OULU UNIVERSITY OF
APPLIED SCIENCES

Alexander Smirnov

MOBILE TERMINAL FRAMEWORK SOFTWARE FOR SUPPORTING SAFETY NAVIGATION

MOBILE TERMINAL FRAMEWORK SOFTWARE FOR SUPPORTING SAFETY NAVIGATION

Alexander Smirnov
Master Thesis
Autumn 2016
Master’s Degree, Wireless Communications
Oulu University of Applied Sciences

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Ylempi korkeakoulututkinto

Tekijä: Alexander Smirnov
Opinnäytetyön nimi: Mobiiliterminaalin ohjelmistoalusta tuekseen turvallista nagivointia
Työn ohjaajat: Timo Vainio, Markku Korhonen
Työn valmistumislukukausi ja -vuosi: Syksy 2016 Sivumäärä: 113

Tämä opinnnöytetyö oli tarkoitettu Ryhti projektin jatkokehitykselle, jossa luottiin älykkäitä kotiympäristöjä ja niihin liittyviä hyvinvointipalveluita. Ryhti-projekti oli toteutettu Pehr Brahen ohjelmistolaboratoriossa vuonna 2011 ja tämän opinnäytetyön tarkoitus oli siirtää Ryhti-projetin konseptit ja ideat Android-mobiilialustalle. Lopputuloksena Androidille piti luoda ohjelmistoalustan jonka kautta voisi tarjota turvallista navigointia puhelinten ja älykkäiden kellojen kautta.

Opinnäytetyössä ensin esitettään mobiiliterveyden ja hyvinvoinnin konseptteja. Esimerkkejä vastaavista olemassa olevista tuotteeista esitettään ja samalla käydään läpi niiden etuja ja haittoja, ottamalla kuvioon terveys- ja hyvinvointipalveluiden saattavuus kehitetysmaissa. Samalla näytetään esimerkkejä älypuhelimeen pohjautuvista terveystyökaluista ja käydään läpi niitä tuloksia mitä oli saattuu niiden käytöstä reaalimaailmassa. Seuraavaksi esitetään konspetit modulaarisesta mobiililaitteistosta ja ohjelmistosta.

Keskustelu modulaarisista ratkaisuista tuo puheenaiheeksi konseptit älykkäistä hyvinvointiympäristöistä ja esineiden Internetistä (Internet of Things). Näiden älykkäiden järjeselmijen pohjalta esitetään turvallisen navigoinnin konsepttejä ja kerrotaan mobiiliterminaalin ohjelmistoalustan (MTFS) pääominaisuuksista. Systeemin tärkeimät käyttötapaukset esitetään lukijalle ja samalla käydään läpi esimerkkejä vastaavista olemassa olevista tuotteista, ja vertaillaan niiden etuja ja haittoja. Puheenaiheeksi myös otetaan mobiilikäyttäjän tietoturvasuojaus ja lukijalle esitetään kaikki päätökset ja ratkaisut mitä tämän opinnäytetyön tekijän piti tehdä käyttäjän tietoturvallisuudelle mobiiliterminaalin ohjelmistoalustan kehityksessä.

Seuraavat kappaleet esittävät kehitetyn ohjelmistoalustan teknillisiä toteutuksia ja komponenttien ominaisuuksia. Lukijalle kerrotaan miten alustan komponentit oli testaattu ja mitä parannuksia niille oli tehty. Loppukappaleissa alustalle esitetään kehitysehdotuksia ja samalla näytetään esimerkin muusta mobiilisovelluksista mitä nyt olisi mahdollista toteuttaa kehitetyn mobiiliohjelmistoalustan pohjalta.

Asiasanat: terveys- ja hyvinvointipalvelut, paikannus, mobiilipalvelut, palvelukeskeinen arkkitehtuuri, Android

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Information Technology

Author: Alexander Smirnov
Title of the Master Thesis: Mobile Terminal Framework Software for Supporting Safety Navigation
Supervisors: Timo Vainio, Markku Korhonen
Term and year of completion: Autumn 2016 Number of pages: 113

This Master's Thesis was done as a continuation of the ideas originally outlined during the Ryhti project, performed in the Pehr Brahe Software Laboratory back in 2011. One of the aims of the Ryhti project was to create ubiquitous and intelligent home environments for elderly people, while this Thesis was aimed at creating a special software framework for enabling mobile well-being services on Android terminals, such as smartphones and wearables.

Keeping that task in mind, the author introduces the concepts of mobile health-care and well-being services to the reader. The examples of real-life mobile health-care products and services are presented and reviewed. The author describes the benefits and challenges of applying mobile technologies to the health-care domain, especially in the scope of enabling access to health-care services in developing countries.

A number of smartphone-based health-care equipment and applications are also presented to the reader. The results of in-field testing of such equipment are also discussed, together with reviewing the modularity of such solutions.

The concepts of Safety Navigation system are presented to the reader, and the main functionality of the Mobile Terminal Framework Software (MTFS) is described. The most important use-cases of the Safety Navigation system are presented and a number of similar real-life products are reviewed, comparing their benefits and drawbacks.

Some end-user's privacy issues are also reviewed, helping the reader to understand the decisions and compromises that the author made within the MTFS framework regarding data security.

The following chapters describe all technical details about implementing, testing and improving the components in the MTFS framework, including other mobile applications that could be developed with the MTFS framework.

Keywords: health-care, well-being, geographic information system, SOA, Android

PREFACE

The Mobile Terminal Framework Software (MTFS) described and discussed in this Master’s Thesis was supposed to extend some well-being services outlined and developed in a research project Ryhti at Pehr Brahe Software Laboratory back in 2011. At its core the developed mobile framework is an attempt to bring some of Ryhti's well-being services to Android-based mobile terminals.

While this thesis work was primarily targeted at mobile terminals and wireless communications, the corresponding back-end system was also developed and discussed as a part of the final solution.

The original aims for this Master’s Thesis were outlined by researches at Pehr Brahe Software Laboratory, namely Vadym Kramar, Markku Korhonen and Yury Sergeev. However, due to the prolonged Thesis work, the original Ryhti project ended without the creation of the MTFS framework. The developments of the MTFS framework were continued on author’s own time schedules and resources.

The author is thankful to all above mentioned researches at Pehr Brahe Software Laboratory for the subject of this Master’s Thesis and for getting a full freedom to turn these ideas into reality.

The author is also thankful to Timo Vainio and Kaija Posio for their supervision and valuable comments which helped to considerably improve the Thesis.

Jyväskylä, 29.11.2016
Alexander Smirnov

TABLE OF CONTENTS

VOCABULARY

AI – Artificial Intelligence – The intelligence exhibited by software or hardware systems

Android – Open-source operating systems developed by the Google company

Apache HTTP Server – Popular web server software used to build Internet servers and proxies

ASCII – American Standard Code for Information Interchange – A character encoding standard for presenting electronic text symbols in computers, communication networks and other devices

AU – Aware User – Member of the family or a relative to a mobile terminal user

Authentication – the process of verifying identity of end-user or a system by checking their credentials; most commonly authentication is a process of logging of end-users to some system with given user-name and password

Authorization – the process of verifying that an end-user or a system have right to access some functionality of other systems; most commonly authorization is associated with checking rights and privileges for accessing some system resources and services

Base64 – A group of binary-to-text encoding algorithms that allow represent electronic binary data in ASCII string format by translating such binary data into radix-64 representation

Biosensor – An analytical device that is able to trace and convert biological processes into electronic signals

Bluetooth – A wireless technology for exchanging digital data over short distances, commonly within a few meters long ranges

CSS – Cascading Style Sheets – A style sheet language for describing look and feel of web-based documents, pages and user interfaces

Dalvik – Process virtual machine in Android operating system

EXIF – Exchangeable Image File Format – A standardized set of tags used for describing metadata about multimedia files

GNU – GNU is Not Unix – A free software collaboration project aimed at giving computer users full freedom in use of their computers and software

Go – or Golang – An open-source, general-purpose programming language originally developed by Google in attempt to resolve complex issues with C++ systems inside their infrastructures. Golang is compiled, statically typed and garbage collected language which also has built-in support for concurrency and many Internet protocols

GPS – Global Positioning System – A space-based satellite navigational system that provides location and time information to its end-users on Earth

GPX – The GPS Exchange Format – A light-weight XML exchange format for the interchange of GPS data between applications and Internet services.

GUI – Graphical User Interface – A type of interface that allows users to interact with electronic device or software application through graphical elements and visual indicators

HTML – Hyper Text Markup Language – The standard markup language for creating web pages and web-based user interfaces

HTML5 – The fifth revision of the HTML standard markup language aimed at best possible support for digital interactive multimedia while keeping readable by humans

HTTP – Hypertext Transfer Protocol – is an application protocol for distributed, collaborative and hypermedia information systems. The HTTP protocol is the foundation for the World Wide Web

HTTPS – HTTP over TLS (also known as HTTP Secure) – is an application protocol for secure data communications over unsecured computer networks. HTTPS uses HTTP for data communications with TLS used for data encryption

IDE – Integrated Development Environment – A set of software applications, tools and utilities that provides developers with comprehensive facilities to software development and testing

IoT – Internet of Things – Interconnection of uniquely identifiable computing devices within the existing Internet infrastructure.

Java – General-purpose programming language and software platform originally developed by the Sun Microsystems Inc. and aimed at maximal source code and binary portability across a range of hardware platforms and computing environments

JavaScript – A dynamic programming language originally aimed at enriching functionality of the web pages

JVM – Java Virtual Machine – An abstract computing machine able to execute computer programs compiled into Java byte-code

JSON – JavaScript Object Notation – A open standard format for representing JavaScript objects in textual form, which was also adopted by many other computing environments as language and system independent format for representing electronic data

LAMP – An architectural solution stack, formed from open-source components, typically suitable for web applications. The LAMP is an acronym from the names of its four components: the Linux operating system, the Apache HTTP server, the MySQL database system and PHP programming language. Currently the LAMP model has absorbed many other open-source components, but still keeping the same functional idea

Leaflet – An open-source JavaScript library for creating mobile-friendly interactive maps

Linux – Open-source operating system kernel compatible with Unix and POSIX APIs

M2M – Machine-to-machine – A broad term that refers to technologies enabling communication between different devices of the same type

MQTT – MQ Telemetry Transport – A machine-to-machine and Internet of things lightweight connectivity protocol based on “publish-subscribe” pattern

MT – Mobile Terminal – A mobile device used by the end-user of described system

MTFS – Mobile Terminal Framework Software – A special software library for mobile devices that enables creation of mobile health-care and well-being applications

MTU – Mobile Terminal User – A user of the mobile terminal or device

MySQL – Popular open-source relational database management system

NFC – Near Field Communication – A set of technologies that enable mobile devices to establish radio communications with each other by bringing them into proximity or close to each other

NDK – Native Development Kit – A toolset for Android platform that enables creation of applications with native-code languages such as C and C++

OAuth – An open protocol for allowing secure authorization of users in a simple and standard method from web, desktop and mobile applications

OGS – Open Geospatial Consortium – An international industry consortium of commercial companies, government agencies and universities participating in developing public interface standards for geographical and geospatial technologies.

ORDBMS – Object-relational Database Management System – A database management system similar to relational database, but with additional support of objects, classes and inheritance in the database schemes

PC – Personal Computer – A term for describing any commonly available desktop or laptop computer

PostgreSQL – A popular open-source object-relational database management system with emphasis on extendability and compliance with the standards

Publish-subscribe – A messaging pattern where “publishers” are sending messages to dispatchers, while “subscribers” are retrieving such messages only after a notification from the dispatcher, based on message topic, meta-data or contents as examined by the dispatcher

QR Code – Quick Response Code – A two-dimensional graphical bar-code used for machine-readable optical labels

RDBMS – Relational Database Management System – A database management system based on the relational model

REST – Representational State Transfer – A software architectural style used in the World Wide Web systems and a way to offer network APIs over HTTP protocol

SDK – Software Development Kit – A set of software development tools that enables creation of applications for certain software or hardware platform

SFTP – Secure File Transfer Protocol (also known as SSH File Transfer Protocol) – is a file access, management and transfer network protocol over reliable and secured data streams, like provided by the TLS or SSH protocols

SMS – Short Message Service – A text messaging service available for GSM networks end-users

SOA – Service-Oriented Architecture – A set of principles and methodologies for designing and developing complex software in form or interoperable services

SP – Service Person – A worker of the public or commercial company that offer monitoring and rescue services to the mobile terminal users

Spring Framework – A Java software platform that provides infrastructure support for developing enterprise applications

SQL – Structured Query Language – A special-purpose programming language designed for manipulating and extracting data stored in database systems.

SQLite – An embeddable relational database management system provided as a C programming library

SSH – Secure Shell – is a cryptographic network protocol used for operating securely network services over totally unsecured networks. SSH provides a secured data channel over unsecured network in a client-server architecture

TLS – Transport Layer Security – is a set of cryptographic protocols that provide secured data communications over computer networks and Internet

UHE – Ubiquitous Home Environment – A user-centric system deployed to user's home living environment or area

Unicode – An international standard for encoding and representation of electronic texts and symbols expressed in most of the world's writing systems

Unix – A family of multitasking and multi-user operating systems that have been originally developed by Ken Thompson and Dennis Ritchie at the AT&T Bell Labs research center back in 1970 and has pioneered many of today's OS and application programming technologies

UPnP – Universal Plug and Play – A set of networking protocols that allow networked devices to seamlessly discover each other and establish functional network services for data sharing and communications

URI – Uniform Resource Identifier – is a string of characters used for unique identification of electronic resource

URL – Uniform Resource Locator – is a reference to a web resource that specifies its location on a computer network together with a mechanism for its retrieving. URL is a specific form of the URI

UTF-8 – A character encoding standard for encoding Unicode code points in variable-length 8-bit code units

Wi-Fi – A local area wireless technology that allows electronic devices to exchange data or connect to the Internet

Wi-Fi Direct – A Wi-Fi standard that enables easy connectivity between devices without creating a wireless access point

WPA – Wireless Access Point – A device that allows wireless devices to connect to a wired network using the Wi-Fi wireless communication technology

WSG84 – World Geodesic System 1984 – A standard coordinate system for the Earth used in cartography, geodesy, and navigation including by GPS.

WWW – World Wide Web – is an electronic information space where documents and other web resources are identified by URLs and can be accessed by the Internet

XML – Extensible Markup Language – A standardized format for representing self-describing documents and data in textual form

1 INTRODUCTION

The Mobile Terminal Framework Software (MTFS) described and discussed in this Master’s Thesis was supposed to extend some well-being services outlined and developed in the research project Ryhti at Pehr Brahe Software Laboratory back in 2011 (1).

At its core the developed mobile framework is an attempt to bring some of Ryhti's well-being services to Android-based mobile devices, such as smartphones and wearables. While this work was primarily targeted at extending the functionality of the mobile terminals, the supportive back-end system was also developed as a part of the final solution.

1.1 The original goals

Scientists and demographers are predicting that the European Union will have a demographic aging challenge in near decades. An average European now lives longer than one's predecessors 100 years ago and by 2050 the number of people aged 65 and more is expected to raise sharply by more than 50%, making about 30% of the total population of the Europe (2).

Such diversity in population will definitely bring additional challenges to health-care and well-being domains where significant increase of patients and customers of senior ages will demand more human and financial resources in order to provide at least the same amount of services of affordable quality.

The utilization of advanced information technologies in health-care and well-being domains, including smart monitoring systems, mobile medical equipment and even robotics, could help with ensuring the availability of some medical and well-being services to the grown number of patients, while at the same time preserving the efforts of medical personnel and keeping involved costs at reasonably low levels.

The information and knowledge obtained from such intelligent medical equipment and smart monitoring systems could help doctors and nursing staff to get a better view of cared-for patients, possibly leading to improved treatments and faster patient recoveries. That, in its turn, could prevent disease complications and make medical treatments more effective and less costly.

At the same time patients could start benefiting from a better knowledge about their health state, and make better efforts in disease treatments and in turning down bad habits of unhealthy lifestyle.

If health-care and medical resources will be cutting under ever growing number of patients, such intelligent medical systems could start playing a significant role in saving and increasing the efforts of medical personnel. At the same time patients could start benefiting from automatic nursing services, based on more extensive and broader personal data.

It is worth to mention that any automation of medical data processing or information analyzing will not replace a doctor or a nurse from the decision making process, but rather will enrich their awareness about the situation and thus it will help them to make better decisions about cared-for patients. And in much the same way, doctors and nurses could get better insights of outlined treatment plans, based on a much broader and detailed view of the patient's data.

Such approach, when medical staff can get some insights of patient's data from the artificial intelligence systems, has already become an accepted practice, like in various cases from the IBM Watson Health portfolio (3).

Therefore, from much similar point of view, this Thesis work is aimed to study how easy it would be to offer mobile health-care and well-being services through commonly available mobile devices, such as smartphones, with the support from intelligent back-end systems and ubiquitous home environments.

1.2 Enabling mobile health-care

The quite rapid expansion of mobile computing technology and mobile networks that we have seen during the last decades has enabled the access to the mobile Internet and data services for a quite vast amount of human population (4).

Such rapid expansion of mobile technology is giving the opportunity to apply it also to the health-care and well-being domains, where mobile technology could bring medical services closer to patients and increase the productivity of medical personnel.

It is quite important to mention that most of the mobile technology expansion is currently happening in the developing countries of Latin America, Asia and Africa (5), while large portions of population in these countries still have limited access to medical services due to economical, social or geographical factors.

Probably the most interesting aspect of this mobile expansion is that quite often the mobile device, such as a smartphone, is becoming the first computing device that a person actually has access to in the above mentioned regions.

Later on, that smartphone starts serving as the only de-facto hardware and software platform for accessing the Internet and exchanging data. This makes smartphones a quite important computing platform in terms of enabling personal health-care services and providing well-being applications.

The smartphones of today are quite powerful computing and communication devices that can run rich multimedia applications and access wireless networks at quite impressive data rates. The pace for more powerful mobile devices and faster wireless networks has never actually slowed down, as presented in the Figure 1.

Instead, the mobile devices has a tendency to integrate into more computing and connectivity technologies while constantly reducing the cost and difficulty of their production.

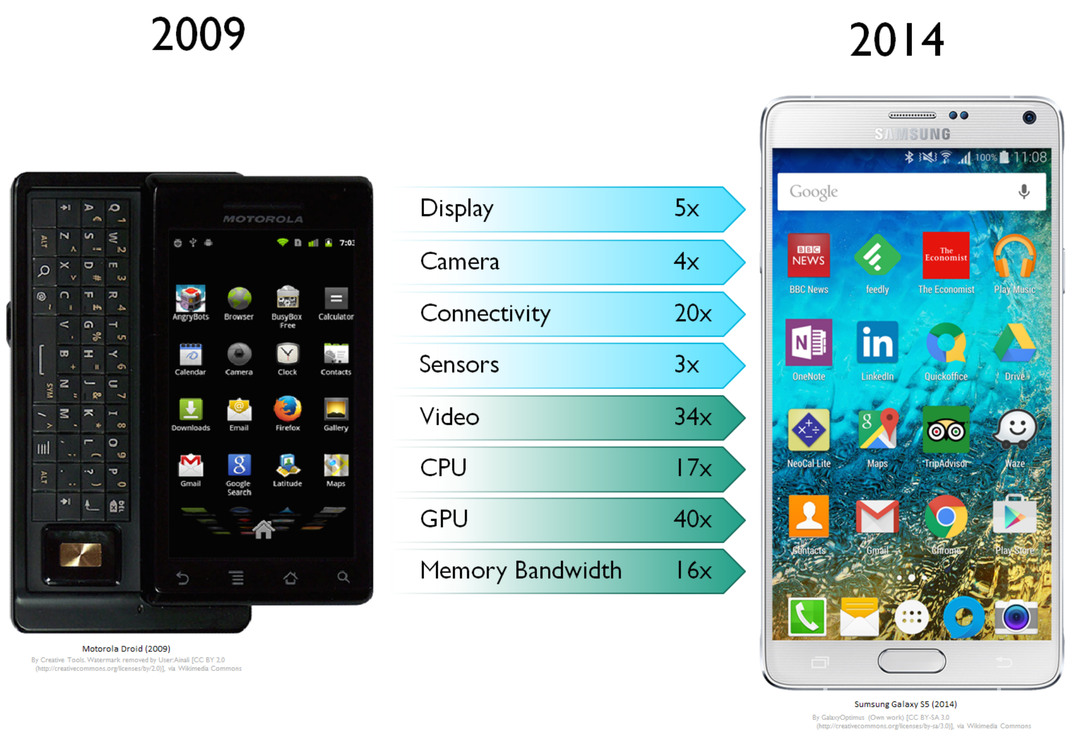


FIGURE 1. Overall increases in smartphone performances (6)

All this enabled a growing utilization of smartphones in the health-care system and well-being domains (7). And while currently mobile devices are most commonly used by doctors and carrying personnel as interactive terminals to the hospital information systems and electronic data records, there is also a growing trend towards mobile medical equipment running on smartphones.

Such trend is strengthened not only with a wide availability of the smartphones, but rather because of their extendability through custom software applications and hardware add-ons.

For example, the ViSi Mobile device in Figure 2 could read a wide range of medical parameters from the patient and instantly deliver that information to hospital information systems over a Wi-Fi connection (8).

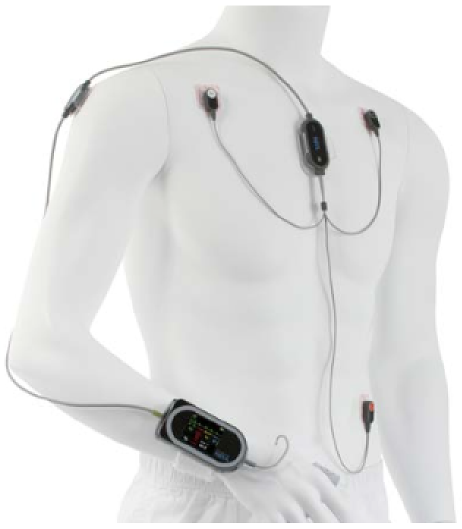


FIGURE 2. The ViSi Mobile device installed on a patient (8)
But despite all advantages of such mobile equipment, its utilization is requiring a proper infrastructure around them, such as special supportive appliances, carefully preconfigured wireless networks and vendor-specific back-end information systems.

In addition to that, such devices are usually representing closed products and solutions that are not aiming at supporting extension or customization from the third-party developers. One of the reasons for such restrictions are in vendors' business models that are typically relying on selling such products and related supportive systems to end-users on exclusive rights.

The other limitation, which is also partly affecting smartphone-based appliances, is in costly and quite often prolonged certification processes that specialized medical devices have to pass in order to be allowed for utilization at hospitals and medical centers (9).

There is, however, a technology trend towards wearable fitness and well-being electronics. For example, Sony's SmartWatch 3 and Fitbit's Surge wearable devices (presented in Figures 3 and 4 correspondingly) can measure pulse rates, read geographical position and deliver that information for further processing through the standard wireless communication protocols to a mobile phone or local PC (10, 11).

FIGURE 3. The Sony SmartWatch3 wearable (12)

And while the features offered by these wearables could vary from model to model, quite noticeable here is that they can share obtained health and location data though the standard communication protocols such as Wi-Fi, Bluetooth and NFC, and so enable further processing of such data by the third party applications.

FIGURE 4. The Fitbit Surge wearable (13)

Some vendors, e.g. Sony, are developing their wearables on top of open software platforms, such as Android Wear – the open source initiative from Google regarding the wearable devices (14).

Such approach enables the further customization of device functionality by the third-party applications and even through custom hardware extensions. The openness at the system and application level enables users to integrate such devices into other environments, such as smart homes or hospital information systems in a much interoperable and less costly manner, thus avoiding many inconveniences related to the integration of closed systems.

Another technological trend is driven by special mobile hardware and software appliances that extend smartphone's basic functionality and turn it into mobile medical equipment. One example of such mobile medical extension is the Mobile Colposcope presented in Figure 5.

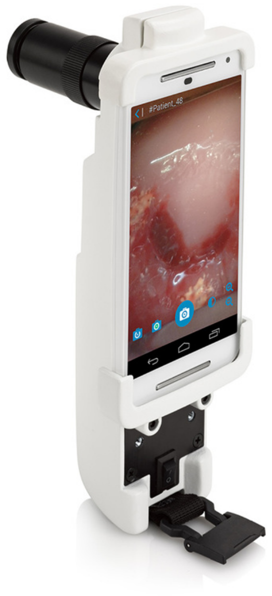


FIGURE 5. The Mobile Colposcope from MobileODT company (15)

The Mobile Colposcope enables colposcopy through the standard smartphone and special add-on hardware. With the specially developed mobile application it turns a smartphone into a fully functional colposcopy tool that could also be easily integrated into hospital information systems and patient electronic record facilities.

Another example of such add-on solution was developed by the Peek Vision company. It turns Android and Apple smartphones into fully functional ophthalmoscope devices, as presented in Figure 6.

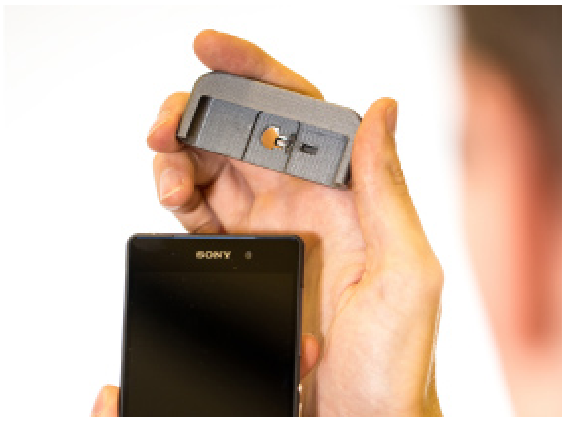


FIGURE 6. Mobile camera enhancement from the Peek Vision (16)

Such extension allows to perform eyesight tests and take high quality retinal images by a smartphone virtually anywhere in the world (16).

In addition, such retinal images could be remotely examined by experts all around the world, virtually bringing ophthalmology services to the places where access to basic medical services is limited or challenging due to economical or geographical circumstances (see Figure 7 below).

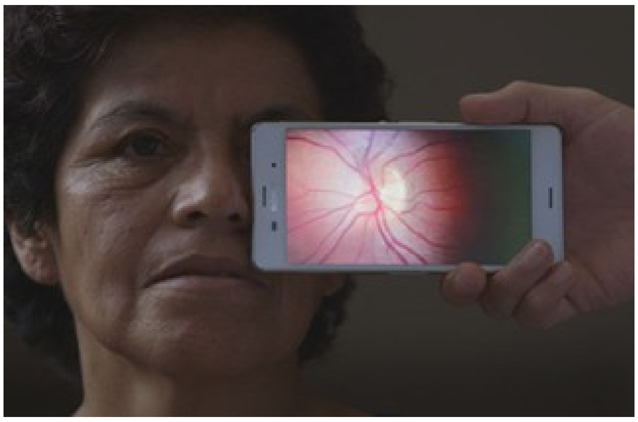


FIGURE 7. Example of taking retinal image by a smartphone with the Peek Vision appliances (16)

Another prominent example of turning smartphones into medical equipment is a special dongle for performing blood tests and diagnosis developed in the Columbia University by Samuel Sia's research group and presented in Figure 8.

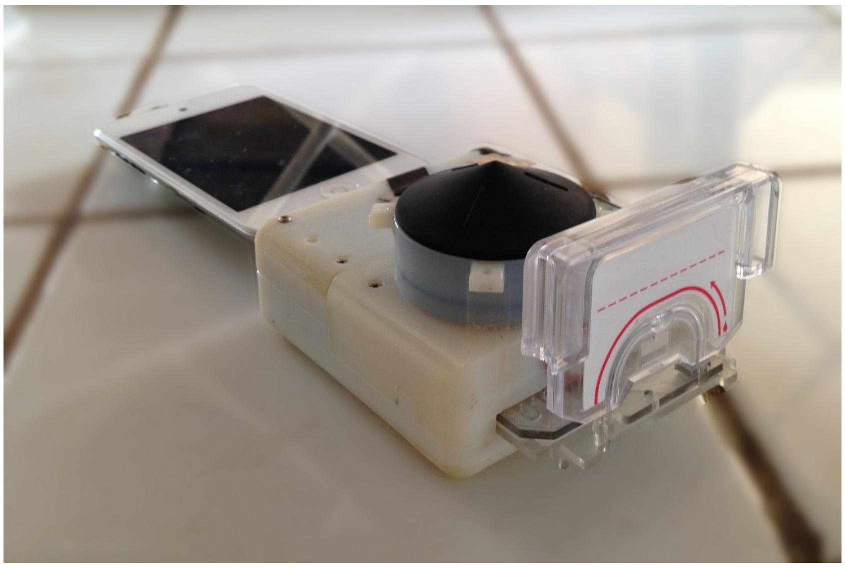


FIGURE 8. The lab on chip dongle developed by Samuel Sia's research group (17)

According to Sia’s report (17), this hardware accessory was intensively used in hospitals and villages of Rwanda and has enabled fast and reliable HIV and other disease diagnostics through the 15-minute blood tests. A schematic process of taking such blood tests with the developed dongle is presented in Figure 9.

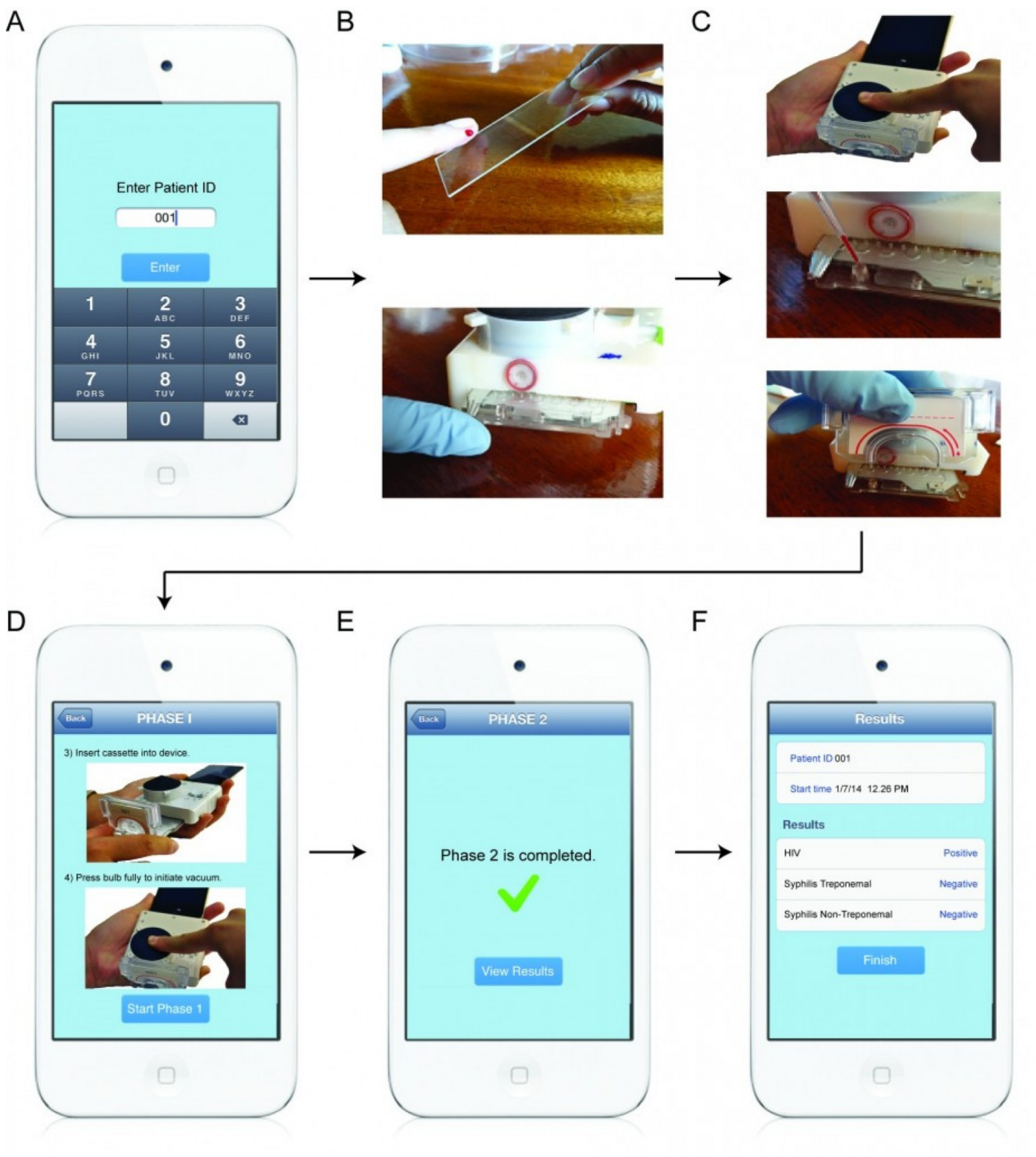

A
B
C
D
E
F
Enter Patient ID
001
Enter
PHASE I
3) Insert cassette into device.
4) Press bulb fully to initiate vacuum.
Start Phase 1
PHASE 2
Phase 2 is completed.
View Results
Results
Patient ID 001
Start time 1/7/14 12.26 PM
Results
HIV Positive
Syphilis Treponemal Negative
Syphilis Non-Treponemal Negative
Finish

FIGURE 9. Process of taking blood tests with the Sia's diagnostic dongle (17)
The results show that "a full laboratory-quality immunoassay can be run on a smartphone accessory" (18). At the same time Samuel Sia is estimating that the dongle's production will cost only $34, which is much cheaper than the price of $18.450 for a typical enzyme-linked immunosorbent assay equipment used to perform similar blood tests (19).

As an additional benefit, such simple equipment has required only 30 minutes of training for local medical personnel, while 97% of patients were pleased with the simplicity, speed and reliability of the provided diagnostics in quite challenging environments where access to medical services was very limited.

Further developments of the dongle by Samuel Sia's group has lead to a solution where digital data about blood tests was transferred into a smartphone application through the standardized audio jack interface, thus enabling the utilization of the dongle in smartphones across multiple vendors in a simple plug-and-play manner.

Some smartphone health-care applications, however, do not require any special hardware extensions at all.

For example, the HemaApp mobile application can measure human blood hemoglobin concentration without invasions using just a smartphone camera, as presented in Figure 10.

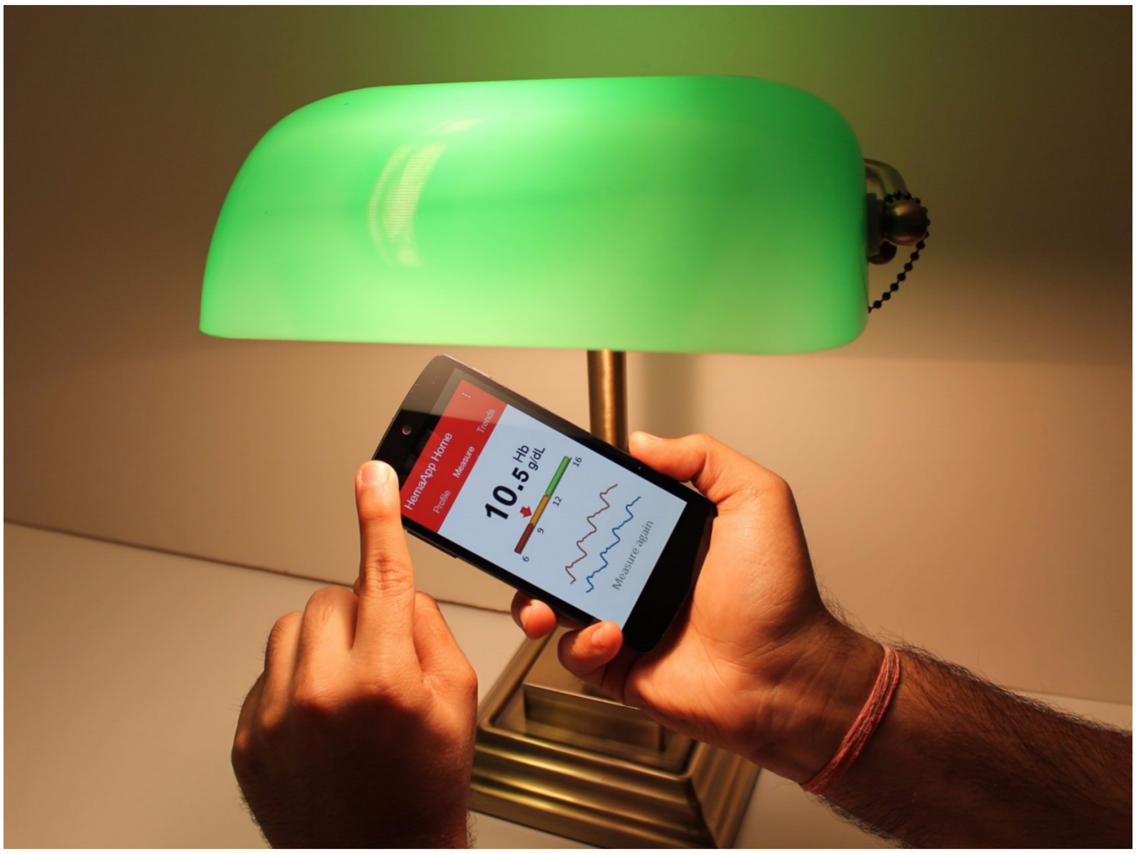


FIGURE 10. Example of measuring hemoglobin level using the HemaApp application (20)

The HemaApp application was developed by a research group in University of Washington and tested in Seattle Children's Hospital of Washington state (20).

The core idea behind HemaApp is to obtain estimations of users hemoglobin level by taking an image of their finger with a smartphone camera and LED flash, and studying the color of the blood by a special algorithm (as presented in Figure 11).

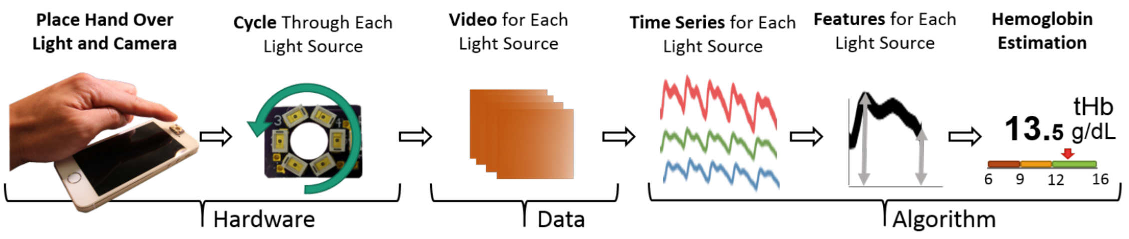

Place Hand Over Light and Camera
Cycle Through Each Light Source
Video for Each Light Source
Time Series for Each Light Source
Features for Each Light Source
Hemoglobin Estimation
tHb
13.5 g/dL
6 9 12 16
Hardware
Data
Algorithm

FIGURE 11. Obtaining hemoglobin levels in the HemaApp application (20)

While these are just a few examples of creating medical equipment from the smartphones, these examples show a few aspects that are making such solutions very promising.

First of all, the smartphones may offer a truly extendable computing platform for mobile medical equipment in much the same manner as portable computers in the 1990s. The hardware capabilities of the smartphones may be extended through the standard audio jack or micro-USB interfaces, which could supply add-on hardware modules with electronic data communications and power supply.

Secondly, the creation and production of medical add-on modules for the smartphones could be a much cheaper option than the creation of full-scaled, but similar medical tools. Moreover, the medical add-ons for smartphones could be produced faster and in much larger quantities than their typical full-scaled analogues. That could be very helpful when medical equipment must be quickly produced and delivered to areas of natural or environmental disasters, especially due to war conflicts or epidemics.

Thirdly, the smartphone’s system software is quite often based on open-source components and runtime environments, for example GNU/Linux and Android. These software environments could be easily extended by custom applications written in high-level programming languages, such as Java and Python.

By accessing the Internet and communicating over standard Wi-Fi, Bluetooth or cellular network interfaces, such custom applications could be relatively easy integrated with other medical systems and infrastructures, be it a hospital information system or some worldwide health-care organization’s data systems.

Additionally, the smartphone-based medical equipment may offer a reasonable option for diagnostics and preventive health-care, especially in the developing countries or at some disaster location. Moreover, such solutions would not require any expensive or proprietary infrastructure around them, thus their integration into local infrastructures will be much simpler and cheaper.

In addition to that, most of the potential end-users of such smartphone-based medical equipment are quite familiar with smartphones and the concepts of mobile applications. This can also significantly simplify and shorten the training requirements for medical personnel and speed up the utilization of such medical equipment when it is necessary.

All these above mentioned aspects turn mobile technology into a quite promising tool and platform in health-care and well-being domains, especially in attempts to bring much needed health-care services closer to patients in developing countries or challenging environments.

1.3 From open modular software to open modular hardware

Quite interestingly the ideas behind openness and modularity in the software have been recently applied to the hardware systems as well, including the smartphones. For example, one of such ideas was that the mobile phone should be made of interchangeable and replaceable hardware modules which could be easily added or replaced in the phone for obtaining some new or improved functionality without replacing the whole device.

The idea behind the mobile hardware modularity is not quite new and it may be dated back to the year 1999 when Palm Inc. released Springboard Expansion Slot for its PDAs for extending the default PDA functionality with various hardware modules (21).

Quite soon similar ideas were also applied to the smartphones, like in case of Phonebloks (22), that originally tried to enable the upgrades of mobile hardware for decreasing amounts of electronic waste in the world.

Some mobile vendors, such as Google, have taken these ideas even further and planned to create and standardize a hardware and software platform for producing truly modular mobile devices (23).

The idea behind Project Ara is to create a highly modular hardware and software platform that enables to create, extend and modify mobile devices to any particular need or use-case in a dynamical and highly customized manner.

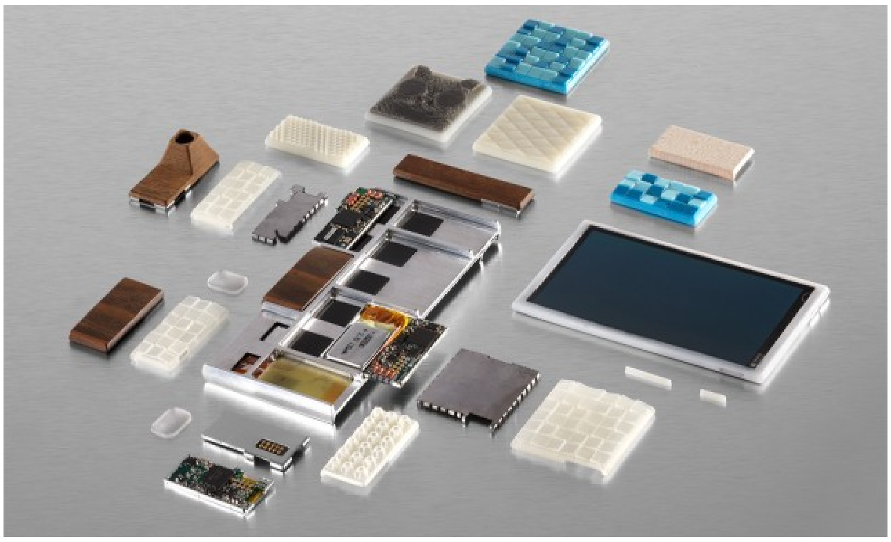


FIGURE 12. An example of Project Ara's endoskeleton with various modules (23)

The Project Ara hardware platform consists of replaceable hardware blocks – the modules – that could be put in a common frame – the endoskeleton, presented in Figure 12. Combined together, the modules start to co-work with each other through a special communication protocol, thus building a fully functional mobile device, as presented in Figure 13.

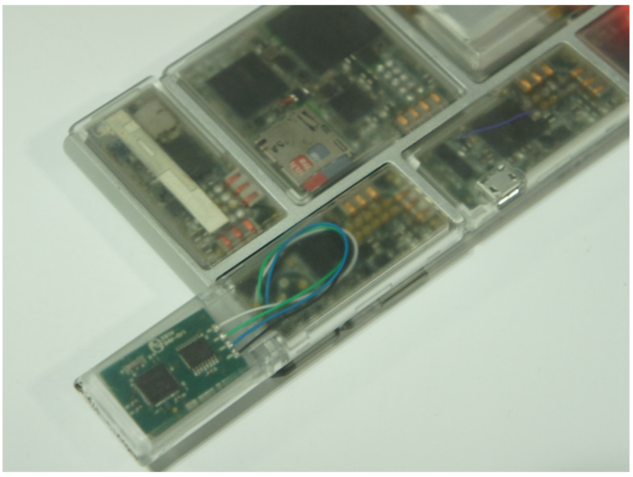


FIGURE 13. A functional prototype of Project Ara smartphone (23)

Such approach indeed enables to create highly customized mobile devices, and since the Project Ara's modules are supposed to be fully functional and replaceable hardware blocks with all the necessary software on-board, it would also be possible to create a hardware module for reading some medical parameters from the end-user of the phone.

For example, the Project Ara's mobile phone could be turned into a pulse oximeter if it will be equipped with the proper hardware module, as presented in Figure 14.

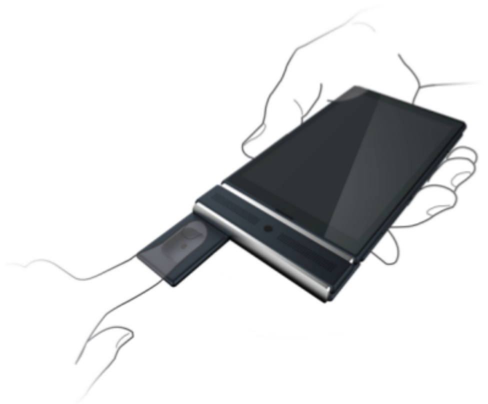


FIGURE 14. An example of Project Ara device with the pulse oximeter module (23)

With the data processing and communication modules on-board, such mobile devices may represent a similar, but more adaptable and even cheaper version of the smartphone extensions that we have reviewed in the previous chapter.

Moreover, customizing mobile devices through various medical hardware modules could enable the creation of highly dynamic and adjustable mobile medical devices that could provide health-care services and diagnostics more economically and exactly as it was demanded.

For example, a patient may be tracked for blood oxygen, heart rates and other parameters instantly through such modular mobile device without the need to stay in hospital all the time. Or for example, various disease diagnostic modules could be freely distributed in the location of disaster for some preventive actions.

With the medical diagnostics obtained from artificial intelligence systems over the Internet or through a health organization's data systems, such modular medical hardware may indeed bring health-care services much closer to patients than it was possible before.

In its turn, the medical data obtained through such modular solutions could provide health-care organizations and companies with an instant view of the situation in the region or location, thus helping to respond to the health-care challenges more adequately, much faster and efficiently than before.

In addition to that, such modular solutions would enable to create truly personalized, well monitored and more affordable treatment plans for the patients. Medical staff would also obtain a much better insight of the patients and of the efficiency of the applied medicine.

1.4 The network of smart things

Probably one of the most interesting aspects of current information systems is in their accelerated integrity and interoperability with other information systems over the Internet in so-called "cloud-based" and service-oriented architectures.

The "cloud" in fact is nothing more than a buzzword for various computing facilities accessible from the network, be it just a faster and more powerful computer, a larger storage for electronic data or a software environment suitable for one's needs. In the simplest way, the "cloud" may be described as a preconfigured and fully functional computing environment accessible over the Internet (24).

The service-oriented architecture (SOA) is simply a way to create a software system which is offering its functionality in the form of the services – distinguished and self-contained business and data processing activities – rather than in the form of a single monolithic application (25).

The SOA-based approach allows the decoupling and encapsulation of an ad-hoc functionality inside self-contained software components, i.e. subdividing complex problems and knowledge domains into much simpler and manageable software modules.

The main benefit of SOA-based architectures and systems is the possibility to combine, intermix and reuse their services in various ways, thus enabling the creation of much more sophisticated data processing applications from a much simpler computing and functional base.

From this point of view, SOA could be considered as a continuation of the Unix philosophy (26), but at much higher levels, where different systems may reuse services from each other for achieving some complicated goals in cooperation.

And just like in the Unix philosophy, where each of the programs was supposed to perform a single task (and perform it well) and be able to cooperate with the other programs through a universal text-based interface, in the SOA-based systems each service is supposed to provide some clearly defined functionality and be ready for cooperation with other services by messaging or through clearly defined interfaces.

Such loose coupling of services and systems presented in the SOA-based solutions has been already widely adapted by the mobile software technology, where mobile applications were typically backed up by the data processing services running somewhere else on remote systems.

In fact, such approach has simplified mobile applications and allowed increasing their scalability, flexibility and interoperability on the constantly changing mobile markets. Also, it has laid the path for smooth improvements in the functionality of mobile and back-end sides without large re-investments, maintenance costs or breaks in the operation.

Therefore, for example within the SOA approach, a mobile application could take care of GUI and the interactivity with the user, while the back-end services would take care of all tasks related to the mobile data analysis and information processing.

Such division of functionality between mobile applications and back-ends has also been recently adapted in the mobile health-care domain, where mobile devices were utilizing the same back-end APIs of medical expert systems as other medical systems.

One of the quite interesting examples of such solution is the mobile gadget that helps to tackle diabetic patients by using glucose monitors and insulin pumps with the data obtained from the IBM's Watson Health system (27).

In the Medtronic and IBM's project the intelligent back-end system was instantly studying the obtained medical parameters and adjusting patient’s mobile medical equipment according to the person’s lifestyle, habits and daily activities.

In this approach, much like in the system consisting of inter-networked intelligent components, each company and system could concentrate on the tasks that they do better, obtaining together much better results than they would apart.

1.5 The network of services

While the back-ends may indeed offer quite impressive computing and data processing capabilities and simplify the creation of mobile applications, the development of mobile software applications still remains a quite challenging and time and budget consuming process (28).

Additionally, due to the fragmentation in mobile ecosystems, mobile developers could meet even more challenges than developers of classical desktop or web-based applications.

Developing complex and monolithic mobile software applications in such circumstances is not just the best possible way to enrich the mobile device functionality, since developers will have to predict all possible use-cases and supply well-implemented and tested software beforehand.

Apart from the development challenges, such approach is also raising concerns about application's maintenance, support and portability issues across multiple mobile platforms and vendors, which indeed could simply drain development resources without much of benefits or returns in investments.

From this point of view, moving the complex data processing and functionality from a mobile device to the networked services can bring some critical advantages over a classical monolithic software approach.

For example, such separation of functionality may significantly simplify developments and testing of a mobile application and speed up its delivery to the end-users. As an additional benefit, keeping data processing services inside the SOA-based system allows the company or developer to reuse such services in other applications or business activities, thus decreasing total development costs and ensuring the system interoperability.

And with the utilization of cloud-based computing facilities, such service-oriented systems could be deployed in a much scalable manner, meeting up live demands and workloads from the real end-users.

By utilizing services and applications over a network, mobile devices could be integrated into various intelligent environments as interactive remote controls, like for example in case of home intelligent environments (29), cars infotainment systems (30) or just as a remote control for a person's electronic devices (31).

But today's smartphones have become quite powerful computing platforms which could work as service providers, too. For example, the BOINC mobile application (32) can dedicate some of mobile phone's computing resources to medical research and investigating such diseases as AIDS, Zika, tuberculosis and many others (33).

And while such smartphone-based computing nodes are used in addition to classical computing clusters based on PCs and supercomputers, it is very interesting to see that today's mobile platforms may offer enough computing power to medical research and computationally intensive tasks, such as defining the user's emotional state (34).

From this perspective, we can just see that it becomes possible to create heterogeneous systems that may utilize a mix of data processing and computing services, available both from the back-ends and from mobile devices.

Such heterogeneous systems, based on powerful mobile devices and networked back-end systems, allow the creation of highly customized, personalized and intelligent surroundings around mobile users which could bring ideas and concepts of the Internet of Things and ubiquitous computing much closer to reality (35, 36).

All this could lead to the creation of smart and dynamic medical environments at hospitals and patient's home, bringing health-care and medical data processing services virtually anywhere where it will be needed.

1.6 The Safety Navigation system

The Safety Navigation system is aimed at producing a tool for monitoring, nursing and helping people suffering from dementia and other mental diseases, such as disseminated sclerosis or light forms of amnesia.

In normal circumstances such patients are not required to stay at hospitals or nursing houses all the time, therefore they may keep just their usual lifestyle, being visited by nurses or supervisors on a timed basis for medical checks and some small talks.

A person-to-person communications is quite important for helping and curing such patients (37), but if the local nursing services are limited or somehow restricted, such patients are simply getting phone calls from the nurses or supervisors and thus being checked in that way.

Such remote calls could probably be considered as the simplest solution to the problem of lacking nursing services, but it stops working when the patient is not answering the phone or ignores the calls from supervisors.

An additional problem may come after the person’s amnesia strike, occurred mental handicaps or wandering. In such cases the patient may simply get lost somewhere without any note or simply forget where to they were actually going.

If the patient does not answer the phone or is not able to describe their current location, nurses and supervisors will have to search that person in the streets, either by guessing their current location or by invoking the police.

The Safety Navigation concept, originally outlined in the Ryhti project, is aimed at addressing these problems by allowing such patients to keep their usual lifestyle, while at the same time enabling instant multi-channel communications with their supervisors or nurses whenever and wherever it will be needed.

Technically speaking, the whole idea is to enable and provide such communicating, supervising and nursing services through Android-based mobile devices, such as smartphones and wearables.

In the Safety Navigation system any patient would be able to get in touch with a personal supervisor through a special mobile application, just when they might need it, and thus become less dependent on the physical presence of nursing staff in the place.

At the same time such mobile application could track the person's location and possibly measure some other medical and environmental parameters, such as movements, heart beat rates, weather conditions, for their further analysis in the intelligent back-end system and thus preventing accidents.

Such solution could automate at least some part of nursing and supervising work and it would supply patients with an accident prevention system, while real nurses and medical staff would just get a better view of patients and their activities.

The Safety Navigation systems should consist of a dynamic set of self-contained mobile and back-end services, each responsible for its own clearly defined task.

The mobile components of the Safety Navigation system are combined into a single software library called the Mobile Terminal Framework Software, or MTFS. The MTFS components should enable a further research and study of mobile health-care and well-being services at Oulu University of Applied Sciences, possibly in combination with other already developed systems.

As an additional requirement, the mobile Safety Navigation application should be portable to Android-based wearable devices, since wearables may have some benefits over smartphones, for example their availability to the end-users almost in all situations, and their possibility to deliver critical notifications virtually any moment when it will be necessary.

1.7 Solutions similar to Safety Navigation system

The tracking and supervising solutions, similar to the Safety Navigation system, are not actually something new in the world, and there are various applications, gadgets, appliances and systems developed for similar purposes.

For example, the King Pigeon Communication Co. Ltd. produces various elderly care products which the company describes as GSM “tele-care” and “tele-health” helpers. Some of such products might be described as interactive and automatic terminals that could track a person's environment and provide an easy tool for the communication with relatives over the GSM network (38).

One of such products is the T3 Senior Telecare System, presented in Figure 15.

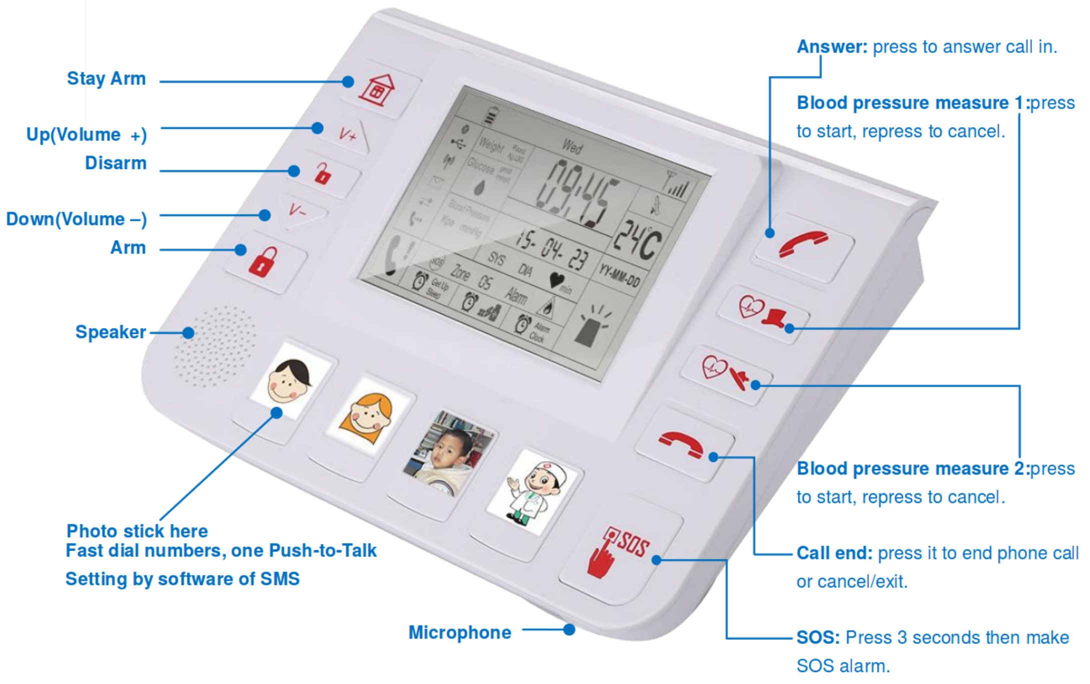

Stay Arm
Up(Volume +)
Disarm
Down(Volume –)
Arm
Speaker
Photo stick here
Fast dial numbers, one Push-to-Talk
Setting by software of SMS
Microphone
Answer: press to answer call in.
Blood pressure measure 1:press to start, repress to cancel.
Blood pressure measure 2:press to start, repress to cancel.
Call end: press it to end phone call or cancel/exit.
SOS: Press 3 seconds then make SOS alarm.

FIGURE 15. Functional description of the T3 Senior Telecare System (38)

The T3 Telecare System may also be supplied with accessories for detecting fire, smoke, gas or water leakages, door contact sensors or a wearable SOS-button, presented in Figure 16.

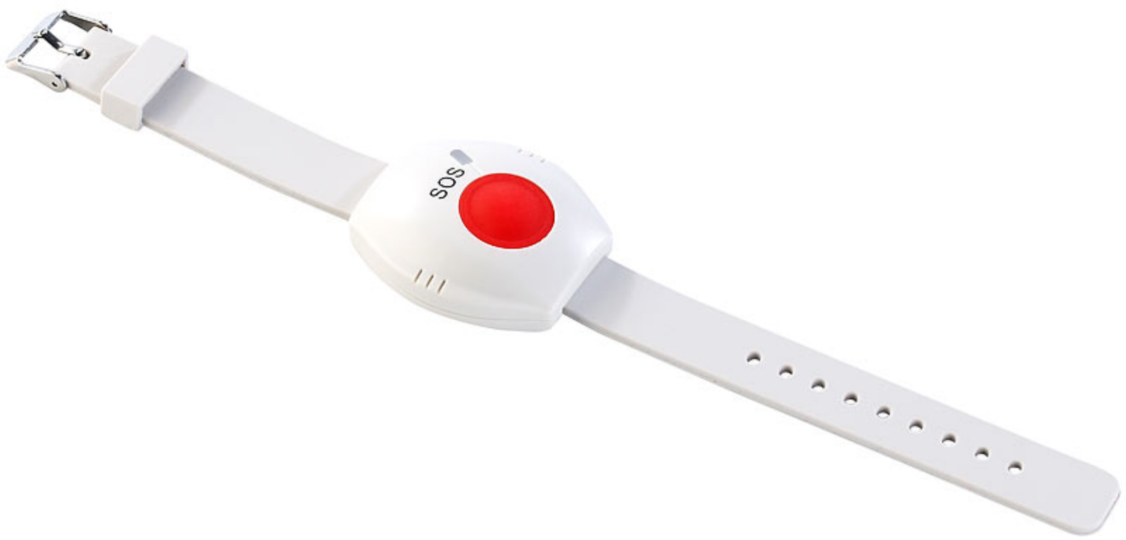
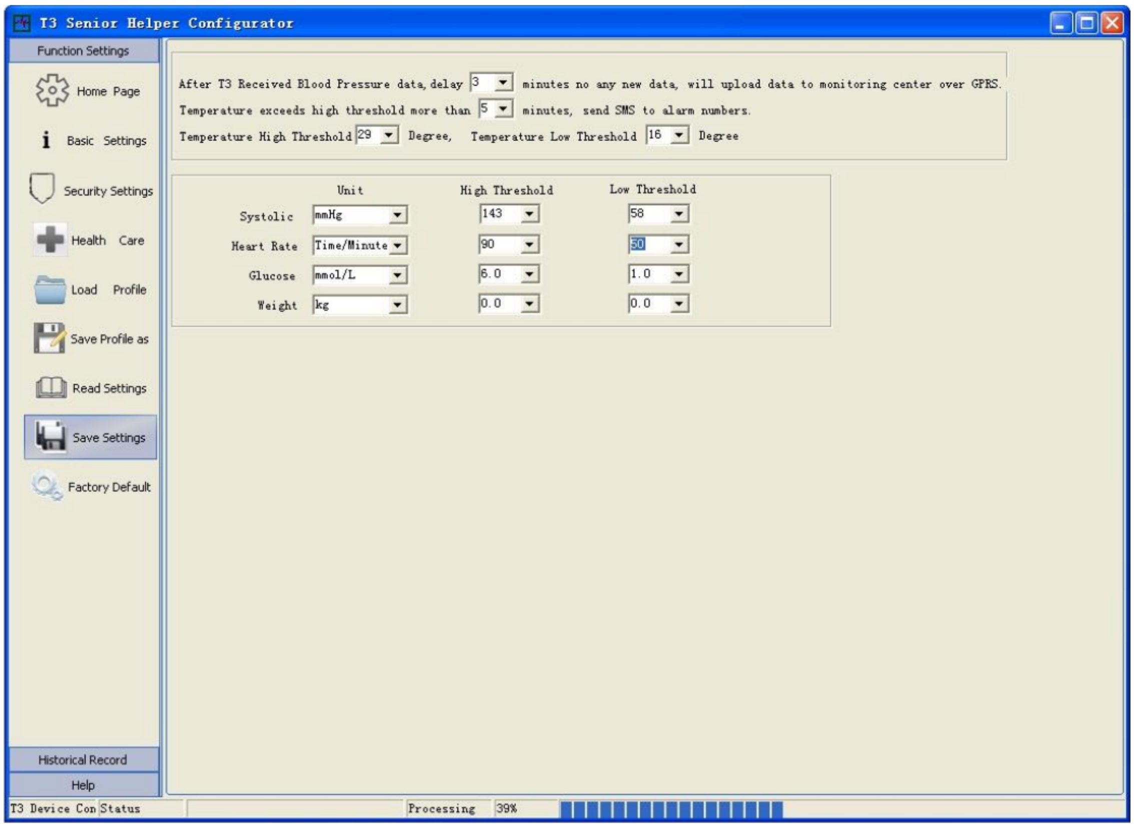


FIGURE 16. King Pigeon's EM-70 wearable SOS-button (39)

The wearable SOS-button works as an appliance to the T3 Telecare System and communicates with it over a wireless interface. The working range for such button is from 30 to 100 meters, thus a person could utilize it only near the home (39).

With the mentioned equipment from King Pigeon Ltd. a person could get a safer living environment with the possibility to get in touch with relatives or supervisors more easily when it is needed, and to get better chances of surviving in accidents or at disease attacks.

Additionally, the T3 Telecare System could be configured and administrated through a special application on a local PC, thus enabling the customization and tuning of the Telecare device. For example, a supervisor may set phone numbers and emails for notifications regarding blood pressure and glucose tests, as well as acceptable parameter ranges for such tests, as presented in the Figure 17.



After T3 Received Blood Pressure data, delay 3 minutes no any new data, will upload data to monitoring center over GPRS.
Temperature exceeds high threshold more than 5 minutes, send SMS to alarm numbers.
Temperature High Threshold 29 Degree, Temperature Low Threshold 16 Degree

	Unit	High Threshold	Low Threshold
Systolic	mmHg	143	58
Heart Rate	Time/Minute	90	50
Glucose	mmol/L	6.0	1.0
Weight	kg	0.0	0.0

T3 Device Con Status Processing 39%

FIGURE 17. *King Pigeon T3 configuration application (38)*

But despite all benefits that such solution is bringing, it comes at quite a price, and for example a simplified tele-care product from King Pigeon Ltd. could cost more than 200 euros in the Finnish Internet shop Verkkokauppa (40).

As an additional limitation, such solution is offered as a closed product, thus limiting its integration and adaptation into other health-care or well-being systems and making monitoring of wider user groups quite challenging and expensive.

Opposite to such products, numerous mobile and smartphone-based health-care, well-being and health tracking solutions were created (7), including the iWander mobile application from the Florida State University (41).

The iWander mobile application allows supervisors and relatives to monitor and nurse dementia patients through Android-based smartphones in a cost effective way. The various data collected from the mobile device is evaluated using the Bayesian network, thus estimating the probability of wandering behavior.

Up on evaluation results, based on the user's profile, the current situation or probability of accident, the supervisors may take various actions. For example, the audible prompts to the patient could be issued, offering directions to navigate the patient home; or the patient's location could be detected and a phone line could be established between the patient and the supervisor; or a party call may be performed, invoking help both from supervisors and from local emergency services.

Quite noticeable is that the iWander solution was targeting at and relying only on commonly available Android smartphones, making such solution much more affordable by a wider range of people. The data about mobile users' activities and actions was used to teach the system and to improve its performances when considering the real-life situations and patient behaviors.

Developing the Android-based smartphones has also allowed Sposaro's team to port the iWander application to wearable devices, thus combining both the SOS-button functionality and the real-time patient tracking.

The Safety Navigation system shares some similarities with the iWander solution, but its primary goal is to provide a common framework for creating other mobile health-care and well-being services in a simple and modular way. Therefore, the iWander application could be considered as just one example of the mobile applications that Safety Navigation should make possible.

1.8 Privacy and awareness

As the correct diagnosis and treatments are heavily relied on the valid and sensitive personal data, thus the automatic gathering and processing of medical data brings some questions regarding medical ethics and confidentiality between cared-for patients and medical personnel involved in this.

In most cases there is an agreement between the patient and doctors about nondisclosure of any details related to that patient's health state or its dynamics without any clear allowance from the patient. However, with the use of automatic medical data gathering and processing systems, it is possible to unintentionally violate such nondisclosure agreements due to improper system configurations or security breaks.

Regarding the Safety Navigation system, it is planned to obtain a clear agreement from end-users or their relatives about what type of data is allowed to be gathered, processed and under which circumstances. These agreements should also be made available to the involved personnel, thus avoiding any potential misunderstandings or privacy violation claims.

Additionally, just like in any other situation that rises ethical questions, any possible confrontation between the mobile patient's privacy and the awareness of carrying personnel in the Safety Navigation system should be probably solved with just the same medical treatment principles that were proposed by the ancient Greek physician Hippocrates in the 5th century BC in his famous Oath, sometimes simplified to a phrase: “Don't harm more than it is truly necessary” (42).

Keeping such principles in mind, the carrying personnel should simply get as much information about their patients as it would be necessary from medical perspectives or the person's well-being, but without breaking anyone’s privacy or minimizing the risks of such breaks.

1.9 Primary aims for the MTFS framework

The major task settled for this Thesis work was the creation of a mobile toolkit – the MTFS framework – that would enable the creation of various well-being and health-care services and applications from the same basic functionality.

This requirement has settled a few primary aims for the MTFS, such as the modularity of software components, employment of SOA concepts, loose coupling between mobile and back-end services, interchangeability of service implementations and possibilities for an easy integration with the third-party data systems.

2 THE WORK ENVIRONMENT

One of the personal aims that the author settled for this project was developing all the required functionality exclusively with the open-source technologies, tools and applications. Such aim arose from the author's personal work experience where many successful commercial projects were performed and implemented in similar environments (43).

The commercially available tools and technologies were not particularly bad in the sense of provided features or functionality, but the open-sourced ones seemed to be more suitable for research and developing needs. Additionally, the open-sourced technologies are usually well accepted by academia and research communities, and they typically provide a better foundation for creating well integrated and solid products, which is a common goal for successful commercial projects, too.

For just the same reasons many successful commercial software solutions and products are fully or partially based on open-source technologies, for example the RedHat's portfolio (44) that is a mix of various tools and software components initially developed by the GNU/Linux, JBoss (45) and Apache Software Foundation communities (46).

However, it is worth to mention that an open-source technology is not by any mean a “silver bullet” solution to software creation issues, but rather a very fruitful foundation that could save time, budgets and development efforts, especially in the research and academy domains.

At the same time it is good to notice that such software foundation is coming with the price, and depending on the applied open-source licenses, the resulting source code and improvements might be obligated for sharing with the development communities and end-users (47).

Additionally, the openness of source code and technologies are not the guarantee for a successful product, functionality, documentation, performance and reliability of the final solution. While indeed, “giving enough eyes, all bugs are shallow" (48), there are still many other issues to be considered while picking up open-sourced solutions in favor of commercial ones especially in a sense of producing commercial products and tools.

For example, sometimes the technology simply cannot be made open due to legal, security or dual-use issues. This is quite often the case with firmware and drivers for the medical equipment, but such closed-source approach has its drawbacks and it simply cannot protect its end-users from design and security flaws in mission critical applications and devices (49).

The design and security flaws may also as well appear in the open-source projects and products, but due to openness and public availability, there are better chances that such defects will be fixed rather than hidden from their end-users.

In the scope of this work the author selected a number of technologies, tools and techniques that were quite popular and well accepted both by open-source development communities and in the commercial world. Such choices were made in a hope that the developed solution would be useful both for the research and for the needs of practical business.

2.1 Service-oriented approach

The core idea behind service-orientation is building complex systems out of self-constrained, loosely coupled and interactive functional modules – the services. Such approach is usually described as the SOA-based approach. However, it is worth to mention that SOA is not a particularly well defined specification for implementing systems but rather an architectural style in which systems could evolve (25).

The Safety Navigation system utilizes the SOA-based approach as a tool for managing the complexity and decoupling functionality of developed components.

The service-oriented approach is used on both sides of the Safety Navigation system: the mobile framework (MTFS) and in the back-ends.

The mobile application is designed as a bunch of independent services running on a mobile terminal and being responsible for specific tasks, such as e.g. tracking the end-user’s current geographic location, checking network and battery states, reading data from acceleration sensors and interacting with the user.

The back-end system is designed as well as a set of independent and self-constrained services, supplying the MTFS mobile application with all necessary data over the REST interfaces.

Such approach has enabled an incremental development and testing of the system, when the most critical services, such as the geographical location tracking, user authentication and basic end-user interactivity were developed first, while the rest of services were added later, thus enriching the system’s functionality incrementally.

As an additional benefit of the SOA-based approach, dividing the whole system into independent and loosely coupled services has simplified implementations and enabled many improvements in the system in an evolutionary and almost seamless manner.

2.2 Android platform

The Android platform was selected as a foundation of the mobile framework for a number of reasons. First of all, Android is the open-source OS that originally aimed at mobile phones and tables. Since its first public versions released in 2007, the Android has gained more than 80% of the smartphone market shares globally (50) and it has been expanded to TV, automotive infotainment and wearable devices.

Based on the Linux kernel, the Android platform offers developers a standard runtime and programming environment on various device classes and domains, such as smartphones, tablets, wearable devices, TVs, or car infotainment systems (51).

Secondly, the Android platform offers various Java, C and C++ programming interfaces, covering all aspects of creating and managing mobile applications.

The Android platform's functionality is divided across modules and provided through Android's standard API and system frameworks. One of the most important aspects of these APIs is their compatibility with the previous versions of the Android platform. Thus, developers might be sure that the functional changes in the new versions of the platform will not break the existing code (52).

Also, the Android platform offers a seamless integration with Google's Internet-services, such as the support for Google Maps, location detection, messaging between users, analysis, reporting (53).

And finally, all Android functionality comes with free and open-sourced SDKs and IDE tools, making the creation of Android applications quite an easy task (54).

The Android platform selected as a primary runtime for the MTFS framework was of Android’s release 4.4 (55).

2.3 Back-end environment

The back-end environment for the Safety Navigation system was also based on commonly available open-source tools. Due to its proven modularity, the back-end took the same approach as in the so-called LAMP software stack, consisting of various open-source components (56).

The basic idea was to create a runtime that anyone could reconstruct on a commodity hardware without much of investments.

In accordance to the LAMP approach the default hosting OS was selected from the GNU/Linux Debian family: namely, the Debian 8.6 and Ubuntu 14.04. The final solution, however, has not used any functionality exclusively specific to GNU/Linux OS and thus it could be as well hosted on other operating systems, e.g. Windows or Unix families.

The services forming a SOA system are usually running behind a properly configured HTTP proxy, such as Apache or Nginx. For the needs of this project the Nginx HTTP server was selected due to its easy reconfiguration, small system requirements. Also it provides load balancing and reverse-proxy functionality out of the box (57).

However, none of Nginx-specific functionality was added to the final solution, therefore it might be considered as a useful but not critical part of the back-end system.

2.4 Data repository

Like any other information processing system, the Safety Navigation system relies on its data repository. Due to its critical role, the data repository had to meet a few requirements as presented in the following list:

- Support for geospatial data types and operations on them
- Support for transactions in data modifications
- Support for hashing and encryption functions
- Support for role-based access to data entries
- A sufficient set of tools for administrating and debugging data repository

A need for a native support of geospatial data types quite naturally come from the concepts of Safety Navigation. While it is possible to represent a geographic location as a set of numbers (e.g. latitude and longitude), the database's support for geospatial data types could give some significant benefits, e.g. querying geographical locations on some criteria or modifying them according to some location-specific rules.

In addition to that, some other geospatial functionality could be useful to the project as well, e.g. measuring distances between geographical points or verifying coordinates against specified geographical fences and borders.

Since the data in the repository is quite sensitive in nature (such as personal geographic location, medical sensors readings, personal geofences), the repository should be able to restrict and authorize access to such data based on users roles and their responsibilities in the whole system.

For example, the repository should prohibit supervisors from changing and altering geographical locations or medical data obtained from mobile users, while at the same time relatives or carrying personnel should get all necessary details regarding the state and location of mobile users.

Since the Safety Navigation system aims at supporting multiple mobile users at the same time, the repository should support concurrent modifications of the data, avoiding any possible corruptions. Therefore, the data repository should support transactions.

And since such multiple-user data repository most probably will work as an independent software component on the remote computer, it should provide all necessary tools and facilities for the administration, management and data archiving.

Based on the mentioned criteria, a number of free and open-source database systems were reviewed for implementing this work. At the time of writing this thesis, the most popular among such database systems were the MySQL, MariaDB and PostgreSQL.

All these systems are well known and accepted in the research and business communities and they have also their benefits and drawbacks (58), but the author's personal experiences led to choosing PostgreSQL as the back-end's database system.

The main reasons of picking up PostgreSQL over other systems were in the PostgreSQL's conformance to the SQL standards and its functional extendability by add-on packages (59).

Additionally, PostgreSQL provides support for user roles and privileges on its core level thus ensuring the application-level security and strict access to its data. The roles could be altered and assigned to database users on the fly without disturbing the running components of the system (60).

PostgreSQL is an object-oriented RDBMS which supports custom data types in parallel to the SQL's standard ones. Comparing with MySQL, PostgreSQL has much striker checks against data validity and integrity. And due to optimizations of the single database engine used inside PostgreSQL, most of table joints, unions and indexes usually work much faster than in MySQL (61).

From the very beginning of its development, the PostgreSQL focused on the data integrity on the transaction level, which lead to a very optimal solution in terms of computing resources required for a table, row or column locking.

In addition to that, PostgreSQL supports various constraints related to table, table attribute or data uniqueness checks that are very useful in distributed and multi-user systems (62).

The basic functionality of PostgreSQL system could be enriched by installing additional functional packages. For the needs of the Safety Navigation framework, the PostgreSQL system was equipped with an open-source PostGIS spatial package that added support for the geographic functionality and objects in the PostgreSQL databases (63), and the open-source pgRouting package that added a geospatial routing functionality to the PostgreSQL database (64).

The PostGIS and pgRouting packages are the world-class projects that are used in such well known open-source project as OpenStreetMap (65). Additionally, these packages are following the Simple Features for the SQL specification from the Open Geospatial Consortium (OGS), making the data and functionality integrity on the level of the SQL code (66).

2.5 The middleware

The middleware is supposed to provide a sufficient runtime and programming environments for the back-end services. Moreover, it had also the task of gluing things together, e.g. providing decision making processes with the information from data repositories, as well as enabling the connectivity with the outside world, such as integration and system users.

With such priorities in mind, two technologies were primarily reviewed for implementing the middleware functionality: the Java-based Spring framework (67) on top of OpenJDK 8 (68), and the Golang programming language and the corresponding runtime from Google (69).

Since each of them could be used as a solid foundation for the SOA system, there were no bigger contradictions between them. Both provided enough of tools and functionality for implementing the REST services, and both were well known and accepted by developing communities at the time of writing this thesis.

However, after a few initial trials, the author decided to utilize exclusively the Spring framework. The reason for that was the quite dramatic time saving that the Spring framework offered to the project through its features and available modules.

Additionally, the utilization of Java programming language on both end-points of the system has significantly simplified the implementation of mobile and back-end services, since they could share common data models and libraries.

The Golang, at the same time, was an excellent experience and had all the necessary components for implementing the back-end system, but comparing with Java and the Spring framework it felt much like a lower-level technology which was aimed at solving the efficiency and manageability problems of some very large projects rather than simplifying the developments of a SOA solution.

2.6 Front-end technologies

Although the primary task of this thesis was in the development of the MTFS framework, it was still quite necessary to visualize obtained data and possibly edit mobile application configurations from the GUI.

For such developments the operator's GUI was implemented by utilizing HTML5 and the map visualizing JavaScript library Leaflet (70).

The web GUI was provided by utilizing the standard HTML template engine called FreeMarker, which was provided by the Spring framework out of the box (71).

However, the author should note that despite all easiness of today’s web developments, the original aim was just to visualize the obtained data rather than to create a fully scaled web GUI.

2.7 Cloud-oriented solution

The overall architecture of final solution was constantly reviewed for its suitability in the cloud computing environments, such as Amazon AWS and Google Cloud Platform.

From such perspective the only requirement for the MTFS mobile part was in the utilization of the HTTP and HTTPS protocols and avoiding any hard-coded end-point URLs. That was handled through the mobile application configurations.

On the back-end side configurations were handled through the Spring framework's environment profiles so that all involved components could be integrated through URIs.

3 DEFINITION

The primary aim of the Thesis work was to design and implement the MTFS framework as a set of reusable and self-sufficient mobile software components – the services – that in their turn would enable the creation of other mobile health-care and well-being applications for Android-based terminals.

The term “self-sufficiency” in this case means that each of such software service would provide a domain-specific set of functionality, for example a location detection, networking and sensors, while the “reusability” stands for the possibility to utilize these services in other applications without too much of refactoring or modification.

The software components developed for the MTFS framework should be tested in an example mobile application, called the Safety Navigation, that would provide its end-users with remote monitoring and support services.

The most critical facilities provided by the Safety Navigation application would be location detection and telemetry data services, therefore it should utilize the location detection capabilities of the mobile device, available sensors and networking.

The application should also provide its mobile users with a very simple graphical user interface (GUI), sufficient both for the elderly and for the young people.

The mobile Safety Navigation application should be able to work independently from the back-end services utilizing some embedded intelligence when a mobile device is offline or when back-end services are not responding.

3.1 The main use-cases in the system

The most important feature of the system is to let mobile users to contact their supervisors in case of accident or after getting lost. At the same time the supervisors should know exactly where the cared-for people are located and what is their condition.

This particularly implies that all required information about mobile users, such as their current locations and sensor data, should be tracked with a sufficient precision and delivered in some form to the supervisors.

Since there could be many mobile users in the system, possibly with unique behaviors and habits, the MTFS framework should be able to identify each user and supply them with personal settings and configurations to enable a better tracking and supporting.

All together, it should form a robust and dynamic system where many users could be served over the network by the supervising professional in a seamless, but precise manner (see Figure 18).

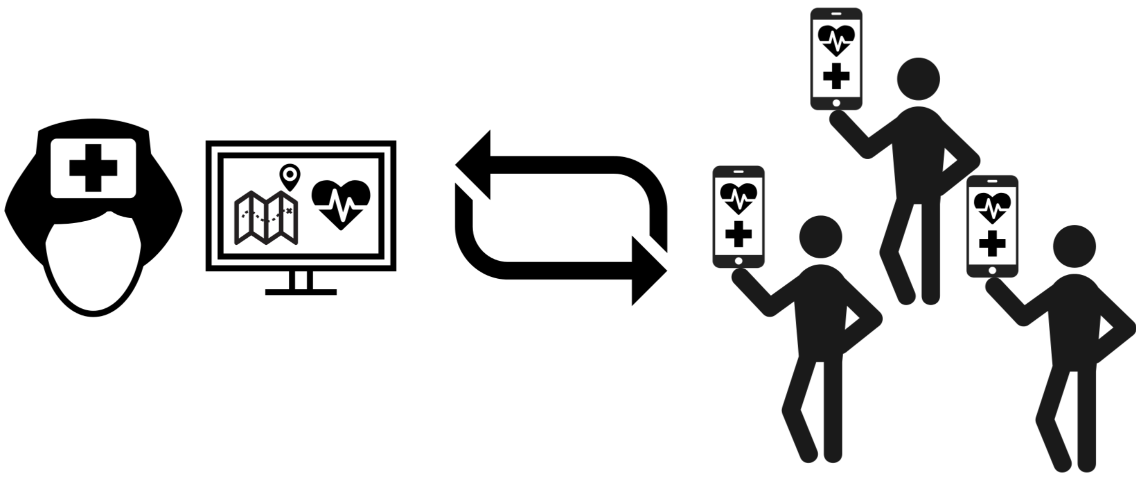


FIGURE 18. Schematic presentation of the core idea behind the whole MTFS system

The MTFS system should allow its end-users to interact with their supervisors when a dangerous situation will be detected. This could be done either by initiating an automatic phone call with the supervisors or by some other means, e.g. through utilizing multimedia and communication applications presented on the mobile device.

The MTFS system should be able to detect when the end-user is leaving or entering some predefined geographical locations or areas. Also, the so-called geofencing functionality should be provided by the system.

Moreover, since possible end-users of the MTFS system could be actually suffering from memory and dementia diseases, the mobile MTFS application should warn the users up on the detected leave of such geofenced location, as schematically presented in Figure 19.

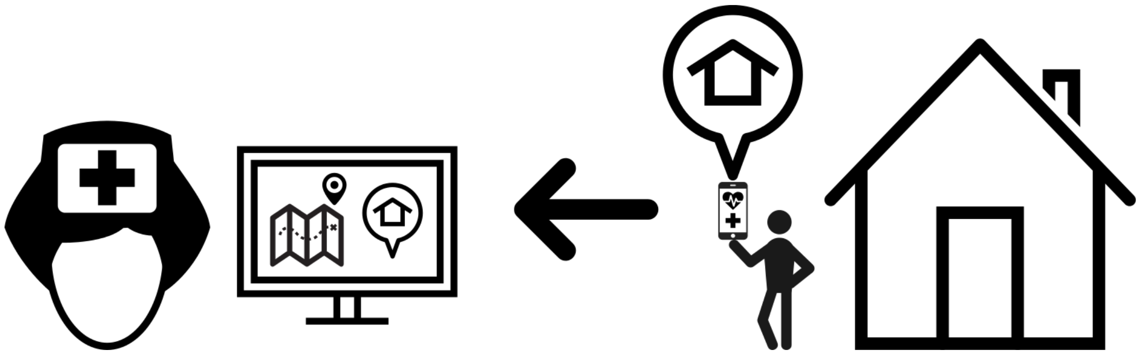


FIGURE 19. Supporting end-user's and supervisor's awareness in the MTFS system

At the moment of leaving some preconfigured safe location, a warning message should be presented to the end-user, possibly asking some questions regarding the situation. The warning message could also be made interactive to confirm the end-user's consciousness and awareness of the situation. This could be done either by presenting a special confirmation dialog or by automatically initiating a direct phone call with the supervisor.

Regardless to the end-user's responses, the up-to-date information about the user's location and activities should be automatically delivered to the supervisors for a further review and confirmation. In this case the carrying personnel could decide what to do about the situation if some anomalies in the end-user's behaviors will be detected.

In addition to location tracking, the MTFS mobile application should supply the supervisors with all necessary information about the end-user's environment and surroundings, taken through pictures from the mobile camera or obtained by recording audio samples or by reading data from available sensors, as presented in Figure 20.

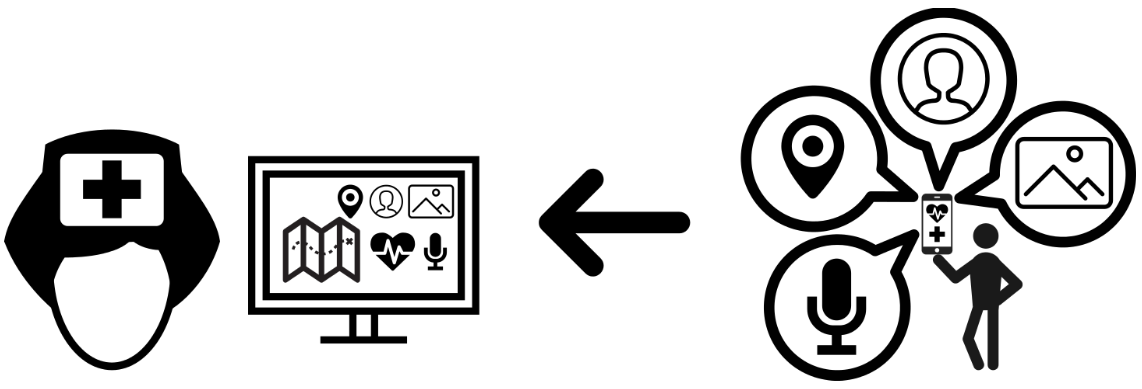


FIGURE 20. Automatic extraction of information about end-user’s environment at accidents

These features should be used with great care and only be activated when the user will be in some dangerous situation or if an accident is detected.

The automatic extraction of information about the end-user’s environment could be activated either by obtaining permissions from the back-ends or independently, after detecting some events or anomalies in the user’s behavior.

Such dynamic functionality will require that all involved MTFS components will support the reconfiguration on the fly, either by receiving customized configurations from the network or from the mobile application’s internal service orchestrating component.

Since the connectivity of a mobile device is always challenged by environments and network covering issues, the MTFS components should be able to work independently from the back-end services during offline periods, still providing the most critical services to the end-user, such as geofencing and accident detection functionality.

The mobile Safety Navigation application may utilize multiple components and sub-systems presented on the mobile device, but it should be made sustainable against software and hardware failures, thus preserving its functionality over the time.

This also implies that MTFS components should utilize available mobile components quite wisely, for example, avoiding quick drains of the device battery or minimizing the communication expenses in cellular networks.

The resulting system should be both modular and flexible, thus enabling its future improvements and reuse in other systems and applications.

3.2 End-users

The primary end-users of the MTFS mobile application are elderly and younger people who are suffering from memory or dementia diseases or having some wandering behaviors. Such people may live quite independently for most of the time, but they could need some help when their disease strikes back.

The other user group is formed by the actual supervisors and caregivers, who are simply interested in receiving up to date information about cared-for people, such as mentioned before, and make sure that they are just fine. This group of users may be called the “operators”, and it may also include professionals who are offering medical, guidance or pick-up services to cared-for people on a commercial or volunteering basis.

The third group consists of technical supervisors of the system, who are performing various technical tasks and making sure that the system is running as supposed.

3.3 Mobile Safety Navigation application

The mobile Safety Navigation application should run on commonly available Android smartphones. The mobile end-user should keep such smartphone on a chest, with the main camera pointed outwards.

The mobile end-user should be able to pick up that smartphone at any time, and use it for receiving phone calls or getting help and guidance from the Safety Navigation system.

The Safety Navigation application should rationally utilize all required features and resources of the smartphone, such as its sensors, networking capabilities and battery, it should also reconfigure mobile services according to the device state and its capabilities.

In case of a hardware or software failure, e.g. a battery drain or an insufficient memory on the device, the mobile Safety Navigation application should notify supervisors about the situation and supply a back-end system with all necessary information about the person's current location.

The mobile Safety Navigation application should provide and perform the following functionality:

- Authenticate the mobile user to the Safety Navigation system
- Keep track on the user's location and activities by utilizing sensors and location detection services of the mobile device
- Keep track of the mobile terminal's state, such as its battery and network status, and possibly other parameters for a better awareness of supervisors and involved personnel
- Alarm the mobile user about any dangerous situation, e.g. leaving safe locations or entering dangerous areas
- Read and apply configurations, guidance and updates from the back-end system for adjusting performance and changing working modes of the mobile software
- Guide the mobile end-user back to a safe location or home by utilizing the GUI and multimedia capabilities of the mobile device, including a direct phone call with the supervisors

The overall architecture of the mobile Safety Navigation application should be modular and support extending the functionality without a complex refactoring. The developed solution should support portability to other Android-based terminals, such as wearables and infotainment devices.

3.4 Back-end system

The back-end system works primarily as a data processing and decision making system, offering various services for the mobile Safety Navigation application.

The most critical services that the back-end is offering to mobile end-users are geofencing functionality, mobile user tracking and supplying the mobile Safety Navigation application with updated configurations.

In addition to that the back-end system should provide the following functionality:

- Enable a reliable alarming of mobile end-users and their supervisors in case of accident
- Provide support for user-specific geofences and personalized configurations for mobile services
- Enable the tracking of end-user activities and habits for preventing possible accidents
- Provide a data repository for a further information processing, analysis and study
- Provide authentication and authorization services for mobile end-users and supervisors
- Secure the data communication between mobile terminals and back-end services

In addition to the above mentioned objectives the back-end system should keep all its functionality in the form of self-constrained and loosely coupled services and enable an easy integration of the third party tools and components for extending its functionality.

4 IMPLEMENTATION

The MTFS framework and corresponding back-end system were implemented with a number of open source technologies and tools that at the time of thesis writing were well accepted by developing communities and software developers.

The primary aim behind all developments was to create a reliable and flexible architecture for the MTFS framework and Safety Navigation system that would be easy to use, maintain and improve over the time.

4.1 Main components of the system

The whole system is subdivided into mobile components, running on Android terminals, and the back-end components, executing on a dedicated application server. The mobile part of the solution is formed by the MTFS framework, executing inside the mobile Safety Navigation application. The back-end components are running independently from the mobile part and they are offered as a bunch of loosely coupled RESTfull services. Both above mentioned parties are interacting with each other by invoking HTTP methods and exchanging JSON data messages, as presented in the Figure 21.

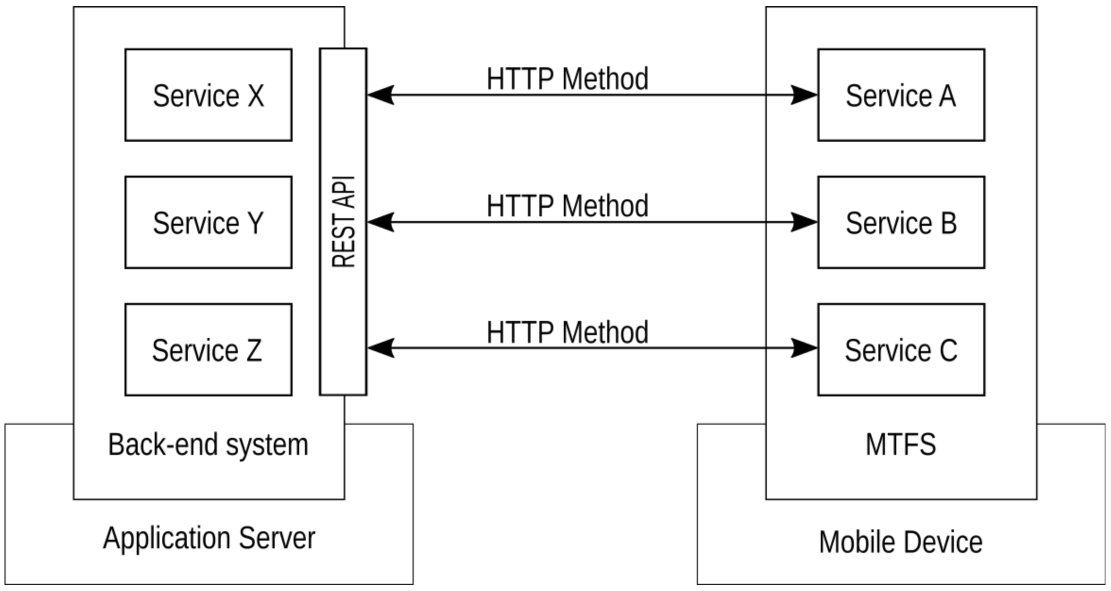


FIGURE 21. Overall architecture of the Safety Navigation system

In the RESTfull approach (72) the services are talking to each other by invoking the corresponding HTTP methods over application specific URIs. The actual logic behind HTTP request processing is hidden from the caller, thus there could be many services and sub-systems involved into handling a single RESTfull API call, just like in any other remote procedure call system.

The MTFS framework is composed of various mobile services running on an Android device and performing specific tasks, such as detecting the end-user's current location, checking geofences, reading sensors, keeping track of mobile device resources, and communicating with back-ends.

The mobile Safety Navigation application is using the MTFS framework as a library, it performs the orchestration, tuning and control of used MTFS services through its Main Service, as presented in the Figure 22.

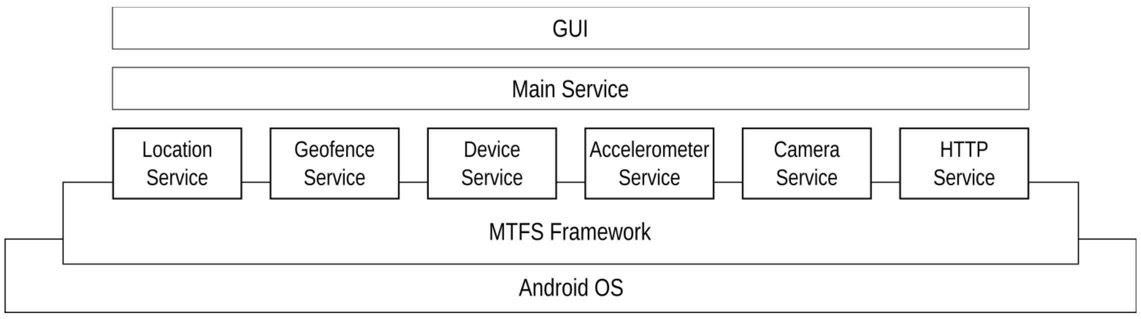


FIGURE 22. The overall structure of MTFS framework and Safety Navigation application

The Main Service is part of the mobile Safety Navigation application and it is independent from the MTFS implementations.

All MTFS services are producing, exchanging and consuming information through messages, implemented as special Java objects. The Java objects may represent a distinguished event in the system or a change in the application's activity, for example a new sample from the accelerometer sensor, a new detected location of a mobile user, or just an HTTP call performed by the HTTP Service.

When MTFS service detects a new event or change in its environment, it creates a new Java message object and broadcasts it across the whole system through Android's information bus, called the intent system (73).

In Android's intent system any software component, subscribed to some specific type of the MTFS messaging object will receive a copy of that object immediately after it was generated and broadcasted by the corresponding MTFS service.

This simple, yet powerful mechanism is utilized by the MTFS framework for decoupling mobile services and components from each other in a quite elegant way, allowing to extend the MTFS functionality without breaking the existing services and components.

Moreover, the MTFS service configurations are also delivered to the services by Android's intent system inside Java messaging objects. That turns the service reconfiguration into a simple process of receiving configuration objects from the intent system and applying received configuration parameters to the service functionality.

The data messages between mobile and back-end services are transferred in the form of JSON objects. These JSON objects are just simple text representations of the mentioned Java objects generated by the MTFS services. The JSON format for a message representation was chosen for its simplicity, easy extendability and quite outstanding support in the world of web-based data systems.

While most of gathered data is transferred to the back-end services over network, some data is first processed locally.

For example, signals from accelerometer sensors are first compared inside the mobile application against preconfigured threshold values, thus helping to determine the person's activities and tune the mobile application's functionality accordingly.

Another example of such local data pre-processing is in the Geofence Service functionality, where the end-user's locations are studied first to determine if the user has left some preconfigured geofences.

Such preliminary local data processing is helping to solve two major problems of the MTFS framework: at first, it helps to detect dangerous situations much faster than if it would be done through the back-end systems; and secondly, it helps to protect end-users while the mobile device is offline or when back-ends are not responding.

4.2 MTFS implementation

The MTFS framework was implemented in the Java 8 programming language on top of the standard Android 4.4 APIs. The framework's functionality was subdivided into independent packages, each representing a number of related services together with their supplementary objects.

Each implemented service played a distinguished role in the system and supplied back-ends with its own kind of information. All implemented services were subdivided into the following functional categories:

- Device monitoring – the software components in this group are notifying the Safety Navigation system about all events related to device resources and its capabilities, e.g. changes in the network connectivity, battery levels, mobile storage capacity, or the end-user's presence to the mobile system
- Sensing – the services in this group are reading data from available mobile sensors and obtaining information about the user’s activities and their environment
- Location – the components in this group are detecting the end-user's current location and interacting with Android’s geopositional subsystems
- Networking – the services in this group are communicating with remote systems over the HTTP or TCP/IP based protocols
- Geofencing – the services in this group are providing the geofencing functionality and keeping track of the user’s presence at geofence locations

- Utilities – the components in this group simplify and automate some MTFS-specific tasks, such as creating Android intents or putting data into the HTTP methods

All the components in the above mentioned functional groups were developed to work independently from each other, so that MTFS services and related components could be easily picked up and reused in other mobile applications without any heavy refactoring.

4.3 MTFS service architecture

All developed MTFS services share some similarities in their architecture and internal structure. Such similarities in the service structure were figured out after some reviews and refactory, based on testing in real-life circumstances.

Originally many of MTFS services were based on Android's *android.app.IntentService* class. However, although that simplified service implementations and automated service activation up on some events in the system, the performances and reliability of such services were quite far from ideal.

For example, the HTTP method objects for network communications could be buffered by the Android system at some point without a notice, therefore it was hard to predict when the HTTP method will be executed and how long it will remain in the buffering queue.

Additionally, intensive communications and object exchanges between such IntentService-based components were getting into similar troubles. Therefore, for example the events detected by the accelerometer sensor could not be delivered to and handled on time by other services.

All this lead to a re-implementation of all MTFS services on top of Android's standard *android.app.Service* class. In this case the MTFS services took more responsibilities for their life-cycle and resource management, but that was basically the only requirement for their utilization.

The Android’s standard *android.app.Service* class provided enough of basic functionality for the MTFS framework services, thus the author could concentrate more on their MTFS-specific functionality.

Quite right from the beginning of developments, it came clear that probably the most beneficial aspect of developing services and applications in the Android environment was the rich infrastructure that Android APIs were offering for implementing inter-process and inter-service communications.

The utilization of Android's intents as the base for the MTFS framework's inter-service communications significantly simplified the implementations and logic of the services and helped to spend more time on the domain-specific developments, rather than on the environment in which mobile MTFS services were supposed to be running.

A schematic description of the Android intent-based communications between MTFS services is presented in Figure 23 below.

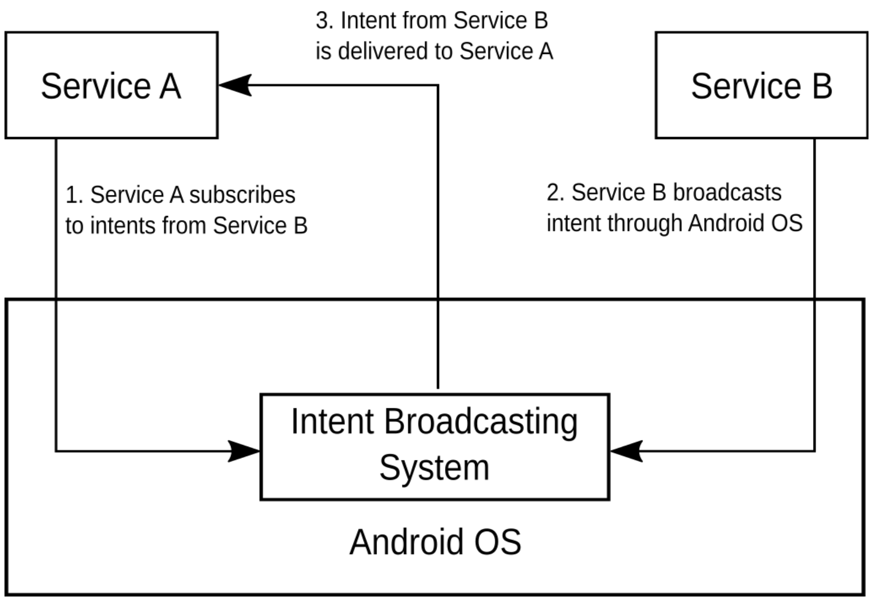

3. Intent from Service B is delivered to Service A
Service A
Service B
1. Service A subscribes to intents from Service B
2. Service B broadcasts intent through Android OS
Intent Broadcasting System
Android OS

FIGURE 23. Utilization of Android's intent broadcasting system in the MTFS framework

As you can see in Figure 23, the Android's intent broadcasting system takes care of all technical aspects of the communication between the processes and services, leaving developers only to decide what type of objects should be exchanged between the services.

The working principles behind Android's intent broadcasting system are similar to the ones used in the publish-and-subscribe system.

In the example presented in Figure 23 Service A subscribes to objects from the Service B by issuing a special request to Android's intent broadcasting system. Such request contains a description of objects that Service A is willing to receive, without actually telling which component should produce it, and a link to a callback function which will be invoked up on delivery of requested objects.

When Service B will send its objects through Android's intent broadcasting system, the broadcasting system will automatically notify Service A through a specified callback function, supplied with a copy of object generated by Service B.

Such simple yet powerful publish-and-subscribe system helps to decouple almost all services and software components in the Android system. Probably the best aspect of such approach is that Service A and Service B do not have to be aware of each other’s implementation, internal activities or even presence in the system. The only thing that a consuming component needs to be aware of is simply the data models that a producing component is generating.

As an additional benefit, the Android intent broadcasting system may transfer custom Java objects of virtually any complexity, thus allowing an adaptation of service data models for any particular need.

For these reasons all data objects generated by the MTFS services were made compatible with the Android intent broadcasting system. This virtually enabled the utilization of MTFS functionality in any other Android-based environment, be it a mobile phone, wearable or some embedded system.

Quite naturally after the MTFS data objects were made exchangeable between Android-based components, the configurations for the MTFS services were also adapted to Android’s intent broadcasting system.

Since the MTFS service configurations are merely sets of parameters in the form of name-value pairs, it was quite easy to turn them into a special type of Java objects that make the reconfiguration of the MTFS services an easy task.

The MTFS services, in their turn, were made ready to listen and accept such Java-based configuration objects from Android’s intent broadcasting system, and update their functionality accordingly. Such feature enabled the dynamic orchestration and tuning of MTFS services on the fly, making them adaptable to the current state of device or changes in the end-user’s environment.

For example, the frequency of location detections in the MTFS Location Service was made adjustable to the events detected by the Accelerometer Service and thus, the number of location queries and updates was increasing after a person started walking or running, and it was decreasing when no end-user’s motions were detected for some time.

All this allowed producing a quite dynamic and flexible system where each part could be improved or modified without breaking the other components.

After adding support for the JSON serialization to all MTFS data and MTFS service configuration objects, it became possible to transfer and retrieve such objects over the network, thus enabling the remote configuration and tuning of MTFS services on mobile devices.

4.3.1 Location Service

The location detection functionality might be considered as the core part of the MTFS framework, thus its implementation involved a signification portion of development and testing.

The MTFS Location Service is defined in the *fi.oamk.mtfs.service.location.LocationService* class. Just as all the other MTFS services, the location service was made reconfigurable on the fly, therefore its performances could be adjusted according to the user activities or state of the mobile device.

The configuration of Location Service is wrapped into the *fi.oamk.mtfs.service.location.LocationServiceConfiguration* class and supports the following parameters:

- The minimal distance change to be tracked by Android’s Location Manager, in meters
- The minimal update interval to be used by the Location Manager, in milliseconds
- The battery power consumption plan for Location Manager, defined by accuracy and power criteria

The Location Service interacts directly with Android's standard Location Manager, which communicates with geopositional hardware presented on the mobile device.

The location data obtained from Android's Location Manager is wrapped into the *android.location.Location* objects, containing latitude, longitude, speed, bearing, accuracy and other parameters. The latitude and longitude coordinates are represented within the standard WGS84 datum, usually called simply as the GPS coordinate system.

The Android’s standard location objects are turned by Location Service into MTFS-specific *fi.oamk.mtfs.service.location.Location* objects and broadcasted across all subscribers by Android’s intent broadcasting system.

The primary consumers of these location objects are the Safety Navigation application’s Main Service and MTFS Geofence Service.

The default configuration for Location Service is set to 10 meters of geographical accuracy and to 15 seconds between each location update. The power consuming plan is set to the most rational utilization of the battery.

The Safety Navigation application automatically updates the Location Service configurations when it detects that device (and also, most probably, a person too) is moving. In such case the Safety Navigation application sets the geographical accuracy to 2 meters and the location update interval to 5 seconds.

Later on, when the application will detect that the device is not moving for more than 30 seconds, the Location Service configurations are restored back to defaults.

4.3.2 Geofence Service

The Geofence Service keeps track on the user’s presence in the specified geofences. The geofences are special locations and areas related to end-user’s interests or activities, such as for example, home, work or hobby.

The geofences in the MTFS framework are defined by the *fi.oamk.mtfs.service.geofence.Geofence* objects. Such geofence object contains the geographical location of place, the radius of geofencing circle in meters, the name of a Wi-Fi network presented in place, or an emergency contact associated with the geofence location.

This simple set of properties allows defining geofence as a circle on the map, thus simplifying its utilization and processing inside both the mobile application and back-end system. An example of a geofence location is displayed in Figure 24.

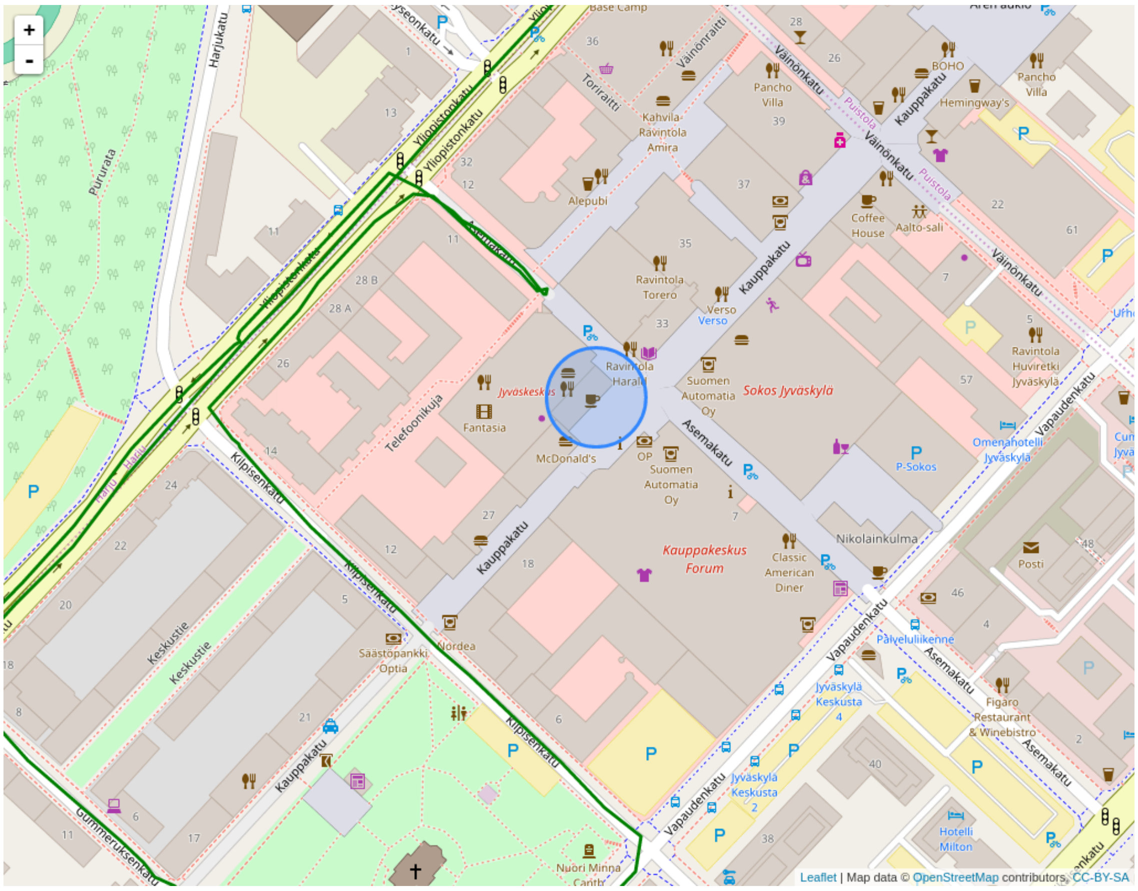


FIGURE 24. A geofence location associated with author's work office

Each user of the Safety Navigation system could be supplied with a dynamic and personalized set of geofences. The user-specific geofences are delivered to the Geofence Service inside the configuration object, defined in the *fi.oamk.mtfs.service.geofence.GeofenceServiceConfiguration* class.

Once the geofences have been provided, the Geofence Service will start checking them up on each location update. They are obtained either by listening to notifications from the Location Service or by receiving such updates directly from some component in the mobile application.

When the Geofence Service receives a new location update, it starts comparing it against all specified geofences. It simply calculates the geographical distance between the obtained location and geofence position, and if the new location will be within the geofence circle, the end-user will be considered as located inside that geofence area.

In the MTFS context all geofences are interpreted as safe locations, thus if the end-user is not detected in one of them this will be considered as an alarming event for both the end-user and their supervisor.

At the same time the MTFS Geofence Service is able to track events of entering, leaving and staying within the geofence location. If such an event is detected, it will be broadcasted by the Geofence Service in the form of the *fi.oamk.mtfs.service.geofence.GeofenceEvent* object.

The generated geofence events will have all attributes of the related geofence and the corresponding event ID describing what has happened. Currently, MTFS supports four types of such geofence events:

- *fi.oamk.mtfs.service.geofence.GeofenceEvent.UNKNOWN* – is assigned by default and tells that the end-user is most probably outside the geofence location
- *fi.oamk.mtfs.service.geofence.GeofenceEvent.ENTER* – tells that the end-user has entered into the geofence location
- *fi.oamk.mtfs.service.geofence.GeofenceEvent.STAY* – tells that the end-user is still present in the geofence location
- fi.oamk.mtfs.service.geofence.GeofenceEvent.EXIT – tells that the end-user has left the geofence location

A schematic description of the geofence event generation is presented in the Figure 25. The circle around the house in Figure 25 represents the home geofence location. When the end-user enters into that geofence circle, the mobile application interprets it as a geofence entering event, notifying the back-end system and supervisors that the user is now in safety.

When the end-user is still presented at the geofence location, it is considered as the user’s stay in safety. Only when the user moves away from the geofence location, the geofence exiting event will be generated, telling the back-end and supervisors that from now the user could be in danger.

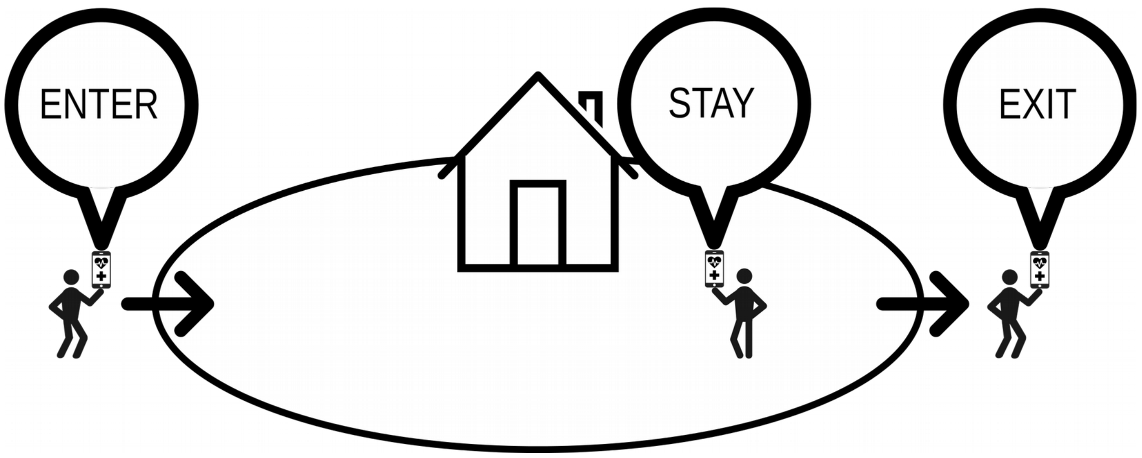
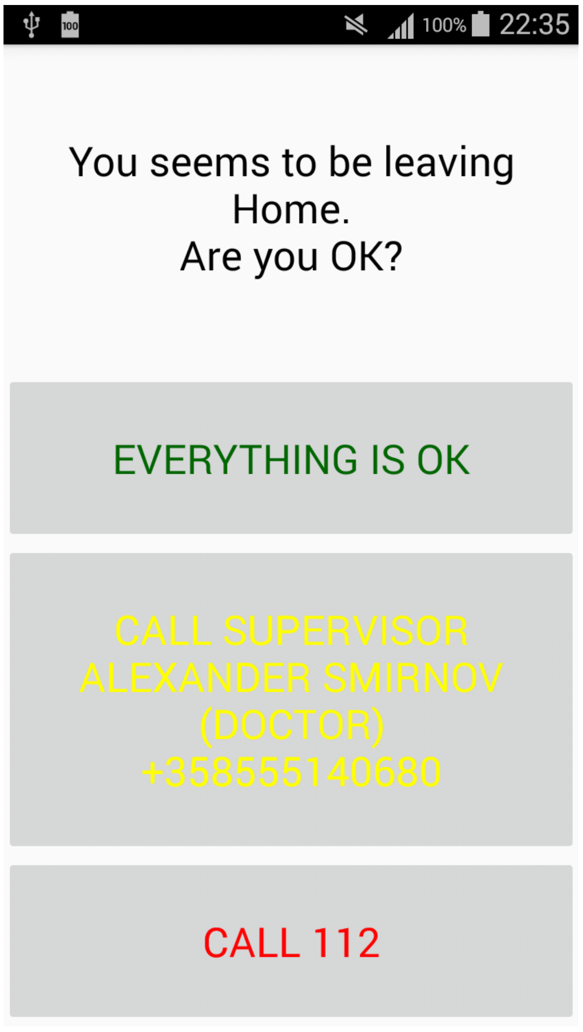


FIGURE 25. Schematic description on generated geofence events

Depending on the Safety Navigation application's configuration, the geofence exiting event will be processed either loudly or silently, leading to a notification send to the supervisors or an alarm dialog present to the end-user, as shown in Figure 26.

The MTFS geofence could be also associated with some emergency contact, representing a person or an organization that might help the end-user in case of an accident or some challenge. Such contacts are defined by the *fi.oamk.mtfs.service.identity.EmergencyContact* class and they could be uniquely specified for each available geofence location.

In the current implementation the details of geofence's emergency contact are presented in an alarm dialog which is activated when the user leaves from a safe location or when a possible accident is detected inside the geofenced area.



100% 22:35

You seems to be leaving Home.
Are you OK?

EVERYTHING IS OK

CALL SUPERVISOR ALEXANDER SMIRNOV (DOCTOR) +358555140680

CALL 112

FIGURE 26. An example of the alarm dialog presented to user after leaving geofence location

The MTFS geofences could also be optionally associated with the Wi-Fi network presented at the geofence location. Thus, the end-user's presence in the geofence area could also be tracked by a connectivity with the associated Wi-Fi network. Such feature helps to amend the location detection inside the buildings where the satellite-based location detection could be quite challenging.

If the end-user will connect to the Wi-Fi network associated with the geofence location, it will be considered as a geofence entering event, while disconnecting from the associated Wi-Fi network will be automatically interpreted as a geofence leaving event.

4.3.3 Device Service

The Device Service, implemented by the *fi.oamk.mtfs.service.device.DeviceService* class, keeps track of all system events related to the resources of a mobile device and notifies about them.

The most important events that the Device Service is tracking are changes in the network connectivity, obtaining current battery levels, checking the presence of the end-user to the mobile system and receiving notes about upcoming reboots or shutdowns of the mobile device.

The Android-specific list of tracked event IDs is presented in the following list:

- *ConnectivityManager.CONNECTIVITY_ACTION* – issued up on changes in the network connectivity
- *Intent.ACTION_USER_PRESENT* – issued on the end-user's presence to the mobile system
- *Intent.ACTION_BATTERY_CHANGED, Intent.ACTION_BATTERY_LOW* and *Intent.ACTION_BATTERY_OKAY* – issued on changes in the capacity of a mobile battery
- *Intent.ACTION_POWER_CONNECTED* and *Intent.ACTION_POWER_DISCONNECTED* and *Intent.ACTION_DOCK_EVENT* – issued on connecting an external power supply, or a docking station to the mobile device
- *Intent.ACTION_DEVICE_STORAGE_LOW* – issued when the device storage will have not enough of free space required for a normal function of the Android system
- *Intent.ACTION_REBOOT, Intent.ACTION_SHUTDOWN, Intent.ACTION_BOOT_COMPLETED, Intent.ACTION_LOCKED_BOOT_COMPLETED* – issued at various stages of a device reboot and shutdown

- *Intent.ACTION_CALL, Intent.ACTION_NEW_OUTGOING_CALL, Intent.ACTION_ANSWER, Intent.ACTION_PROVIDER_CHANGED* and *Intent.ACTION_AIRPLANE_MODE_CHANGED* – issued during phone calls and at various events during the end-user's activities in a cellular network

All these detected events are wrapped into MTFS-specific Java objects and propagated across all the Android system. For example, the network connectivity events are wrapped into the *fi.oamk.mtfs.service.device.NetworkEvent* objects, while battery events are transformed into the corresponding *fi.oamk.mtfs.service.device.BatteryEvent* objects. All other events are carried inside the *fi.oamk.mtfs.service.device.DeviceEvent* objects.

4.3.4 Accelerometer Service

The data from a mobile accelerometer sensor is processed by an independently running Accelerometer Service, defined in the *fi.oamk.mtfs.service.sensor.accelerometer.AccelerometerService* class. The main task of this service is to detect whenever a mobile device, and thus, a person, is moving, running or has fallen.

For determining such events, the service reads device acceleration values from three axes: X, Y and Z, and compares them against user-specific threshold values. The values coming from the accelerometer sensor are represented in the standard SI m/s^2 units.

The threshold values for the service are provided in the configuration objects, defined by the *fi.oamk.mtfs.service.sensor.accelerometer.AccelerometerServiceConfiguration* class. In the current implementation the following configuration parameters are supported:

- Move detection threshold – a float number telling a typical acceleration value that a mobile device will have when a person is walking. The default value for such a move threshold is selected as 0.3 m/s², picked up after some trials and testing

- Run detection threshold – a float number telling a typical acceleration value that a mobile device will have when a person is running. The default value for such a run threshold is selected as 2.5 m/s²
- Fall detection threshold – a float number telling a typical acceleration value that a mobile device will have when it is dropped by a person, or has fallen with the person. The default value for such a fall threshold is selected as 8.5 m/s²

The Accelerometer Service configuration has also a special time window parameter, declaring the minimal time period during which accelerometer samples should be read and studied before the Accelerometer Service event will be generated.

The default value for this parameter is set to 5 seconds, which means that a person must walk or run for at least 5 seconds before such activity will be considered as walking or running. The fall detection routine does not use this time parameter due to quite high acceleration values, typically presented at the device or person's fall, and forces immediate a generation of the corresponding events out of the service.

All detected accelerometer events are propagated across the Android system inside the corresponding *fi.oamk.mtfs.service.sensor.accelerometer.AccelerometerEvent* objects. The accelerometer event objects have a time parameter, describing when exactly an event was detected, and a set of three Boolean parameters, clearly identifying if the end-user was walking, running or has fallen.

4.3.5 HTTP Service

The HTTP Service performs all HTTP method calls issued by the MTFS components. The HTTP Service is defined by the *fi.oamk.mtfs.service.http.HttpService* class and runs independently from all other services and application’s components.

It listens to and processes the HTTP method call objects, defined by the *fi.oamk.mtfs.service.http.HttpMethod* class, containing all information required to make an HTTP call to some end-point.

For each HTTP method call object the HTTP Server creates a special asynchronous handler, defined in the *fi.oamk.mtfs.service.http.HttpMethodCall* class. The *HttpMethodCall* is based on Android’s so-called *AsyncTask* type, which allows its execution as an independent thread, thus avoiding any blocking in the mobile application and other MTFS services.

Up on a successful HTTP call to a specified end-point, the results are put back into the original *HttpMethod* object and forwarded back to the issuer of the HTTP call through a callback method, defined in the *fi.oamk.mtfs.service.http.HttpMethodCallResponse* interface.

Each *HttpMethod* object may specify a number of trials which can be taken before the HTTP call will be considered unsuccessful. Such trials could be taken when the mobile device will have some HTTP failures due to a poor network coverage, or because of some limitations in the provider’s network.

In case of an unsuccessful HTTP call the *HttpMethod* object will be processed one more time by the HTTP Service, until its trial limit will be exhausted. In such case the HTTP call will be supplied with an occurred error description and forwarded back to the issuer.

The HTTP Service keeps track of the current network state, thus if the mobile device will go offline, the HTTP Service will start buffering HTTP calls without their execution. When the device will go online, all the buffered HTTP calls will start executing one by one in the received order, thus helping to utilize the network when it is actually available.

4.3.6 Camera Service

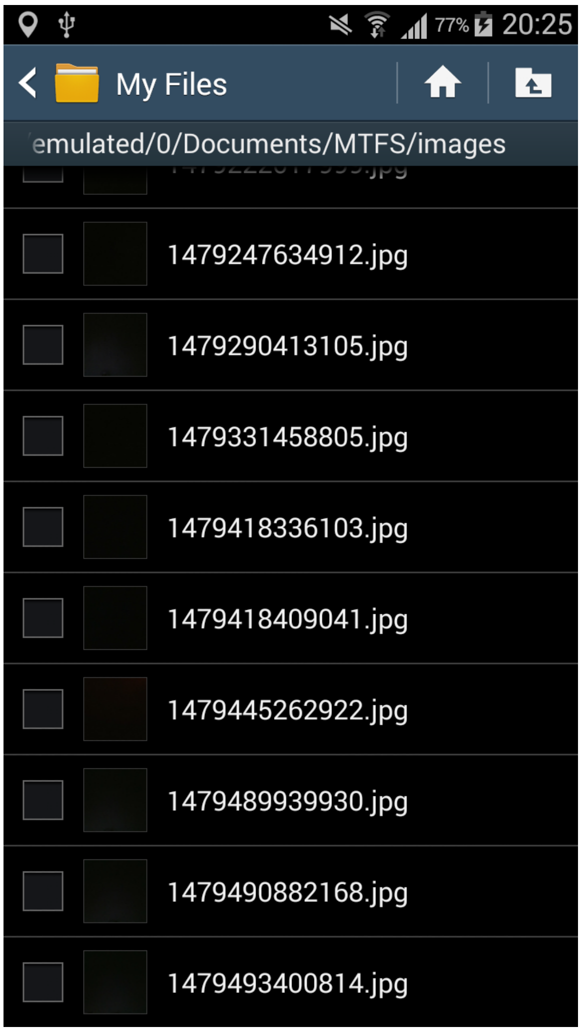
The Camera Service takes pictures from the mobile device cameras. Up on its startup the service studies how many cameras are presented on the mobile device and what are their properties, such as supported resolutions and image formats.

For avoiding any device storage exhausting the Camera Service chooses the smallest available resolution and the JPEG format for all taken pictures.

The Camera Service is defined in the *fi.oamk.mtfs.service.camera.CameraService* class, while a single request for taking a picture is encoded by the *fi.oamk.mtfs.service.camera.CameraAction* object. The Camera Service runs in the background and takes pictures up on received *CameraAction* objects.

A single *CameraAction* contains all the parameters required for taking a single picture. By default, it contains the name of the folder where an image file should be stored, the name of the image file, the ID of the camera to be used for taking an image, and the last detected location of the user.

By default, all taken images are stored in the public */Documents/MTFS/images/* folder, while image files are automatically given the name in the format *<UNIX timestamp of shooting moment>.jpg*, as presented in Figure 27.

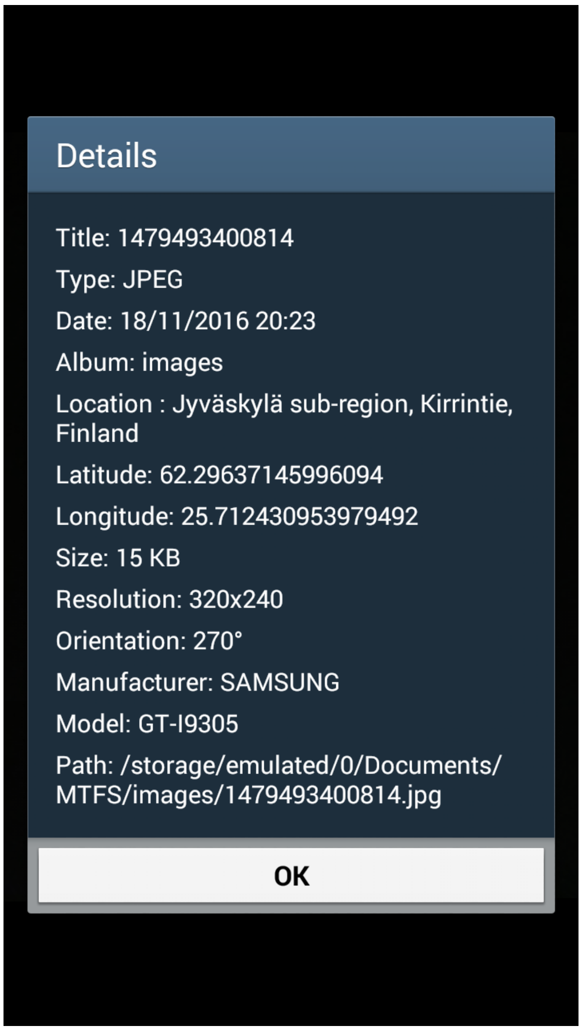


emulated/0/Documents/MTFS/images

1479247634912.jpg
1479290413105.jpg
1479331458805.jpg
1479418336103.jpg
1479418409041.jpg
1479445262922.jpg
1479489939930.jpg
1479490882168.jpg
1479493400814.jpg

FIGURE 27. A list of images taken by the Camera Service

If details of the user's location are provided with the *CameraAction* object, the Camera Service will automatically update image file's location metadata, stored in the EXIF format. This helps to supply the taken images both according to the date and time of the shooting moment and according to the details about a location where the image was taken (see Figure 28).


Details

Title: 1479493400814

Type: JPEG

Date: 18/11/2016 20:23

Album: images

Location : Jyväskylä sub-region, Kirrintie, Finland

Latitude: 62.29637145996094

Longitude: 25.712430953979492

Size: 15 KB

Resolution: 320x240

Orientation: 270°

Manufacturer: SAMSUNG

Model: GT-I9305

Path: /storage/emulated/0/Documents/MTFS/images/1479493400814.jpg

FIGURE 28. An example of EXIF meta-data stored with image file

The taken images are kept on the mobile device as files and they could also be sent to back-end repositories over the HTTP Service. During the image transmission over HTTP, the binary contents of the image file are encoded into a Base64 string representation. Such approach allows easily to envelope images into JSON and XML messages, to transfer them over the network and store them in the platform-independent format inside the data repositories.

4.3.7 Securing the mobile traffic

The back-end services were developed with the Spring framework, which also included the support for the secured authentication and authorization functionality out of the box through its security module (74).

As the usual practice, the traffic between the mobile Safety Navigation application and back-end services was protected by the HTTPS (HTTP over TLS) connections, although with a self-signed certificate.

This simple technique enabled secured communications between the mobile application and the back-end services, virtually preventing hijacking of end-user’s data, such as location and telemetry by so-called “men in the middle” security attacks.

At the same time access to mobile end-user’s data in the back-end’s web GUI was also secured by HTTPS and enhanced by the role-based authorization, thus enforcing application-level security in a typical enterprise application manner.

4.4 The Safety Navigation application

All the earlier mentioned MTFS services were used in a mobile application called the Safety Navigation. The Safety Navigation application is supposed to keep track on the end-user’s activities and location and notify supervisors if something suspicious will happen.

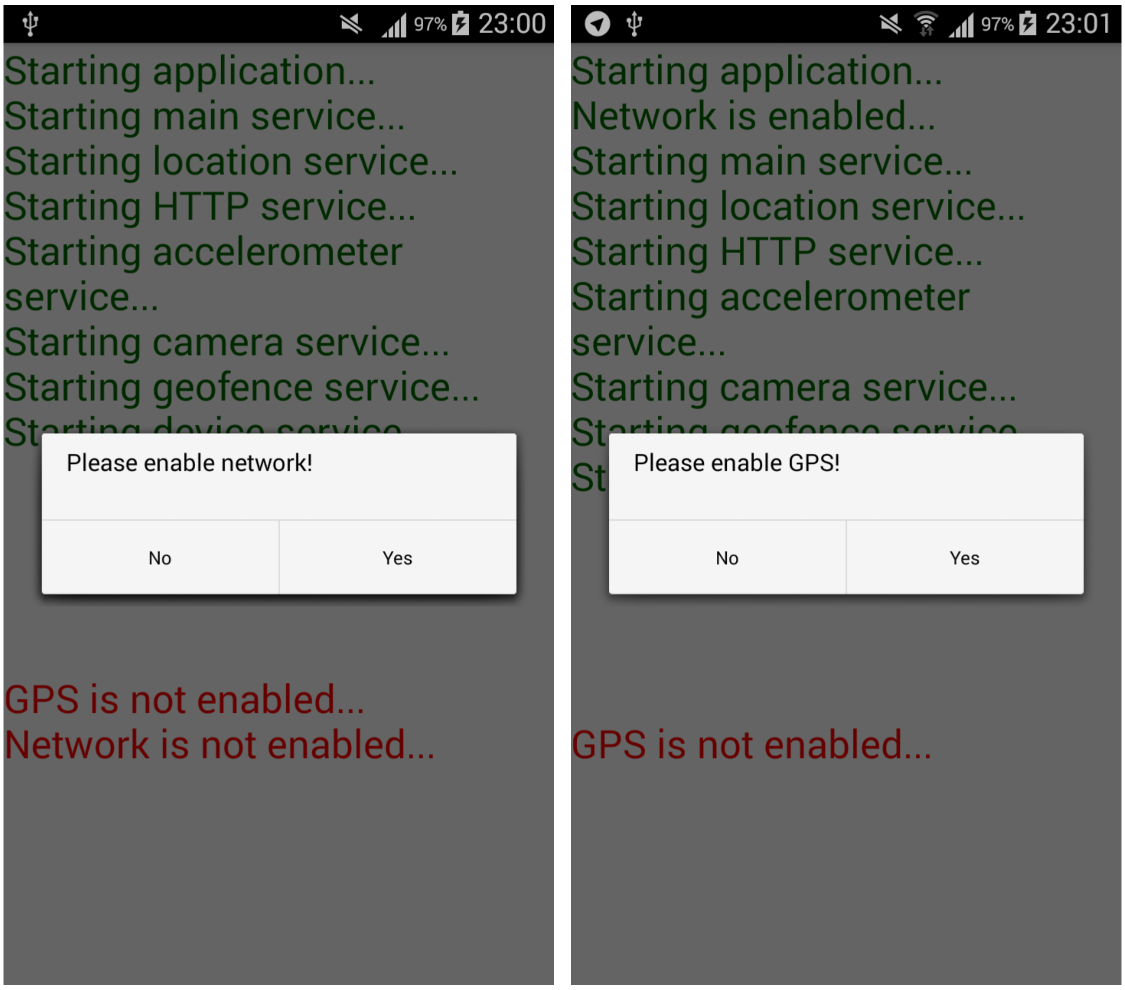
The mobile Safety Navigation application was designed to work in both the online and offline modes and support the end-user in various circumstances. Even if the user will turn off network and GPS module on the mobile device, the Safety Navigation is still capable of doing some work and gather some information about the end-user and their surrounding.

Most of the Safety Navigation’s functionality is coming from the MTFS framework and its application’s Main Service just performs tuning and orchestrating of MTFS services according to the current situation and end-user’s activities.

For example, the Accelerometer Service never stops to track the user’s movements and supplies the Safety Navigation with all information about their activities. The Camera Service is configured by the Safety Navigation for automatically taking a picture about the end-user’s environments each hour while the device is not moving. And the Location Service is tuned by the application to increase location detections when the user walks or runs.

All this is done through configuration objects send to the MTFS services and the corresponding events detected by the MTFS components.

When the mobile Safety Navigation application starts, it checks that the network and GPS modules are enabled on the mobile device. If they are switched off, it asks the mobile user to enable them for a proper work, as presented in the Figure 29.



97% 23:00

Starting application...
Starting main service...
Starting location service...
Starting HTTP service...
Starting accelerometer service...
Starting camera service...
Starting geofence service...
Starting device service

Please enable network!

No Yes

GPS is not enabled...
Network is not enabled...

97% 23:01

Starting application...
Network is enabled...
Starting main service...
Starting location service...
Starting HTTP service...
Starting accelerometer service...
Starting camera service...
Starting geofence service
St

Please enable GPS!

No Yes

GPS is not enabled...

FIGURE 29. Startup dialogs in the mobile Safety Navigation application

When everything is properly initialized, the Safety Navigation application starts running in the background and gather all information that could help the end-user to prevent accidents or to support them when something has happened.

The end-user may keep their usual daily activities and move freely at their location, while the Safety Navigation system will be silently checking their locations and verify them against all specified geofence locations.

When the Safety Navigation detects that the end-user's has left from the geofence location, it will present the user an alarm dialog as shown in Figure 30.

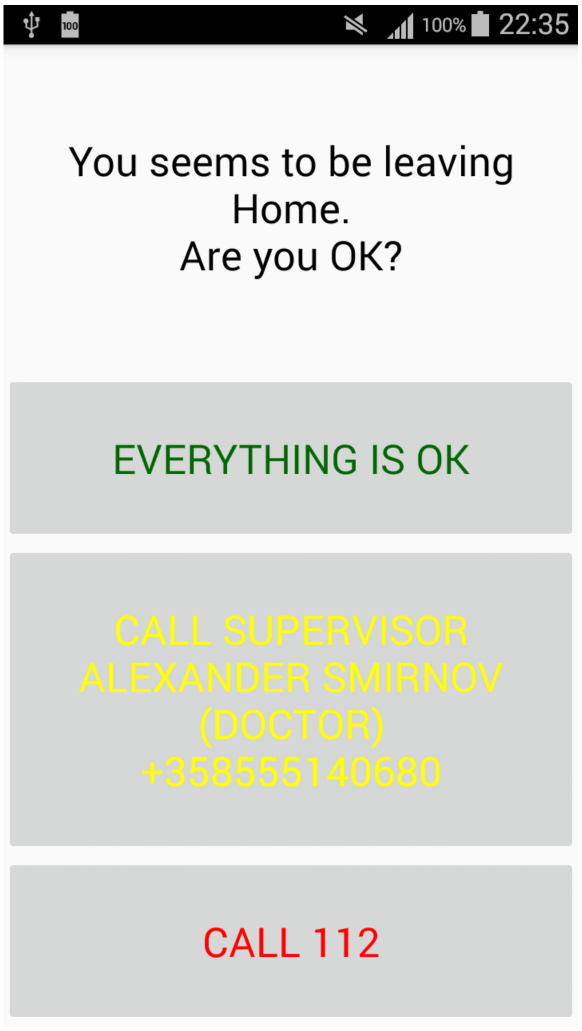

100% 22:35
You seems to be leaving Home.
Are you OK?
EVERYTHING IS OK
CALL SUPERVISOR ALEXANDER SMIRNOV (DOCTOR) +358555140680
CALL 112

FIGURE 30. Alarm dialog presented to end-user after leaving geofence location

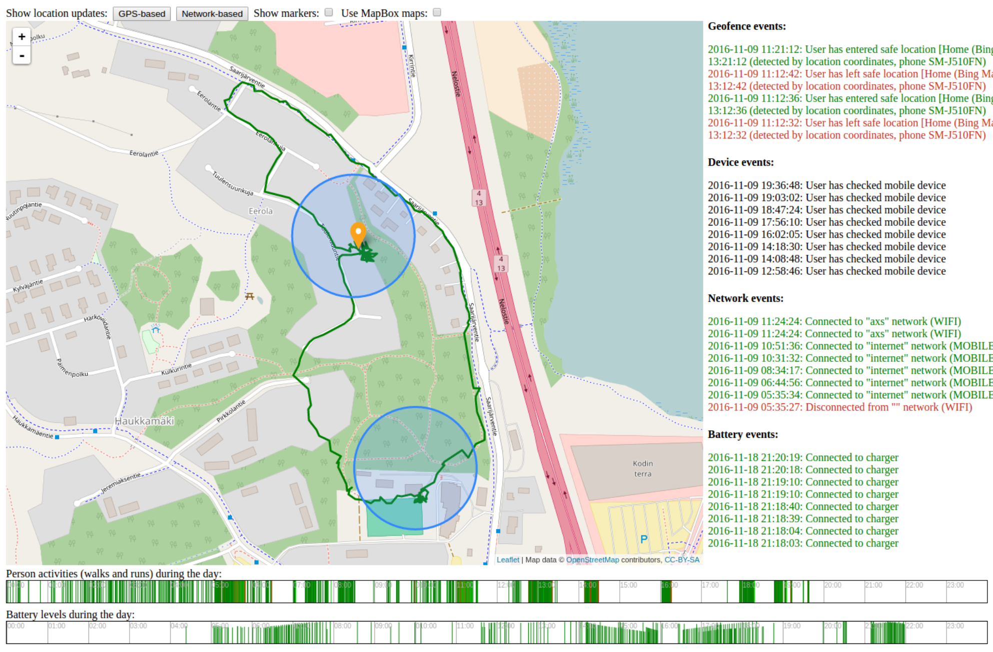
Although such dialog may seem unnecessary or too excessive, it should be remembered that the primary user target for the mobile Safety Navigation application are people suffering from the dementia and memory diseases, thus it could just happen that they will leave their locations without any idea where they are actually going or what they are doing.

The alarm dialog, presented in Figure 30 has a short description of the detected event and three buttons, colored as semaphore lights.

By clicking the first button, called “Everything is OK”, the user is confirming that they just know what they are doing. If the user has some doubts, then the supervisor might be called by clicking the button in the middle. The supervisor’s contact information is obtained either from the default emergency contact encoded into the mobile Safety Navigation application, or extracted from the geofence where this accident has occurred. As the third option, the user may simply invoke the local 112 emergency service.

Whatever button is clicked, the corresponding message will be delivered to the supervisors to be examined and verified.

The information about the user’s current location and mobile device resources is also automatically delivered to the back-end services, or stored into a special log file on the mobile device for later processing if the device is offline. All this information may be easily seen from the supervisor’s UI, handled by the back-ends and presented in Figure 31.



Show location updates:
GPS-based
Network-based
Show markers:
Use MapBox maps:
+
-
Haukkamäki
Eerola
Kodin terra
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA
Person activities (walks and runs) during the day:
Battery levels during the day:

Geofence events:

2016-11-09 11:21:12: User has entered safe location [Home (Bing Maps)] at 2016.11.09 13:21:12 (detected by location coordinates, phone SM-J510FN)
2016-11-09 11:12:42: User has left safe location [Home (Bing Maps)] at 2016.11.09 13:12:42 (detected by location coordinates, phone SM-J510FN)
2016-11-09 11:12:36: User has entered safe location [Home (Bing Maps)] at 2016.11.09 13:12:36 (detected by location coordinates, phone SM-J510FN)
2016-11-09 11:12:32: User has left safe location [Home (Bing Maps)] at 2016.11.09 13:12:32 (detected by location coordinates, phone SM-J510FN)

Device events:

2016-11-09 19:36:48: User has checked mobile device
2016-11-09 19:03:02: User has checked mobile device
2016-11-09 18:47:24: User has checked mobile device
2016-11-09 17:56:10: User has checked mobile device
2016-11-09 16:02:05: User has checked mobile device
2016-11-09 14:18:30: User has checked mobile device
2016-11-09 14:08:48: User has checked mobile device
2016-11-09 12:58:46: User has checked mobile device

Network events:

2016-11-09 11:24:24: Connected to "axs" network (WIFI)
2016-11-09 11:24:24: Connected to "axs" network (WIFI)
2016-11-09 10:51:36: Connected to "internet" network (MOBILE)
2016-11-09 10:31:32: Connected to "internet" network (MOBILE)
2016-11-09 08:34:17: Connected to "internet" network (MOBILE)
2016-11-09 06:44:56: Connected to "internet" network (MOBILE)
2016-11-09 05:35:34: Connected to "internet" network (MOBILE)
2016-11-09 05:35:27: Disconnected from "" network (WIFI)

Battery events:

2016-11-18 21:20:19: Connected to charger
2016-11-18 21:20:18: Connected to charger
2016-11-18 21:19:10: Connected to charger
2016-11-18 21:19:10: Connected to charger
2016-11-18 21:18:40: Connected to charger
2016-11-18 21:18:39: Connected to charger
2016-11-18 21:18:04: Connected to charger
2016-11-18 21:18:03: Connected to charger

FIGURE 31. User’s activities presented on supervisor’s web page

The supervisor’s view represents the most important information about the corresponding end-user and their activities, such as the current location, obtained tracks, geofence events, network connectivity, as well as the user's walks and runs during the day and changes in other mobile device resources or capabilities.

All this helps the supervisor to determine if the user is just fine or if the user needs some help. With the latest location available to the supervisor, the caregivers could coordinate their efforts to find the user if they get lost.

The end-users' movements are also detected and studied by the mobile Safety Navigation application, be it online or in offline mode. When the mobile application detects that the user has fallen, it will activate another alarm dialog shown in Figure 32, asking about what has happened.

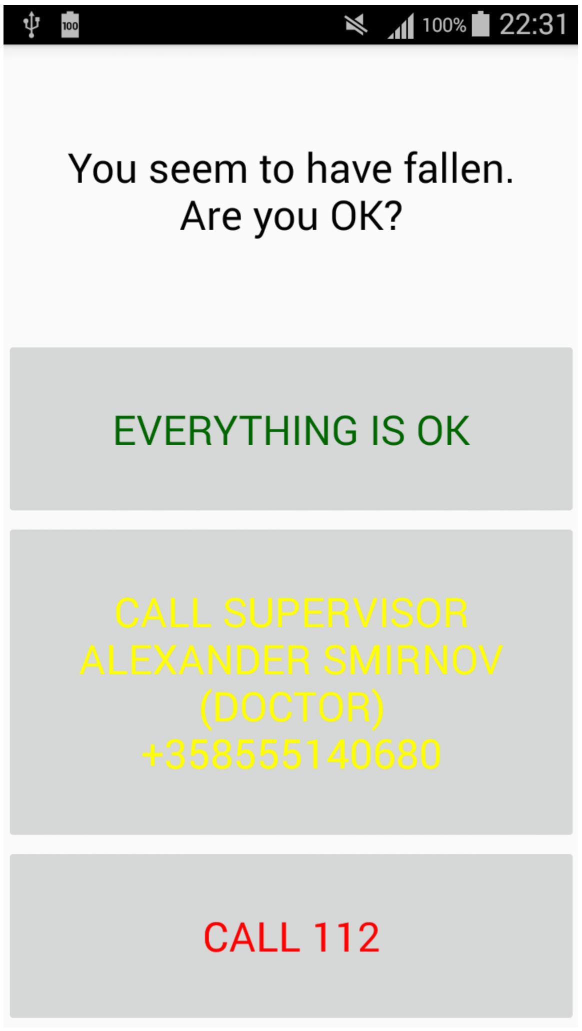

You seem to have fallen.
Are you OK?

FIGURE 32. Alarm dialog presented to the user when falling is detected

Just like in the geofence alarm dialog, the end-user may pickup three options by clicking on the corresponding buttons. The processing logic behind this alarm dialog is exactly the same as described earlier, except that in addition to a mandatory notification message for supervisors, the back-end will draw a vertical red line on the user's physical activity bar at the bottom of the supervisor's UI.

The mobile Safety Navigation application may also work in so-called "stealth" mode, when no alarm dialogs are presented to the user and all information about their activities will be silently forwarded to the supervisors. In this case all estimations about the end-user's health state and conditions will be totally in the responsibility of the supervisors.

The configurations for the mobile Safety Navigation application could be encoded directly into the application or obtained from the network. The configurations might include the following parameters:

- Application's working mode
- MTFS back-end URL
- User contact information
- Default emergency contact for the user
- Location Service settings
- Accelerometer Service settings

A detailed view about all current Safety Navigation settings is presented in Figure 33.

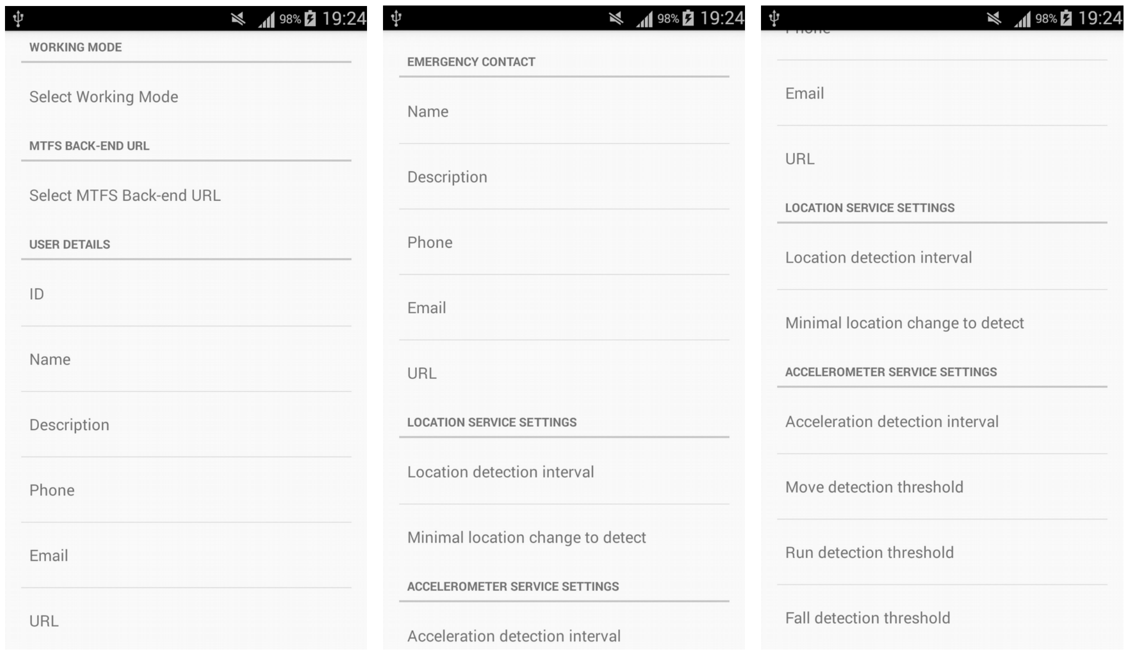

19:24
WORKING MODE
Select Working Mode
MTFS BACK-END URL
Select MTFS Back-end URL
USER DETAILS
ID
Name
Description
Phone
Email
URL
19:24
EMERGENCY CONTACT
Name
Description
Phone
Email
URL
LOCATION SERVICE SETTINGS
Location detection interval
Minimal location change to detect
ACCELEROMETER SERVICE SETTINGS
Acceleration detection interval
19:24
Email
URL
LOCATION SERVICE SETTINGS
Location detection interval
Minimal location change to detect
ACCELEROMETER SERVICE SETTINGS
Acceleration detection interval
Move detection threshold
Run detection threshold
Fall detection threshold

FIGURE 33. The Safety Navigation application settings

After the Safety Navigation application has been installed, the supervisor should go through the application settings and change them accordingly. However, with the settings obtained directly from the network, the Safety Navigation could immediately get some very optimal parameters created for each user in a personalized manner.

In both cases the two most important Safety Navigation settings should be currently defined manually: the MTFS back-end's URL, which defines an entry point for all back-end API calls, and the user ID, which uniquely identifies each user in the MTFS system.

In the current version users' phone numbers are used as unique identifiers. However, the user identifier in the Safety Navigation system could just be any unique set of URI-friendly characters, be it a phone number, an email address or some universally unique identifier (UUID).

If the MTFS back-end's URL will be left unspecified or will have an invalid address, the Safety Navigation application will automatically switch to the back-end offline mode, where all end-user's activities and location updates will be stored into a log file on the mobile device, created in a public directory under the

/Documents/MTFS/MTFS_year.month.day.log path. In this case the log files will be automatically created for each day that back-ends are offline.

Such log files could help to preserve quite critical data about user activities when the device goes offline or has some connectivity issues. These log files may also work as the backup copies of user’s data in case of a device failure or during accident investigations.

4.5 Back-end system

As it was said earlier, the back-end system was not a part of the original plan but due to the need to store, view and debug data, obtained from the mobile Safety Navigation application and MTFS framework, the author decided to create one with the Java 8 programming language and the Spring framework.

The back-end’s services were quite simple Spring controllers, offering their functionality through RESTfull APIs and offering means to store, update, retrieve and delete the MTFS and Safety Navigation entities with the standard HTTP method calls.

4.5.1 Data repository

The Spring framework’s CRUD repositories, available out of the box, have significantly simplified the interactivity with the database, allowing to concentrate more on the REST-based implementation of back-end services for the Safety Navigation system.

The PostgreSQL database instance used to store the Safety Navigation and MTFS data was provided by a stand-alone server and accessed by the back-end services through the Spring framework’s CRUD repositories, backed up by Hibernate entity managers.

For the sake of the author’s technical interests, the back-ends were also configured for the utilization of so-called in-memory H2 database, running purely inside the hosting Java Virtual Machine in PostgreSQL-compatible mode (75).

Such solution allows deploying the database functionality inside the Safety Navigation back-end’s binaries without a need to install dedicated PostgreSQL servers, thus virtually allowing an easy installation of back-end instances in cloud-based or other platform-as-a-service hosting environments.

The role-based access to database entries and back-end’s functionalities was also enabled through the Spring framework’s standard security configurations. The back-end supports the same roles as were outlined in the Safety Navigation concept, namely the *USER* and the *SUPERVISOR*. The *ADMIN* role was created especially for the back-end’s specific technical tasks.

4.5.2 Front-end

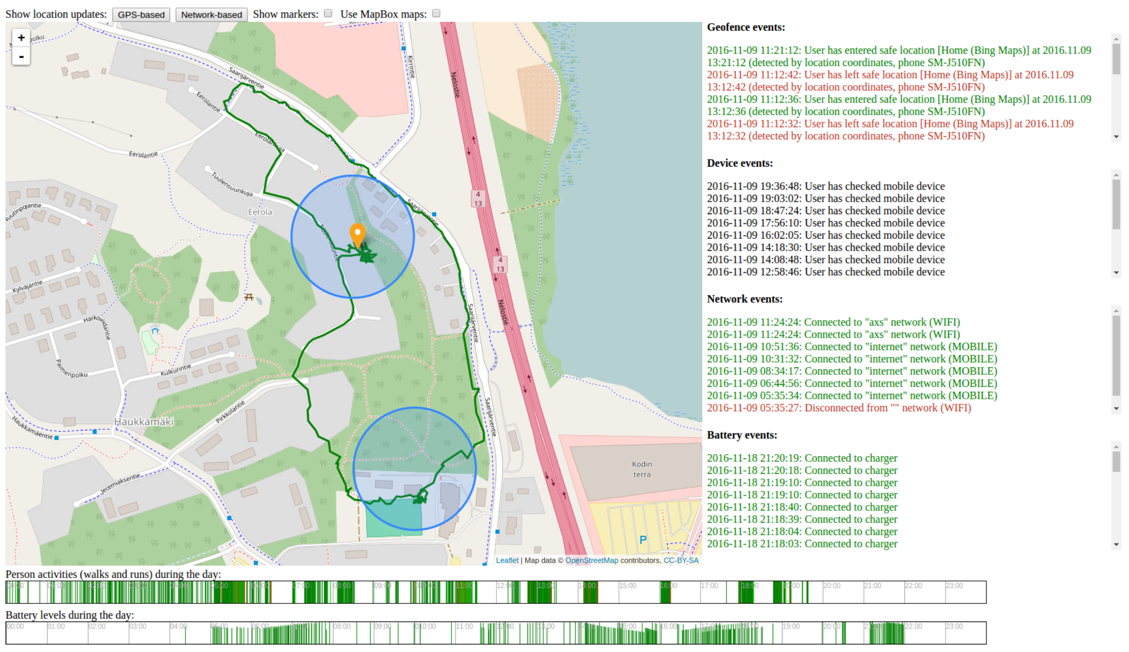
Although most of the back-end’s data could be obtained through RESTfull API calls, the author decided to create a very simple UI for supervisors, just to display all important data obtained from mobile user activities.

Since the back-end services were implemented with the Spring framework, there were a few HTML template engines bundled with Spring and available out of the box. Therefore, a generation of dynamic HTML pages was enabled at the back-ends without any additional libraries or web runtime environments.

After a short review and trials, the author selected the Apache FreeMarker template engine for generating a dynamic HTML content (71). The FreeMarker engine provided enough of capabilities for representing all mobile user’s activities and location updates on a relatively simple supervisor’s UI.

For representing the end-user’s location updates on a map and enabling some interactivity, the author selected the Leaflet JavaScript library (70). The Leaflet library allowed a direct utilization of MTFS and Safety Navigation location data, converted to JSON objects, so that the user’s location history, geofences and last known location were seamlessly integrated into the interactive map view.

The map view’s layers, tiles and location features were provided by the OpenStreetMap (65) and the MapBox servers (76), selected up on the viewer’s preferences.


Show location updates:
GPS-based
Network-based
Show markers:
Use MapBox maps:
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA
Person activities (walks and runs) during the day:
Battery levels during the day:

Geofence events:

2016-11-09 11:21:12: User has entered safe location [Home (Bing Maps)] at 2016.11.09 13:21:12 (detected by location coordinates, phone SM-J510FN)
2016-11-09 11:12:42: User has left safe location [Home (Bing Maps)] at 2016.11.09 13:12:42 (detected by location coordinates, phone SM-J510FN)
2016-11-09 11:12:36: User has entered safe location [Home (Bing Maps)] at 2016.11.09 13:12:36 (detected by location coordinates, phone SM-J510FN)
2016-11-09 11:12:32: User has left safe location [Home (Bing Maps)] at 2016.11.09 13:12:32 (detected by location coordinates, phone SM-J510FN)

Device events:

2016-11-09 19:36:48: User has checked mobile device
2016-11-09 19:03:02: User has checked mobile device
2016-11-09 18:47:24: User has checked mobile device
2016-11-09 17:56:10: User has checked mobile device
2016-11-09 16:02:05: User has checked mobile device
2016-11-09 14:18:30: User has checked mobile device
2016-11-09 14:08:48: User has checked mobile device
2016-11-09 12:58:46: User has checked mobile device

Network events:

2016-11-09 11:24:24: Connected to "axs" network (WIFI)
2016-11-09 11:24:24: Connected to "axs" network (WIFI)
2016-11-09 10:51:36: Connected to "internet" network (MOBILE)
2016-11-09 10:31:32: Connected to "internet" network (MOBILE)
2016-11-09 08:34:17: Connected to "internet" network (MOBILE)
2016-11-09 06:44:56: Connected to "internet" network (MOBILE)
2016-11-09 05:35:34: Connected to "internet" network (MOBILE)
2016-11-09 05:35:27: Disconnected from "" network (WIFI)

Battery events:

2016-11-18 21:20:19: Connected to charger
2016-11-18 21:20:18: Connected to charger
2016-11-18 21:19:10: Connected to charger
2016-11-18 21:19:10: Connected to charger
2016-11-18 21:18:40: Connected to charger
2016-11-18 21:18:39: Connected to charger
2016-11-18 21:18:04: Connected to charger
2016-11-18 21:18:03: Connected to charger

FIGURE 34. Supervisor's UI in the Safety Navigation system

As can be seen in Figure 34, the most important elements of the supervisor's UI are an interactive map, activity bars and event descriptions. The map view scales according to the user's location history and their current location on the map. The end-user's location tracks are presented as green-colored paths, while geofence locations are presented by blue circles and the user's current location is represented by the orange marker in the same map view.

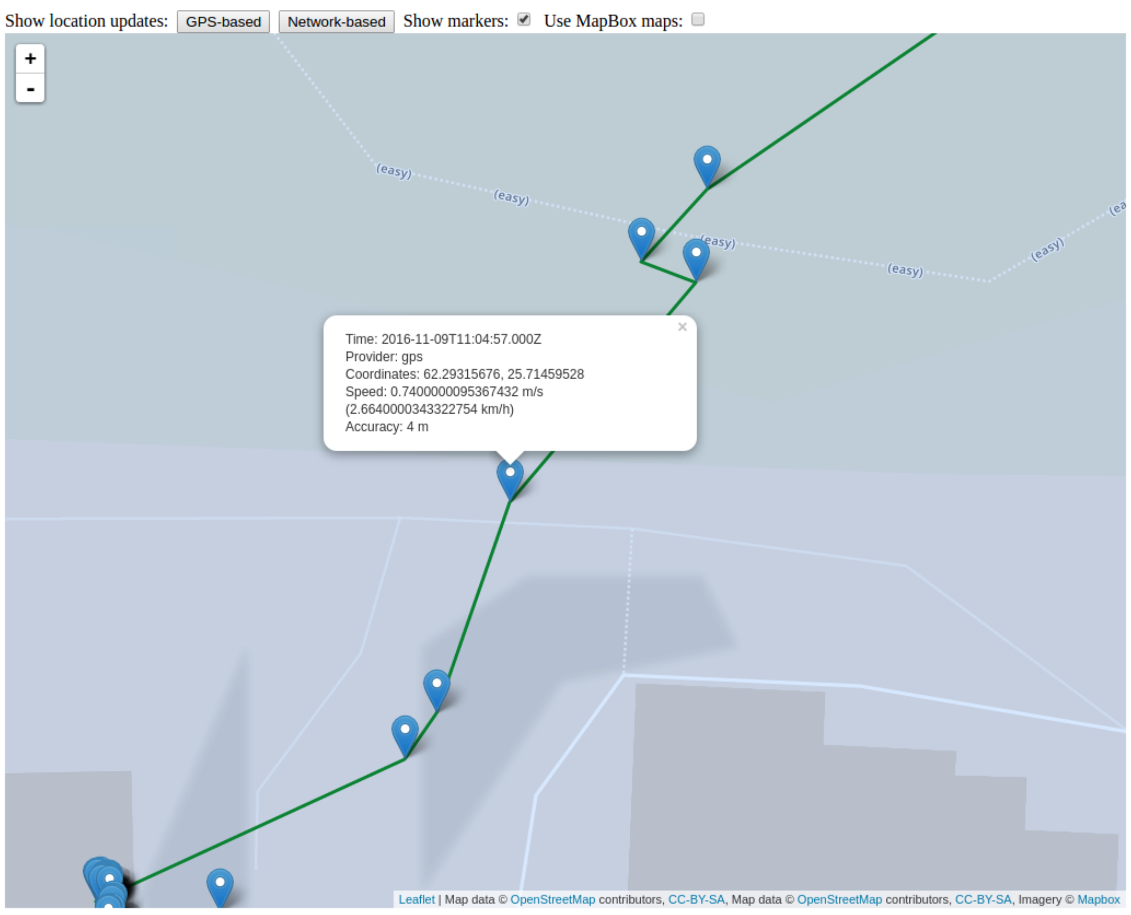
The most important events are displayed on the supervisor's UI on the right side. There the supervisor can find the latest geofence events, such as entering or exiting safe locations. The next are the device events, describing what has happened to the end-user's mobile device, e.g. if the GPS module has been switched on or off. Below are the networks events, showing connections or disconnections from the wireless networks. The last block shows battery events, describing when was the last time the user has charged the mobile battery.

At the bottom of the supervisor's UI there are two histograms, showing the end-user's physical activity and device battery levels during the day.

The end-user's walks are presented on a histogram as vertical green lines, positioned exactly at the time when these walks were detected. The light green lines are drawn at the moments when the user was running, and red lines are drawn at the moments when the user's falls were detected. Such histogram can easily show if some anomalies in the user’s physical activities were presented, possibly speeding up the supervisor’s decisions.

The map view may display both the locations obtained from the GPS system and the ones obtained from the wireless and cellular networks. The locations obtained from the networks could be of less precision compared to those from the GPS system, but they could help to trace the end-user if the GPS module on the mobile device were switched off.

The “Show markers” option helps to display markers on the map view, showing some more details about each location update. An example of such marker is presented in the Figure 35.


Show location updates: GPS-based Network-based Show markers: ✓ Use MapBox maps: ☐

Time: 2016-11-09T11:04:57.000Z
Provider: gps
Coordinates: 62.29315676, 25.71459528
Speed: 0.7400000095367432 m/s
(2.6640000343322754 km/h)
Accuracy: 4 m

+
-
(easy)
(easy)
(easy)
(easy)
(easy)
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA, Map data © OpenStreetMap contributors, CC-BY-SA, Imagery © Mapbox

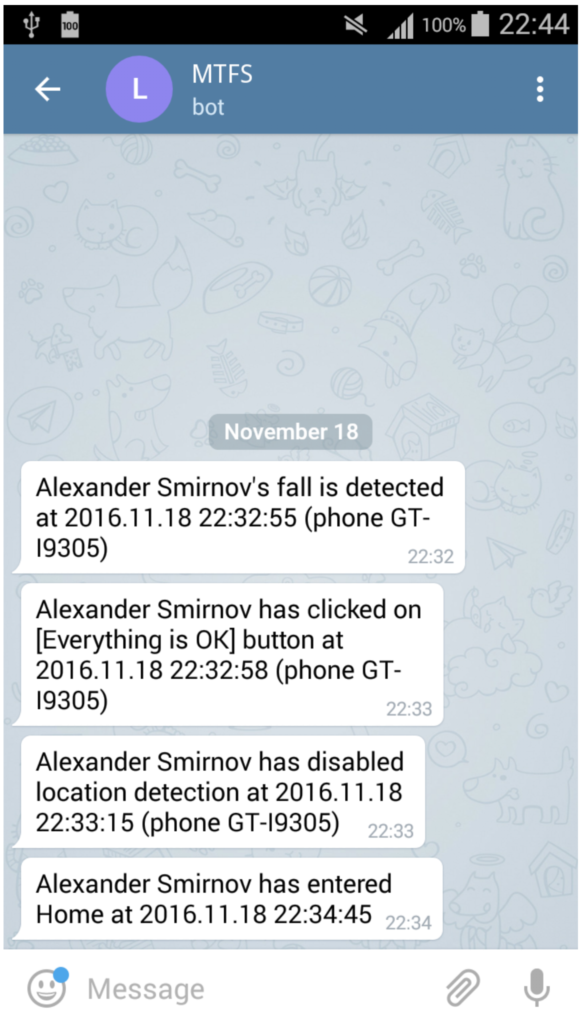
FIGURE 35. Marker detailed view

By clicking on the marker, the supervisor may see more details regarding that particular detected location, such as its longitude and latitude, time stamp, the user’s speed, accuracy and how the location was obtained. Such details may help the supervisors to verify the quality of location data and review the end-user's behaviors.

4.5.3 Integrity with the third party systems

Just being too curious about the integration capabilities of both the mobile application and back-end services, the author decided to integrate the Safety Navigation with the Telegram messaging system (77).

The Figure 36 shows some messages issued by the Safety Navigation system to the author's Telegram's account through the MTFS Telegram's bot.



Alexander Smirnov's fall is detected at 2016.11.18 22:32:55 (phone GT-I9305) 22:32

Alexander Smirnov has clicked on [Everything is OK] button at 2016.11.18 22:32:58 (phone GT-I9305) 22:33

Alexander Smirnov has disabled location detection at 2016.11.18 22:33:15 (phone GT-I9305) 22:33

Alexander Smirnov has entered Home at 2016.11.18 22:34:45 22:34

FIGURE 36. Safety Navigation notifications delivered through Telegram API

Such integration with the messaging systems could bring some benefits to the supervisors and end-users, since it virtually shortens the links between people and systems, and allows communicating over messaging clients rather than through custom made communication systems.

In such case both parties could benefit from the rich infrastructure that messaging systems are offering and, for example, minimize the need to use special web sites or clients to review the end-user’s state and location.

5 TESTING

Right from the very beginning all developed software components were tested on real Android devices and in real-life circumstances. The author’s aim was to develop reliable, flexible and battery efficient software which other mobile applications could benefit from.

The mobile Safety Navigation application was tested primarily by the author and his 8-year old son. Although originally the Safety Navigation application was aimed at elderly users, the author’s experience showed that there are quite a lot of behavioral similarities between the elderly and young people, leading to a conclusion that the testing coverage was intensive and sufficient.

5.1 Improving the location detection

The testing of the Safety Navigation application and MTFS components in the real circumstances almost immediately started by supplying a project with an important feedback on the reliability and performance of developed software. The obtained results helped to significantly improve the most critical MTFS services right from the very beginning, namely the Location, Accelerometer and the HTTP Services.

For example, the location detection was not always of sufficient precision, leading to error-prone results due to various factors, such as weather, network coverage or the structure of the building where the end-user was located. Such erroneous location detections might be easily seen in Figure 37, where the GPS coordinates obtained from the Android’s Location Manager were quite disturbed, leading to a fuzzy overall picture.

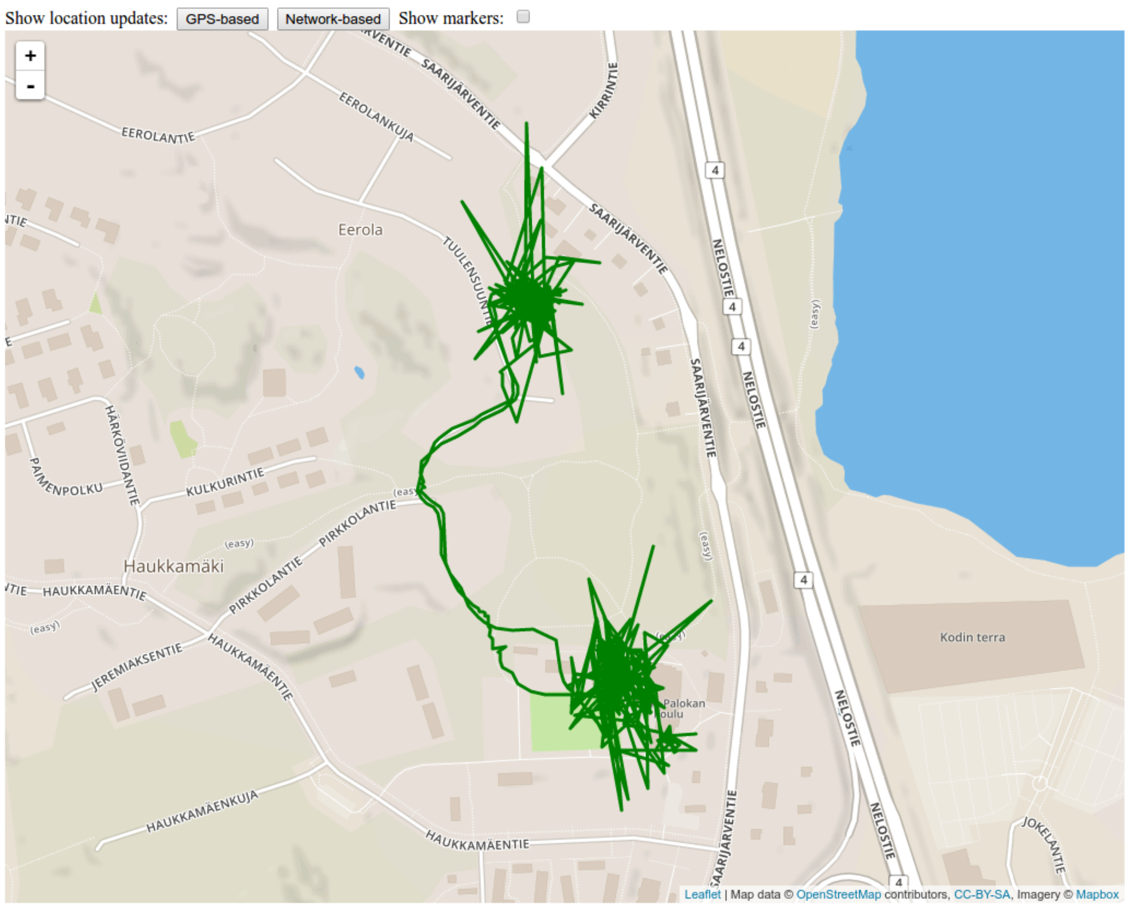


FIGURE 37. Quality of location detection in MTFS framework at the earlier stages of the project

Such disturbance came from the lack of GPS signal inside the buildings, thus the Android's location detection subsystem tried its best to guess where the device was actually located. At the same time the location detection was significantly better outside the buildings, especially when the end-user was moving, leading to a smooth track between two locations on the map.

Realizing that, the precision of location detection in the MTFS framework was improved by activating the GPS module only when the device was actually moving. The detection of device movements was performed by the Accelerometer Service, which could identify when a person with the device was most likely walking, running or staying at some place.

Each time the Accelerometer Service was detecting the movement of the device, the corresponding signal was propagated across all MTFS services, including the Safety Navigation application's Main Service. After receiving such signal, the Main Service was reconfiguring the Location Service to an increased frequency of location detections, thus obtaining a much better precision in detected locations.

Soon after the person had stopped walking or running, the Accelerometer Service was broadcasting another corresponding signal, and the application's Main Service was reconfiguring the Location Service to a decreased frequency of location detections, thus significantly minimizing the amount of erroneous location detections.

As a result, such approach lead to a much better quality of detected locations, as presented in the example in Figure 38. Additionally, the Location Service was activated at right time and in right situations, helping thus to prolong the battery life and significantly decreasing the amount of incorrect data about the user.

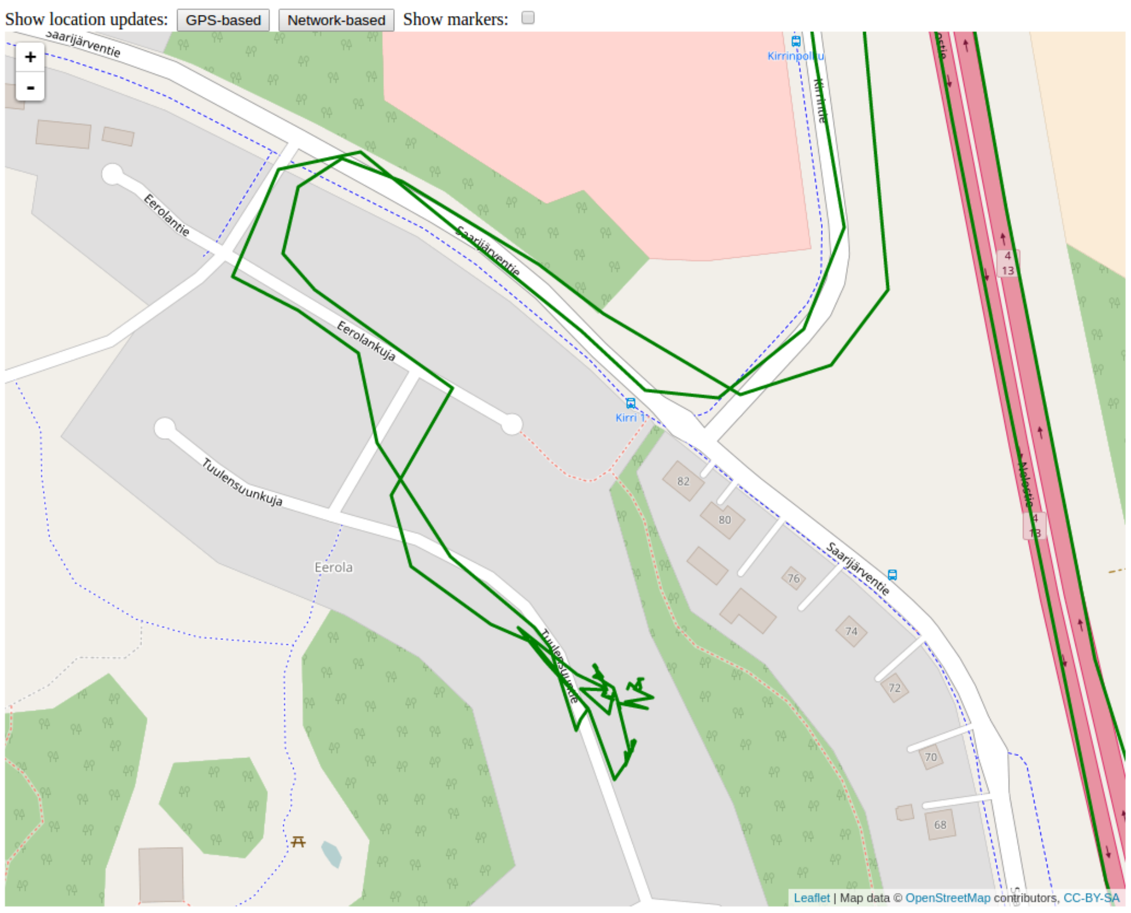


FIGURE 38. Improved quality of location detection after proper MTFS service orchestration

At the same time it is worth to mention that the subject of GPS accuracy is just too wide and depends on many factors, ranging from the GPS chip-set of the mobile device and the quality of system software, to the environmental circumstances and the default accuracy of civil services in the GPS system, which is typically around 5-8 meters (78).

5.2 Adding robustness to the HTTP Service

The HTTP Service was originally implemented as the Android's intent service (79), but quite soon it became obvious that such implementation was not sufficient for MTFS needs.

For example, while being an easy tool for performing tasks in parallel to the application's main thread, the intent-based HTTP Service lacked a required robustness and could not perform multiple simultaneous HTTP requests, thus prolonging and sometimes even blocking communications with the back-end services.

Additionally, the intent-based HTTP Service could not properly buffer HTTP tasks, sometimes leading to an improper order of HTTP requests send to the back-end services.

As an improvement, the HTTP Service was just re-implemented as a stand-alone service, accepting and performing HTTP tasks in an asynchronous multi-threaded manner. This way each HTTP task was enveloped into Android's standard *AsyncTask*, keeping all other requests unblocked and independent from each other.

Additionally, the HTTP Service was made aware of the network state by accepting the corresponding network state notifications from the Device Service. When the network was not available, the HTTP Service could simply start buffering HTTP tasks and execute them later when the mobile device was online again.

However, even when the network was available to the application's needs, some HTTP calls were failing due to various reasons, ranging from the quality of the provider’s network services in some geographic locations, to some data processing issues at back-end services.

These issues were solved with the number of trials that the HTTP call could take before it was considered unsuccessful. Such simple solution allowed retrying HTTP calls inside the HTTP Service without disturbing the application’s Main Service or end-user.

As the real testing showed, it helped to overcome some network issues, typically presented at the moments when the device switches between the mobile and Wi-Fi network or when GSM towers are switching frequently while the user is driving in the car.

5.3 Orchestration of the services

Quite soon after the beginning of implementation, it became clear that services should be made re-configurable on the fly.

For example, the Accelerometer Service was supposed to compare the values from the accelerometer sensor to the user-specific threshold values, which were identifying the walking or running of that particular person. The Geofence Service had to support a dynamic set of geofences, updated either by the back-end services or obtained from mobile application configurations.

The solution to this problem quite naturally came from the Android’s information bus system, called an intent broadcasting system, which was also utilized by the MTFS framework.

Since MTFS services were exchanging information with each other over specially composed Java objects, the configurations could be made exchangeable if wrapped into similar Java object structures.

With this simple approach all the critical parameters of each reconfigurable service were encoded into the corresponding configuration Java class. Additionally, such classes were made marshable into their JSON representations, allowing an easy transferring of such configurations over the network.

This simple solution allowed both the reconfiguration of the services on the fly and obtaining such configurations from the back-ends and configuration files, thus leveraging the true power of service decoupling and user-specific tuning of the MTFS services.

6 POSSIBILITIES OF FURTHER DEVELOPMENT

Probably the most important property of the MTFS framework is its modular architecture which enables an easy extension and improving of the MTFS functionality without loosing the already developed features.

Practically, it was just impossible to create a fully scaled framework that would fulfill absolutely all use-cases and needs in the mobile health-care or well-being domains inside a single project. However, anyone can grasp simple ideas behind the MTFS framework will be able to add more functionality to the framework up on some real practical needs.

6.1 Guiding Service

According to the author the most important missing feature of the current MTFS framework is a guiding service, which could help a person to return to a safe location by following a visual and acoustical guidance offered by some Guiding Service.

Just presenting a map view with the path to a destination point might be quite sufficient for the ones who are familiar with the digital maps and who can easily orient on location, but users from the Safety Navigation's target group – the elderly and young people suffering from dementia or wandering behaviors – are not necessarily capable of orienting themselves in the street just by looking at some map.

In this sense an intelligent Guiding Service, which can speak to end-users in a natural language and orientate them to a destination, could be simply the best possible option.

6.2 Supporting other sensors

Although the Accelerometer Service has provided most of the required functionality for MTFS basic use-cases, supporting other types of sensors could enable other functionality and use-cases.

For example, tracking an environmental temperature through the corresponding sensors from a large user group may enable to create live and dynamic temperature maps of the city. Such temperature maps could help in preventing insults and heart attacks, which are usually increasing in hot weather (80).

Similar benefits could be obtained from the humidity sensors, since humidity could as well quite dramatically affect human well-being.

The utilization of the magnetic field sensor could also be useful. Although strong magnetic fields are not that common in rural areas, they are quite common in urban locations, and their presence could dramatically affect such vital devices as artificial heart pacemakers. By reading the surrounding magnetic field, a mobile device could alarm its user just before it will start affecting the user's vital electronic devices.

6.3 Reading user's emotional state

The smartphones of today could provide face recognition services either through an embedded functionality or through some external services, such as Google's Mobile Vision (81). This could give a chance to track the user's emotional state, which might be useful in case of people suffering from mental diseases.

Simply by viewing at the patient's face, the specialist could detect if they have some mental disorder, such as panic attack or anxiety. Maybe a face recognition system could also be taught to perform similar tasks and thus, detect such disorders preemptively. This would help a suffering person to get professional help just on time and possibly avoid any unfortunate consequences of such disorders.

6.4 The 112 volunteering application

While the MTFS framework and Safety Navigation application were developing, the author figured out another possible application for the MTFS functionality – the 112 volunteering mobile application.

The core idea of such 112 volunteering mobile application is to create a virtual network for medical professionals or otherwise properly trained people who may offer first aid to nearby people before an ambulance or police is arriving at the scene of accident.

Although ambulances can truly save lives, unfortunately sometimes they simply cannot arrive just on time to rescue people. In such situation the emergency center could simply check if some properly trained people are near the scene of accident or suffering person and thus invoke them for offering first aid before it will be just too late.

In the 112 volunteering mobile application each such first aid person would specify their contact details, and possibly other medical skills. The mobile application should allow a volunteer to specify when they are available for supporting local emergency centers, so that it would be a truly volunteering experience.

After all preparations, the 112 mobile application would be regularly checking the volunteer’s availability and their current location and updating special database in the local emergency center.

When something unfortunate happens, e.g. a heart attack or some injury, the operator at the emergency center will be able to check if there are any 112 volunteers near the scene of accident and ask them to offer first aid before an ambulance or police arrives to the place.

The emergency center’s operator could see the location of 112 volunteers through the special map view, as presented in the Figure 39.

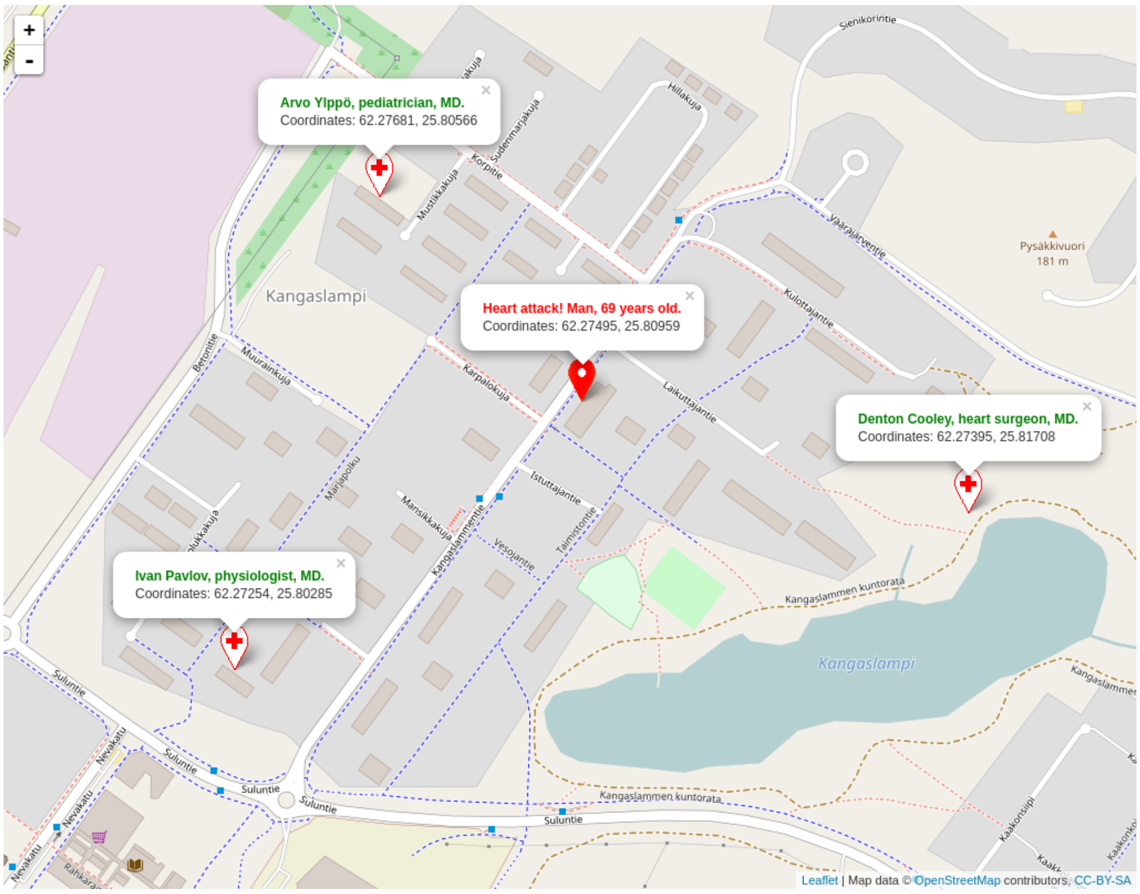

Arvo Ylppö, pediatrician, MD.
Coordinates: 62.27681, 25.80566
Heart attack! Man, 69 years old.
Coordinates: 62.27495, 25.80959
Denton Cooley, heart surgeon, MD.
Coordinates: 62.27395, 25.81708
Ivan Pavlov, physiologist, MD.
Coordinates: 62.27254, 25.80285
Kangaslampi
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA

FIGURE 39. The 112 volunteering application based on MTFS framework

In the example presented in Figure 39, an emergency call about a man suffering from a heart attack has arrived together with a description about his location (presented in Figure with the red-colored marker in the center). The emergency operator sends an ambulance to the scene and also queries in the volunteering database for anyone who may just offer first aid before the medical team is there.

Fortunately, there are three volunteers who may offer such help, presented in Figure 39 by the red-crossed markers around, thus the operator can negotiate with the closest one and send him to the suffering person before the ambulance will be there.

Such 112 volunteering mobile application could also help in other cases, where for example a relatively large amount of people should be organized and coordinated on the scene of accident for some rescuing or searching operation.

Thus, in a much similar way some MTFS functionality could be reused for enabling different use-cases, but with the same simple aim of rescuing people and saving human lives.

7 CONCLUSION

The mobile technology is truly offering some opportunities for the health-care and well-being domains. As the mobile devices become the only de-facto computing devices that people are carrying around, they are representing a truly promising platform for enabling mobile medical services.

With the modular hardware and software such mobile devices could be relatively easy to turn into full-functional medical equipment. The intelligent medical systems, accessible from the Internet, could indeed enable much of the expertise and interactivity in such mobile medicine tools, e.g. providing diagnostics and preventive and monitoring services. Therefore, the combination of mobile computing, medicine and the Internet could indeed bring health-care services virtually to any place on the Earth.

The concepts of the Safety Navigation system were outlined and developed from just similar views. Much of the Safety Navigation's properties were obtained by using modular and interoperable components which were relying on the cooperation between services and systems.

The utilization of the SOA-based approach and information bus concepts in the mobile Safety Navigation application and back-ends has significantly simplified their developments, bringing flexibility and robustness to the whole solution.

Choosing the Android platform as the primary target for the developed solution was the right choice, since it has enabled many features in the MTFS framework with a less development effort.

The resulting system showed enough of flexibility and reliability in some real-life testing, thus proving the right architectural and technological choices made during the project. Hopefully it has also obtained enough of critical features to be reused in other mobile health-care and well-being projects at Oulu University of Applied Sciences or other research companies.

REFERENCES

1. Pehr Brahe Software Laboratory, 2011. Cited 25.02.2013, http://pbol.org/fi/index.jsp?link=projektit

2. Carone, G., Costello, D. 2006. Can Europe Afford to Grow Old? Finance and Development magazine of the International Monetary Fund, September 2006, Volume 43, Number 3. Cited 14.06.2011, http://www.imf.org/external/pubs/ft/fandd/2006/09/carone.htm

3. IBM, Watson Health, 2016. Cited 22.11.2016, https://www.ibm.com/watson/health

4. Smartphone historical sales figures in millions of units, Wikipedia, 2015. Cited 05.01.2014, http://en.wikipedia.org/wiki/Smartphone#Historical_sales_figures_.28in_millions_of_units.29

5. TechCrunch, 1.2B Smartphones Sold In 2014, Led By Larger Screens And Latin America. Cited 21.02.2015, http://techcrunch.com/2015/02/16/1-2b-smartphones-sold-in-2014-led-by-larger-screens-and-latin-america/

6. ARM, System IP, 2015. System IP For 2016 Premium Mobile Systems. Cited 14.02.2015, http://community.arm.com/groups/processors/blog/2015/02/12/system-ip-for-2016-premium-mobile-systems

7. Ventola Lee C., Mobile Devices and Apps for Health Care Professionals: Uses and Benefits, 2014. Cited 21.02.2016, http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4029126/

8. Sotera Wireless, ViSi Mobile 2015. Cited 24.01.2015, http://www.visimobile.com/visi-product-info

9. Makower, J., Meer, A., Denend, L. 2010. FDA Impact on U.S. Medical Technology Innovation: A Survey of Over 200 Medical Technology Companies. Advanced Medical Technology Association, November 2010.

10. Sony, SmartWatch 3 User Manual 2015. Cited 14.02.2015, http://support.sonymobile.com/global-en/swr50/userguide

11. Fitbit, Surge Product Manual 2015. Cited 14.02.2015, https://help.fitbit.com/resource/manual_surge_en_US

12. Sony, SmartWatch 3 2015. Cited 10.01.2015, http://www.sonymobile.com/global-en/products/smartwear/smartwatch-3-swr50

13. Fitbit, Surge 2015. Cited 21.01.2015, https://www.fitbit.com/uk/surge

14. Google, Android Wear 2015. Cited 14.02.2015, http://www.android.com/wear/

15. MobileODT, 2015. Cited 22.04.2015, https://www.mobileodt.com/mobile-colposcope.html

16. Peek Vision 2016. Cited 01.05.2016, http://www.peekvision.org

17. Sia, S. 2015. A smartphone dongle for diagnosis of infectious diseases at the point of care. Science Translational Medicine. Vol. 7 (273).

18. Columbia Engineering 2015. Cited 21.02.2016, http://engineering.columbia.edu/smartphone-finger-prick-15-minutes-diagnosis%E2%80%94done-0

19. ELISA Biocompare 2016. Cited 21.02.2016, http://www.biocompare.com/Immunochemicals/7185-ELISA/

20. Wang E. J., Li W., Hawkins D., Gernsheimer T., Norby-Slycord C. N. Patel S., 2016. Cited 01.10.2015, http://homes.cs.washington.edu/~ejaywang/documents/HemaApp.pdf

21. Wikipedia, Springboard Expansion Slot, 2016. Cited 31.07.2016, https://en.wikipedia.org/wiki/Springboard_Expansion_Slot

22. Phonebloks 2016. Cited 31.07.2016, https://phonebloks.com

23. Project Ara, Google, 2015. Cited 09.01.2015, http://www.projectara.com

24. Armbrust M., Fox A., Griffith R., Joseph A., Katz R., Konwinski A., Lee G., Patterson D., Rabkin A., Stoica I., Zaharia M., 2009. Above the Clouds: A Berkeley View of Cloud Computing. Berkeley Technical Report No. UCB/EECS-2009-28. Cited 31.07.2016, http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

25. The SOA Source Book, The Open Group, 2016. Cited 31.07.2016, http://www.opengroup.org/soa/source-book/intro/

26. Unix Programming Environment, Kernighan B., Pike R., 1984. Prentice-Hall Software Series

27. Medtronic Inc. and IBM Watson, 2016. Medronic and IBM Watson Health partner to develop new ways to tackle diabetes. Cited 06.08.2016, http://www.medtronic.com/us-en/about-3/medtronic-ibm-watson-health.html

28. IBM Center for Applied Insights 2015. Star qualities: What it takes for mobile development projects to succeed? Cited 31.07.2016, http://ibm.com/ibmcai

29. Samsung, SmartThings 2016. Cited 28.02.2016, https://www.smartthings.com

30. Volvo, Volvo on Call 2016. Cited 28.02.2016, http://www.volvocars.com/intl/own/owner-info/volvo-on-call

31. Wi-Fi Alliance, Wi-Fi Direct 2015. Cited 07.01.2015, http://www.wi-fi.org/discover-wi-fi/wi-fi-direct

32. Berkeley, BOINC, 2016. Open-source software for volunteer computing. Cited 31.07.2016, http://boinc.berkeley.edu

33. IBM, World Community Grid 2016. Cited 31.07.2016, https://www.worldcommunitygrid.org

34. Samsung, U.S. Patent 20130316684, 2013. Method for providing phone book service including emotional information and an electronic device thereof. Cited 31.07.2016, http://google.com/patents/US20130316684

35. Weiser, M. 1991. The Computer for the 21st Century. Scientific American 265 (3), 94–104.

36. Friedemann, M., Floerkemeier, C. 2010. From the Internet of Computers to the Internet of Things. Informatik-Spektrum 33 (2), 107–121.

37. Lauerma, H. 2002. Dissosiaatiohäiriöt ja niiden hoito. Lääketieteellinen aikakausikirja Duodecim. Vol. 118 (21), 2199–2205. Cited 08.10.2014, http://www.terveysportti.fi/xmedia/duo/duo93238.pdf

38. King Pigeon Communication Co. Ltd., T3 Senior Telecare System Manual, 2016. Cited 07.08.2016, http://www.gsmalarmsystem.com/UploadFiles/20150627/T3%20GSM%203G%20Telehealth%20User%20Manual%20V1.2.pdf

39. King Pigeon Communication Co. Ltd., EM-70 wearable SOS-button's description, 2016. Cited 07.08.2016, http://www.gsmalarmsystem.com/EnproductShow.asp?ID=278

40. Verkkokauppa, King Pigeon A10 GSM-vanhusvahti, 2016. Cited 07.08.2016, https://www.verkkokauppa.com/fi/product/34722/dkmvx/King-Pigeon-A10-GSM-vanhusvahti

41. Sposaro, F., Danielson, J., Tyson, G., 2010. iWander: An Android Application for Dementia Patients, Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE, pp. 3875-3878

42. Wikipedia, Hippocratic Oath, 2016. Cited 07.08.2016, https://en.wikipedia.org/wiki/Hippocratic_Oath

43. FifthElement, Products 2016. Cited 31.07.2016, http://www.fifthelement.fi/tuotteet

44. RedHat Products, 2015. Cited 14.01.2015, http://www.redhat.com/en/technologies/all-products

45. JBoss Projects 2015. Cited 14.01.2015, http://www.jboss.org/projects

46. Apache Projects, 2015. Cited 14.01.2015, http://projects.apache.org

47. Open Source Initiative, Licenses & Standards, 2016. Cited 14.08.2016, https://opensource.org/licenses

48. Raymond, E. 1999. The Cathedral and the Bazaar. O'Reilly Media, 30.

49. ICS Alert 2013. Industrial Control Systems Cyber Emergency Response Team, Alert 13-164-01. Cited 19.01.2015, https://ics-cert.us-cert.gov/alerts/ICS-ALERT-13-164-01

50. Strategy Analytics 2014. Android Captures 84% Share of Global Smartphone Shipments in Q3 2014. Cited 09.12.2014, http://blogs.strategyanalytics.com/WSS/post/2014/10/31/Android-Captures-84-Share-of-Global-Smartphone-Shipments-in-Q3-2014.aspx

51. Android, 2016. Cited 13.08.2016, https://www.android.com

52. Android Device Compatibility, 2016. Cited 13.08.2016, https://developer.android.com/guide/practices/compatibility.html

53. Google Cloud Platform, 2016. Cited 13.08.2016, https://cloud.google.com

54. Android Tools, 2016. Cited 21.02.2016, https://developer.android.com/sdk/index.html

55. Android KitKat, 2016. Cited 13.08.2016, https://developer.android.com/about/versions/kitkat.html

56. LAMP Software Bundle, Wikipedia, 2016. Cited 13.08.2016, https://en.wikipedia.org/wiki/LAMP_(software_bundle)

57. Nginx, 2016. Cited 14.08.2016, https://nginx.org/en/

58. SQLite vs MySQL vs PostgreSQL, DigitalOcean 2016. Cited 14.08.2016, https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems

59. SQL conformance, PostgreSQL 2016. Cited 14.01.2016, http://www.postgresql.org/docs/current/static/features.html

60. SQL Role Privileges, PostgreSQL 2016. Cited 14.08.2016, https://www.postgresql.org/docs/current/static/ddl-priv.html

61. PostgreSQL, Migration from MySQL to PostgreSQL 2015. Cited 07.01.2015, https://wiki.postgresql.org/wiki/How_to_make_a_proper_migration_from_MySQL_to_PostgreSQL

62. PostgreSQL, Why PostgreSQL instead of MySQL 2015. Cited 07.01.2015, https://wiki.postgresql.org/wiki/Why_PostgreSQL_Instead_of_MySQL_2009

63. PostGIS 2016. Cited 06.01.2016, http://postgis.net

64. PgRouting 2016. Cited 06.01.2016, http://pgrouting.org

65. OpenStreetMap 2015. Cited 07.01.2015, http://www.openstreetmap.org/about

66. OSG, Simple Features 2015. Cited 07.01.2015, http://www.opengeospatial.org/standards/sfa

67. Spring Project, 2016. Cited 06.10.2016, http://www.spring.io

68. OpenJDK 8 2015. Cited 19.01.2015, http://openjdk.java.net/projects/jdk8/features

69. Golang, 2016. Cited 13.08.2016, https://golang.org

70. Leaflet 2015. Cited 07.01.2015, http://leafletjs.com

71. Apache FreeMarker, 2016. Cited 06.11.2016, http://freemarker.org

72. Wikipedia, Representational state transfer 2016. Cited 29.10.2016, https://en.wikipedia.org/wiki/Representational_state_transfer

73. Android Intents, 2016. Cited 16.11.2016, https://developer.android.com/guide/components/intents-filters.html

74. Spring Security, 2016. Cited 06.11.2016, http://projects.spring.io/spring-security

75. H2 Database Engine, 2016. Cited 20.11.2016, http://www.h2database.com/html/main.html

76. MapBox, 2016. Cited 06.11.2016, https://www.mapbox.com

77. Telegram API, 2016. Cited 19.11.2016, https://core.telegram.org/api

78. GPS Performance Standard, 4th Edition, 2008. Cited 06.11.2016, http://www.gps.gov/technical/ps/2008-SPS-performance-standard.pdf

79. Android Services, 2016. Cited 06.10.2016, https://developer.android.com/guide/components/services.html

80. British Heart Foundation, 2016. Hot weather and your heart. Cited 20.11.2016, https://www.bhf.org.uk/heart-health/living-with-a-heart-condition/weather-and-your-heart

81. Google, Mobile Vision, 2016. Cited 20.11.2016, https://developers.google.com/vision

82. Android Jelly Bean, 2016. Cited 13.08.2016, https://developer.android.com/about/versions/jelly-bean.html

83. Android Lollipop, 2016. Cited 13.08.2016, https://developer.android.com/about/versions/lollipop.html

84. Android Wi-Fi Peer-to-Peer, 2016. Cited 28.02.2016, http://developer.android.com/guide/topics/connectivity/wifip2p.html

85. Bastawrous, A. 2014. Get your next eye exam on a smartphone. Cited 01.05.2016, https://www.ted.com/talks/andrew_bastawrous_get_your_next_eye_exam_on_a_smartphone

86. CIA, Long-Term Global Demographic Trends 2001. Cited 08.10.2014, https://www.cia.gov/library/reports/general-reports-1/Demo_Trends_For_Web.pdf

87. Chin, C. Linder, V. Sia, S. 2012. Commercialization of microfluidic point-of-care diagnostic devices. The Royal Society of Chemistry, Lab on a Chip, 2012.

88. Finlex, Laki potilaan asemasta ja oikeuksista 2015. Cited 11.02.2015, http://www.finlex.fi/fi/laki/ajantasa/1992/19920785

89. Google, Android 2015. Cited 05.01.2015, http://www.android.com

90. Google, Eddystone Protocol Specification 2016. Cited 28.02.2016, https://github.com/google/eddystone/blob/master/protocol-specification.md

91. Google Firebase Android Packages, 2016. Cited 13.08.2016, https://firebase.google.com/docs/reference/android/packages

92. Google, Gson Library, 2016. Cited 30.10.2016, https://github.com/google/gson

93. Google, Physical Web 2015. Cited 06.01.2015, https://google.github.io/physical-web

94. Google, Physical Web Cookbook 2016. Cited 28.02.2016, http://google.github.io/physical-web/cookbook/

95. GPX, 2016. Cited 14.02.2016, http://www.topografix.com/gpx.asp

96. IBM, Quarks Project 2016. Cited 13.03.2016, http://quarks-edge.github.io

97. IDC 2014. Smartphone Market Share, Q3 2014. Cited 05.12.2014, http://www.idc.com/prodserv/smartphone-os-market-share.jsp

98. Intel, Intel Curie Module 2015. Cited 14.01.2015, http://www.intel.com/content/www/us/en/wearables/wearable-soc.html

99. JCP 2015. Cited 19.01.2015, https://jcp.org/en/introduction/overview

100. Mann, S. 1997. Wearable computing: A first step toward personal imaging. Computer, Vol. 20, No. 2, February 1997.

101. Mieli, Finnish Mental Health Association 2015. Cited 11.02.2015, http://www.mielenterveysseura.fi/en

102. MQTT, 2016. Cited 18.09.2016, http://mqtt.org

103. OAuth 2.0 Authorization Framework, 2016. Cited 31.07.2016, https://tools.ietf.org/html/rfc6749

104. Oljaca, N. 2014. Advances in bio-inspired sensing help people lead healthier lives. Worldwide Medical Business Development, Texas Instruments. Cited 20.08.2014.

105. Pananek, J. 2015. How wearable startups can win big in the medical industry. Cited 21.02.2015, http://techcrunch.com/2015/02/19/how-wearable-startups-can-win-big-in-the-medical-industry

106. Preventice, Remote Patient Monitoring Solutions 2015. Cited 21.02.2015, http://www.preventice.com/mobilehealthsolutions/remotemonitoringsolutions

107. Sia Lab, A smartphone dongle to diagnose sexually transmitted infections. Cited 11.02.2015, https://www.youtube.com/watch?v=TC9XNqSgj4w

108. Simon, M. 2015. The Incredible Hospital Robot Is Saving Lives. Also, I Hate It. Cited 10.02.2015, http://www.wired.com/2015/02/incredible-hospital-robot-saving-lives-also-hate

109. Sony, Open Devices 2016. Cited 21.02.2016, http://developer.sonymobile.com/knowledge-base/open-source/open-devices/

110. Sotera Wireless, Wi-Fi in Healthcare 2011. Cited 28.01.2015, http://www.visimobile.com/wp-content/uploads/2011/11/wp_Wi-Fi_in_Healthcare_20110217-2.pdf

111. Strack, R. The work force crisis of 2030 – and how to start solving it now. Cited 02.12.2014, https://www.ted.com/talks/rainer_strack_the_surprising_workforce_crisis_of_2030_and_how_to_start_solving_it_now

112. Universal Plug and Play Forum 2015. Cited 07.01.2015, http://www.upnp.org

113. Ubiquitous Home Environment 2015. Cited 14.01.2015, http://fruct.org/sites/default/files/files/conference12/UbiHomeServer%20Front-end%20to%20the%20Ubiquitous%20Home%20Environment%20slide%20set.pdf

114. Mobile Augmented Teleguidance-based Safety Navigation concept for seniour Citizens 2015. Cited 14.01.2015, https://arkisto.uasjournal.fi/uasjournal_2012-2/1392-2946-1-CE.pdf