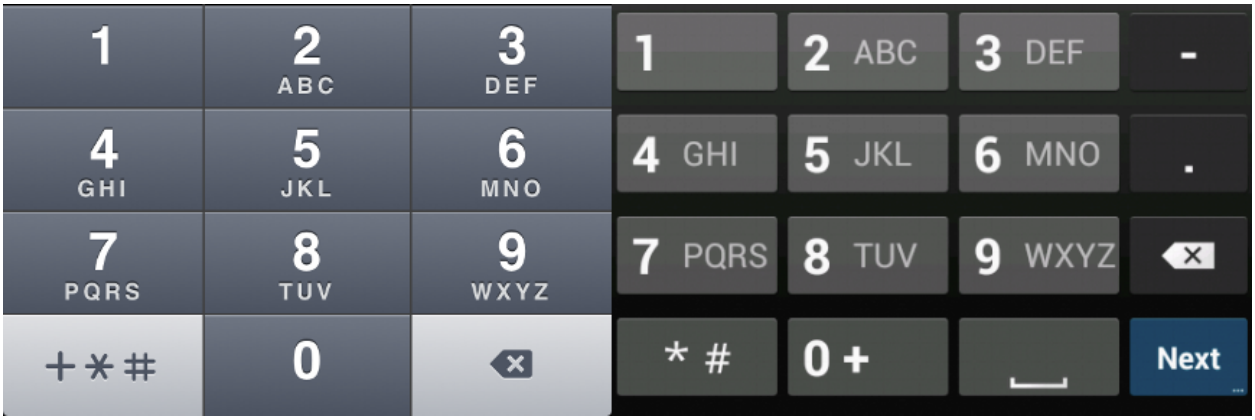


Figure 20 - Screenshot of smartphone keyboard GUI (iOS + Android)

Another interesting property in the PersonIds table is the Expiry property. This was created to make sure that id's with lower combinatory possibilities are not allowed to live forever. This decision will also be explained in the orchestration service chapter.

The devices table contain two dates called "ActiveFrom" and "ActiveTo". These are not used in the project, but they were initially intended to limit the number of active devices to one per person at the time. By having these dates a database constraint could be set up to not allow two entries with the same personId where the "ActiveTo" was null which would indicate that it was still active.

[37] http://blogs.msdn.com/b/sqlserverfaq/archive/2010/05/27/guid-vs-int-debate.aspx

"LocationLogs" holds the data that the devices are logging. This table simply contains the required fields as explained in the analysis section. The data types chosen use the lowest space consuming option that do not compromise data precision[38]. Float with double precision have been chosen for distances, speed and rotation where angles are truncated to an integer.

3.3. Orchestration service

The orchestration service is a web service that performs operations on the data and stores it in the database. The orchestration service is in charge of registering homes (premises), persons, devices and logs. Some of the methods will require authorization of an administrator and methods regarding a single entity require authorization by the home administrator.

3.3.1. Home

The operations related to the object Home are listed here:

Purpose	Parameters	Returns	Description
Add home		Created Id	Adds a new house to the system
Delete home	Id, password	Indication of success	Removes the house from the system (incl related Persons, devices, sensors, and measurements) Requires administrator privileges
Get homes		A list of persons related to the home	-
Get home	Id (of home), Password (of home)	The person identified by the Id.	-

Table 4 - Home operations

3.3.2. Persons

Name	Parameters	Returns	Description
Add person (to home)	Name, Password (of home), Id (of home)	Created id	Creates a new person and adds it to the home
Edit person	Name, Password (of home), Id (of person)	Indication of success	Edits the person

[38] http://msdn.microsoft.com/en-us/library/ms173773.aspx

Delete person	Id (of person)	Indication of success	Deletes the person and the related entities Requires administrator privileges
Get persons	Id (of home), Password (of home)	A list of persons related to the home	-
Get person	Id (of person), Password (of home)	The person identified by the Id.	-

Table 5 - Person operations

3.3.3. Devices

Name	Parameters	Returns	Description
Add device (to person)	Device id, description, Password (of home), Id (of home), Id (of person)	Created id	Creates a new device and adds it to the person
Edit device	Description, Password (of home), Id (of home), Id (of device)	Indication of success	Edits the device
Delete device	Id (of device)	Indication of success	Deletes the device and the related entities Requires administrator privileges
Get devices	Id (of person), Id (of home), Password (of home)	A list of persons related to the home	-
Get device	Id (of person), Password (of home), Id (of home)	The person identified by the Id.	-

Table 6 - Device operations

The device id is generated according to the person id and a non-changing value in the device. Therefore it should not be changed.

3.3.4. Rooms

Name	Parameters	Returns	Description
Add room (to home)	Name, Is inside, Password (of home), Id (of home)	Created id	Creates a new room and adds it to the home
Edit room	Name, Is inside, Id (of room), Password (of home), Id (of home)	Indication of success	Edits the room

Delete room	Password (of home), Id (of home), Id (of device)	Indication of success	Deletes the device and the related entities
Add room relation	Relation type, Id (of room 1), Id (of room 2) Password (of home), Id (of home)	Created id	Adds a room relation
Delete room	Password (of home), Id (of home), Id (of relation)	Indication of success	Deletes the room relation
Get rooms	Password (of home), Id (of home)	List of rooms in the home	-
Get room relations	Password (of home), Id (of home)	List of room relations in the home	-
Get room	Id (of room), Password (of home), Id (of home)	The room identified by the id	-
Get room relation	Id (of room relation), Password (of home), Id (of home)	The room relation identified by the id	-
Get room relation types	-	A list of valid room relation types	-

Table 7 - Room operations

The room relation type indicates how the rooms influence each other. In practise only rooms that are accessible from a room should have a relation to that room, however in thermodynamics two rooms can influence each other even when there are no pathway between those two. For simplicity this has been omitted.

3.3.5. Sensors

Sensor types are global and therefore no sensor types can be added. When adding a sensor a list of valid sensor types can be retrieved from the Sensors service.

Name	Parameters	Returns	Description
Add sensor (to room)	Id (of room), Id (of sensor type), Password (of home)	Created id	Creates a new sensor and adds it to a room
Delete sensor	Id (of sensor), Password (of home)	Indication of success	Deletes the sensor from the room
Get sensor	Id (of sensor), Password (of home)	The sensor identified by the id	-
Get sensors	Id (of room), Password (of home)	List of sensors in the room	-
Get sensor types	-	A list of valid sensor types	-

Get sensor type values	Id (of room), Password (of home)	List of sensors in the room	-

Table 8 - Sensor operations

3.3.6. Logs

Logs are the realization of the static data. It uses the existing sensors and registered smart phones to create logs. When logging sensor events the valid values for each sensor can be retrieved from "Get sensor type values" in sensors.

Name	Parameters	Returns	Description
Log sensor event	Id (of sensor), Password (of home), Id (of sensor), Id (of sensor event), Timestamp	Indication of success	Reports a change in a sensor value.
Log from device	DeviceId, Distance to home, Location precision, Relative direction, Distance travelled, Speed, Shaking, Is charging, Is on home Wi-Fi, Timestamp	Indication of success	Reports full state of smartphone sensors.

Table 9 - Log operations

3.4. Web app

The orchestration service is used by the mobile apps developed, but as stated in the analysis it is not beneficial to make generic implementations on several platforms. Therefore the premise administration is done from a website where homes and persons can be registered and administered. The web app will use the orchestration interface to perform operation on the data.

Application logic

As mentioned in the analysis chapter there are 2 actors that use this interface: the home administrator and the research conductor. Let's first take a look at what the home administrator can do and how this functionality is facilitated.

Add premise

To create a home no user supplied information is needed, as a home is just a shell or a container for child elements. The outcome of having created a home in the service is the newly created home's ID and therefore creating a home must provide the user with that id. Creating a home should present the id to the user and ask him/her to save it. Further it should present the possibility to add persons to the home.

Add person

A person can be added when visiting the main view of the premise. A person knowing the premise id can only access this page. A person is added by providing a name or an alias of the person to distinguish the members of the household.

Update person
Updating a person is similar to add person and the same form can be used. Access to editing person should happen from the premise main view. The edit person view should be prefilled with the existing name and have a button that saves the changes and returns the user to the premise view.

Add device
Add device does not need to be implemented in the web app as this can be done from the app.

Update device
As described in the data model the two dates active from and active to are not used. Therefore this design will not be completed. The device should never change its id, but the default description should be able to be changed.

Research conductor
For the research conductor additional functionality is available. The research conductor has to authorize him/herself to get access to this functionality. The main difference is that the research conductor can delete resources. Due to the cascading deletions this any deletion of a parent entity will delete any logs belonging to this premise, person or device.

Delete premise
Deletion of a premise should happen from the Premises List that only the research conductor could access.

Add person
A person can be added when visiting the main view of the premise. This page can only be accessed by a person knowing the premise id. A person is added by providing a name or an alias of the person to distinguish the members of the household.

3.5. Mobile app

The mobile app is the central component of this project. The mobile app is in charge of gathering information and reporting it to the service. To make sure that apps developed across different platforms act in the same way this detailed design-guide has been made. The design will explain which views, events, values etc. that the app should log and report. The requirements for the platforms are stated in the analysis section.

General events

App.OnInitialize

The app should start with the operating system and immediately start logging if the device has been set up. Use the storage with the key named K_SETUP_COMPLETED to determine if setup has completed. If setup is completed the timer event should be set to happen every 10 minutes.

User interface

The app consist of two views

1. Status (welcome view)
   Responsible for telling the state of the system
2. Settings view
   Responsible for setting up the app

Status screen

The status screen serves as the entry point of the app and has two states. Before and after the app has been set up.

Figure 21 - AtHome homescreen

- Header with the text “Welcome to AtHome”
- Welcome text depending on whether the device has been set up
- Settings button is a link to the “Settings View”
- The “Logging state button” should allow the user to turn on and off logging. When the device has not been set up it should be inactive and off. The actual control should therefore have the possibility to be active/inactive as well as on/off. If the collection of UI elements does not include such a control, simply hide the control before setup has been completed.

Figure 22 - AtHome homescreen logging on

- Header saying “Welcome to AtHome”
- Welcome text depending on whether the device has been set up
- Settings button is a link to the “Settings View”
- The “Logging state button” should allow the user to turn on and off logging. When the device has not been set up it should be inactive and off. The actual control should therefore have the possibility to be active/inactive as well as on/off. If the collection of UI elements does not include such a control, simply hide the control before setup has been completed.

Events

On load

Check if the device has been set up (key “K_SETUP_COMPLETED” in local storage). If that is the case the “Logging is on” toggle button should be activated and the “Welcome text” should be replaced with:

“You are now logging when the toggle button below is set to 'ON'.\nThe last log was sent at {K_LAST_LOG_TIME}”

Where {K_LAST_LOG_TIME} is the readable representation of how long time it has been since the AtHome app last time logged successfully.
Before first log and after setup has completed this text should be set to “no logs yet”.

Check if logging is enabled (in local storage). If that is the case set the toggle to On and start the “recurring task”.

On toggle (Logging is on)

Set the value for the key “LOGGING_ENABLED” to the value of the Toggle Button, in local storage (On -> true, Off -> false).

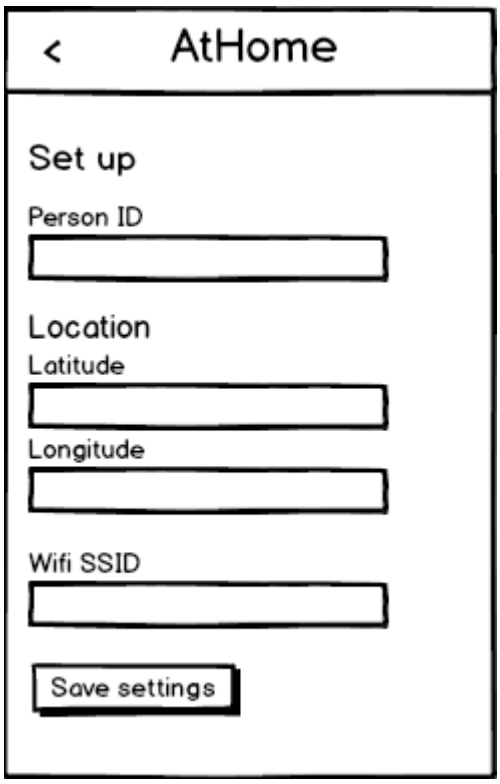


Figure 23 - AtHome Set up screen

- Header saying “Settings”
- Label for text field saying “Home Id”
  Hidden when resolved to an “App Id”
- Text field for Person App Id (editable)
  Hidden when resolved to an “App Id”
- Label for home location
- Labels and text fields for latitude and longitude
  These fields are editable and must be constrained to a floating point number
- Header for Wi-Fi SSID
- Text field for Wi-Fi SSID (not editable)
  The actual SSID is not saved but in stead the unique BSSID is saved. The SSID is shown for the user to confirm the SSID*

Events

On load:

1. Subscribe on network events (when Wi-Fi is turned on/off and when a new network is connected). If this is not possible in the OS make a polling mechanism that checks this periodically.
2. Check that Wi-Fi is on - if not ask the user to turn it on*
3. When the network event fires do the following:
   a. Save the MAC address** of the device locally
   b. Save the Wi-Fi BSSID that the phone is currently connected to locally
   c. Fill in the “SSID field” with the SSID that the phone is currently connected to
   d. Retrieve the most recent location data and fill it into the lat and long fields respectively

* There are 2 reasons Wi-Fi should be on: First and foremost to identify which Wi-Fi should be considered "Home". Secondly, when connected to any WIFI the location can be determined quite precisely, therefore the base point for comparing future locations can be set.

** In iOS and possibly also other devices the mac address is not available to the developer. It is not a problem, instead make sure to use a unique, eternally non-changing string. In iOS use the identifierForVendor which gives a unique ID that stays the same even when reinstalling the app. https://developer.apple.com/library/ios/documentation/uikit/reference/UIDevice_Class/Reference/UIDevice.html#//apple_ref/occ/instp/UIDevice/identifierForVendor

Click of button "Save settings":
Home location (lat,long) and BSSID is saved in the phone storage (see table 10)
Home location is copied to Last location (lat, long)
If there is already a device id present (i.e. device was set up previously) the re-occurring timer is started, the two keys "logging enabled" and "setup completed" are set to true and the view is forced back to the status screen

If there was no device id it means that the devices was not previously registered.
In that case the home id needs to be translated to an app id which need to be registered with the service:

To get the device id the following needs to be done:

1. Send a HTTP GET request to http://athome.solardecathlon.dk/api/persons/enroll/{person_id} where {person_id} is the 12 digit code retrieved from the registration web site.
2. The response is plain text containing the person id quoted ("00000000-0000-0000-0000-000000000000") save it locally (without quotes)
3. Calculate the device id as sha-1(MAC Address + person id) and save it locally

To register the device in the service send a HTTP POST request to:
http://athome.solardecathlon.dk/api/persons/{person_id}/devices containing the json body:
{description: "*", deviceId: "the calculated deviceId"}
Remember to set the HTTP header Content-Type to "application/json".
There is no response body of the request so just make sure it returns status 200 OK. If not the service will present an error code and reason.

When the request completed successfully add the 2 keys "logging enabled" and "setup completed" to local storage and set them to true. Finally start the recurring timer and return to the "status screen".

Logging
A recurring timer should call the logging mechanism every ten minutes. In Android this can be done using AlarmManager. Notice that this timer event has to be set up 2 places:

1. When setup has completed
2. When the application starts

Before starting to gather data the method needs to check if logging is enabled. Use the storage key: K_LOGGING_ENABLED. If this value is false no further actions are required.

Get the current location

Start listening for Location updates. This subscription should last at most LOGGING_RECORDING_WINDOW milliseconds. The default value is 20 seconds which in most cases should be enough to get a fix. More time will consume more battery power.

Location fixes will sometimes include a speed and a precision that if available should also be logged. The distance since last measurement should be calculated using the last recorded location stored in K_LAST_LNG and K_LAST_LAT. The distance can usually be calculated in the phone's location API. Get the relative direction Get the heading and compare it to

Phone state

Read the phone state to get information about if the phone is charging and if the Wi-Fi is enabled and connected to the home Wi-Fi. Use the storage value with the key K_HOME_BSSID to compare. The motion can be measured using the accelerometer or the gyroscope. If the accelerometer is available this is usually the most energy efficient. The way to measure this is by listening to motion events from the sensor in the time given by LOGGING_RECORDING_WINDOW_MOTION. When the first event comes save the x, y and z value in 3 variables that can be read at next event. At all subsequent events compare the new x, y and z to the previous and add them to get deltaXYZ. If this delta is above the value stored under key: SHAKE_THRESHOLD then add it to a sum that represents the total shake during the logging window. 10 seconds is default and should be enough to determine if the device is being moved.

Sending the data

With all the data collected the data needs to be sent. Send a HTTP POST request to the url: http://athome.solardecathlon.dk/api/log/location

Header: Content-Type: application/json

And the data:

{
"timePoint": "2014-07-19T17:35:10.000+02:00",
"distance": "6.470728",
"precision": "98.877998",
"relativeDirection": "56.114189",
"shake": "136.488647",
"distanceTravelled": "0.045899",
"isOnHomeSsid": true,
"deviceId": "dpZ5hpKW7GRp7XzOm5lX2g==",
"isCharging": true
}
	{
	Time with timezone info
	Distance with floating point '.'
	Precision with floating point '.'
	The relative direction
	The sum of shake events
	The distance from last location log
	True if on home WIFI
	The device id
	True if device is charging
	}

Local storage

In the devices implementing the design of the AtHome app some information needs to be stored on the device. This information will make sure that the app works between reboots of the system. This information is important to calculate the information described in the Privacy section of this chapter.

Key	Value type	Description
K_HOME_SSID	String	The SSID of the WIFI in the home the test subject
K_HOME_BSSID	String	The BSSID of the WIFI Access point at the home of the test subject
K_HOME_LAT	float	The latitude of the home of the test subject
K_HOME_LNG	float	The longitude of the home of the test subject
K_LAST_LAT	float	The latitude that was considered last time the app was logging.
K_LAST_LNG	float	The longitude that was considered last time the app was logging.
K_MAC	String	The MAC address of the device. If not applicable for the platform, use any unique, for the specific device recalculatable value.
K_SETUP_COMPLETED	boolean	Indicates if setup is completed
K_LOGGING_ENABLED	boolean	Indicated if logging is enabled
K_DEVICE_ID	String	The calculated device Id. A SHA-1 hash of the concatenation: MAC + person Id
K_PERSON_ID	String	A GUID that represents the person. Obtained from the orchestration web service
K_LAST_LOG_TIME	String	Readable representation of how long time it has been since the AtHome app last time logged successfully. Before first log and after setup has completed this text should be set to “no logs yet”.

Table 10 - Local storage

Constants

In the project a number of constants are used. Some of them are used for static access to keys in the storage and other er values that might be subject to change.

Key	Type	Value
K_HOME_SSID	String	HOME_SSID
K_HOME_BSSID	String	HOME_BSSID
K_PASSWORD	String	PASSWORD

K_HOME_LAT	String	HOME_LATITUDE
K_HOME_LNG	String	HOME_LONGITUDE
K_LAST_LAT	String	LAST_LATITUDE
K_LAST_LNG	String	LAST_LONGITUDE
K_MAC	String	MAC_ADDRESS
ACTION_ALARM_ELAPSED	String	athome.Action.Alarm.Elapsed
SHAKE_THRESHOLD	int	10
K_SETUP_COMPLETED	String	SETUP_COMPLETED
K_LOGGING_ENABLED	String	LOGGING_ENABLED
K_DEVICE_ID	String	APP_ID
K_PERSON_ID	String	PERSON_ID
LOGGING_INTERVAL	int	10 * 60 * 1000
LOGGING_RECORDING_WINDOW	long	20 * 1000
LOGGING_RECORDING_WINDOW_MOTION	long	10 * 1000
K_LAST_LOG_TIME	String	LAST_LOG_TIME
DECIMAL_PLACES	int	6
SERVICE_URL	String	http://athome.solardecathlon.dk/api/
API_LOCATION	String	log/location
API_ENROLL	String	persons/enroll/
API_DEVICES	String	persons/%s/devices

Table 11 - Constants

Data processing: Determination and prediction

Due to time reasons the prediction framework will consist of evaluating the features in the collected data with regards to presence prediction and determination. As explained in the beginning of this chapter the full prediction framework will have to include home automation systems to provide a base truth.

In this section the data processing part of the project will be designed. Considerations of the home automation scenario will also be presented, but not implemented.

The first part of the data processing will be to validate the features to see if there are errors in the collection. The validation should happen using common sense and knowing what values to expect:

TimePoint
Expected to be sequential such that a device logging two events e1 and e2 will be stored where:

$$id(e2) > id(e1) \Rightarrow time(e2) > time(e1)$$

DistanceToHome
The distances should be very close to 0.0 when the persons are at home. If not the setup has failed and the data is useless. Further the max distances should not exceed half the circumference of the earth (40,075/2).

LocationPrecision
The precision depends on the sensor providing location data. The precision should never exceed the distance to home if it is relatively high[39]. When this is the case the inference of presence can be difficult because the radius becomes so high that false positive results might be recorded.

RelativeDirection
The relative direction is expected from]-180;180[according to the Privacy section of this chapter.

DistanceTravelled
The distance travelled should not usually be higher than normal travelling speeds multiplied with the logging interval. An example of an expected high value is 120km/t * (10 minutes/60) = 20km.

Speed
Speeds are only registered when using GPS and given that AtHome only uses Wi-Fi and cell information to get location data this value is not recorded at the moment.

Shaking
These values are expected to be positive. It is expected that when the device is in the pocket of the test subject that biking will produce a larger value than sitting in a chair. This thesis will not be verified.

IsCharging
Expected boolean - true or false.

WifiOnHomeSSID
Expected boolean - true or false.

In this project data will only be collected over a couple of months which will not reflect the real scenario in two very significant ways. First of all the amount of data collected can be processed in a matter of

[39] Relative to the locations the persons is visiting

seconds for all persons. If a real implementation of this system has been running for years the data collected will take long time to process and old data might not be representable for the current situation. Therefore some kind of weight needs to be applied in a real scenario. For the use in this project all data is recent and therefore equally interesting. The other way is yearly periodicity which is of course not possible when data only reflect a couple of months. In a real scenario it would be interesting to compare a summer week of one year to another. By doing this, information about holidays can be used to infer that greater uncertainties apply in those periods. Further the circadian rhythm and desire to be outside is expected to depend on the season. The trade-off in this situation is more information vs. longer processing time.

Knowing that this project can not deal with large periodicities it will be interesting to look into weekly periodicities to see if there is a weekly pattern. From the data collected a feature called weekday needs to be extracted from the TimePoint. The prediction aspect of this project will only deal with weekly periods to see if a pattern applies.

Feature extraction

The data collected can hold latent information that can be inferred using the data itself or other sources. Of the features suggested in the analysis only weekday and speed will be used. The expectation of speed values will be normal travelling speeds.

Weekly plan

Using distances and indication of Wi-Fi is will be possible to make a weekly plan with probabilities that a person is AtHome. These weekly plans will be a naive input to the prediction framework.

A weekly plan will group the measurements over the entire period into buckets by weekday and full hour intervals. Percentiles of these measurements will show how likely it is for each person to be home at a given point of a given weekday. If a person is away during work hours these schedules can also provide information about predicted presence in the future.

4. Implementation

This chapter is dedicated to presenting the implementation details of the different components developed in the project. The chapter will not cover every method or assignment in the code but provide an explanation for the concepts and algorithms used. Especially those that is not obvious. For fully understanding the implementation details it is suggested to consult the actual code.

4.1. Data model

The data model was presented in the design chapter and the implementation section will cover constraints and population of the database. The database creation script reveals

The construction of the many-to-many relation currently allows invalid events to be logged for the respective sensors. The constraint below has been made to cope with that:

```
CREATE FUNCTION dbo.CheckSensorConstraintFunction()
 RETURNS INT
AS BEGIN
  RETURN (SELECT COUNT(*) FROM dbo.SensorLogs sl
            INNER JOIN dbo.Sensors s      ON sl.SensorId = s.ID
            INNER JOIN dbo.SensorEvents se ON sl.EventId = se.ID
       WHERE s.SensorTypeId != se.SensorTypeId)
END;
```

The SQL function is called on INSERT on SensorLogs and will prevent insertion if larger than 0 which means that the entry attempted to insert created an invalid entry in the database where the sensor event did not match the sensor that reported the event.

As mentioned the events and sensors are predefined and at this time it is not possible for users to add new sensors, but that might be subject to change. The advantage of having predefined sensors and states is that they can be compared across installations to make predictions more accurate. More about this in the prediction chapter.

In this iteration of the project a couple of sensors and Sensor Events have been added to show the concept.

ID	Type
1	Motion
2	Presence
3	Lock
4	Window

ID	SensorTypeId	Event
1	1	Detected
2	2	Off
3	2	On
4	3	Unlocked

5	Alarm

Table 12 - Sensor types

5	3	Locked
6	4	Opened
7	4	Closed
8	4	OpenedSafe
9	5	Disarmed
10	5	Armed

Table 13 - Sensor events

With the types and events given above a room “1” would look like this with two motion sensors and a window magnet sensor:

ID	SensorTypeId	RoomId
1	1	1
2	1	1
3	4	1

Table 14 - Sensor examples

An example of the SensorLogs table where a person went into the room and opened the window could look like this:

ID	Time	EventId	SensorId
1	2014-09-20 22:29:19.643	1	1
2	2014-09-20 22:29:21.613	1	1
3	2014-09-20 22:29:22.241	6	3
4	2014-09-20 22:29:23.123	1	2

Table 15 - SensorLogs table

In the example above two motion events happen, then the window is registered as being open and another motion event is registered, but from the other sensor in the room.

4.1.1. Authentication data model

Authentication is necessary to protect data against unauthorized access. For the use cases that the Administrator has access to and the home administrator is not allowed authentications is needed. For operations that the home administrator has to authorize it will be sufficient knowing the Guid of the premise he/she lives in.

Authentication is done using ASP.NET identity[40] as it provides a simple and easy to set up framework that fits perfectly in the orchestration service. More on that later, let's take a look at the schema:

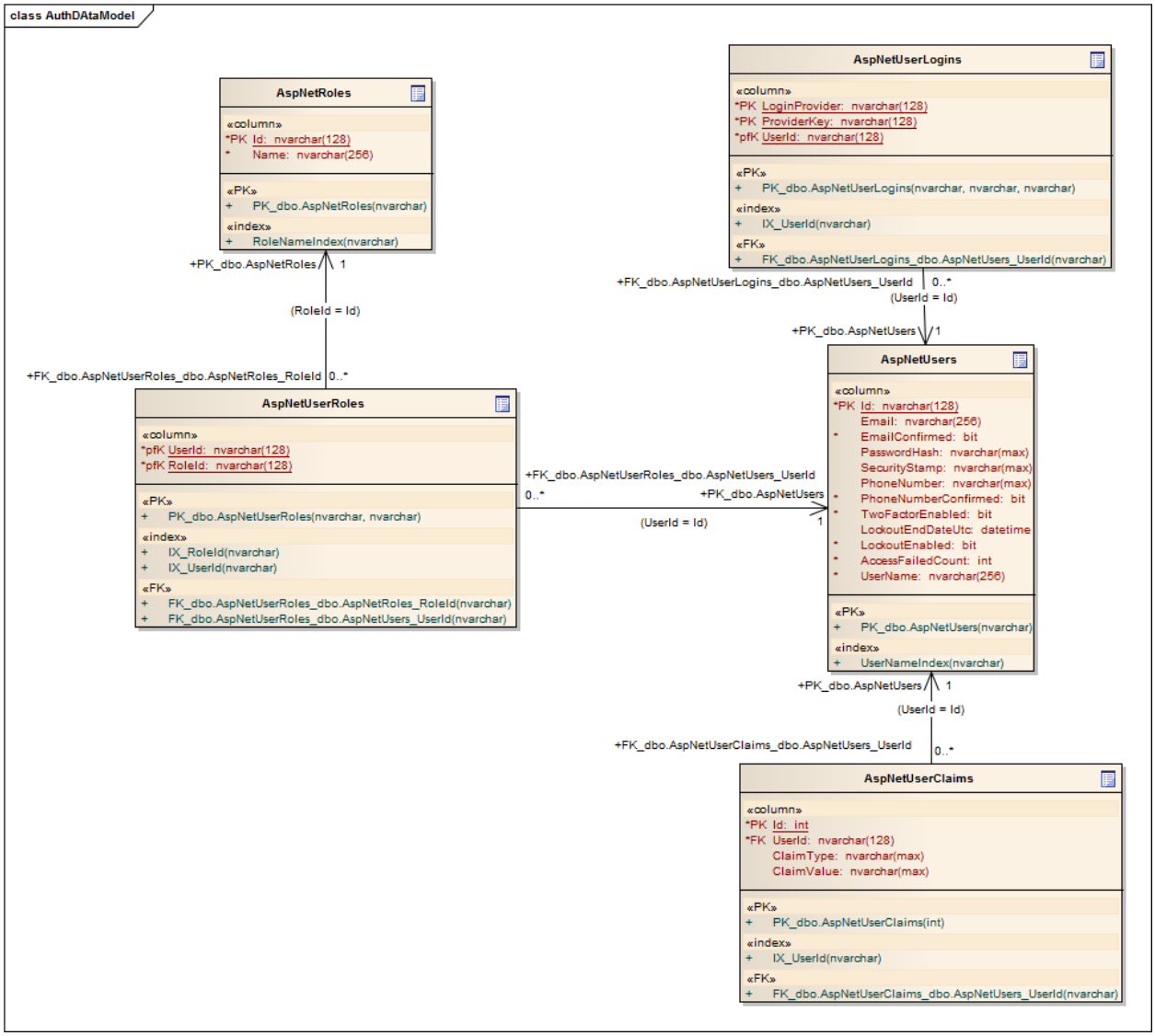


Figure 24 - class AuthDataModel

[40] www.asp.net/identity

The schema is auto generated by the ASP.NET identity framework and just needs to be populated with users and roles. On this project only one user is used: Administrator and no roles. It might seem as a large effort to implement a full identity framework for just one person, but it has been done for 2 reasons: In the future every home administrator might get his/her own login. The second reason is that when using the ASP.NET Web API it would be obvious to use the associated framework instead of reinventing the wheel.

4.1.2. Orchestration web service

The role of the orchestration service is to establish the relations between the homes of the persons, the devices and the logs being sent. The design suggested using passwords for accessing privileged data, but for some entities in the implementation a different approach was used.

The web service was implemented using ASP.NET Web API[41]. This framework is a well tested robust framework designed for creating RESTful HTTP web services. The framework supports both XML and JSON for sending and receiving data, but in this project only JSON is used. JSON has been chosen as it is object oriented and uses less space because there are no closing tag such as there is in XML[42].

The orchestration web service implements the functionality described in the design chapter and using ASP.NET Entity Framework[43] it makes it possible to facilitate a HTTP interface to the database. In the following section the implementation of this interface will be explained.

After the data model has been created in the database using the SQL script developed the Database Schema can be imported into the Entity Framework. Classes and relations are generated for use in the code. As this report is not a presentation of the Entity Framework only the non-standard elements will be explained.

The authentication has been created to prevent unauthorized access to the data. The Identity Framework is initialized in the Orchestration service when the application starts. The framework uses a data connection from the environment to gain access to the database.

[41] http://www.asp.net/web-api
[42] http://www.json.org/xml.html
[43] http://msdn.microsoft.com/en-us/data/ef.aspx

```
static Startup()
{
    PublicClientId = "self";

    UserManagerFactory = () => new UserManager<IdentityUser>(new UserStore<IdentityUser>());

    OAuthOptions = new OAuthAuthorizationServerOptions
    {
        TokenEndpointPath = new PathString("/Token"),
        Provider = new ApplicationOAuthProvider(PublicClientId, UserManagerFactory),
        AuthorizeEndpointPath = new PathString("/api/Account/ExternalLogin"),
        AccessTokenExpireTimeSpan = TimeSpan.FromDays(14),
        AllowInsecureHttp = true
    };
}
```

In the example above the authorization endpoint is defined and the expiry of the tokens are set. The choice of 14 days is actually too much and needs to be significantly smaller such as 1 hour. The final statement allows the authentication to be used on HTTP without SSL.

In the API the authorization can be enforced by annotating the method like this:

```
[Route("api/premises"), HttpGet, Authorize(Users = "Administrator")]
public IHttpActionResult GetPremise()
{
    try
    {
        using (var db = new AtHomeEntities())
        {
            var list = db.Premises.ToList();
            return Ok(db.Premises.ToList());
        }
    }
    catch (EntityException e)
    {
        return ErrorNoDb(e);
    }
}
```

The figure above shows how the methods relate to HTTP requests coming to the web server. The route annotation is the path that is appended to the applications root URL as defined in the IIS web server. The HttpGet annotation tells the framework that it is expecting a HTTP GET request.

Below is the implementation details of the API with example requests and responses.

Authentication

Some functionality in the API should not be accessible to everyone and therefore

DESCRIPTION Get token

URL http://athome.solardecathlon.dk/Token

METHOD POST

PARAMETERS Content-Type:application/x-www-form-urlencoded
grant_type:password
username:<username>
password:<password>

RETURNS Content-Type:application/json
{"access_token":"<Token>","token_type":"bearer","expires_in":1209599,"userName":"<Username>",".issued":"Sun , 17 Aug 2014 16:32:52 GMT",".expires":"Sun, 31 Aug 2014 16:32:52 GMT"}

ERRORS HTTP 400 Bad request: The user name or password is incorrect

NOTES In subsequent requests the following header has to be set:
Authorization:Bearer <Token>

Premises

GET /premises

DESCRIPTION Get a list of all premises in the system

URL http://athome.solardecathlon.dk/api/premises

METHOD GET

PARAMETERS -

RETURNS List<Premise>

ERRORS HTTP 500: Data not available

NOTES Requires login (see authenticate)

GET /premises/{id}

DESCRIPTION Get a single premise

URL http://athome.solardecathlon.dk/api/premises/{id}

METHOD GET

PARAMETERS The id of the premise as a string.

RETURNS Premise

ERRORS HTTP 500: No connection to the database
HTTP 404: Premise not found

NOTES Requires login (see authenticate)

POST /premises

DESCRIPTION Add a premise

URL http://athome.solardecathlon.dk/api/premises

METHOD POST

PARAMETERS -

RETURNS New Id
Example response:


```
{
   "$id": "1",
   "NewId": "5988db90-dce7-478f-aaea-026e8179b300",
   "Message": null
}
```


ERRORS HTTP 500: No connection to the database

NOTES Requires login (see authenticate)

DELETE /premises/{id}

DESCRIPTION Delete the premise identi

URL http://athome.solardecathlon.dk/api/premises

METHOD POST

PARAMETERS -

RETURNS New Id
Example response:


```
{
   "$id": "1",
   "NewId": "5988db90-dce7-478f-aaea-026e8179b300",
   "Message": null
}
```


ERRORS HTTP 500: No connection to the database

NOTES Requires login (see authenticate)

Persons

GET /persons

DESCRIPTION	Get a list of all persons in the system
URL	http://athome.solardecathlon.dk/api/persons
METHOD	GET
PARAMETERS	-
RETURNS	List<Person>
ERRORS	HTTP 500: Data not available
NOTES	Requires login (see authenticate)

GET /premises/{id}/persons

DESCRIPTION	Get all persons related to a specific premise
URL	http://athome.solardecathlon.dk/api/premises/{id}/persons
METHOD	GET
PARAMETERS	The id of the premise as a string.
RETURNS	List<Person>
ERRORS	HTTP 500: No connection to the database HTTP 404: Premise not found
NOTES	

GET /persons/{id}

DESCRIPTION	Get a single person
URL	http://athome.solardecathlon.dk/api/persons/{id}
METHOD	GET
PARAMETERS	The id of the person as a string.
RETURNS	Person

ERRORS HTTP 500: No connection to the database
HTTP 404: Premise not found

NOTES Requires login (see authenticate)

GET /persons/enroll/{id}

DESCRIPTION Activate the person and get the person's id.

URL http://athome.solardecathlon.dk/api/persons/enroll/{id}

METHOD GET

PARAMETERS The activation-id of the person as 64bit integer (long).

RETURNS The id of the activated person

```
"cb362efd-17a6-4993-a2e4-69500785ca5b"
```

ERRORS HTTP 500: No connection to the database
HTTP 404: Person not found
HTTP 400 Bad Request Too long time has passed since enrollment

NOTES The enrollment has to happen lo nater than 30 days from creation of the person.

GET /persons/reactivate/{id}

DESCRIPTION Assigns a new activation id for the person

URL http://athome.solardecathlon.dk/api/persons/reactivate/{id}

METHOD GET

PARAMETERS The id of the person as a string.

RETURNS Premise

ERRORS HTTP 500: No connection to the database
HTTP 404: Premise not found

NOTES

POST /premises/{id}/persons

DESCRIPTION Add a person to the premise identified by the id.

URL http://athome.solardecathlon.dk/api/premises/{id}/persons

/METHOD POST

PARAMETERS Content-Type:application/json
{ name: "test123" }

RETURNS New Id
Example response:


```
{
  "$id": "1",
  "NewId": "5988db90-dce7-478f-aaea-026e8179b300",
  "Message": null
}
```


ERRORS HTTP 500: No connection to the database
HTTP 404: Not found.

NOTES

PUT /premises/{premiseId}/persons/{personId}

DESCRIPTION Edit a person

URL http://athome.solardecathlon.dk/api/premises/{premiseId}/persons/{personId}

METHOD PUT

PARAMETERS Content-Type:application/json
{ name: "test123" }

RETURNS Indication of whether the operation was successful.

ERRORS HTTP 500: No connection to the database
HTTP 404: Not Found

NOTES

DELETE /premises/{premiseId}/persons/{personId}

DESCRIPTION Delete the person identified by the id

URL http://athome.solardecathlon.dk/api/premises/{premiseId}/persons/{personId}

METHOD DELETE

PARAMETERS -

RETURNS Indication of whether the operation was successful.

ERRORS HTTP 500: No connection to the database
HTTP 404: Not found

NOTES

Devices

GET /persons/{personId}/devices/{deviceId}

DESCRIPTION Get a list of all premises in the system

URL http://athome.solardecathlon.dk/api/premises

METHOD GET

PARAMETERS -

RETURNS List<Premise>

ERRORS HTTP 500: Data not available

NOTES Requires login (see authenticate)

GET /premises/{id}

DESCRIPTION Get a single premise

URL http://athome.solardecathlon.dk/api/premises/{id}

METHOD GET

PARAMETERS The id of the premise as a string.

RETURNS Premise

ERRORS HTTP 500: No connection to the database
HTTP 404: Premise not found

NOTES Requires login (see authenticate)

POST /premises

DESCRIPTION Add a premise

URL http://athome.solardecathlon.dk/api/premises

METHOD POST

PARAMETERS -

RETURNS New Id
Example response:


```
{
  "$id": "1",
  "NewId": "5988db90-dce7-478f-aaea-026e8179b300",
  "Message": null
}
```


ERRORS HTTP 500: No connection to the database

NOTES Requires login (see authenticate)

DELETE /premises/{id}

DESCRIPTION Delete the premise identi

URL http://athome.solardecathlon.dk/api/premises

METHOD POST

PARAMETERS -

RETURNS New Id
Example response:


```
{
  "$id": "1",
  "NewId": "5988db90-dce7-478f-aaea-026e8179b300",
  "Message": null
}
```


ERRORS HTTP 500: No connection to the database

NOTES Requires login (see authenticate)

4.1.3. Web app

Having the API and the design of the web app in place it needs to be implemented. As explained earlier it would not make sense to make this interface as a part of the app because then this has to be done on all platforms even though the action it is not specific for a platform. Instead a web interface would utilize the API and make registration easier. The immediate benefit of a website is that it is installed

centrally and thus new versions can be deployed without involving the user and it can be deployed alongside the web service. Actually the website is just a HTML front for the API.

The website follows the same URL structure as the API so that a premise is retrieved like this:

Website:	GET http://athome.solardecathlon.dk/#/premises/{ID}
API:	GET http://athome.solardecathlon.dk/api/premises/{ID}

The pound-sign '#' is intentional and is a consequence of what is most commonly known as a "Single page application" or SPA for short[44]. A single page application is a website that simulates the use of a local application by modifying the page using client side scripting. In the diagram below traditional web sites fall into the leftmost category where the SPA approach simulate the example in the middle. The word simulating is used purposely because the user interface and application logic is still stored on the server, but downloaded when the application is started (browsed). Thereby achieving the advantages of central deployment and client side processing at the same time.

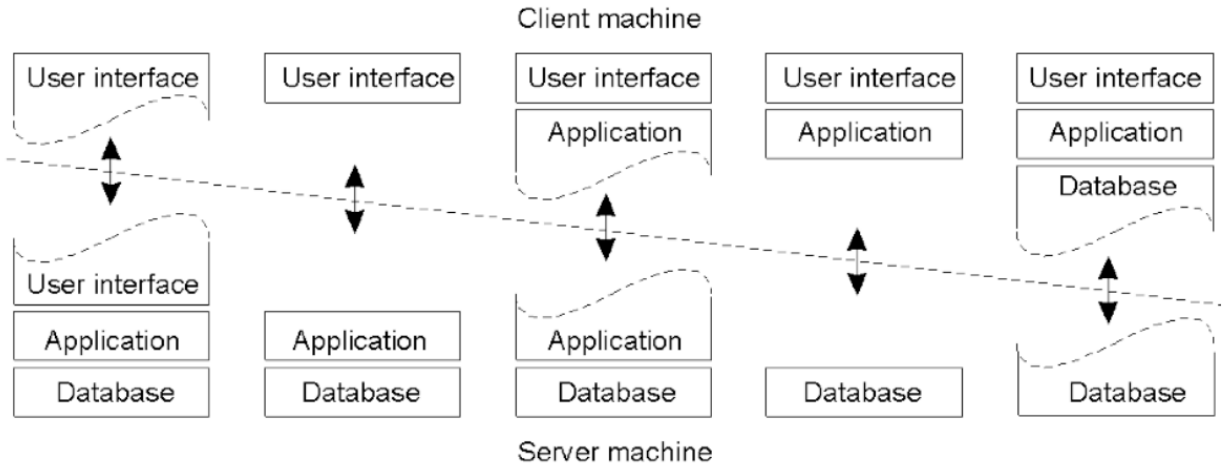


Figure 25 - Client server application responsibility distribution[45]

In the web app developed for this project a JavaScript framework called Backbone JS[46] was used. Backbone was the choice because of its built in functionality to utilize a RESTFUL web service. An example of this will be shown in the following chapter.

Backbone uses MVC to create and update the UI. The entities from the data model are created as models with references to the API for implicit data access, and then appropriate views and controllers are made to show the data and handle events. Backbone calls its controllers for views and uses HTML-templates as the views.

[44] http://msdn.microsoft.com/en-us/magazine/dn605877.aspx
[45] Tanenbaum et al. pp 41
[46] http://backbonejs.org/

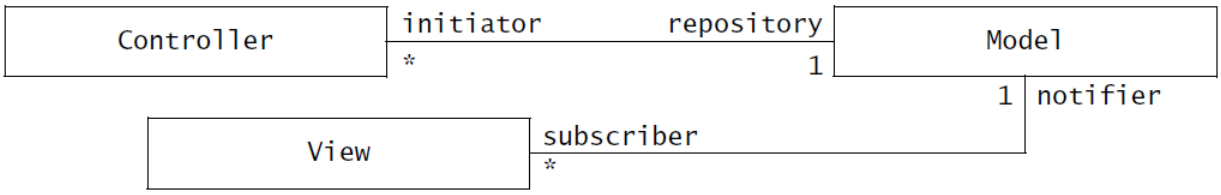


Figure 26 - UML Model of the MVC architectural style[47]

In this project two basic kinds of views are used: Single and Edit. Single shows a single element with its child elements in a list and Edit shows a form to edit the entity. The diagram below shows the design:

[47] Bruegge et al. pp 240

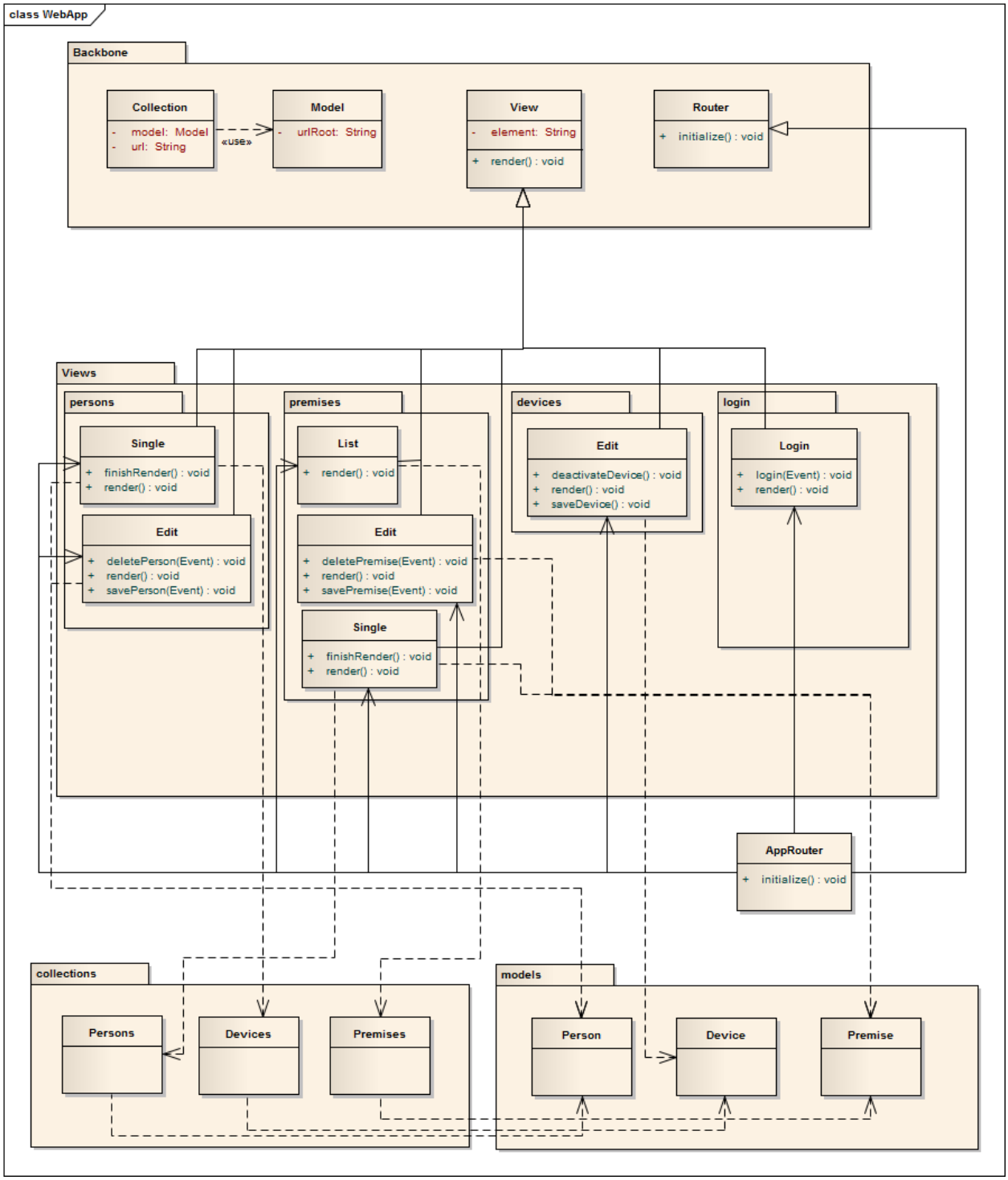


Figure 27 - class WebApp

The initialization is done from a script being included in the single HTML page the site consist of. Using a dynamic module loader called Require.js[48] the different components are loaded and the initialize

[48] http://requirejs.org/

method of the router is called. The details of initialization and the inner workings of Require.js are not relevant to this project and will not be explained.

There are in addition to the class diagram above also some HTML templates (the actual views). Those views mirror the controller's 1:1 and only the controllers knows them. These templates have been left out of the diagram to reduce complexity.

Page navigation and the pound char '#'

"The character "#" … is used in World Wide Web and in other systems to delimit a URL from a fragment/anchor identifier that might follow it[49]. In the orchestration web app the pound char takes advantage of the different browsers' implementation of RFC 1738. Since everything after a '#' sign should be considered as a fragment or anchor and not another resource the page is not reloaded but scripts are still able to receive the event of the anchor changing and can manipulate the DOM[50]. By manipulating the DOM using JavaScript the page does not need to be reloaded when switching between resources saving time and traffic costs[51].

The navigation in this project is facilitated by the router which is a component in the Backbone framework that listens to changes in the anchor and presents the content related to the path after the pound char. In the previous example (repeated below), the path is premises followed by an id which is a string containing anything but '/' and of course invalid URL characters. In this case it is a GUID.

Show a premise: GET http://athome.solardecathlon.dk/#/premises/{ID}

The allowed paths are set up in the initialization of the router. The paths are written in plain text with placeholders as the ID above defined with a colon as prefix. Below is a list of the mappings used in this project. The mapping is done from a path to an already defined function.

```
routes: {
    "": "home"
    ,"edit/:id": "premise_edit"
    ,"new": "premise_edit"
    ,"premises/:id": "premise"
    ,"premises/:premiseId/persons/new": "person_edit"
    ,"premises/:premiseId/persons/:id/edit": "person_edit"
    ,"persons/:id": "person"
    ,"persons/:personId/devices/new": "device_edit"
    ,"persons/:personId/reactivate": "reactivate_person"
    ,"login": "login"
}
```

Allowed routes in the web app

[49] RFC 1738 Berners-Lee et al. https://www.ietf.org/rfc/rfc1738.txt
[50] http://www.w3.org/DOM/
[51] Provided that the browser did not cache the resources

The functions in green types are defined in the same constructor and they can either be an initialization of a view or an action that doesn't necessarily change the visible content. The example below initializes the PremiseListVeiw and renders it in the content container. The content container is a named element that is a part of the page such as a HTML element referenced by its ID.

```
var router = new AppRouter;
router.on('route:home', function() {
    var listView = new PremiseListView();
    listView.render();
});
```

The example above calls the render method in the view. This views render method is then responsible for generating content and populate the container. Below is an example of this. This example is the one of Premise-view. All views almost perform the same logic but the PremiseView is the most important view and the most complicated and is therefore a perfect candidate for explanation:

```
var PremiseView = Backbone.View.extend({
    el: '.page',
    premise: null,
    persons: null,
    events: {
        'click .reactivate': 'reactivate'
    },
    render: function (options) {
        var that = this;
        var tempPremise = new Premise({ id: options.ID });
        var tempPersons = new Persons({premiseId: options.ID});
        tempPremise.fetch({
            success: function (premise) {
                that.premise = premise;
                that.finishRender();
            }
        });
        tempPersons.fetch({
            success: function (persons) {
                that.persons = persons.models;
                that.finishRender();
            }
        });
    },
    finishRender: function() {
        if (this.premise && this.persons) {
            var template = _.template(premiseTemplate, { premise: this.premise,
                                                         persons: this.persons });
            this.$el.html(template);
        }
    }
});
```

A lot of things happen in the example above so let's examine key by key. As the extend method is fed with an anonymous object all strings preceding a colon is a field in the object which can point to a variable or a function.

el is a reference to the content container. In the index.html page there is a element with the class "page".

premise and persons are variables to hold the current premise and the related persons.

events is an object containing all the events that the framework is responsible for hooking up. The key is the selector which in this case states that clicking any element with the class "reactivate" will call the function 'reactivate'.

render is the function "derived" from Backbone.View. This is the method called from the router. The functions instantiate an instance of Premise and Persons (collection of Person) with the parameters needed to retrieve the correct data from the API. For Premise the id of the premise is supplied and for Persons the related premise's id is supplied. The fetch methods retrieve the data asynchronously and the call-back functions "success" will save the fetched data and try to render the template in finishRender. If one of the 2 API calls fail nothing will be rendered.

The collections and models contain a path to a resource, in collections it is called url as seen below and in models it is called urlRoot. This can either be a string or as in the case below, an anonymous function that calculates the url based on the presence of an id for a premise. If no id is there all persons will be retrieved, however that requires authentication.

```
var Persons = Backbone.Collection.extend({
    initialize: function(models, options) {
        this.premiseId = options.premiseId;
    },
    url: function() {
        return (this.premiseId)
               ? 'api/premises/'+this.premiseId+'/persons'
               : 'api/persons';
    },
    model: Person
});
```

The url is used when fetch is called on the object. The base url that these local urls are appended to is automatically determined from the url preceding the pound char.

finishRender is the function called by the callback methods. This method will, if both models have loaded, render the received content into the HTML template. The HTML template is just a partial HTML

page meaning that the document is not a complete valid HTML document[52] but simply mark-up that is to be injected into an existing page. The rendering is done by another JavaScript framework called Underscore.js[53] where references in an HTML document is filled by the supplied data model. In this example the premise and the list of persons are provided. This is an example of how it works:

_.template('<h1><%= heading %></h1> <%= text %> ', { heading: "text1", text: "text2" });

Which will render to:

<h1>text1<h1> text2

As mentioned earlier the router can also be used to handle functionality besides changing content. Below is an example of reactivating a person for registration:

```
router.on('route:reactivate_person', function (personId) {
    $.ajax({
        url: "/api/persons/reactivate/"+personId,
        type: "GET",
        success: function() {
            window.history.back();
        },
        error: function(obj) {
            var message = (obj.message) ? "\n" + obj.message : "";
            alert("Person could not be reactivated!"+message);
        }
    });
});
```

Reactivating a person from the Router (router.js)

The reactivation method in the API is called and if successful the user is redirected to the page the event originated from. If it fails the user is presented with an alert and a reason from the API.

4.1.3.1. Authentication

For certain functionality knowing the id of the resource is not enough. Operations in the API that require Administrator privileges has to receive the Token received from the service. Let's take a look at how authentication is done in the web app and how the obtained token is used subsequently.

[52] http://www.w3.org/TR/html401/struct/global.html
[53] http://underscorejs.org/

```
login: function (ev) {
    var userDetails = Utils.serializeObject($(ev.currentTarget));
    $.ajax({
        url: "/Token",
        type: "POST",
        data: "grant_type=password&username=" + encodeURIComponent(userDetails.username)
                                + "&password=" + encodeURIComponent(userDetails.password),
        success: function(token) {
            var TokenModel = Backbone.Model.extend({
                auth_header: null
                });
            // Authorization: Bearer XXXX
            var header = "Bearer " + token.access_token;
            TokenModel.auth_header = header;
            if (!Backbone.Models) {
                Backbone.Models = {Token: TokenModel};
            } else {Backbone.Models.Token = TokenModel;}
            location.href = "/#"
        },
        error: function() {
            alert("Invalid login");
        }
    });
    return false;
```

In the code above an asynchronous request is sent using jQuery[54] which produces the HTTP request seen below (irrelevant headers omitted). The token is stored as a Backbone model inside a, for the user, globally accessible collection for later use.

POST
Accept:*/*
Content-Length:47
Content-Type:application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With:XMLHttpRequest

Data:
grant_type=password&username=demouser&password=demopassword

All views that make privileged API calls insert this token as an extra header on the request. In the example below the HTTP Authorization header is set to the value stored earlier. If the value is not present the header is set to null and the user will not receive the response from the API.

[54] http://jquery.com/

```
var header = (Backbone.Models.Token)
        ? {'Authorization': Backbone.Models.Token.auth_header }
        : null;
premises.fetch({
    headers: header,
    success: function (premises) {
        var template = _.template(premiseListTemplate, { premises: premises.models });
        that.$el.html(template);
    }
});
```

All methods that require Administrator privileges has to include a valid token, otherwise a 401 Unauthorized response will be returned.

4.1.4. Android app

The Android app is implemented to conform to the requirements in the Generic app design. The two screens have been designed as seen below.

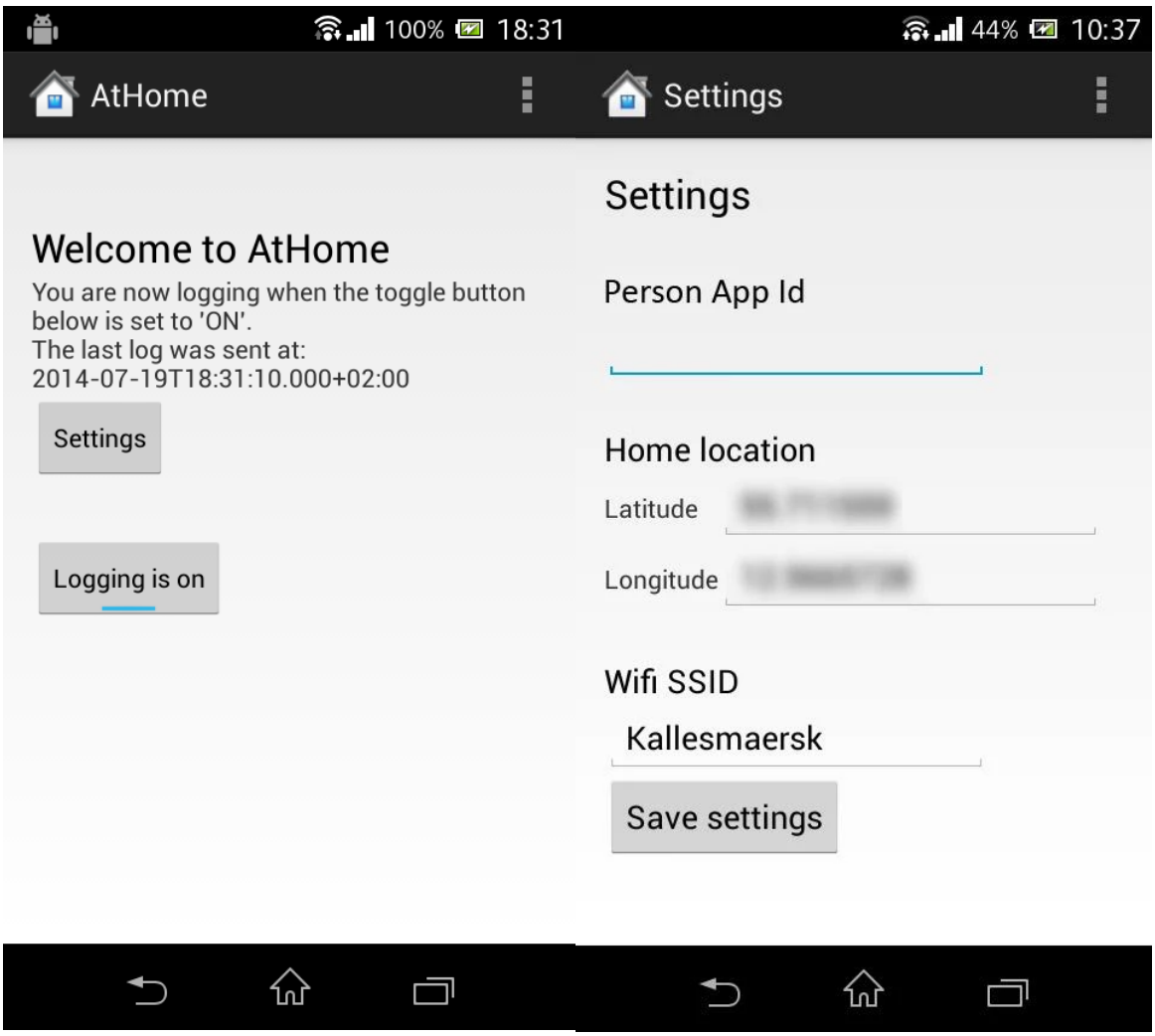


Figure 28 - Android requirement screens

Save settings button is pressed

When the save settings button is pressed the latitude and longitude is saved as the home latitude and longitude. This location is also saved as the last location so that the first comparison can be made when first log occurs.

If there is no device id in storage yet it means that this is the first time the app is being set up. When that is the case the person app id needs to be resolved to the person's Guid and from that Guid the device id can be calculated.

When the device id has been generated and saved to the local storage along with logging being enabled and setup being completed, the bootstrapper is started and the timer event is activated.

Device is booted

When the device is booted the app is automatically started. This is due to the fact that the app configuration file states that it requests this permission:

```
<uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="19" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

When the boot occurs another entry in the application configuration file directs the call to a specific class:

```
<receiver android:name="dk.solardecathlon.athome.service.Bootstrapper" >
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
</receiver>
```

Classes that are listed as receivers has to be inheriting from BroadcastReceiver which has the abstract method onReceive. Below is the onReceive method of the Bootstrapper:

```
@Override
public void onReceive(Context context, Intent intent) {
    Log.i(this.getClass().getName(), "Received intent: "+intent.getAction());

    if (!intent.getAction().equals(Intent.ACTION_BOOT_COMPLETED))
    {
        Log.e(this.getClass().getName(), "Booststrapper received and invalid intent: "+intent.getAction());
        return;
    }
    setTimer(context);
}

public void setTimer(Context context) {
    Log.i(this.getClass().getName(), "setting timer");
    AlarmManager am = (AlarmManager)context.getSystemService(Context.ALARM_SERVICE);
    Resources res = context.getResources();
    String action_alarm_elapsed = res.getString(R.string.action_alarm_elapsed);
    Intent intent = new Intent(action_alarm_elapsed, null, context, LoggingService.class);
    PendingIntent pi = PendingIntent.getBroadcast(context, 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
    am.setRepeating(AlarmManager.RTC_WAKEUP, SystemClock.elapsedRealtime() + 5000, Const.LOGGING_INTERVAL, pi);
    Log.i(this.getClass().getName(), "Timer was set");
}
```

The bootstrapper starts the timer that makes it possible to log every ten minutes. It actually uses another subscription technique using a PendingIntent. A PendingIntent is a way to delegate a call for others to make on your behalf at a later state. The alarm manager is performing this call when the timer event occurs.

Timer event occurs

Every ten minutes the timer event occurs. The timer event creates a broadcast with the event named “athome.Action.Alarm.Elapsed”. The method checks if logging is enabled using a value in local storage

and if it is enabled it creates a new instance of the sensors shown below. These sensors have been developed to include the information needed to make a logging. They get their data from sensors in the Android API.

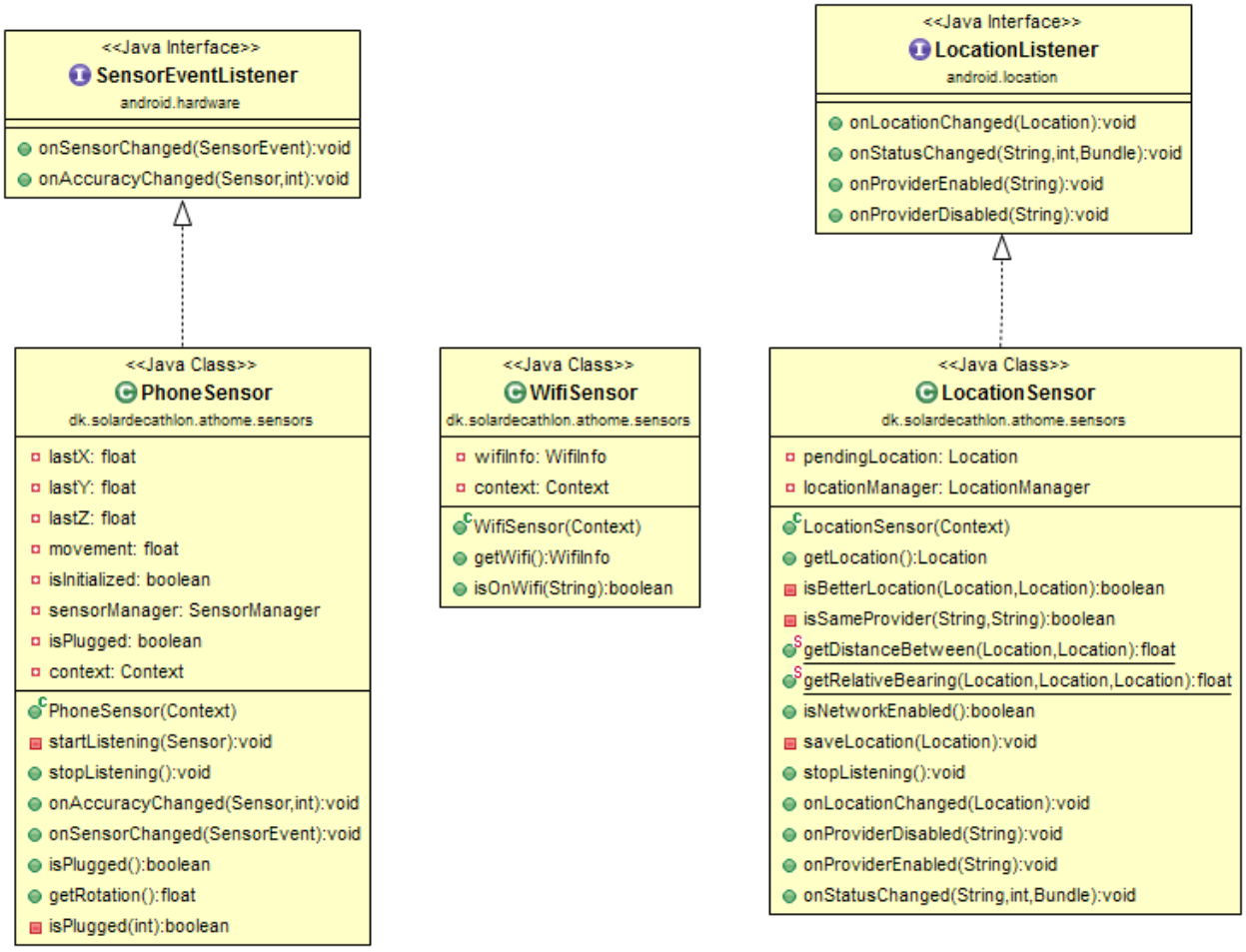


Figure 29 - Timer event occurs

The instances of the sensors or knowledge providers as one might call them is passed to a new instance of HttpLoggingModule which is an extension of AsyncTask[55] that is a component in the framework used for asynchronous processing. Since the logging is making a blocking call while making the HTTP request and since the sensors require processing to fetch their data this asynchronous approach is neccesary. Below is a sequence diagram that shows how the logging is initiated.

[55] http://developer.android.com/reference/android/os/AsyncTask.html

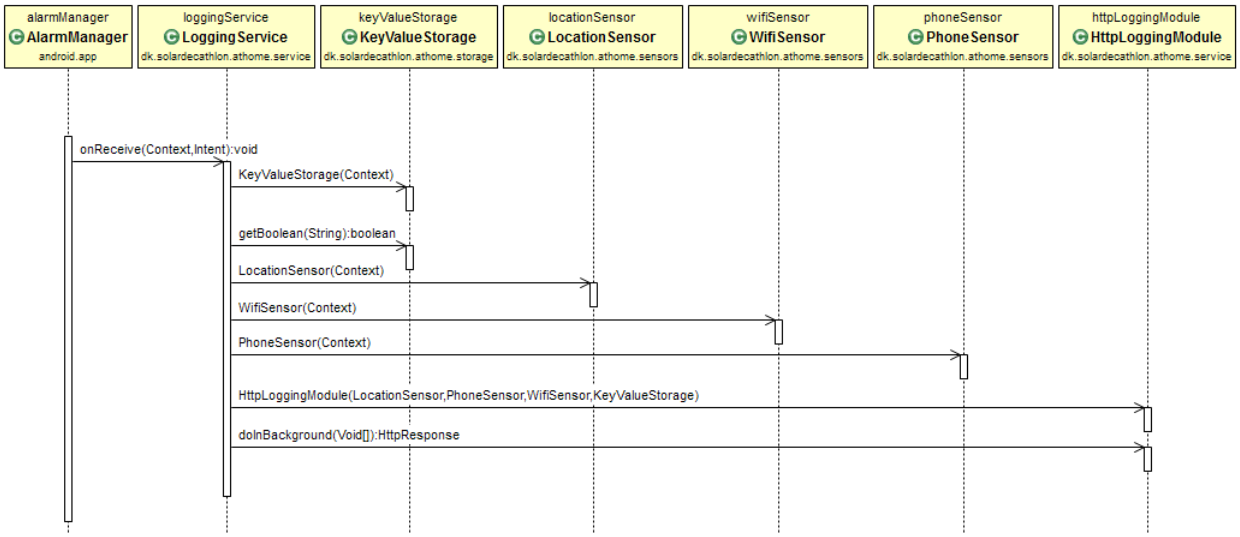


Figure 30 - Logging continues

The logging continues inside the HttpLoggingModule. The module uses its information sources to gather the data needed to make the log. Before sending the HTTP POST request to the service the location values are stored in the local storage for comparison with the next log.

4.1.5. iOS development

At the beginning of the project it was the intention to develop an iOS app for use on iPhones to include more test subjects and thus have more data to look at. Another reason was to be able to compare sensor measurements across different operating systems particularly with regards to motion. Due to complications, the actual app was never released and therefore no data for iOS devices was collected. During the development, a lot of considerations regarding how to implement the functionality was made and this section will seek to describe the design process of the iOS app. The chapter will also describe possibilities and limitations of the OS that relates to this particular project.

User interface

Following the mockup shown in the design chapter, a user interface has been created for iOS. The user interface uses the original components from the toolbox in xcode which is the de facto IDE for developing apps for iOS.

There only deviation from the mockup is the “Settings View” that has been made as an overlay rather than a separate screen. When the OS allows for an easy way to shift between views while still keeping a good overview this seems like an obvious choice. The wire between the two screens shows the transition and the icon shows that it uses a so called modal which is an overlay.

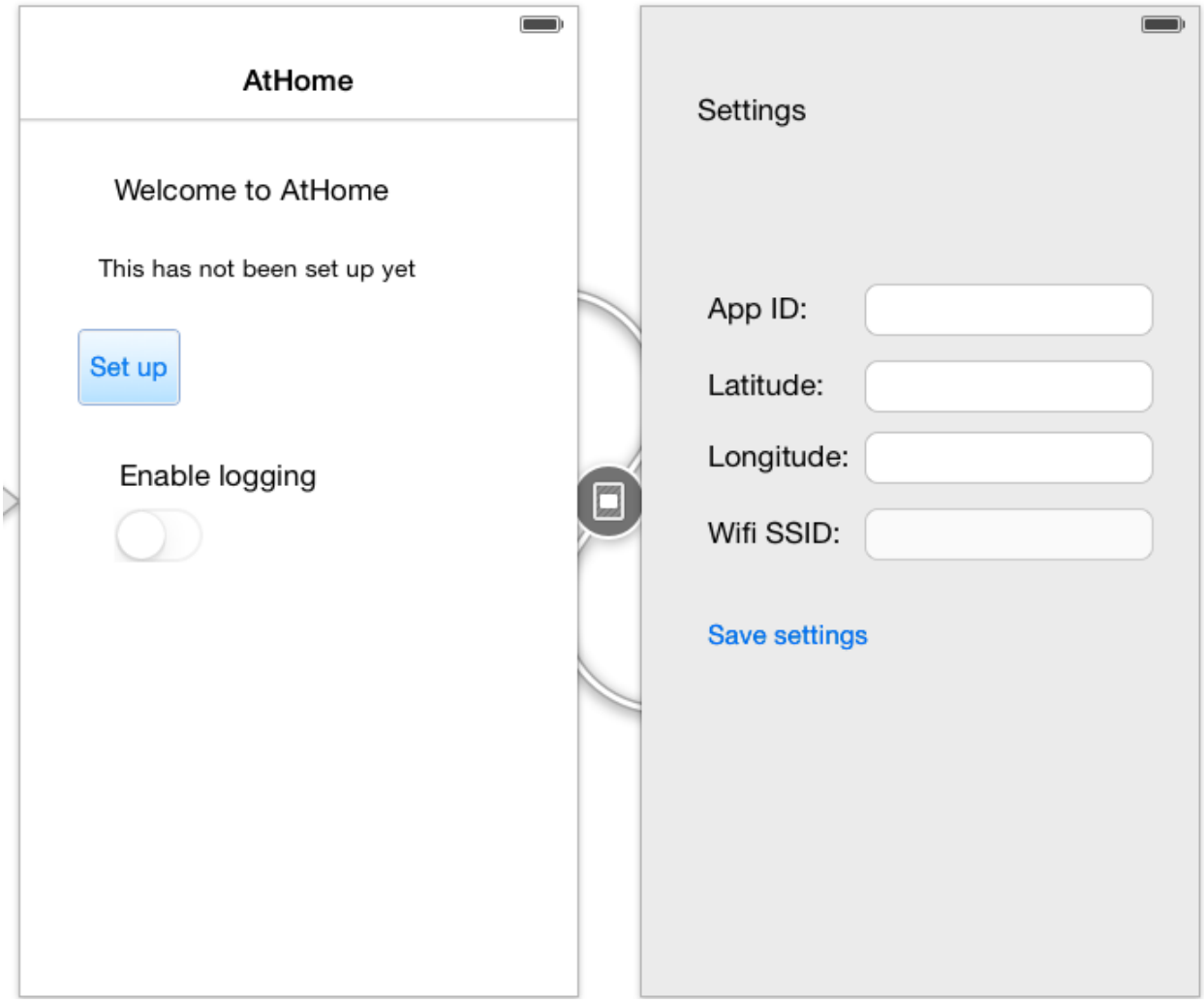


Figure 31- Mock up of user interface for iOS

The events and logic of the application was only implemented partly and will therefore be omitted.

5. Evaluation

Having collected data since mid July and forward it can be evaluated if the application is producing meaningful data and if that data can tell anything about daily routines and travel times. The first observations is that only a few people managed to register and install the app. Below is a chart showing the occurrences of logs through the period 20 July - 4 Sept 2014

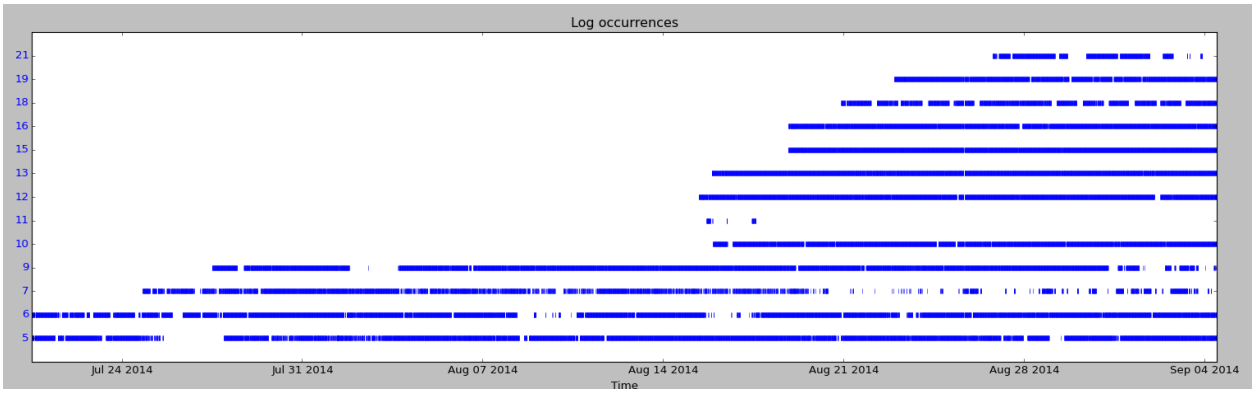


Figure 32 - Data from 20 July - 4 Sept 2014

Data will be collected until the end of the project and this subset will evaluation will be based on the data gathered in this period. The number of logs is not the same for all persons due to several reasons. First of all the test persons did not sign up and start logging at the same time. Secondly some of the persons show periods of inactivity that could be due to the threats listed in the analysis section. For test person 18 it looks as if data connectivity is periodically switched off and for test person 7 it looks as if he/she lives in an area with bad coverage. These theories can never be verified as the data is collected anonymously, but since these are real persons the patterns they show are, although not statistically representative, still representing the population and address some of the problems that AtHome might face in the future. For something to be statistically evident the population needs to be a lot bigger and randomly selected. This population is neither of the two.

5.1. Reliability

Looking at the gaps in the graph above it would be relevant to study the reliability of the logging. One definition of reliability in software engineering is the one stated in Bruegge et al.: "Reliability is the ability of a system or component to perform its required functions
under stated conditions for a specified period of time. Reliability requirements include,
for example, an acceptable mean time to failure and the ability to detect specified faults
or to withstand specified security attacks."[56] Although it describes a software component it can be used to measure the reliability in the app. The so-called stated conditions being that it should log every ten seconds and should not retry if it fails. The reliability can in that case be estimated by taking the ratio

[56] Bruegge et al. pp 126

between the actual number of logs and the expected number of logs in a fixed time period. Since not all devices were added at the same time, and since some devices stopped logging before the end of the data set, periods are calculated individually by the first and latest log received for the specific device. To calculate the ratio the following formula has been used:

count(logs) / hours(Time of first log - Time of latest log) * 6

Although this is not measured in “mean time to failure” these metrics are roughly translatable. The problem is that failure is only evaluated every ten minutes due to the nature of these periodic logs.

ID	Phone Description (model)	Actual / Expected Logs
15	Samsung GT-I9506	0.994275
12	Samsung GT-I9505	0.980378
19	Samsung GT-I9100	0.967963
10	Samsung GT-I8190N	0.965182
16	Samsung GT-I9295	0.95474
13	Samsung GT-I9100	0.933101
6	HTC One	0.881605
5	Sony C2105	0.860934
9	LGE Nexus 5	0.857114
18	HTC One mini	0.809711
7	Samsung Galaxy Nexus	0.586046
11	HTC Wildfire S A510e	0.199882
21	Samsung GT-S6500	0.194264

Table 16 - Phone model and actual/expected logs

The results are for most devices acceptable with rates as high as .99. For the devices with a rate of less than .80 it can be hard to make proper use of the data. For test subject 7 it seems, looking at the log occurrences, that a change happened around the 20. August and the reliability is significantly lower after that date. The test subject could have moved or installed new software that sporadically turns off the data connection. For test subjects 11 and 21 it is doubtful that the data can provide any valuable information.
The reasons for the failed logs can be many.

5.1.1. Outage periods

The observed reliability of the measurements in the database does not provide the full picture of reliability. The environment also has to be considered. Below is a screenshot from the Windows Server 2012 event log filtering to see reboots of the machine. The list is sorted with Date and Time in descending order and shows that no restarts were performed after June 14 which means that no downtime of the server occurred.

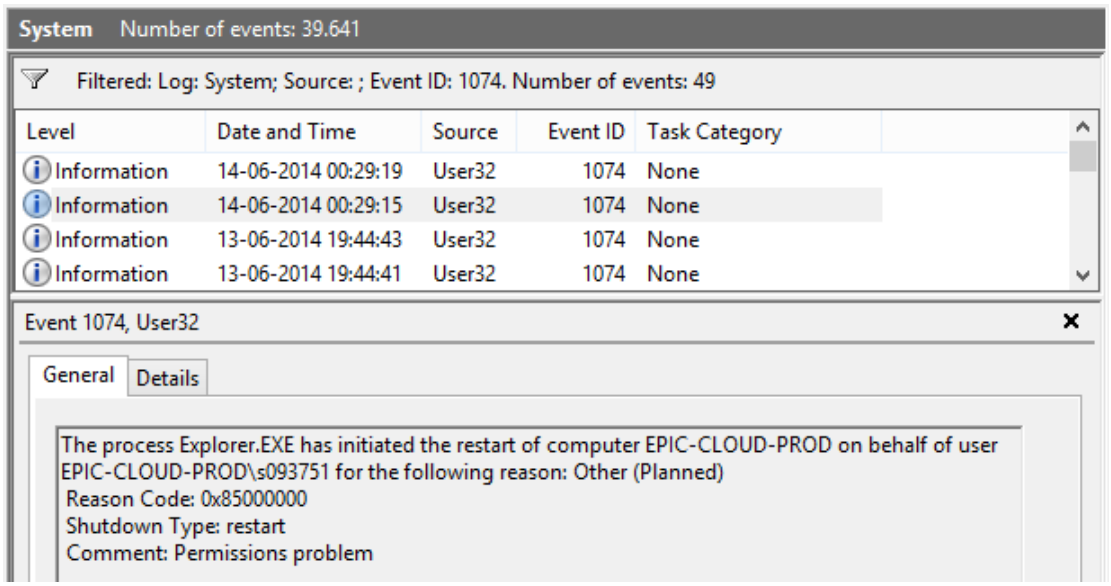


Table 17 - Screenshot from Windows to see reboots of machine

The AtHome service that is responsible for receiving requests and saving them in the database is running on the IIS webserver in Windows. This process also logs to the event viewer and can be filtered using the following query:

```
<QueryList>
 <Query Id="0" Path="System">
  <Select Path="System">
    *[EventData[Data[@Name='AppPoolId'] and (Data='AtHome')]]
  </Select>
 </Query>
</QueryList>
```

Since the service was first deployed on June 10, 79 logs were recorded and none of them was warnings or errors. The logs are either to inform about automatic recycling of the application or that the process running the application was shut down due to inactivity. In the last case a new process will be started by the IIS when needed.

The next step is to look at the application itself. As explained in the implementation section a fault handler was hooked up to catch faults in the Web Service. These faults were recorded and saved in the AuditTrail. The distribution of outcomes from the Service is as follows:

Outcome	Count
200	73967
400 Bad Request	41
401 Unauthorized	1
404 Not Found	2
OperationCanceledException	12

Figure 33 - Faults in the Web Service

The table shows that most of the requests were successful and a few 4xx meaning a user provoked fault such as attempting unauthorized access or non-existing resources. The final category "OperationCanceledException" is unaccounted for. This exception seems to come from the WebAPI when the browser cancels the request. Due to the small rate this fault has not been investigated further.

5.1.2. Availability of the service

Since the environment has not reported any fault it will be interesting to look at periods with no logs in the service. Using the AuditTrail which is a table that records all data in the database it is possible to see gaps in the logs:

Start	End	Outage
25-08-2014 16:18	25-08-2014 15:48	00:30:03
07-08-2014 23:09	07-08-2014 22:45	00:24:40
02-08-2014 03:40	02-08-2014 03:21	00:18:13
02-08-2014 00:12	01-08-2014 23:57	00:15:03
08-08-2014 10:26	08-08-2014 10:11	00:14:47
09-08-2014 04:40	09-08-2014 04:26	00:13:39
02-08-2014 00:55	02-08-2014 00:42	00:13:17
04-08-2014 09:34	04-08-2014 09:20	00:13:16
02-08-2014 02:10	02-08-2014 01:57	00:12:42
08-08-2014 11:12	08-08-2014 11:00	00:12:30

Table 18 - Availability of the service

These gaps reveal that there were periods where no loggings were performed. The reason for this is unknown since it is unlikely at a late stage with 13 persons logging within 10 minutes that no logs were recorded in a period of 30 minutes.

Android app

VERSION	UPLOADED ON	STATUS	ACTIONS
7 (7)	Aug 20, 2014	in Prod	

OTHER APKS Hide

VERSION	UPLOADED ON	STATUS	
6 (6)	Aug 15, 2014	Unpublished	Show details
5 (5)	Aug 15, 2014	Unpublished	Show details
4 (4)	Aug 3, 2014	Unpublished	Show details
3 (3)	Jul 21, 2014	Unpublished	Show details
2 (2)	Jul 20, 2014	Unpublished	Show details
1 (1.0)	Jul 19, 2014	Unpublished	Show details

Table 19 - Snapshot from the Google Play developer console[57]

The app was released in 7 releases each new version solving problems that occurred at one or mores users. Google provides a developer console that can be used to receive bug reports and detailed information about exceptions. Below is a list of the bugs experienced in the system and when they were solved.

57 https://play.google.com/apps/publish/?dev_acc=11902827434078924379#ApkPlace:p=dk.solardecathlon.athome

NAME	NEW	REPORTS THIS WEEK	REPORTS TOTAL	LAST REPORTED
dk.solardecathlon.athome.excepti... in dk.solardecathlon.athome.sensors...		0	6	Aug 20 11:43 AM
java.lang.NullPointerException in dk.solardecathlon.athome.sensors...		0	1	Aug 16 9:21 PM
java.lang.ArrayIndexOutOfBounds... in dk.solardecathlon.athome.service....		0	2	Aug 15 10:57 PM
java.lang.NumberFormatException in java.lang.StringToReal.invalidReal		0	1	Aug 15 2:25 PM
android.content.ReceiverCallNotA... in android.app.ReceiverRestrictedCo...		0	3	Aug 15 10:27 AM
java.lang.NullPointerException in dk.solardecathlon.athome.Settings...		0	8	Aug 14 9:50 PM
java.lang.RuntimeException in dk.solardecathlon.athome.storage....		0	1	Aug 4 4:04 PM

Figure 34 - Bugs experienced in the system

In the screenshot above from the Google Play Developer Console a list of the reported errors appear. These errors are reported by the users from the app when it crashes and requires the consent of the user before they are sent. So far there is no reported but in version 7 that is the current version in production. In the other versions different problems were reported and subsequently solved.

Version 6:
dk.solardecathlon.athome.exceptions.NoCurrentLocationException
in dk.solardecathlon.athome.sensors.LocationSensor.getDistanceBetween

The location was not found within reasonable time. Null check was made and method was not called it no location was there.

java.lang.NullPointerException
in dk.solardecathlon.athome.sensors.WifiSensor.getWifi

On one device the WifiSensor did not receive the Broadcast and was null even though it should never be null.

Version 5:
java.lang.ArrayIndexOutOfBoundsException
in dk.solardecathlon.athome.service.HttpGetPersonIdModule.doInBackground

Initialization of the module was done with an empty array. Method signature was changed so that the information needed for processing was fed to the constructor in stead.

java.lang.NumberFormatException
in java.lang.StringToReal.invalidReal
On one device the Home Latitude and Longitude was not set and therefore not a floating point number.

Version 4:
android.content.ReceiverCallNotAllowedException
in android.app.ReceiverRestrictedContext.registerReceiver
In older versions of Android BroadcastReceivers were not allowed to inherit from LocationListener. Solved by removing dependency on BroadcastReceiver which wasn't needed.

java.lang.NullPointerException
in dk.solardecathlon.athome.SettingsActivity.updateWifiInfo
No location fix was registered and when trying to read latitude and longitude values this exception happened. Solved by making null check and retrying until success.

java.lang.RuntimeException
in dk.solardecathlon.athome.storage.KeyValueStorage.getString
It was possible to skip setting up some of the values resulting in the first log reading a non-existing value.

Version 3:
java.lang.NullPointerException
in dk.solardecathlon.athome.SettingsActivity.updateWifiInfo
Same as v4

The developer console also shows statistics about Android versions and countries in which the app was downloaded. The Android version can help target the devices with most people in a situation where a newer version of the API offers benefits for the way AtHome uses it.

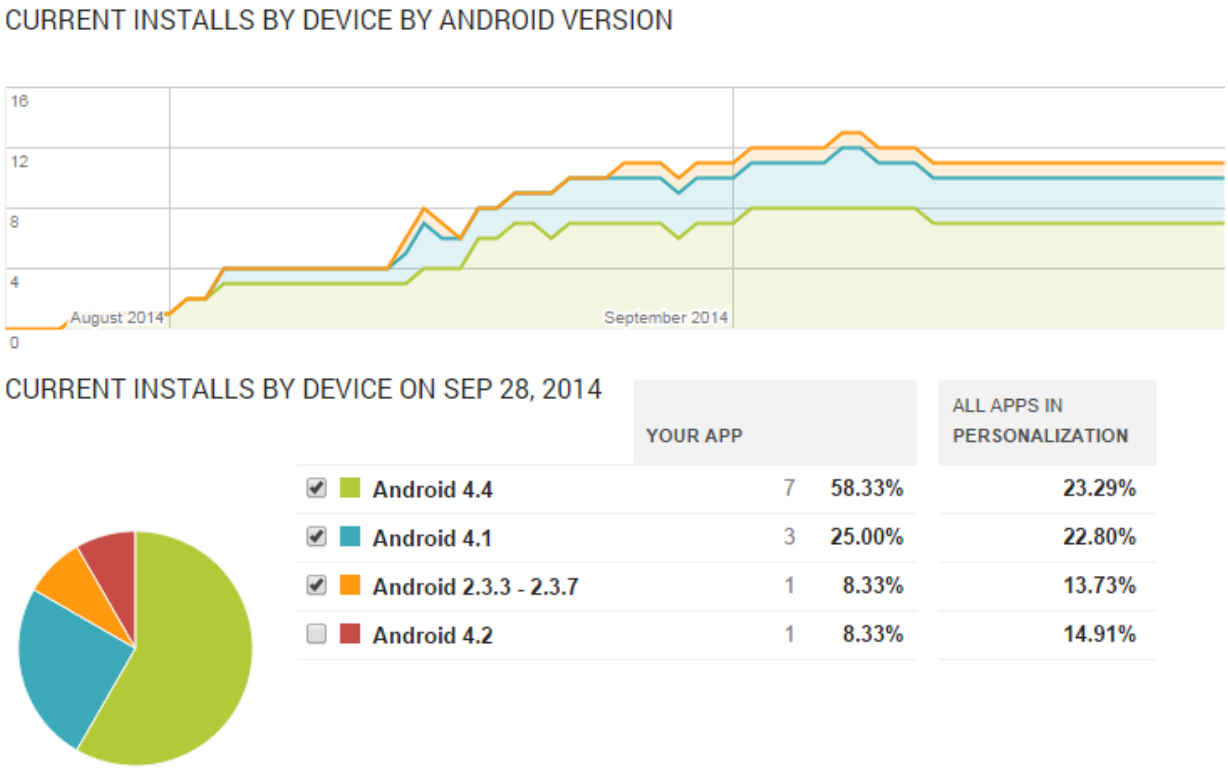


Figure 35 - Current installs by device by Android version

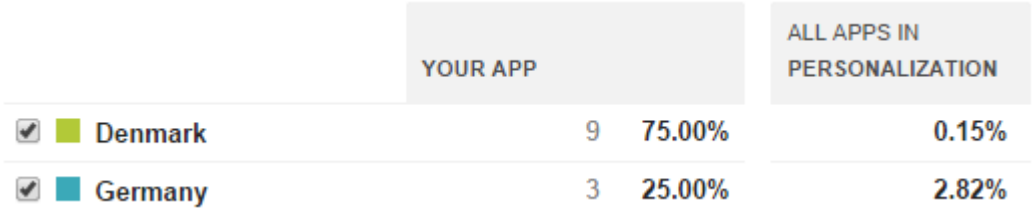


Figure 36 - Installs sorted by country

The overview also shows the distribution on versions of the app. It was a positive surprise that only one test subject was running an older version of the app. The apps are currently not logging their own version which would be a great benefit if changes to logging intervals or logging windows change.

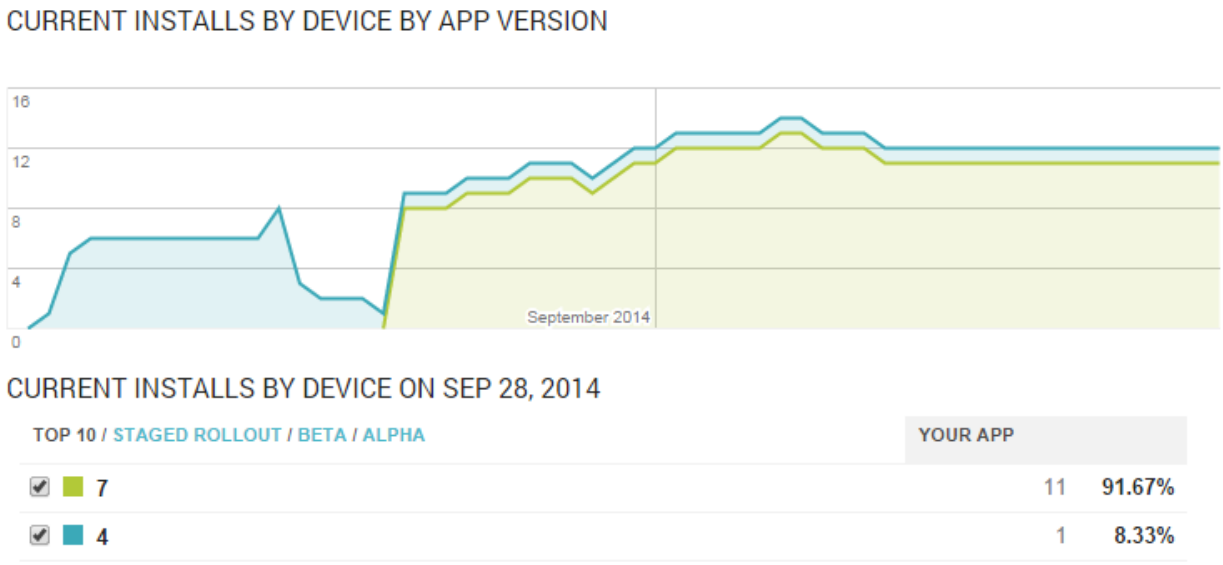


Figure 37 - Current installs by device by app version

5.2. Data collected

Unfortunately time did not allow for much data processing, but the quality of data needs to be validated in order to know if it can be used for meaningful prediction and determination of presence. Before looking at patterns it would be sensible to validate the different features of the data individually.

Devices

In total 13 devices registered and sent at least one log. This is far less that initially anticipated, and makes it hard to make any statistically evident conclusions, however it is enough to validate the components and the way of gathering the data.

TimePoint

The time points range from the day the service was added to the present day and all time points are logged to that if time(2) > time(1) then ID(2) > ID(1) which means that it works as expected. One problem is that no time zone data is saved which would be a benefit. In SQL Server there is a data type called datetimeoffset[58] that can store this value. At the moment the app in sending the date and time with timezone info but the service converts it to local Danish time and stores it without time zone information. At the moment it is not a problem because the data is still logged chronologically, but important information is lost.

[58] http://msdn.microsoft.com/en-us/library/bb630289.aspx

Distance

The distances in the system range from less than one meter to 9000km which seems plausible since it has been a period of holidays. The low minimum distances also indicate that the initial positioning of home is accurate.

ID	Min	Max
5	0.349839	279117
6	0.066775	163350
7	1.881584	8805258
9	0.173608	1386081
10	0.489904	120210
11	3.389812	349
12	0.035057	224551
13	0.560397	977519
15	0.099794	8929348
16	0.021537	1611197
18	0.005229	6233702
19	0.027751	678994
21	0.122934	506758

Figure 38 - Minimum and maximum log distances

The distribution of distances looking at all test subjects also looks reasonable. The conclusion at this point is that people are AtHome a lot:

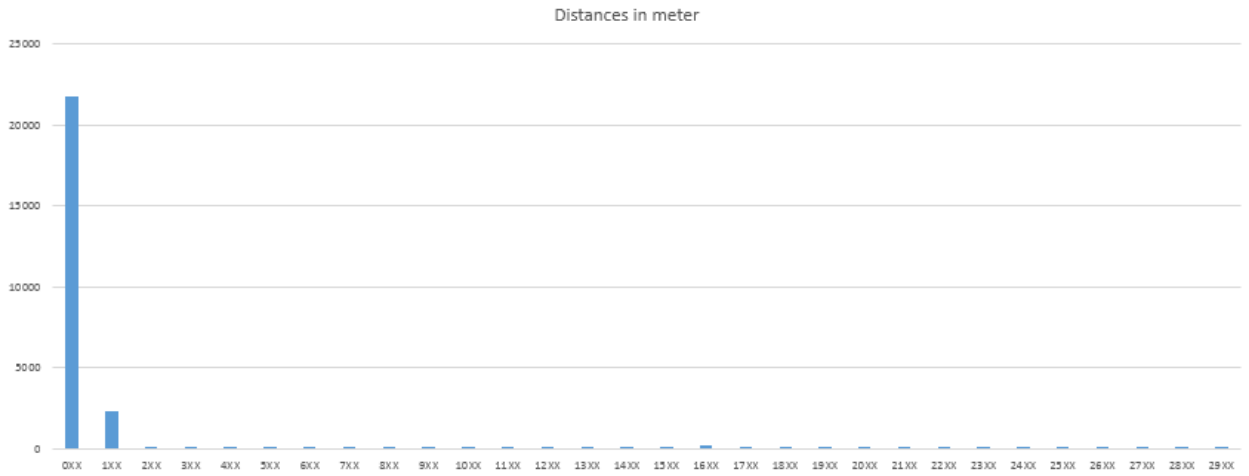


Figure 39 - Distribution of distances

The graph shows that most logs were made less than 100m from home. The observations above 3km has been collected into one.

LocationPrecision

The location precisions reflect the location source. If the accuracy is around 50 it is usually from Wi-Fi. The yellow bars are logs when the test subjects are on home Wi-Fi and the orange ones are all logs. The values have been normalized to percentage of total to be comparable.

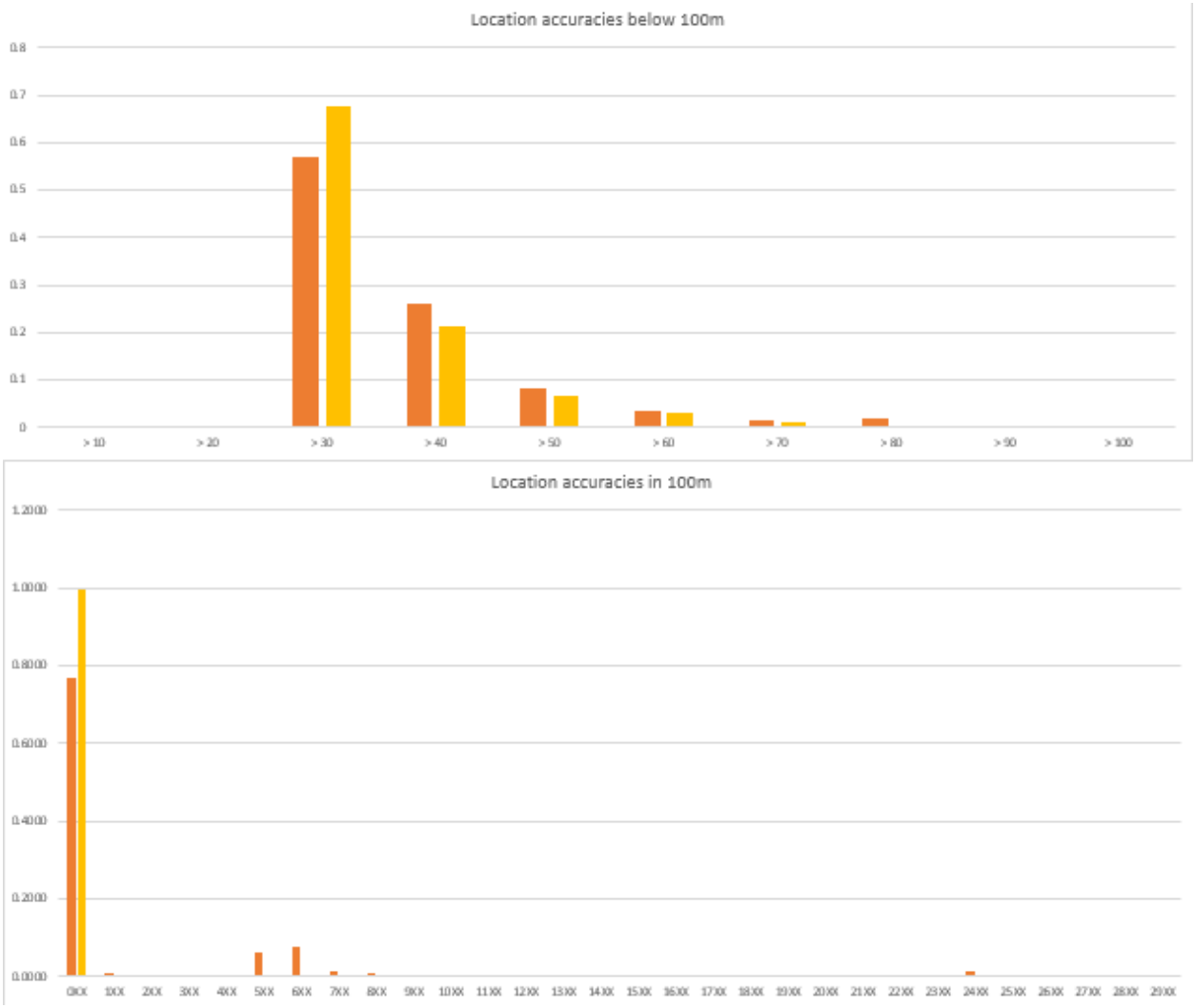


Figure 40 - Location precision

RelativeDirection

The relative directions were not logged correctly. In the database the minimum and maximum are -360 and 358 which is more than the]-180:180] interval that was expected. There was a mistake in the formula in the code. The true formula is the one from the design chapter of this report. This means that relative directions in the dataset are invalid and should be discarded.

DistanceTravelled

Distance travelled is supposed to show the distance the test subject travels between two logs. As it can be seen in this table the max-values are a bit suspicious when considering 10 minute intervals. If the internet connection was gone for a while for instance when in an airplane or a car in area with bad coverage for a long distance these distances might make sense.

ID	Min	Max

5	0.002545	257661
6	0.008149	116394
7	0.004394	8803704
9	0.008147	1378355
10	0.008859	33380
11	0.249254	365.453
12	0.001337	235352
13	0.007834	985293
15	0.002504	8930403
16	0.008031	1611304
18	0.005229	6232830
19	0.001236	72547

Table 20 - Travel distance between logs

Speed

Speed is only collected with the GPS sensor. As GPS is not used in this project due to its high power consumption no speeds were reported. In the dataset there exist a few measurements with a speed value. Those are from a test distribution where the GPS was enabled. Speed can be partially inferred from the "DistanceTravelled" feature and comparing 2 time points, however that exposes some interesting data about DistanceTravelled.

ID	Min	Max
5	0.000003	1546
6	0.000015	530
7	0.000002	45785
9	0.000006	8585
10	0.000053	200
11	0.000019	219
12	0.000007	1669

13	0.000001	6074
15	0.000014	53404
16	0.000043	11589
18	0.000007	47813
19	0.000008	4042

Table 21 - Speed as a factor in the distance travelled

The max speeds observed are doubtful to be correct. A possible explanation for this is caching of locations so that if no location is received then a previous location is sent. If this is the case then a possible solution will be to include the timestamp of the location fix when sending the data to the service. Then these old locations fixed can be disregarded in the data processing.

Shaking

Looking at the individual data from the accelerometer it can be seen that they vary a lot. Sampling on differences on the accelerometer for 10 seconds can create a high number when running or biking.

ID	Min	Max
5	0	43850.55
6	0	4278.15
7	0	5218.57
9	0	35538.91
10	0	2086.06
11	0	202.33
12	0	5166.89
13	0	852.85
15	0	1601.57
16	0	7217.91
18	0	1616.12
19	0	836.25

Table 22 - Minimum and maximum values of the accelerometer

IsCharging

The charging data is interesting to follow as it can be seen on the coloured chart on the right 4-5 people have a nightly charging habit. The differences between the habits when on home Wi-Fi (right) and all measurements combined (left) is not apparent. At this point charging data will not be considered for providing information about presence.

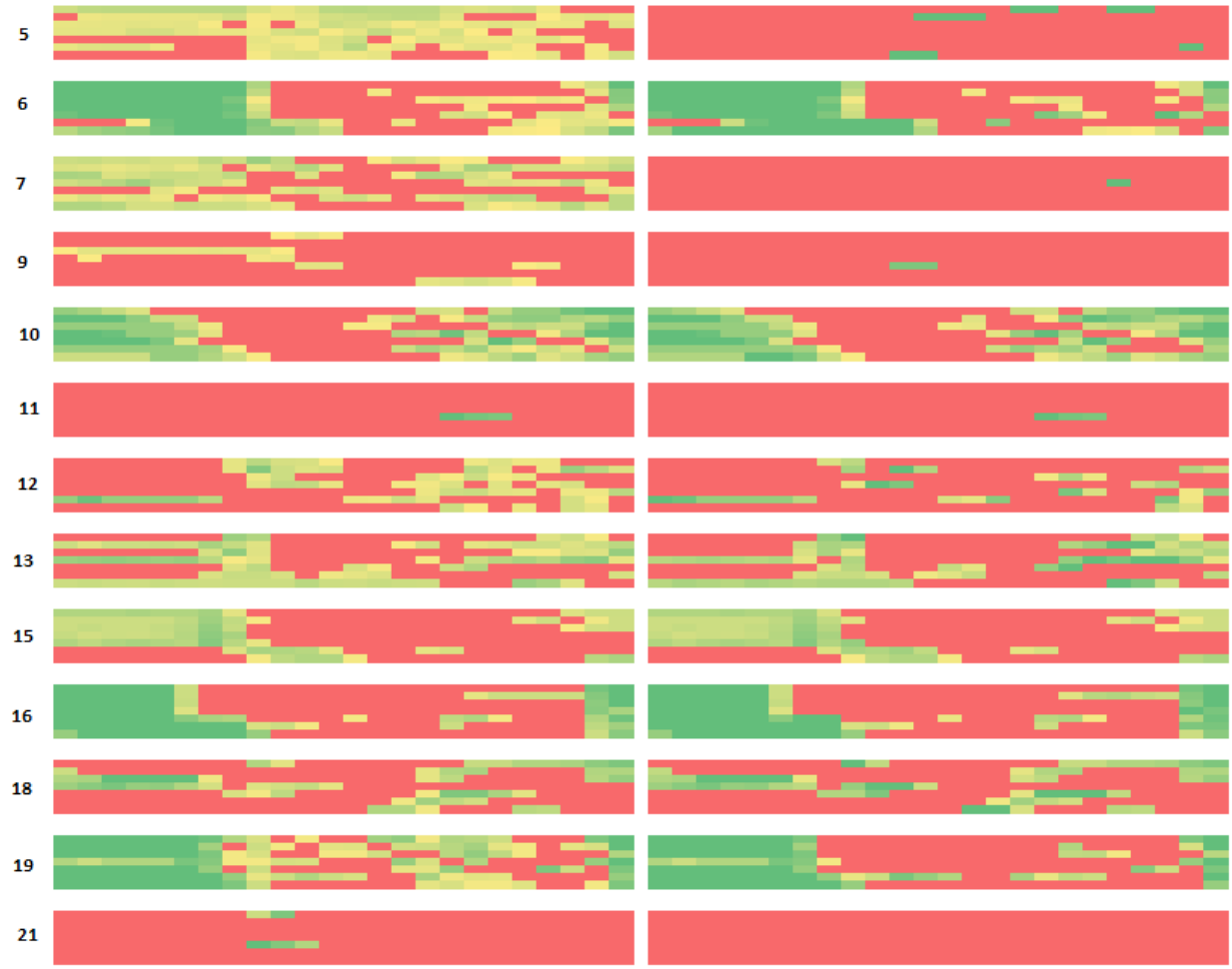


Figure 41 - Charging habits

Charging habits when on and off home Wi-Fi. Right is on Wi-Fi. Values are probability that phone is charging 0-1 with 0 being red, 0.5 yellow, 1.0 green. The individual cells are divided in 7 rows (monday-sunday) and 24 columns indicating hour of day.

WifiOnHomeSSID

When the person is at home on the domestic Wi-Fi the person is considered to be at home. In the graph below

Figure 42 - Probability of a person being on home Wi-Fi

Values are probability 0-1 with 0 being red, 0.5 yellow, 1.0 green. The individual cells are divided in 7 rows (monday-sunday) and 24 columns indicating hour of day.

For test subjects 5, 7 and 21 it is clear that they do not use Wi-Fi that much and therefore it can be hard to make a base truth for those subjects without having data from a home automation system.

5.1.1. Creating a weekly schedule

The data collected shows for some people that there are patterns that occur on a weekly basis. To illustrate this the diagram below shows the distribution of distances. The green line shows the 25 percentile, the blue line shows the median and the red line shows the 75 percentile. In the example below the persons is most likely working on tuesdays and some times having an activity at night around 20. The work place is approximately 6-7km away

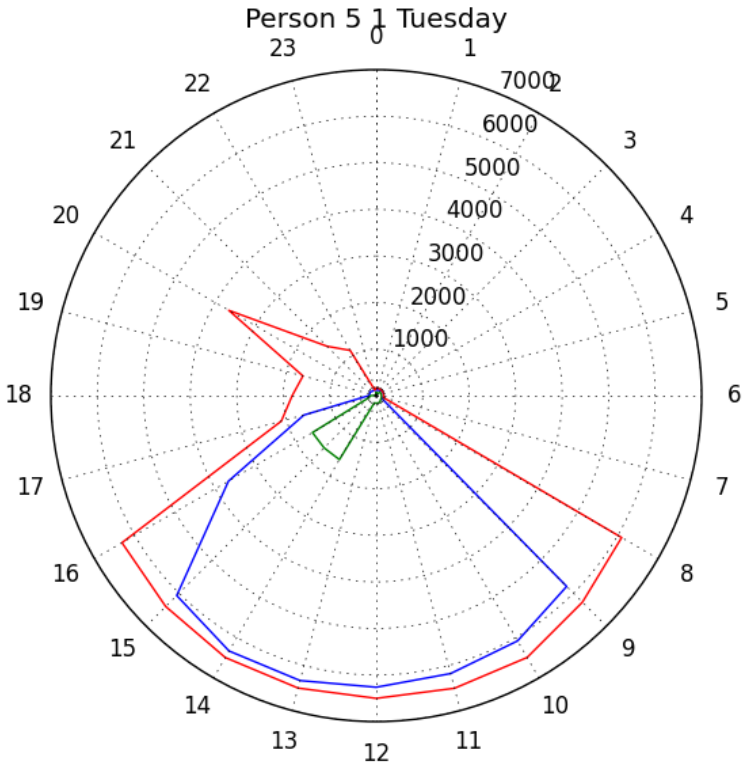


Figure 43 - Circadian rhythm

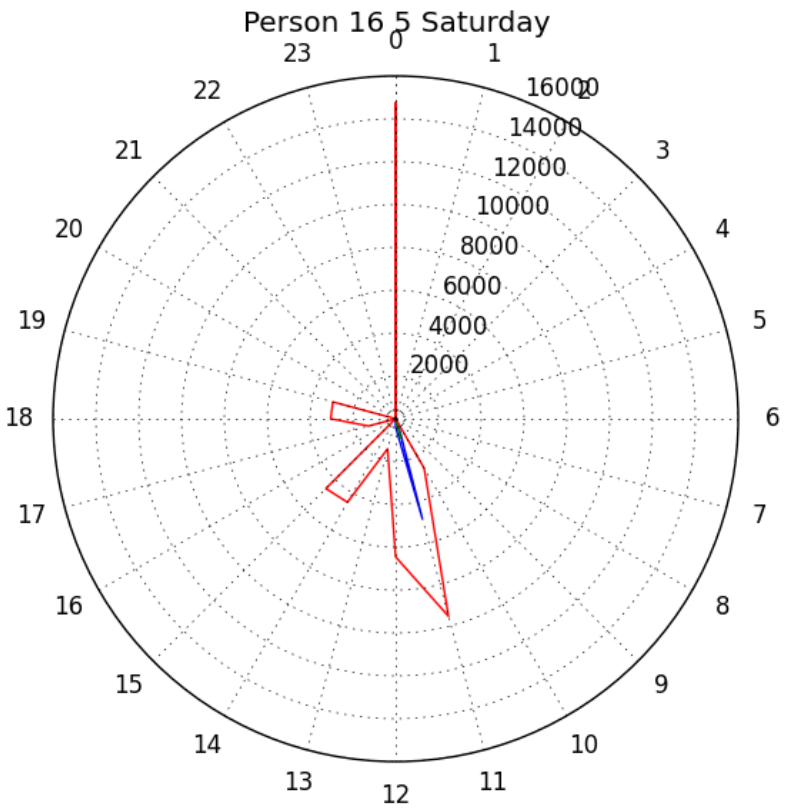


Figure 44 - Circadian rhythm 2

During weekends the patterns are generally towards being at home. In this diagram the green line does not show and the blue line only appears at 11. This means that over 50% of the times the person is at home almost all day Saturday.

Diagrams have been constructed for all persons on all days. They are included in the enclosed CD.

6. Conclusion

The knowledge about current and future presence can help home automation systems save energy by planning the power usage better and by using less energy in an uninhabited situation. By making a dynamic schedule that predicts when persons in a household is at home it will be possible to switch off sockets, floor heating and other domestic devices.
To determine if people are at home a smart phone app was developed. Apps were installed on the smartphones of 13 test subjects. Before being able to log those test subjects had to register their home and the persons using the app so that the system would know how 2 persons related. During a period of three months these test subjects have logged anonymous data to the system. Anonymity was reached though a website where people would register and receive their id for use in the smartphone. After having installed the app they were able to log data anonymously not disclosing any locations or persons. The app developed deals with the tradeoff between location accuracy and battery consumption and does not consume enough power to appear on the list of most battery consuming apps
The data collected was good enough for creating a weekly pattern for some of the test subjects. For other test subjects the patterns was not good enough to make qualified decisions. Some data features were recorded wrongly resulting in a feature being unusable. Some of the data was questionable and needs more attention before it can be used.
The prediction framework was only implemented to the state that it could detect weekly schedules. The prediction of coming home from a trip that is not part of the weekly schedule remains to be solved. The data quality was evaluated and it was possible to make a system that uses anonymous logs from smart phones to determine and predict presence.
The modelling of the system and the logical components made it possible to register people anonymously and register people living together as one household. This decision cost a lot of time and since this information was not used it ended up costing on the time that could be spent on treating the data.
During the project a lot of different measures were taken to ensure correctness and to add features with little or no value to the final product. Example of this is generation of ids on a device ensuring that two installs would produce the same id. The protection of administrator interface where the database could have been used to perform these operations. The construction of an orchestration layer that could have been replaced with a simple script doing an insert in a single table in the database.
Even though these extra efforts does not create value to the project at this point they are essential to the project if it ever were to become a working software system. The design was made as a proof of concept that it is possible and that it can be done using the proposed components. In a project where the focus is not holistic and tries to encapsulate the entire problem domain a simpler solution would have left more time for processing data from the devices.
On a personal note this project has taught me a lot. Having worked on something that you are so enthusiastic about has its pros and cons. A personal project where the desire to do it exactly right overshadows the ability to make cold business decisions about what is feasible challenges you. Looking back on the project I now know that every decision has be to evaluated in terms of estimated time it takes versus the value it adds to the project.

7. Bibliography

Distributed systems: Principles and Paradigms by Andrew S. Tanenbaum (Second edition)

Object-Oriented Software Engineering Using UML, Patterns, and Java by Bernd Bruegge & Allen H. Dutoit (Third edition)

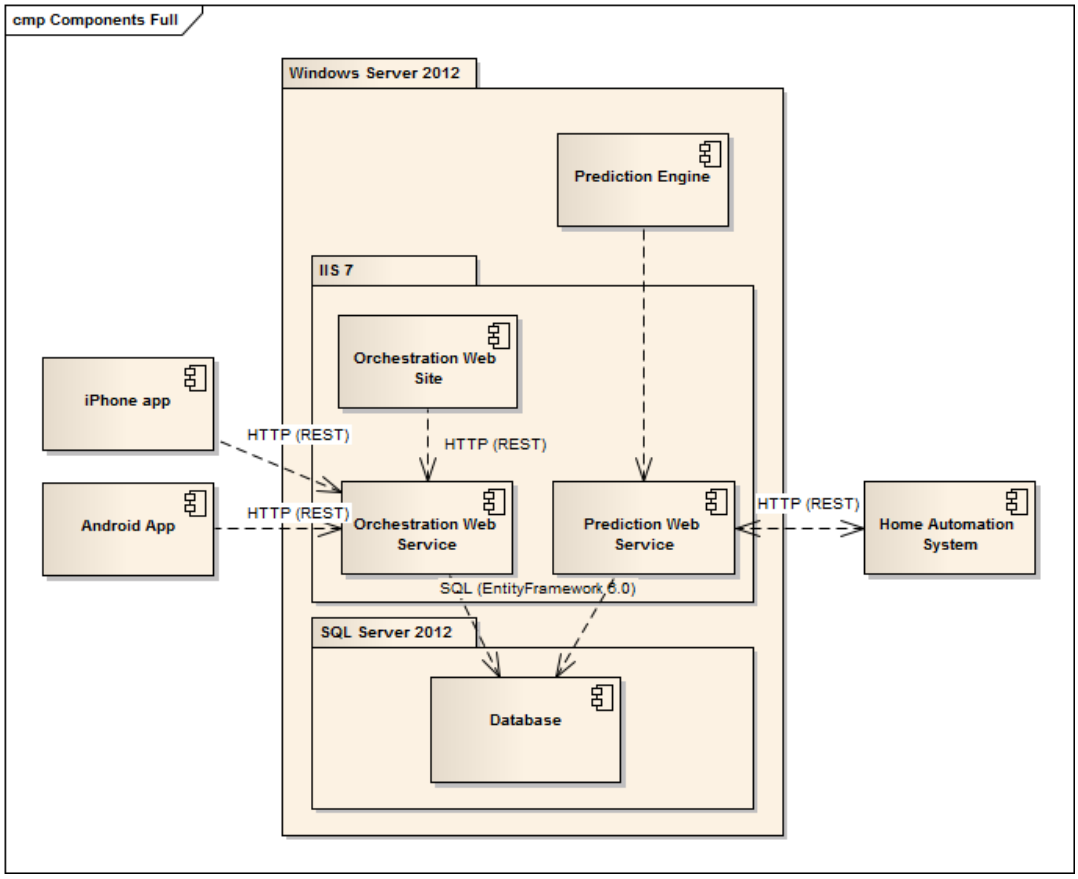
8. Appendix

8.1. Deployment diagram (Home Automation)

The diagram in this appendix shows how the real solution should be implemented. An extra web service is added, the so-called Prediction Web Service. This web service is responsible for providing information upon request to the Prediction engine and the home automation systems.

The prediction engine is a piece of software that runs periodically and checks if events will happen in the near future. The prediction engine will most likely use external libraries to perform machine learning algorithms on the data in order to find patterns. The prediction engine is also responsible for extracting knowledge from the gathered data and to calculate and keep intermediate results. The establishment of the service and the engine will facilitate information to be gathered and sent to the home automation system.

The home automation system can be behind a proxy as shown in cloud deployment example. This system is required to implement the interface defined by the Prediction Web Service



8.2. Complexities in the integration of real home automation systems

Due to a number of complexities no real home automation systems will be considered in this report. The system developed would benefit from this kind of integration and this document will present the reasons for not making the integrations in data gathering and the phase of acting out the plan that could be made from knowing when a home is uninhabited.

Complex and time consuming
Developing an integration for a hardware system is time consuming and requires different integrations for each type of hardware system. These different systems also have different kinds of sensors which in some cases needs to be "normalized" to make sense when gathered across systems.
In addition to developing the integrations they also need to be installed and configured and configuring several home automation systems manually is significantly harder and more time consuming than installing an app on a smartphone.

Small population
There are significantly more people who have a smartphone than homes that have an intelligent home system. This means that gathering enough test subjects to produce enough data will be hard and possibly quite time consuming.

Value of outcome
Although the sensor values would have an immediate value for the project, the output of the system does not have a direct application. Decisions have to be made on how the knowledge of presence should affect the physical installations in a home. These decisions have to be implemented in software which adds time and complexity to #1. In addition to that decisions regarding heating and cooling can't usually be determined by an average user as this requires knowledge about the thermal properties of the building.

8.3. Process

The process of finding test subjects and the communication to these impacts the test results and therefore this method will be described. The process consist of several steps and there are multiple scenarios to describe. The process can be divided into two main parts: 1. Gathering research participants and 2. Setting them up. Before explaining the process the different platforms and actors needs to be described.

Research conductor
The person in charge of the project; in this case myself. The responsibilities involve handling the test subjects with regards to promotion, enrollment, withdrawal, creating terms of data use, enforcing these terms, answering questions.

Research participant
The persons enrolled in the project, gathering and submitting data. The research participant is considered a such after having installed the app and thereby accepted the terms of use. The research participant or test subject is responsible for enrolling and withdrawing and to ask questions if he/she has any. The research participants can have their personal details deleted at any time, but the anonymized data collected from his/her smartphone can never be subject to such requests.

Registration website
This website is responsible for providing information about the project and to collect contact details of people interested in the project. Further the website should inform about the terms of participation.

Email provider
The research participants email provider. This actor or instance is included to show the full complexity of the process.

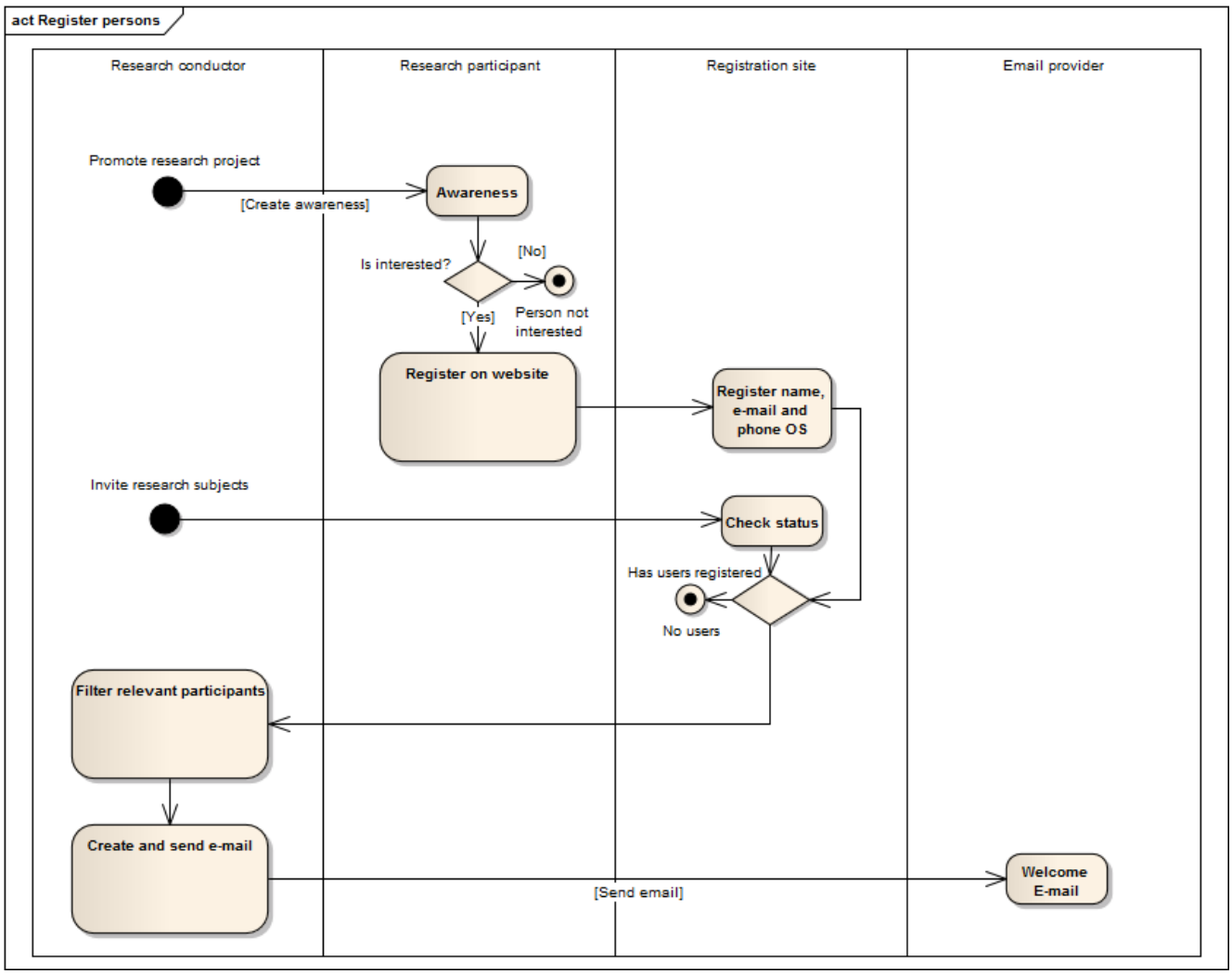
Orchestration website
The second website developed during this project is responsible for facilitating the requirements set up by the domain model. The website is responsible for the creation and linking of objects on behalf of the research participant. The website also serves the purpose of providing an overview for the research conductor. This information requires authentication so that only authorized persons can gain access to this information.

App store (Google Play)
Google Play is the de-facto store for the Android operating system. It is responsible for providing the AtHome app to interested users. For other operating systems a similar facility or store should provide the app for other platforms in the future.
The research participants are expected to know how to use this service.

AtHome app

The AtHome app is downloaded and installed by the research participant from the respective app store. The app is responsible for monitoring the sensors on the phone and submitting these values to the orchestration service according to the requirement specification.

The activity diagram below gives an overview of the processes:



Creating awareness

During the project, different approaches for creating awareness has been used. First, a website was created with the purpose of explaining the research project and requesting people to sign up as being interested in participating in the project. The people interested has registered with their name, e-mail address and the operating system of their phone. This data is separate from any other data in the project as this is personal information. Read more about this in the privacy section.

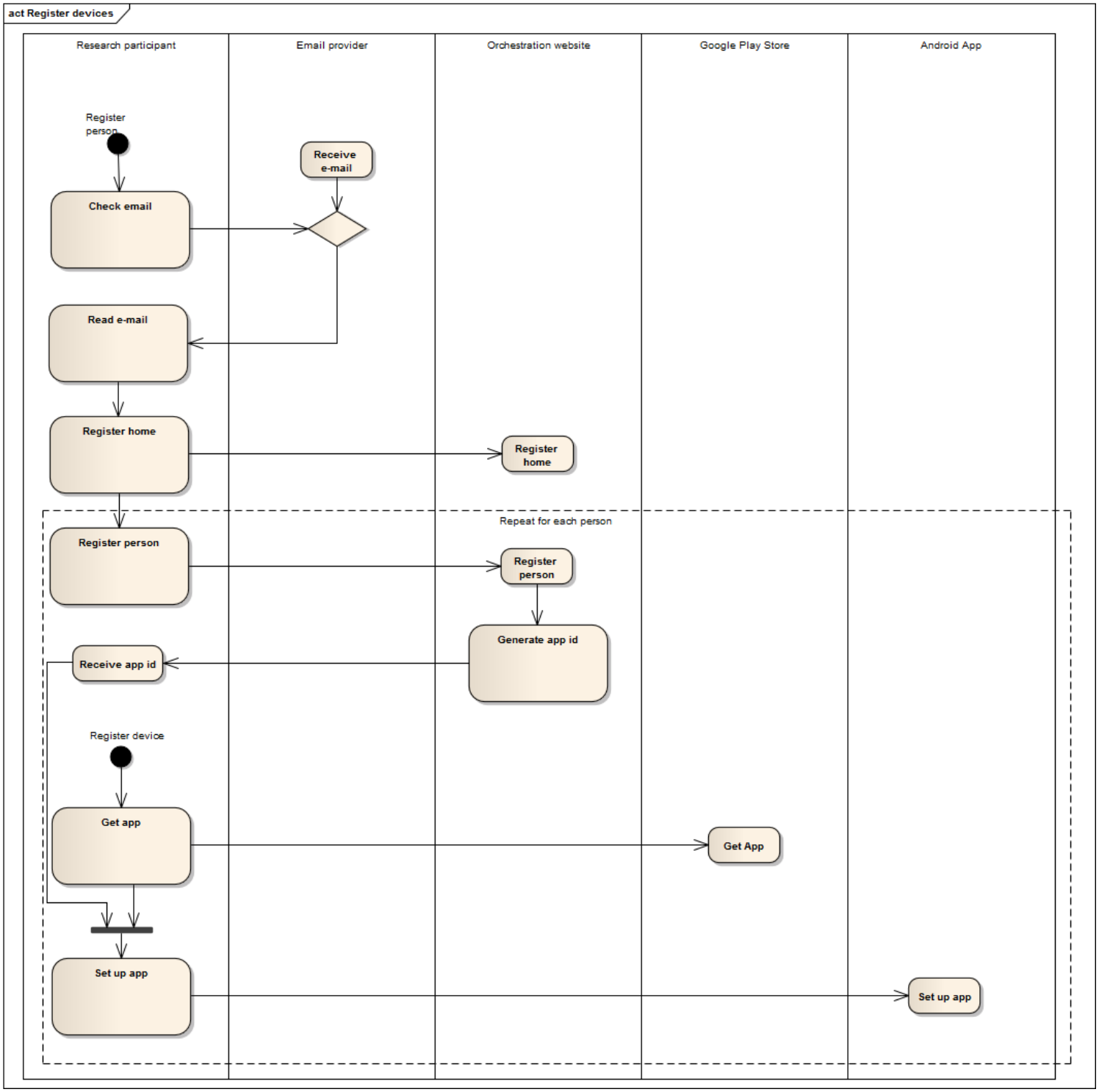
This link has been distributed via social networks both physically and on the internet. The advantage of the web based social networks is that people can easily share it in their social circles.

Invite test subjects

During the recruitment process test subjects were invited to participate by registering their household and their personal phones. As the Android app was the first and only app to be developed the filtering consisted solely of sending e-mails to people who indicated that they own an Android phone. This e-mail contains a link to the app, a link to the orchestration web site and a guide to help them through the respective processes. At the time this process was designed it was the intention that updates to the app, would be presented to the relevant subjects by e-mail notification. This notification would serve the purpose of getting more people to update their app to the newest version faster. The motivation of making this invitation separate from the setup of the app is to preserve anonymity.

Enrolling persons

When the test subjects have received the email, they have all the information needed to enter the project. The process consist of 2 parts: registering the household and registering the individual devices. The activity diagram below shows the process:



The steps are described in detail with illustrations in the appendix and hence this section will explain the context and reasons behind the process.

Registering household

Because of the data model devices are not registered on the home but on a specific person. The orchestration web site is in charge of creating these relations and thus the household needs to be set up before a device can be set up.

The process includes registering the “root object” which is the home and the persons living in that home. The test subjects are asked to use aliases or simply first names to distinguish the different persons without exposing information that can be used to identify a test subjects identity and place of living.

For each of these persons a socalled app-id is generated. The purpose of this app is is to make registration easier for the test subject. Typing on full GUID (32 hex characters) on the keyboard of the phone is hard and cumbersome and therefore a 12 digit number is generated for use in the app. By only having numbers that are visible on the phones dialer in stead of the full qwerty keyboard makes it easier and more convenient for test subject. The complexity of the id is reduced by the cost of combinatoric possibilities. To accommodate the potential threat of attacks a time limit has been set on the validity of this number.

When the research participant has registered the home and him/herself and gotten the app id, the app can be downloaded and set up by using the persons smartphone. When installing the app, the research participant explicitly accepts the terms of use. To set up the phone, the research participant needs to be at home connected to the residential WIFI. Setup includes registering the location of “home” and the ID of the residential WIFI locally on the phone for later comparison when logging.