University of Hradec Králové

Faculty of Informatics and Management

Department of Information Technologies

Dissertation

Sensorial Networks embedded in Mobile Devices

Mgr. Miroslav Behan

Acknowledgement

Special thanks to my supervisor Ondřej Krejcar for acquired knowledge during my studies by writing articles and by his effective reasonable motivation based on evolutionary mind with possibility of change and brighter vision of future science in the Czech Republic.

Declaration

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources. I understand that my work relates to the rights and obligations in particular the fact that the University of Hradec Králové has the right to conclude a license agreement on the use of this work as a school work.

In Hradec Králové date 2015 Signature …..........................

Abstract / English

TITLE	SENSORIAL NETWORKS EMBEDDED IN PERSONAL MOBILE DEVICES
AUTHOR	Mgr. Miroslav Behan
DEPARTMENT / INSTITUTE	Department of Information Technologies, Faculty of Informatics and Management, University of Hradec Králové
SUPERVISOR OF THE DOCTORAL THESIS	doc. Ing. Ondřej Krejcar, Ph.D., Department of Information Technologies, Faculty of Informatics and Management, University of Hradec Králové
ABSTRACT	The modern society evolves into a sensorial network environment where individual sensor data can be transformed into cumulative and comprehensive representation for human. In a real time, it is independent of current location or behavior. The only limits to increase productivity and to create a smarter surrounding environment are personal habits and technology progress. The paper describes sensorial framework, which is dealing with the current aspects of technology, sociology, and usability in daily life usage of ubiquitous mobile devices with sensors, and arising computed and energy power. Nowadays, most of the common tasks of our lives are mainly influenced by network and social connectivity where infrastructural speed limits are provided by the information flow.
KEYWORDS	Sensor, Android, Mobile

Abstrakt / Česky

NÁZEV PRÁCE	SENZORICKÉ SÍTĚ V OSOBNÍCH MOBILNÍCH ZAŘÍZENÍ
AUTOR	Mgr. Miroslav Behan
KATEDRA / ÚSTAV	Katedra Informačních Technologií, Fakulta Informatiky a Managementu, Universita Hradec Králové
VEDOUCÍ DOKTORSKÉ PRÁCE	doc. Ing. Ondřej Krejcar, Ph.D., Katedra Informačních Technologií, Fakulta Informatiky a Managementu, Universita Hradec Králové
ABSTRAKT	Současný evoluční rozvoj mobilních osobních zařízení přináší nové příležitosti vedoucí k zefektivnění lidských činností. Společnost se z informační posouvá do roviny senzorické, kde je možné na základě aktivity jednotlivce a díky všudypřítomným senzorům extrahovat znalosti o modelech chování, které mohou být prospěšné jak pro jednotlivce, tak pro společnost. Tyto modely mohou přispět k chytré distribuci informací, kde uživatelé definují svým chováním vlastní informační světy, které v důsledku slouží ke kvalitnějšímu životnímu stylu a přispívají k zlepšení procesů ve společnosti.
KLÍČOVÁ SLOVA	Senzor, Android, Mobil

Abbreviation List

AIAI	Artificial Intelligence Applications and Innovation
ANR	Application Not Responding
API	Application Program Interface
APK	Android Package Kit
CI	Continues Integration
CRUD	Create, Read, Update and Delete
DKIM	Domain Keys Identified Mail
DSL	Domain Specific Language
GPS	Global Positioning System
HAL	Hardware Abstraction Layer
HTTP	Hyper Text Transfer Protocol
IDE	Integrated Development Environment
IPC	Inter-Process Communication
JVM	Java Virtual Machine
MAC	Mandatory Access Control
MTA	Mail Transfer Agent
MVC	Model, View and Control
REST	Representational State Transfer
SASL	Simple Authentication and Security Layer
SMTP	Simple Mail Transfer Protocol
SNR	Signal to Noise Ratio
SPF	Sender Policy Forward
SSL	Secure Socket Layer
TCP	Transport Control Protocol
TLS	Transport Layer Security
UML	Unified Modeling Language
URI	Uniform Resource Identifier
VPS	Virtual Private Server
WLAN	Wireless Local Area Network
WYSIWYG	What You See Is What You Get

Content

1. Introduction

We are facing the progressive mobile smart based reality nowadays where all life supportive informational systems are able to propagate effectiveness of human behavior on the Earth. The key for intelligent human behaving is in embracing ecological sustainable solutions which could move people from financial slavery to creative oriented future. Human needs are changing accordingly to human wisdom during each era of evolution and human dreams or desires are the most important catalyst for elevation of effectiveness in knowledge distribution throughout mankind. The knowledge itself consist of informational pieces where some parts are related to physical aspect of reality and others are relevant to relations between those solid aspects which can be gathered by mobile device sensors. The time of plain non-sensorial smart phones is more or less over, and we are facing the future environment where built-in sensors as ambient temperature, magnetic field, accelerometer, gravity, light, humidity, and others are common equipment of current mobile devices [1] and [2]. We assume sensors as basement in our discovery and as mandatory information providers for building future knowledge based sensorial informational system and as providers of physical reality aspects which the most are related to the surrounding environment for single person in current place and time. Such sensor centric view is the key to improve effectiveness of human behavior in relation to actual resources and knowledge of environment.

The history of sensors started around 1883 when the first electric thermostat came up onto market and many people consider this as the first modern, manmade sensor. "*The inventor would be Warren S. Johnson. While it might have seemed crude by the modern standards that we have today, this thermostat was able to keep temperatures within a degree of accuracy – something that's better than some of the low quality thermostats on the market today! The first motion sensor used for an alarm system came about in the early part of the 1950s, and was the invention of Samuel Bagno. His device made use of ultrasonic frequencies as well as the Doppler Effect.*" [3]

Since then the evolutionary roller-coaster of inventions came up and today we have sensors embedded everywhere for instance in smart phones, smart walls or other surrounding technology where information is used as relation between physical reality and system automation or in conjunction of visualization to human for better environment cognition or decision making. The Ubiquitous computing is modern poem which define such informational systems based on relevant inputs from physical reality and which are important for humans based on algorithms and based on knowledge to be able to make decisions automatically. These ubiquitous sensors and the computing power are important for creation effective human life support systems which can in the future facilitate population itself.

In further chapters we will go deeper into ubiquitous computing with consideration of its positive aspects for human evolution. We define goal of research [Chapter 2] accordingly to our interest of effective needs for humankind and also we define the problematic area itself over physical reality and over its property which are also relevant to our discovery [Chapter 3]. Where we go through available sensors and describe them in details with their pros and cons. Furthermore we do not forget on the recognition process from computational side and perception from human side to outline criteria and possibilities of our discovery in comparison with others related works [Chapter 4]. We define [Chapter 5] the ideal sensorial framework as blue print for design model of solution. And lastly we proceed into particular implementation to have a proof of concepts for our research [Chapter 6]. At the end we specify the areas of usage of our sensorial framework at various environments [Chapter 0] and provide discussion over discovery and results gathered from real usage [Chapter 8].

2. Goal of Research

The goal of research is about to discover sensorial based framework which provides information relevant to mobile device users in particular sensorial, location based or device relevant information. The main usage of such framework we consider at home, public and work environment where users became more productive to desired activities where sensorial framework became supportive to everyday users activities. In abstract framework we would be able to translate physical reality cognitively by embedded sensors in mobile devices and provide comprehensive information to users in terms of just in time notification message types or knowledge based messages. Sensorial data should be gathered from masses of users by non-irritating mobile device clients. Data should be sent to cloud and analyzed for environmental and behavioral patterns of individuals to provide intelligent or smart environment capabilities in surrounding environment. In the following list we outlined the main goals to explicitly provide the key points of the whole work on what we are focusing on:

- Discover mobile device embedded sensors area
- Analyze Senzoric framework which exploits embedded sensors capabilities
- Design Senzoric framework with agile development approach
- Implement Senzoric framework with state of art development techniques
- Test Senzoric framework with real users
- Consider areas of usage in real environments
- Evaluate Senzoric framework

To provide such a framework there is necessary to discover physical reality which is relevant to available sensors embedded in mobile device currently available on market. It is needed to comprehend the quality of such sensors and what can influence the gathered data in order to outline pros and cons of each sensor type. Especially found out the real factors which have to be considered as mandatory for real application in real environment and with real users. The challenge would be to design framework itself properly where effectiveness of system architecture, technical capabilities of used components and comprehensive interaction with users are key criteria which we have to consider. We suggest to be selected appropriate programing techniques, used software and hardware technology to reach goals of research inevitably. The knowledge gathered during research would be as guidance for advanced research of similar solutions.

We conclude the implementation of sensorial framework consists of technical and programing resolution with valuable information regarding to the state of art in sensorial mobile device networks useful for others to contribute or obtain summarization information in this problematic. Implemented sensorial framework will be tested in different scenarios by real users willing to provide our test cases. The results would become as bases for creation knowledge based systems with influence of global human effectiveness. Also the results will provide valuable information about possible improvements, application logic and bug fixing. The results of dissertation work would be useful for further discovery within practical usage of sensorial based networks for mobile device users.

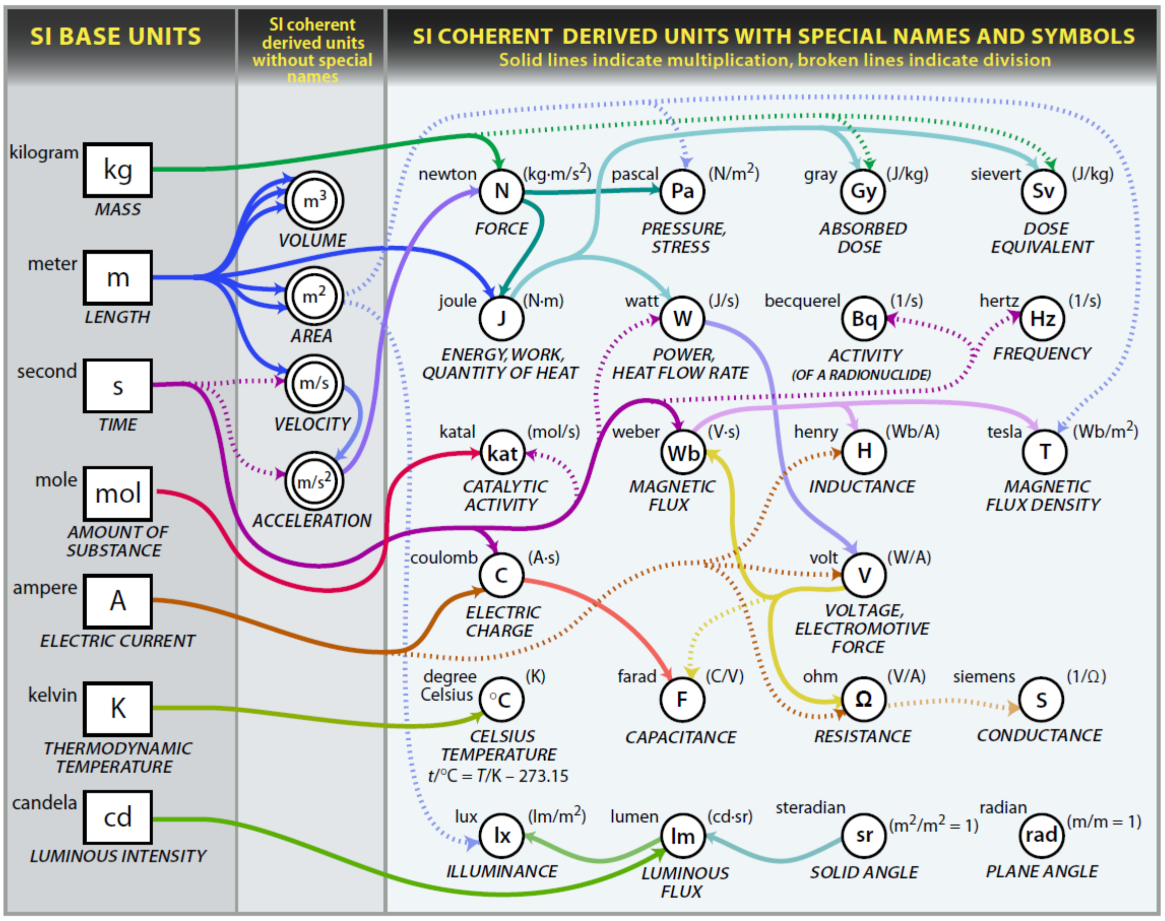
3. Problem definition

There are several areas which have to be defined first to understand core principals of reality to design the Senzoric Framework properly. At first we should know maximal information about embedded sensors in devices, what and how they are measuring, how and what it the output. Then we should consider real mobile devices which are going to be used for experiments. And then we discover how these mobile devices interact with user and environment theoretically and practically. What are the key factors which can influence our project and how. At last we have a discussion at the end of this chapter.

3.1. Physics and property of reality

We will start to define area of our discovery by physics and property of reality. We assume the sensors as convertors which measure a physical quantity and then convert it into a signal. Therefore the understanding of physical reality is relevant to sensors used in smart devices and it is important for comprehension of our possibilities and may conclude readers to vision of future sensors innovation or revolution.

As base part of physics we have to describe system of units SI [4]. The system was established in 1960 with the international support of standardization process where are defined seven basic units and all others are derived from these base units namely Meter, Kilogram, Second, Ampere, Kelvin, Mole, Candela. All other units of reality defined by mankind are derived from these seven basic ones. The relation between them is outlined in the following [Figure 1] where solid lines are used for indication of multiplication and broken lines for indicating of division from basic SI units.



SI BASE UNITS
SI coherent derived units without special names
SI COHERENT DERIVED UNITS WITH SPECIAL NAMES AND SYMBOLS
Solid lines indicate multiplication, broken lines indicate division
kilogram kg MASS
meter m LENGTH
second s TIME
mole mol AMOUNT OF SUBSTANCE
ampere A ELECTRIC CURRENT
kelvin K THERMODYNAMIC TEMPERATURE
candela cd LUMINOUS INTENSITY
m3 VOLUME
m2 AREA
m/s VELOCITY
m/s2 ACCELERATION
newton N (kg·m/s2) FORCE
pascal Pa (N/m2) PRESSURE, STRESS
gray Gy (J/kg) ABSORBED DOSE
sievert Sv (J/kg) DOSE EQUIVALENT
joule J (N·m) ENERGY, WORK, QUANTITY OF HEAT
watt W (J/s) POWER, HEAT FLOW RATE
becquerel Bq (1/s) ACTIVITY (OF A RADIONUCLIDE)
hertz Hz (1/s) FREQUENCY
katal kat (mol/s) CATALYTIC ACTIVITY
weber Wb (V·s) MAGNETIC FLUX
henry H (Wb/A) INDUCTANCE
tesla T (Wb/m2) MAGNETIC FLUX DENSITY
coulomb C (A·s) ELECTRIC CHARGE
volt V (W/A) VOLTAGE, ELECTROMOTIVE FORCE
degree Celsius °C (K) CELSIUS TEMPERATURE t/°C = T/K – 273.15
farad F (C/V) CAPACITANCE
ohm Ω (V/A) RESISTANCE
siemens S (1/Ω) CONDUCTANCE
lux lx (lm/m2) ILLUMINANCE
lumen lm (cd·sr) LUMINOUS FLUX
steradian sr (m2/m2 = 1) SOLID ANGLE
radian rad (m/m = 1) PLANE ANGLE

Figure 1 Relation SI base units [5]

After brief description of system units [Table 1], to remind such statements forming mankind behavior in many directions from economical, intellectual and social point of view and establish essential knowledge of nature so important in society development, we will step further into our discovery area. In last decade we realize to enhance mobile devices with all possible sensors which are capable to connect our applications to external reality which are surrounding us.

We are nowadays conform with situation that all basic sensors are “a must have” in our mobile phones, tablets or wearables for gathering data from physical reality and support our decision making throughout visualization comprehensive form of results. We outline all sensors available on market for daily usage by common mobile devices and their parameters and relation with physical system units in the following [Chapter 3.2].

Table 1 Definition SI base units

QUANTITY	NAME	SYMBOL	DEFINED BY
LENGTH	Meter	m	The length of the path traveled by light in vacuum in 1/299.792.458 of a second. (1983)
MASS	Kilogram	kg	The mass of platinum-iridium prototype (1889)
TIME	Second	S	The duration of 9.192.631.770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the cesium-133 atom (1967)
ELECTRIC CURRENT	Ampere	A	Force equal to 2x10-7 Nm of length exerted on two parallel conductors in vacuum when they carry the current (1946)
THERMO-DYNAMIC TEMPERATURE	Kelvin	K	The fraction 1/273.16 of the thermodynamic temperature of the triple point of water length (1967)
AMOUNT OF SUBSTANCE	Mole	mol	The amount of substance which contains as many elementary entities as there are atoms in 0.012 kg of carbon 12 (1971)
LUMINOUS INTENSITY	Candela	Cd	Intensity in the perpendicular direction of a surface of 1/600.000 m2 of a blackbody at temperature of freezing Pt under pressure of 101.325 Nm2 (1967)

3.2. Embedded Sensors

The commonly embedded sensors in mobile devices nowadays, for instance, temperature, light, gyroscope, etc. [Table 2]; are kind of raw extractors of data from external reality but we also expect to find and include complementary sensors in our analysis, such as microphone, camera, or battery indicator described [Table 3]. The way of gathering sensorial data, we identified in supported system calls dedicated to specific mobile device platform, are based on the study of literature [6], [7], [8] and [9]. We divided the sensors into two groups where the first one is with short-term and second one is with long-term data change characteristics. According to the data amount which sensors possibly producing; the first group required more frequent measurement to gather the correct samples of data for better pattern recognition, while the second group is resistant to infrequent precise measurements. The complementary sensors are more related to pattern environment recognition than behavior. From all available defined sensors, we announced which one of them are contributory to specific behavior and environment pattern recognition with brief description. Behavior patterns are the bases of motion effects of device sensors. That means recognition whether the device is worn and somehow influenced by a human body. The running motion is significant with fast location movements and periodical short shocks, while walking is distinguishable by lower shocks and slower location changes. By standing, we assume a static body position with small movement interferences, while sitting is more stable and longer lasting [10]. Sleeping behavior is recognizable in deferred device position where surrounding specific pattern noise occurs.

Table 2 Native sensors embedded in current Smart Devices [11]

TYPE AND UNIT	DESCRIPTION

ACCELEROMETER (M/S2)	Motion sensor of three dimensional X, Y, Z acceleration includes gravity magnitude which is g = 9.81 m/s2 and is expressed by following equitation:
	Acceleration = -g - ∑F / mass
	To express pure acceleration, the gravity force must be removed from data and the result is basically a linear acceleration.
	from Android 1.5 API 3, from iPhone 3G

TEMPERATURE (°C)	Environmental sensor of one dimensional ambient temperature in degree Celsius. from Android 4.0 API 14
GRAVITY (M/S2)	Motion sensor of three dimensional x, y, z vector indicating the direction and magnitude of gravity. The coordinate system is the same as acceleration. from Android 2.3 API 9
GYROSCOPE (RAD/S)	Motion sensor of three dimensional x, y, z rotation vector around device's local axis. from Android 2.3 API 9, from iPhone 4S
LIGHT (LUX)	Environmental sensor of one dimensional ambient light level indicator. from Android 1.5 API 3, from iPhone 3G

LINEAR ACCELERATION (M/S2)	Motion sensor of three dimensional x, y, z linear acceleration vector indicating acceleration along each device axis, not including gravity. The coordinate system is the same within acceleration. Assume following equation:
	Acceleration = gravity + linear-acceleration
	from Android 2.3 API 9

MAGNETIC FIELD (µT)	Motion sensor of three dimensional ambient magnetic fields measured in micro-Tesla in the x, y, and z axis. from Android 1.5 API 3, from iPhone 3GS
PRESSURE (HPA)	Environmental sensor of one dimensional ambient air pressure in hPa or mBar units. from Android 2.3 API 9
PROXIMITY (CM)	Environmental sensor of one dimensional distance measured in front of the device. from Android 1.5 API 3, from iPhone 3G
HUMIDITY (%)	Environmental sensor of one dimensional relative ambient air humidity. from Android 4.0 API 14

ROTATION VECTOR (SCALAR)	Motion sensor of three dimensional rotation vector expressed by following equation:
	Value = $\sin(\theta / 2)$
	Where θ is angle of device has rotated around specific axe.
	from Android 2.3 API 9, from iPhone 3G

That was a brief description of basic motions, and more will be described in further discovery, as dancing, watching, gym, fun, etc. All of them are based on accelerometer, gyroscope, rotation, or magnetic field. Another point of view is environmental-based resolution which is more limited and relies on external resource and social or network collected group knowledge. We defined the basic environment as home, work, transportation, or others which are less statistically probable. The home environment is recognizable as a place located overnight staying with the most count of occurrences in time. Work environment is a place located over recognizable specific equal time duration consumption over an awaken user state and most likely during a day in a different location than home. The transportation by car, plane, boat, train, bike, etc., is recognizable by the speed of a device, respectively, measured by location differences. Therefore, environment recognitions are essential location sensors and supportive sensors as humidity, temperature, pressure, light, and charging indicator.

Table 3 Complementary sensors derived from Smart Devices

TYPE AND UNIT	DESCRIPTION
GPS (LATITUDE, LONGITUDE)	Location sensor of Global Positioning System where the minimal three different satellite signals are required [12]. The coordinates are calculated by trilateration [13] but outdoors only.
CELL GSM (DB)	Location sensor of Global System for Mobile communication signal measuring where coordinates are estimated over multilateration [14], [15] within outdoors or indoors.
WIRELESS (DB)	Location sensor type, where signal of access point results within external knowledge base in dynamic location monitoring.
MICROPHONE (DB)	Environmental sensor of noise level where a Signal to Noise Ratio (SNR) technique is appropriate for environment evaluation.
CAMERA (PIXEL)	Alternative way of a light sensor, but with another applicability, as face or object recognition is out of energy scope.
BATTERY CAPACITY (%)	Sensor for smart reminding to charge in specific environment and smart battery management. Power plugin (bool) In building environment recognition.

In short [Table 3] above are highlighted complementary sensors which are capable to consider as everyday usage information provider which benefits to quality of gathered data and increase correctness of automation decision in supporting application. In the following chapter we outlined the mobile devices it selves as mobile personal computer unit which is able to gather and analyze sensor data in real time and provide consistent information into cloud or other parties.

3.3. Mobile devices

Cellphones, smart phones, mobile devices and others names those are synonyms for something quite new or do we think it is just old well known personal computer but just mobile? Well that is a good question for each one of us. Someone can tell that these are not the same old crap machines; someone can tell we have still our mobile computers with us on every step. Nevertheless the main reason of arise of mobile devices was actually simply communication and because human is moving mammal the science has to come up with something small and handy what has speaker and microphone and also with some human interface based on simple press buttons. After while the internet as new information and communication medium forced this technology to adopt itself and then came into existence a mobile devices with arm processors and big touch screens capable to handle much more variety of user tasks.

At first we need to analyze what mobile devices are available on the market and which operation system suits for our discovery purposes, therefore we outlined in the following [Figure 2] graph data of smartphones OS market first to announce that the Android OS is very one suitable mobile device operation system which we need to use regardless to others well known operation systems as iOS or Windows Phone.

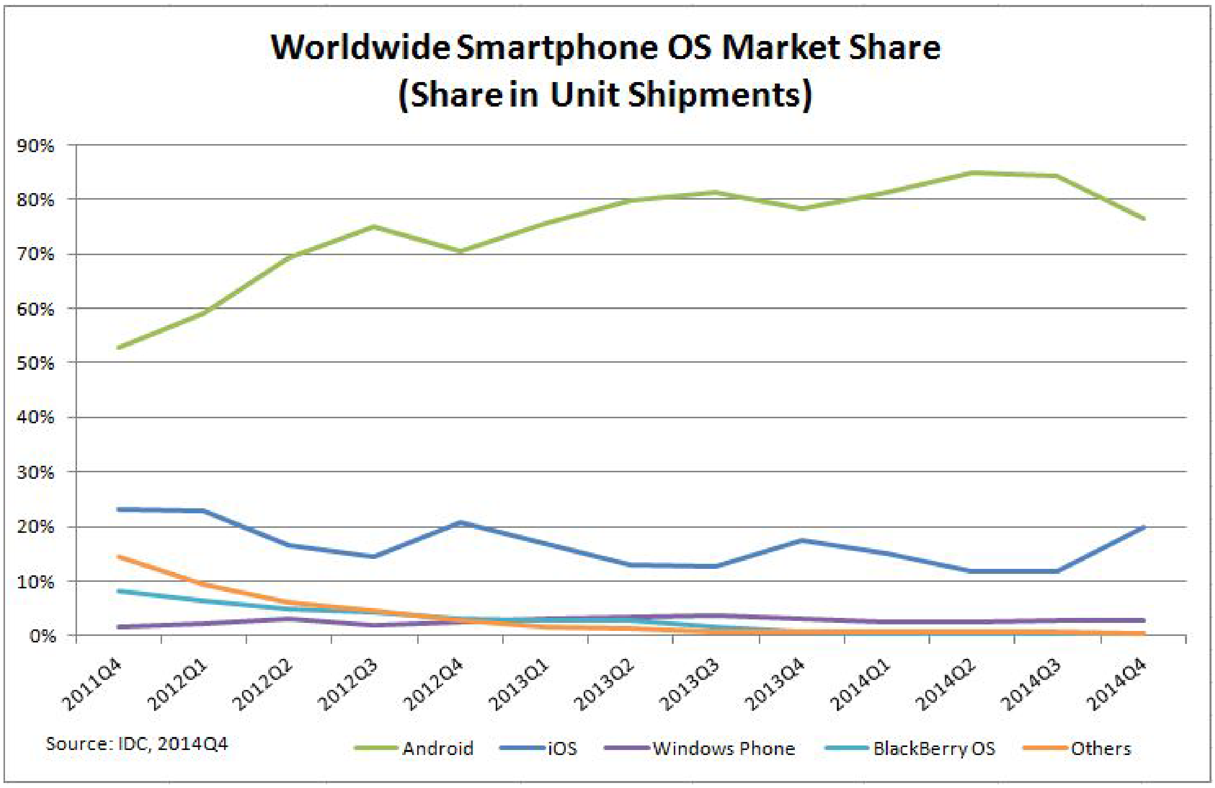


Figure 2 Smartphone OS Market Share [16]

Once we are convinced by the market which platform is the most world widely spread out we assumed that it is a right choice speaking for Android platform. Another argument is open source available on Android Source [17] which can be considered as a main reason for some development projects. Next supportive argument for Android is that it is Java based programing language. The last but not least argument why the Android is the winner one in category of the mobile device operation systems is simply because of its open architecture. Developers are able to publish platform changes independently and push application on the Android market without censorship. Once there is any functionality developed in one application it is possible to be reused by other application by magical intent handling. Therefore it is possible to introduce application community development which actually speeds up and improves the usability in total application on this platform.

As far as we now know that the Java and Android OS is preferred our choice in discovery we also have to use real mobile devices and tablets defined by exact model and brand available on market the nowadays. Also another consideration which has to be taken into account regarding mentioned openness of platform we have to use mobile devices models which actually support customization from the beginning. It means that we have to be able to flash the operation system image by customized the Android source code. There are supported models from Google Company which enable such customization of operation system such as Nexus 7, Nexus 10, Galaxy Nexus, Motorola Xoom, Panda Board, etc. The whole list of devices is available on Android source code [17] in section of customized operation system images. Also to decide what devices are the most relevant and suitable for our discovery we based our choice on the very top of worldwide used mobile devices in last year 2014. Another source is top sales Mobile Phones in the category of personal usage on daily bases world widely [18]. The top most seller Android based mobile device is Samsung Galaxy S4, Samsung Note 3, Samsung Galaxy S4 mini and Xioami Hongmi Redrice from sales numbers February 2014 therefore we would use at least one Samsung device. Another mobile devices representative we also consider Sony Xperia M and Samsung Mini 5. In terms of use of tablets we have to consider relevant Android tablet device which would be Nexus 7 (Wi-Fi) because it enables customization operation system build. Next tablet with the best power capacity of battery is Sony Xperia Z2 also announced as the best product of the year 2014-2015 of EISA Awards. In the following [Table 4] we are providing the list of parameters and features of these mobile devices and tablets to have clear comparison picture of used devices.

Table 4 Comparison of mobile devices [19]

MODEL	SONY XPERIA M2 DUAL	SAMSUMG S5 MINI	TABLET SONY XPERIA Z2	TABLET NEXUS 7
FRONTSIDE				
WIDTH	71.14 mm	64.8 mm	266 mm	200 mm
HEIGHT	139.65 mm	131.1 mm	172 mm	114 mm
THICKNESS	8.64 mm	9.1 mm	6.4 mm	8.65 mm
WEIGHT	148 g	120 g	426 g	290 g
CPU	ARM Cortex-A7	ARM Cortex-A7	Krait 400	Krait 200
CPU CORES	4	4	4	4
CPU FREQUENCY	1200 MHz	1400 MHz	2300 MHz	1500 MHz
GPU	Qualcomm Adreno 305	ARM Mali-400 MP4	Qualcomm Adreno 330	Qualcomm Adreno 320
GPU CORES	1	4	4	4
GPU FREQUENCY	450 MHz	450 MHz	450 MHz	400 MHz
RAM CAPACITY	1 GB	1.5 GB	3 GB	2 GB
ASPECT RATIO	1.778 16:9	1.778 16:9	1.6 16:10	1.6 16:10
RESOLUTION	540 x 960 pixels	720 x 1280 pixels	1920 x 1200 pixels	1920 x 1200 pixels
SENSORS	Proximity Light Accelerometer Compass	Proximity Light Accelerometer Compass Gyroscope	Light Accelerometer Compass Gyroscope	Proximity Light Accelerometer Compass Gyroscope

		Fingerprint Heart rate		
WI-FI	802.11a 802.11b 802.11g 802.11n Wi-Fi Hotspot Wi-Fi Direct	802.11a 802.11b 802.11g 802.11n 802.11n 5GHz Dual band Wi-Fi Hotspot Wi-Fi Direct	802.11a 802.11b 802.11g 802.11n 802.11n 5GHz 802.11ac Dual band Wi-Fi Hotspot Wi-Fi Direct	802.11a 802.11b 802.11g 802.11n 802.11n 5GHz Dual band
CAPACITY	2300 mAh	2100 mAh	6000 mAh	3950 mAh

The most suitable for our discovery is mobile phone with limitless battery and with all sensors possible to be embedded in a single device. But reality is far away from perfectness for our needs and as such we rely on mobile devices which are reasonable on price and also which combines desired parameters with maximal battery capacity and also are in major stream of world widely usage. We consider these devices are only relevant in developments phase and after deployment phase we outline the cross region model representation globally based on real use of Sensorial Framework.

3.4. Usability of embedded sensors in cloud based services

Sensors embedded in mobile devices have additional value in our research. The most important is the influence of power consumption which is given appropriately by specific mobile device based on the process of prioritization and real hardware consumption. Some mobile devices operation system control power consumption provide better customer experience where in case of minimum energy left most of the sensors are powered off. Actually if customer explicitly says that the internet connection or GPS sensor reading is necessary for usage of the system then the device cannot be switched off for battery saving. We assume the sensors which are commonly embedded in nowadays mobile devices as base information channel and according to the current possibility in mobile device segment of sensors we consider available functionalities and capabilities. Basically framework solution has to be capable to involve future types of sensors which will evolve in terms of power consumption. And also we have to consider that any physical characteristics have to be possible to be described in the future therefore we have to keep in mind such functionality in our sensor framework.

3.4.1. Problem of Connectivity and Data Management

The connectivity is a basic factor which influences speed and power consumption of transmitted data where the aggregation ratio between raw sensorial data and representational device state information implicates to deliver a reasonable data stream, which competes to be the most effective informational dataflow in time, energy consumption, and expected system functionality. We consider the minimal granularity of sensor's transmitted information dedicated to maximal network throughput. All it depends on mobile device connectivity over well-known standards such as General Packet Radio Service (GPRS), Enhance Data rates for GSM Evolution (EDGE), Universal Mobile Telecommunication System (UTMS), High Speed Packet Access, Wireless Local Area Network (WLAN), or Worldwide Interoperability for Microwave Access (WiMAX). We also see considerable differences between Quality of Service according to a network type or current distance between mobile device and network access point.

3.4.2. Problem of Networking

We focus on wireless networks where data transfer rate fluctuates more than on wired networks; therefore, there is ongoing challenge with data optimization that has to be considered in application development depends on location and provided technology. Nowadays, wireless networks are being challenged by increasing amount of users and data application dependency. That is why we test the performance of current wireless connectivity [20]. The measurements are provided over Transport Control Protocol (TCP). Backend server is located depending on the tested technology that means using Virtual Private Server (VPS) with backbone connectivity for mobile networks, and using local network server for wireless fidelity (Wi-Fi). The results are outlined in [Table 5]. There are theoretical and practical capabilities according to the types of networks which serve to smarter definition of a development data management concept. Bandwidth emphasizes the possibilities of data amount stream, the distance range recalls the possibility of locational change possibility during transmitting, and measurement overview real data statistics of latency; upload and download provided by Android mobile devices ZT3-Blade by application available on Google Play. The data acquired depend also on the mobile network provider and its implementation of network standard. Therefore, the test results could be different. In addition, the environmental and distance factors influence the network connectivity. For further discovery, we expect more data from crowd with more statistical results according to countries. The test application, where latency, the download and upload measurements, is provided in basic testing flow triggered by a user on the start button. The principles of testing are to disable all remaining network traffic and to locate the server to the closest location to test the device.

We provide only one server in Prague, so the latency could be influenced by the maximal round trip around the world, which is in the worst case around 200 ms and around the Europe 30 ms. The testing process starts from a client who is sending Transport Protocol Packets with minimal size 1B containing urgent processing flags and waits until the response is received from the server, where the time dilation is a real Round Time Trip. The download is requested from the client after latency timeout. The server produces the maximal buffer size output stream which client receives in specific amount sizes, estimates speed as time dilation rate, and receives sizes supported by the average value for fluent result flow. The upload is the same on the server side just when the server is sending back to client a 4B packet with the speed encoded from integer value. The results were gathered from approximately 400 mobile devices by downloading applications from market. We expect to elaborate the statistics over time and cover all available network types in further discovery.

Table 5 Wireless network limits for mobile devices

TECHNOLOGY	BANDWIDTH (D/U MBPS)	DISTANCE RANGE (KM)	LATENCY (MS)	DOWNLOAD (KB/S)	UPLOAD (KB/S)
EDGE (2,5G)	1.3/0.6	10-35	100-1200	3-10	2-5
UTMS (3G)	28/11	5-30	50-500	20-50	10-40
HSDPA (3,5G)	42/12	5-30	-	-	-
LTE (4G)	100/50	5-30	-	-	-
WI-FI	10/54	0.001-0.5	10-100	400-1000	300-900
WIMAX	46/4	5-10	-	-	-
BLUETOOTH	2/2	0.001-0.005	-	-	-

3.4.3. Problem of Battery Management

All sensors are consuming energy to provide measurements and with mobile devices it is especially real challenge to maximize the duration of usage of device without charging. We assumed the energy consumption for localization is the most effective within Global System for Mobile communication (GSM) cell bases evaluation [21]. The energy cost is lower than <20 mW and is followed by WLAN which have around 500 mJ and finally Global Positioning System (GPS) sensor is the most consumer of energy more than >1,000 mJ dependable on mobile device. Therefore, the sensors used for location estimation have to be considered in order relevant to energy efficient factor. Instead of that the accelerometer, gravity, magnetometer have quite minimal energy consumption and sampling of those sensors is not that painful in terms of power management. In the following [Table 6] are measured energy consumptions of Samsung S2 smartphone sensors.

Table 6 Sensors energy consumption Samsung S2 [21], [22]

SENSOR	SAMPLING	IDLE	SWITCH ON / OFF
ACCELEROMETER	21mJ	-	-/-
GRAVITY	25mJ	-	-/-
MAGNETOMETER	48mJ	20mJ	-/-
GYROSCOPE	130mJ	22mJ	-/-
MICROPHONE	101mJ	-	123mJ/36mJ

We can conclude from measurements the efficient sensor for motion detection of smart phone which is accelerometer [22] that means when we need to trigger any action when mobile device start to moving the listening of accelerometer sensorial data is sufficient.

3.5. Discussion

All previous chapters are showing the problems in our discovery area in more details and we can now conclude the suitable solution for our goal of research based on gathered knowledge. We are able to provide effective solution in terms of battery consumption of mobile devices, available network connectivity and necessity of provided information. We consider sensorial informational flow in following [Figure 3] as basement of our Senzoric Framework (SF) where information are delivered effectively across internet and provides end users or external informational system sensing data of user behavior.

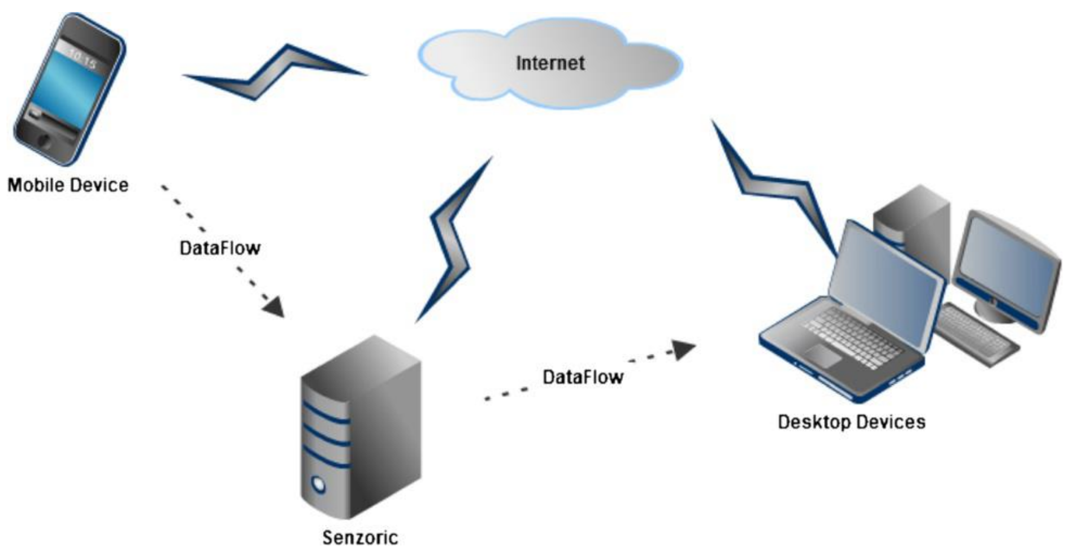


Figure 3 Sensor Informational Flow of Senzoric Framework

All possible sensors of mobile devices are producing a large amount of data in total which are not necessarily to be transferred over network and reasonably shared in a real time. That is why we could consider most of them in raw form as redundant or without meaningful informational value. Also, processing of sensorial values with maximal refresh data rate would be unreasonable in terms of battery management or use of non-real-time system bases. The key to sensorial networks, where mobile devices are an essential part, is in the balanced design of outlined high-level architecture [Figure 3]. There, the device is producing a meaningful amount and type of sensor's data, which depend on current network connectivity and are required to measure representational device states.

To provide the maximal efficient informational stream between the entities of system, we have to consider data granularity and sampling intervals of transmitting data. The reason is to minimalize energy consumption of mobile devices and to maximize provided informational value to our SF. We assume the maximal granularity of sensor's information where the defined behavior and environmental patterns are over time period recognized by mobile device and transmit only a minimal amount of data through network. We outlined defined patterns in [Table 7]. The minimal transition bundle of information has to consist of pattern identification (4B), timestamps (8B*2) of starting and ending time of recognition, location (8B*2) of latitude and longitude and the last device id or temporary token (16B). Therefore, a real-time data transmission has a minimal size (52B) in online mode delivery. We also consider an offline mode for data flow, where the lists of recognition states are transferred over network that is demanded by remote request or periodically uploaded to data distribution server by time period or buffer size limit. The data flow type of service is according to battery management limited in real-time processing; therefore, the historical and current data are flushed to server only if required, with possibilities of in time authorization. Otherwise, in case of real-time processing requirements where the predefined authorization variant exists, the data flow depends on the point of interest and on the status change events with maximal battery effectiveness of sensor data gathering.

Table 7 Activity and environment context

STATES	SENSORS	SAMPLING (HZ)	DURATION (S)
ACTIVITY – WALKING	Accelerometer	20	3 seconds
ACTIVITY – RUNNING	Accelerometer	30	2 seconds
ACTIVITY – SLEEPING	Microphone	1	20 seconds
ACTIVITY – STANDING	Accelerometer	10	3 seconds
ACTIVITY – SITTING	Accelerometer	5	3 seconds
ACTIVITY – DRIVING	Accelerometer, Localization	1	5 seconds
ENVIRONMENT – HOME	Localization, State	0.1	1 week
ENVIRONMENT – CAR	Accelerometer, Localization	1	10 seconds
ENVIRONMENT – WORK	Localization	0.1	1 week

Other aspect, which would be taken in consideration, is network availability where the online mode is in fact commonly expected during the day, but it may also occur on specific occasions when the mobile device is temporarily or in the long-term without network connection. In that case, we expect to continue saving locally the desired sets of sensorial information for correct results in the objective of large scale. Therefore, the data could be stored on the device locally with optional network batch upload performed upon intelligent upload mechanism.

4. Related works

Once we declared the core problems in previous chapter we are now ready to start searching related works in this area. There are plenty of papers which are dealing with sensor based computing but we have to constrict our view only on those which have something in common with mobile devices. Such set of works are rapidly decreased but still we are talking about thousands of articles and works. Therefore we consider other criteria of search and that is framework and informational system related applications. Now we have reasonable amount in our sets which are mainly oriented onto context-awareness area works. We consider such frameworks aiming our area of discovery which are outlined in the following [Table 8] and be part of our comparison analysis.

Table 8 Related works

ID	NAME OF RELATED WORK	TYPE
[7]	(FTW) Feel The World: A Mobile Framework for Participatory Sensing	Sensor development framework
[9]	(MobiSens) A Versatile Mobile Sensing Platform for Real-World Applications	Sensor development platform
[23]	(SensLoc) Sensing Everyday Places and Paths using Less Energy	Location based service
[24]	(ODK) Open Data Kit Sensors: A Sensor Integration Framework for Android at the Application-Level	Sensor development framework
[25]	Implementation of a smartphone sensing system with social networks	Social location aware mobile application
[26]	An activity recognition system for mobile phones	Activity recognition system
[27]	Logging user activities and sensor data on mobile devices	Environment recognition application
[28]	(InContexto) Multisensory architecture to obtain people context from smartphones	Activity recognition system
[29]	Activity recognition for risk management with installed sensor in smart and cell phone	Activity recognition application

We describe in more detail the first four related works in separate subchapter and the rest of related works are summarized in the last subchapter because they are the most closed to our desired goals of

Sensorial Framework. In the last chapter we discuss comparison of works with brief functionalities overview.

4.1. Feel the World Framework

This related work provide embedded sensor middleware with ability for the third party programmers to develop application that enables people to sense, visualize and share information about surrounding environment. This middleware platform is called Feel the World (FTW) and the key contributions of work can be summarized as follows [7]: *"An open source framework for developing people-centric sensing Android applications. Through FTW, developers would be able to exploit and configure all the embedded sensors of mobile phone as well as external sensors. The general configuration properties of FTW are the sampling rate, the duration of each data collection, the priority of data and the running environment (background/foreground). Additionally, developers can specify whether or not the data will be uploaded on a server and how often this will take place. "*
The framework proposing system architecture based on Android SDK and Java Runtime Environment (JRE) with possibility to download source code. The framework introducing universal embedded and external sensors handling based on objective data type BaseSensor outlined in the following [Figure 4].

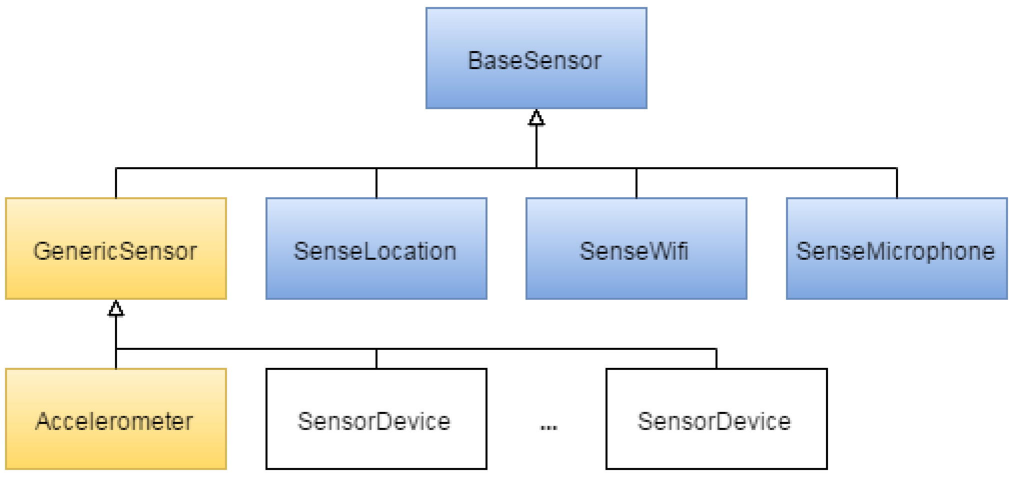


Figure 4 FTW Sensor Class Hierarchy [7]

The FTW allows extending the BaseSensor hierarchy by inherited classes implemented by external developers giving the framework slightly more flexibility for improvements to others. The data itself are stored on mobile devices by default as CSV values in different files. Such implementation is not sufficient as data access where we might consider patterns recognitions based on gathered sample on mobile devices itself of course where it makes sense. The core architecture is outlined in the following [Figure 5] where are all components of the system dedicated to the specific role. The data layer on the client side is responsible for foreground and background data collection as well as external sensors pairing and collection. The computational layer consists of monitoring service and file writer responsible for storage data on mobile device with decision maker in cases such as battery level is below 20%, the sensor sampling is stopped itself.

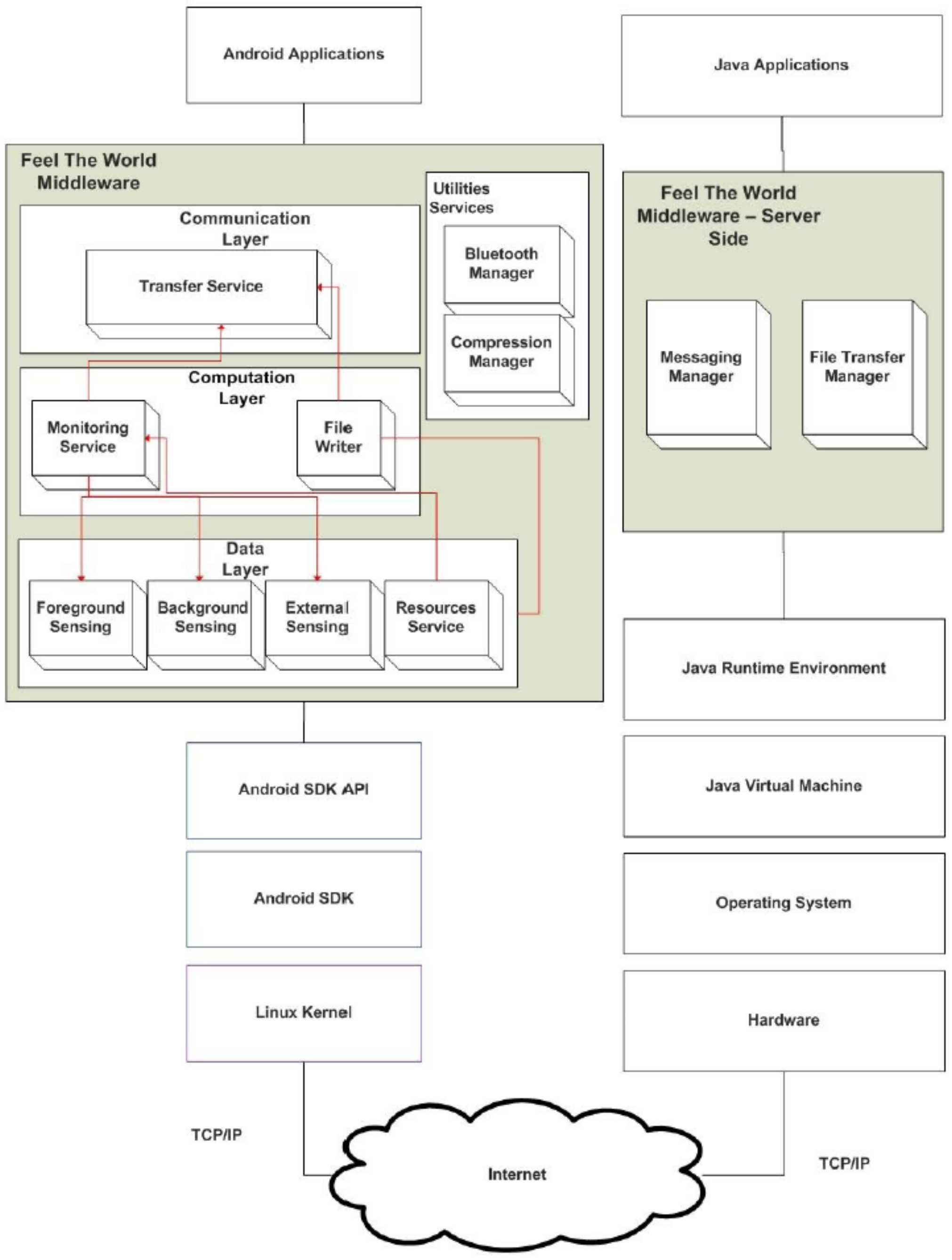


Figure 5 FTW System Architecture [7]

Communication layer is responsible for uploading datasets to server with consideration of network throughput and therefore when Wi-Fi is connected the files are automatically being transferred in maximal speed. And last one on the client side are the utilities services such Bluetooth and Compression manager. The work using compression algorithm Deflate which is the most energy-efficient for compression rather than Huffman coding, combination Huffman and Run-Length or Zip compression.
The FTW work proves by its implementation possibility of the third party support in development process and proposed several optimal sensor data gathering scenarios in background mode which are effectively worked. But there are couple of issues which may be considered in further discovery which is resolving our solution such as security aspect, client side pattern recognition based on knowledge base, effective data storage based on database instead of flat files. Some of pros and cons are outlined in the following summarization as overview to extract possible improvements for our work:

- Pros
  - Extensible potential together with the third parties.
  - External sensors for mobile devices support.
  - Resource monitoring - User friendly policy based on battery capacity, compression tools.
- Cons
  - Local storage CSV on the mobile device.
  - Static data (process, clock, ram) vs Dynamic data (battery level, processes) categorization.
  - No security consideration, no Social networking possibilities.

4.2. MobiSens Platform

The versatile mobile sensing platform for real world application where common requirements of mobile sensing application depend on power consumption, activity segmentation, recognition and annotation based on description provided by group of motivated users who provide activity labels. The framework proved over the time the usability of several applications with auto-segmentation and auto-recognition features which increased the usability of the whole framework. In short by their own achievements is following [9]: "*Based on the MobiSens platform we developed a range of mobile sensing applications including Mobile Lifelogger, SensCare for assisted living, Ground Reporting for* *soldiers to share their positions and actions horizontally and vertically, and CMU SenSec, a behavior driven mobile Security system."* Therefore we are delighted to analyze pros and cons of their solution based on the user behavior patterns to find out some missing gaps. At first we recognize MobiSens system architecture [Figure 6] which is based on the client/server consisting of three main parts such as mobile application which collects sensors data, apply activity segmentation with light-weight algorithms, backend system first tier where data are indexed and processed with heavy-weight algorithms and second tier with application and services.

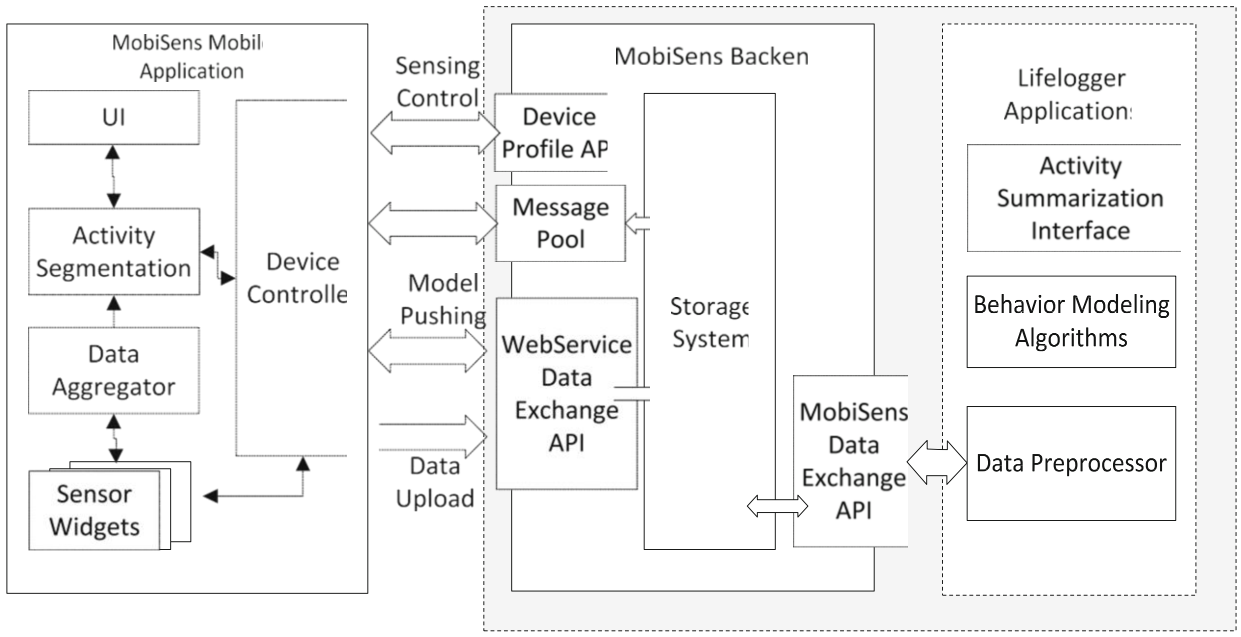


Figure 6 MobiSens System Architecture [9]

Once we understand the client architecture we can assume some advantages of solution in simplification of message based processing of sensors data and using separate sensor widgets but on the other hand there are some implementation disadvantages based on the raw file storage system of sensors data which cannot provide online processing on the client side in case of independent mobile sensor storage provider. The database sensor data handling could be improved in pattern recognition processing to be able to provide for the third party applications sensorial content. Nevertheless we consider implementation of sensing profile pulling as innovative approach where all settings of mobile client application are not tight to application release itself but rather are maintained on backend server and can be pulled to mobile client application upon request. The sensing profile consists of list of sensors which needs to be sampled, sensor sampling rate, sampling strategy, data push intervals and others to provide users maximal efficiency in terms of battery consumption and activity recognition.

On the backend side we consider as improvement database type of sensor data handling together with transmission provided rather than raw files by JSON based format. There are couple of reasons speaking for JSON format such as native implementation available on the most clients and together with GZIP compression creates the effective and fast solution with less power consumption which can be used for data transport. The most significant contribution of MobiSens platform is covered by activity classification based on Hidden Markov Models (HMM) together with adaptive activity recognition based on user annotation interaction with sophisticated User Interface (UI) where end users are willing to annotate their unknown activities. We summarize pros and cons in following points:

- Pros
  - Behavior-based anomaly detection, behavior-driven passive authentication, future activity detection, adaptive activity recognition algorithm
  - Sensing profile pulling from server
  - Real world application implemented based on MobiSens platform
- Cons
  - Raw data file handling of sensor data
  - Database support vs raw data files of sensor data content handling
  - Users privacy & security concerns

4.3. SensLoc Location Service

We conclude the location based services as core functionality of sensorial framework. The SensLoc location service covers innovative approach on mobile device to result the location with minimal battery consumption. Namely authors telling us following: "*SensLoc comprises of a robust place detection algorithm, a sensitive movement detector, and an on demand path tracker. Based on a user's mobility, SensLoc proactively controls active cycle of a GPS receiver, a Wi-Fi scanner, and an accelerometer. Pilot studies show that SensLoc can correctly detect 94% of the place visits, track 95% of the total travel distance, and still only consume 13% of energy than algorithms that periodically collect coordinates to provide the same information.*" Therefore we consider such work as relevant benefit to our discovery since location resolution is mandatory for advance environment and behavior pattern recognition. The proposed service declares the optimization for location resolution gathered from sensors Wi-Fi, GPS and accelerometer with advance technique for indoor location since commonly people spent approximately 89% of their time indoors and only 6% outdoors [30]. The concept of system architecture is illustrated in the following [Figure 7].

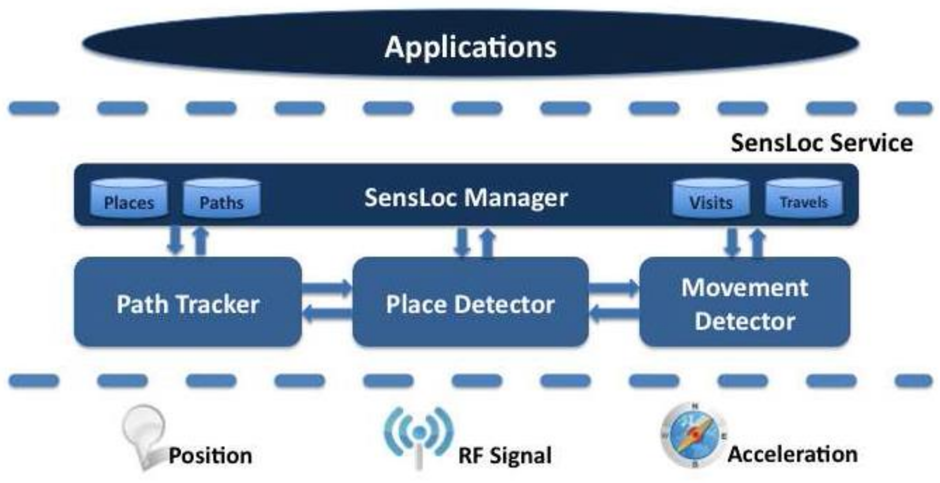


Figure 7 SensLoc System Architecture [23]

Service consists of several parts namely Movement Detector, Place detector and Path tracker responsible of each area. The place detection is provided by scanning surrounding area with Wi-Fi Access Points (APs) and by filtering, finger printing and Tanimoto coefficient based algorithm where entering place is detected by increasing current Wi-Fi signal vector maximum threshold and departure place is oppositely detected by decreasing values down to minimum threshold. The movement detection is bound by accelerometer sensor where magnitude is computed over all tree axes with intelligent double sizing scan interval to reduce battery consumption. Lastly the path tracking is provided by GPS sensor monitoring which is invoked by departure place detection. In case of speed is over threshold 2 m/s the Wi-Fi scans are turned off to save energy and turned back on when speed is under threshold means the user may be entering a new place. We consider couple of ideas as contributory to our work and useful and the pros and cons are outlined in following points:

- Pros
  - Energy efficient location based service which overcomes other approaches
  - Innovative algorithm for Place, Path and Motion detection
- Cons
  - Missing blacklisting Wi-Fi signature based on past experience
  - Server data synchronization is not enabled

4.4. Open Data Kit Framework

Once we are decided to develop application which works with external mobile device sensors connected via Bluetooth or USB this related work Open Data Kit Framework come a handy partner. The authors describe their framework as: "A *framework to simplify the interface between a variety of external sensors and consumer Android devices. The framework simplifies both application and driver development with abstractions that separate responsibilities between the user application, sensor framework, and device driver".* Therefore we acknowledge this framework as modular to adding new sensors into the system with isolation between applications and sensor-specific code. There is available single sensing interface for external and internal sensors to provide low-level abstraction of sensor communication. Typically applications can directly communicate with sensor manager over standard Android service interfaces or content providers. In the following [Figure 8] is outlined architecture of ODK framework for application developers.

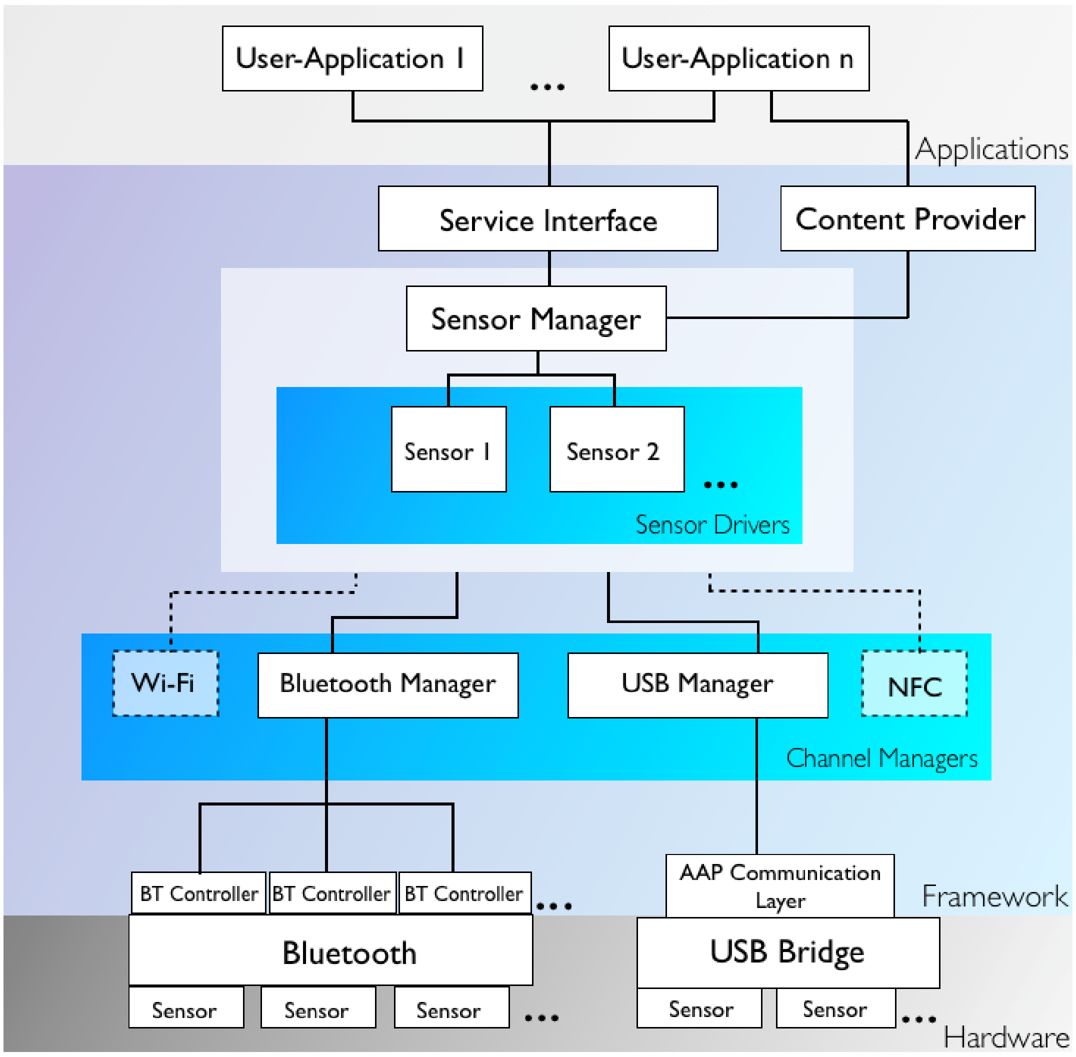


Figure 8 ODK System Architecture [24]

We consider to use this framework and such principals in case of external sensors usage which is currently out of scope of our discovery but can be considered as future work and innovation. The pros and cons are as follows:

- Pros
    - Universal driver support and development
    - Dynamic sampling
    - Android Binder IPC
- Cons
    - Sensor driver has to be explicitly implemented for framework

4.5. Other works

Activity recognition system [26] was developed as a real time monitoring system for mobile devices that embrace neuronal network motion pattern recognition by body accessories, wirelessly connected over Bluetooth to provide sensorial data. The proposed solution requires additional components outside of the mobile device and additional energy consumption for local communication [31]. Another related work [8] considers providing sensorial information by context-aware web browsers which are able to merge web application tags with current mobile device sensorial data but only as a foreground process invoked by user actions, and therefore, for smart environmental-behaving solutions, it is insufficient. On the other hand, InContexto [28] is a solid work that represents background service with still, walk, and run recognition with 97% accuracy, but the only sensor used for gathering these data is accelerometer which is not enough for larger scale of behavior and environment recognition. For environment recognition and connected activities, where data are being minded in a recorded log, the case is to predict or analyze future patterns or to recognize the environment as a great work [27] where the point of view of soft sensors is taken into account, but the hard sensors are considered as a future possible improvement. Another interesting point of view, from activity recognition algorithm classification [29], provides the accuracy comparison and results with 97.7% measured by a mobile device in a pocket but with the fact that the tested data are gathered with high sampling frequency which leads to higher battery consumption. Related works [26], [27], [28] and [29], created to compare the needed solution for the cross features and also to define a more precise goal of developing the application and to define specific requirements. According to the demonstrated summary [Table 9], it is hard to find a complex solution around all the mentioned solutions. This fact leads us to design and develop a new architecture as well as to implement some parts to provide a basic evaluation of the proposed solution.

Table 9 Comparison Context Aware Works

FEATURE	[26]	[27]	[28]	[29]	REQUIRED
SAMPLING DATA/HZ/SEC	1/50/2	-	-/0.6-2/5	6/16/4	1-7/5/5
SAMPLING WINDOW SIZE	100	-	512	384	100
SENSORS	1	1-15	2-10	6	1-7
BACKGROUND PROCESSING	Yes	Yes	Yes	No	Yes
EFFECTIVE BATTERY CONSUMPTION	No	Yes	No	No	Yes
CONNECTIVITY	No	No	WSDL	No	REST / UDP
EMBEDDED SENSORS	No	Yes	Yes	Yes	Yes
PATTERN RECOGNITION TYPE	Neuronal	-	SMA*	SVM**	Intensity
ACT. / ENV. RECOGNITION	Yes/No	Yes/Yes	Yes/No	Yes/No	Yes/Yes

**Signal Magnitude area **Support Vector Machine*

There are others relevant works as Jigsaw [32], Funf [33] or SociableSense [34] which covering part of system features where Funf was even bought by Google due to exceptional design and sensorial unifying data approach. We can use them as think tank for some innovative ideas which we have to resolve during design or implementation of our Senzoric Framework.

4.6. Discussion

Throughout all related works analyzation we consider really contributory many ideas and solutions of problems mentioned. In the following [Table 10] we outlined core features which are relevant to our discovery across others related works. We consider using only embedded sensor in our SF and external sensors connected to mobile device are in scope of future work. Server data synchronization is basement of spreading out dataflow to multiple clients and we consider such feature also as mandatory. Dynamic sampling is required to adjust sensors usage in terms of effectiveness and battery consumption. Decision module on the client side is one of the feature which supports intelligent data transmission and pattern recognition.

Table 10 Sensorial Frameworks Comparison

CORE FEATURES	FTW [7]	MOBISENS [9]	SENSLOC [23]	ODK [28]	REQUIRED
EMBEDDED SENSORS	ALL	ALL	Wi-Fi Accelerometer GPS	ALL	ALL
EXTERNAL SENSORS	YES	NO	NO	YES	NO
SERVER DATA SYNC	YES	YES	NO	NO	YES
DYNAMIC SAMPLING	YES	YES	NO	NO	YES
DECISION MODULE	YES	YES	NO	NO	YES
SECURITY	NO	NO	NO	NO	YES
THIRD PARTIES	YES	YES	NO	YES	YES
SOCIAL CONNECTORS	NO	NO	NO	NO	YES

The related works do not include all of our desired goals and functionality therefore we assumes as really contributory to design and implement such sensorial framework which provides sensorial information related to mobile devices in comprehensive form on mobile device itself or externally in other devices.

5. Design of Solution

In this chapter we define the full model of our solution of Sensorial Framework. To be able to easily and fast develop such solution we decided to use agile techniques which will be described in the first sub chapter. Then we will proceed to definition of our goals with description and considerations about possibilities of our sensorial framework. Then by increasing granularity of our goals we convert ideas into user stories where we would reach the level of project model definition and basic architecture of system as activity flow, class and data model. After we consider some security aspects and possible security threats and lastly we configure deployment model of the system in terms of end user usages. We also describe in minimal details techniques or tools which we are using for modeling and analysis. We will prefer more agile approach for modeling information system and as such less move on modern agile approach.

5.1. Agile development background & used tools

During last decade the software development evolved into agile approach which seems to fit more to the human concept of thinking with more flexibility instead of old schema as waterfall type of development. The agile approach is a type of software development where requirements and solutions evolve thought cross functional teams which are able to self-organize. It promotes flexible planning, evolutionary development, early delivery and continuous improvement where the changes are provided as soon as possible.

We have to outline core principals of agile approach in following list to ensure we have clear vision in how the development of sensorial framework will be driven.

- End user satisfaction by rapid delivery of useful software
- Requirements change are welcomed even at the end of development
- Working software is delivered frequently in weekly based periods
- Closed on daily based cooperation with developers and end users
- Face-to-face conversation is the best communication channel
- Project is built around motivated individuals
- Working software is basic measurement of progress
- Team self-organization

Once we are clear in focus in agile development area we also have to setup timing and format of how our agile process of development is going to be realized. Basically the shortest cycles of delivery working software are better than longer ones but there should not be too much tight schedule without stressful timing. Typical timing for synchronization / brainstorming meetings also called scrums is to be organized on daily bases where duration is maximal 15-30 minutes. The purpose of scrum meeting is about every team member have to call about on what he or she is working on, if there is any problem with that and what is planned for the next day. Problems and technical solutions are not solved on scrums there is space only for discussion who and when can help or how to escalate the problem. The timing for cycle of working software delivery also called sprint is about 1-2 weeks where discussed functionalities are analyzed, implemented, tested and deployed to the solution which can be demonstrated on the end of the cycle. In the following [Figure 9] is overview of the whole process of agile software development where also we define product backlog which typically is created by the end users / product owners and developers on weekly / monthly based periodical meetings called product review. This product backlog gathers all requirements on functionalities of developing system in form of short description also called user story which we will describe in more details in the following chapters.

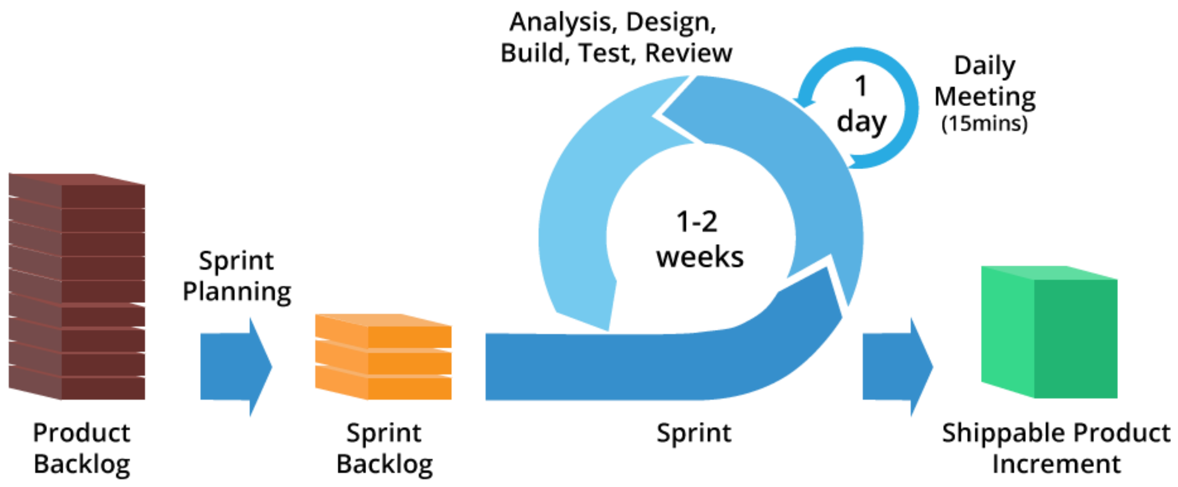


Figure 9 Agile Development Flow [35]

From product backlog we are able to plan the sprint back log by scope and priority which has to be provided during sprint reviews and retrospective meetings. The sprint can start after sprint backlog is planned and discussed on sprint review meeting with each team member and once the sprint ends the functionalities which are not ready for delivery and are moved to next sprint during sprint retrospective meeting. Also in the stage of product presentation and delivery at the end of each sprint the features are reviewed and if they do not fit to the end user needs they are moved back to next sprint iteration for modification. Also there are other agile methodologies for software development good to be mentioned in the following [Table 11] as well as scrum agile approach which is used for our software development phase.

Table 11 Agile methodologies of software development

METHODOLOGIE	DESCRIPTION
ADAPTIVE SOFTWARE DEVELOPMENT	Replaces traditional waterfall cycle with a repeating series speculate, collaborate and learn cycles
AGILE MODELING	Methodology for modeling and documenting software system based on best practices
AGILE UNIFIED PROCESS	The simplified version of the Rational Unified Process
DYNAMIC SYSTEM DEVELOPMENT METHOD	Interactive and incremental approach which embraces principal of agile development including continuous user / customer delivery
EXTREME PROGRAMING	Improve software quality and responsiveness to changing user / customer requirements completed by pair teams
FEATURE DRIVEN DEVELOPMENT	Interactive and incremental software development process which is driven by client valued functionality / features perspective
KABAN	Visual process management system to manage knowledge of work in terms of just-in-time delivery
SCRUM	Interactive and incremental self-organizing within team development method

The user stories captured simplicity of “who” “what” and “why” have to be done by the end user within future information system as a part of requirements specification where in simple short sentences which could be written on small cards is defined what is and what is not the part of the end user’s job function. It is comprehendible expression of common everyday business language and is close to user cases requirement gathering technique. There is a way of how user stories are being captured – it is called the “Scrum” which is basically kind of brainstorming where representative of developers and end users are coupled together and by idea creation process generating user stories by questions from developers. There are maybe required other roles in case of more scrum members such as scrum master who moderate effectively creation process and also role such as product manager who is responsible to capture message of user stories and formulating precise simple sentences which have to be written down for future usage. There are defined formats of user stories outlined in the following formula:

- As a <role>, I want <goal/desire>
- As <who> <when> <where>, I <want> because <why>
- In order to <receive benefit> as s <role>, I want <goal/desire>

After definition of user stories by formulation in short sentences the customer end user side has to prioritize each story accordingly its real value basically in terms of which story values the informational system the most. Based on defined priority the user stories with the highest priority are processed as first in kind of life cycle where each user story has to be defined, implemented, tested and shown to end user if it fulfill the needs. The life cycles may be considered on weekly bases or two week bases dependable on team size of suitable processing habits for providing team. The key of success is velocity and periodicity of standard outcome of providing team which also can be measured by story points where the difficulty of system feature is expressed. Such measurements are having only high level informational value which is increasing in time in preciseness of estimation in case the team has not been changed and their skills are static. But in the real world most of the teams are changing over the time and also skills are changing during the time and team in good environment should be more efficient over time. Therefore the measurement of project finalization is statistical derivation based on previous experience in order to revise story point value and good personal quality estimation. Anyways the measurement of difficulty and timeline consumption is still much better than other techniques due to small well defined chunks of system which are immediately incrementally provided and delivered as real functionality.

After description the software agile development theory and methodologies we start to develop sensorial framework with goals definition in the following chapter.

5.2. Goals & requirements & user stories

We have to specify our goals which we would like to be reached and at what we should be focused on create reality by thinking more sharply therefore following list outlined the mainstreams goals to be considered and repeatedly kept in mind.

- Gather any possible sensorial information from mobile devices
- Provide visualization of gathered sensors data into comprehensive form for the end users
- Add prediction models for consolidated sensorial data

Also we have to mention innovation potential and reasons for such defined goals. At first each voluntary user is able to view history of sensor's data on his/her mobile device. Where it make sense we outlined graphical representation of records based on user's location, time and activity. Users would be able to see sensorial map provided by sensorial framework which can help to automate decision making of any sensorial based electronic. And now let's describe user stories as basement for our models and further analysis.

5.2.1. User stories

For user story definition we are using scrum format where the developer side is represent by me and end users are represented by students who own Android mobile devices with desire to use innovative approach to information distribution. In the following [Table 12] we outlined user stories which are extracted by questionnaires from end user group.

Table 12 User stories defined by end user group

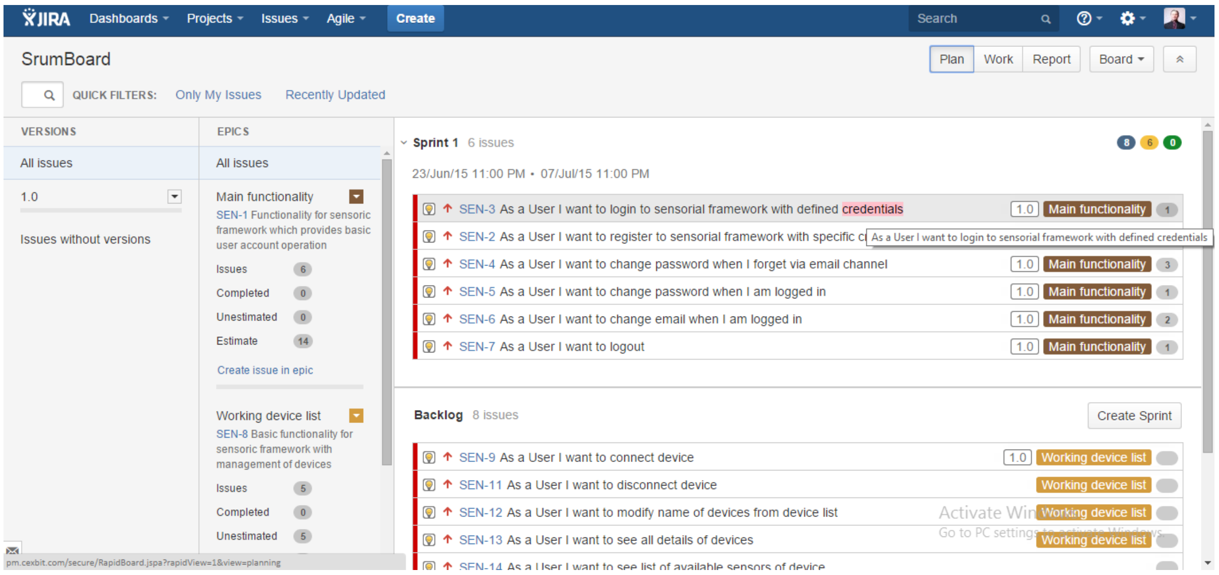
USER STORY IDENTIFIER (USI)	**CONTENT OF USER STORIES**	**RELATED TO**
USI-1-1	As a User I want to register into sensorial framework with specific credentials as email and password	Main
USI-1-2	As a User I want to login to sensorial framework with defined credentials	Main
USI-1-3-1	As a User I want to change password when I forget via email channel	Main
USI-1-3-2	As a User I want to change password when I am logged in	Main /Authorized
USI-1-3-3	As a User I want to change email when I am logged in	Main /Authorized
USI-1-4	As a User I want to logout	Main /Security /Authorized
USI-2-1	As a User I want to connect device to sensorial framework by installing application on device with the same login credentials and by specification basic description as name	Devices /Authorized
USI-2-2	As a User I want to disconnect the device	Devices /Authorized
USI-2-3	As a User I want to modify name of the device	Devices /Authorized

USI-2-4-1	As a User I want to see all connected devices to sensorial framework with basic description as a name, type of device and connection status of device in the device list	Devices /Authorized
USI-2-4-2	As a User I want to see all details of devices by selecting item from the device list	Devices /Authorized
USI-3-1	As a User I want to see list of available sensors of the device with name, type, periodicity/action and data counters	Device /Authorized
USI-3-2	As a User I want to see details of device sensor where I can customize details as period of monitoring, type of action, …	Sensor /Authorized
USI-3-3	As a User I want to see historical data of sensor in timeline charts	Sensor /Authorized

Once we have defined user stories we should be able to prioritize them and make assumption of difficulty in story points. We define such criteria to improve quality of decision making where the most important user stories are developed first and with difficulty we are able to count how much time it takes with how many human resources have to be spent. This kind of measurements increasing the flexibility of project organization and also much easily everyone can have brief insight of current project status. There is specialized form to visualize such overview called story board and we are providing in the following chapter in more detail the user stories with difficulty and priority where we consider as the main reason for that kind of view it is its simplicity for anyone and even view can be provided simply by pencil and stickers on any wall around you without needs to use electronic equipment.

5.2.2. Story board

In our case we are using for agile software development JIRA Agile v6.3.15 which is for non-commercial usage free to use by students or individuals. There are lots of functionalities which empower users in their aim to develop in agile style any software but we will focus on story boards. There are two main types of story boards – KABAN type or SCRUM type where visualized user stories are bounded into epics which are high level boundaries for them. Each member of the team can participate by work spent on specific user story in specific time line unit called sprint which can take days or weeks dependable on team size and its work efficiency. For overview following [Figure 10] outlined how story board of scrum type is possible to be visualized in planning phase before sprint starts.



SrumBoard

QUICK FILTERS: Only My Issues Recently Updated

VERSIONS
All issues
1.0
Issues without versions

EPICS
All issues

Main functionality
SEN-1 Functionality for sensoric framework which provides basic user account operation

Issues	6
Completed	0
Unestimated	0
Estimate	14

Create issue in epic

Working device list
SEN-8 Basic functionality for sensoric framework with management of devices

Issues	5
Completed	0
Unestimated	5

Sprint 1 6 issues 8 6 0

23/Jun/15 11:00 PM • 07/Jul/15 11:00 PM

SEN-3	As a User I want to login to sensorial framework with defined credentials	1.0	Main functionality	1
SEN-2	As a User I want to register to sensorial framework with specific c			
SEN-4	As a User I want to change password when I forget via email channel	1.0	Main functionality	3
SEN-5	As a User I want to change password when I am logged in	1.0	Main functionality	1
SEN-6	As a User I want to change email when I am logged in	1.0	Main functionality	2
SEN-7	As a User I want to logout	1.0	Main functionality	1

As a User I want to login to sensorial framework with defined credentials

Backlog 8 issues Create Sprint

SEN-9	As a User I want to connect device	1.0	Working device list
SEN-11	As a User I want to disconnect device		Working device list
SEN-12	As a User I want to modify name of devices from device list		Working device list
SEN-13	As a User I want to see all details of devices		Working device list
SEN-14	As a User I want to see list of available sensors of device		

Activate Windows
Go to PC settings to activate Windows

pm.cexbit.com/secure/RapidBoard.jspa?rapidView=1&view=planning

Figure 10 Story board – plan

After definition of each user story with required assigned attributes as estimate a story points, description, version of release, epic link, resolver assignment and assignment priority the sprint can be started. The typical trigger to start next sprint is meeting with all participants in form of scrum / work where all team members synchronized their knowledge about issues and planned workload. The simplest workflow consists just from *To Do*, *In Progress* and *Done* state but for our purposes we have to define more sophisticated workflow with a bit higher granularity which should be monitored. More states increase system monitoring ability of different types of participants such as external suppliers or third parties where information of states are required to be monitored. We define states as *Open*, *In Development*, *Waiting for QA*, In QA review and done visualized on the following [Figure 11] where relation between all states are also. All issues which have the initial state *Open* are basically in sprint backlog and also in product backlog and wait until one of the developers choose them.

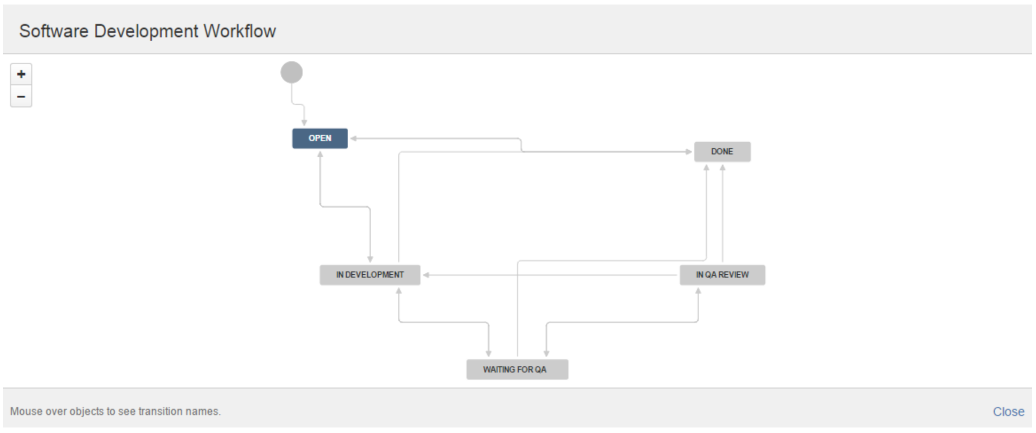
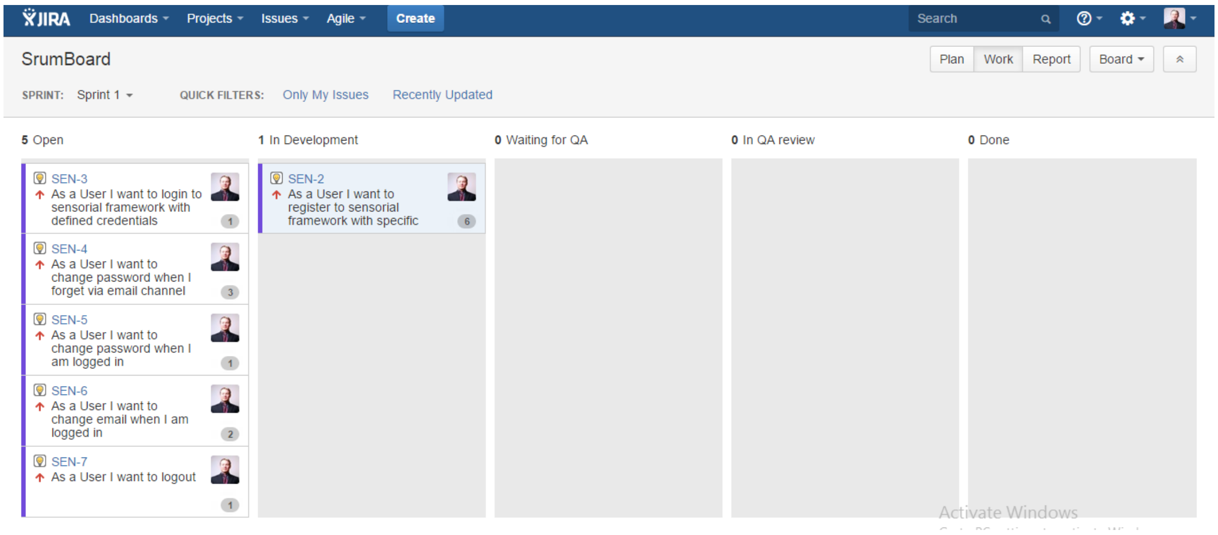

Software Development Workflow
OPEN
DONE
IN DEVELOPMENT
IN QA REVIEW
WAITING FOR QA
Mouse over objects to see transition names.
Close

Figure 11 Software development workflow

Once developer start working on issue the state is changed to state *In development* and after success implementation such user story goes into live system by release and deployment procedure. The end user or Quality Assurance (QA) tester reviewed them if it fulfills requirements defined in user story and if review of the user story is success the issues state is state *Done* otherwise ends up on the beginning in state *Open* with updated comments what else should be changed. Following [Figure 12] highlights story board of work in progress context where each user story or issue goes through defined states which are defined by particular workflow. Such view is really simple and easy to monitor in which state issue is at that time and also who is working on the issue with time consumption. We start sprint with bundle of user stories where estimation of difficulty in total do not exceed week limit of sprint. The week limit of sprint is defined by multiplication of daily limit which estimates capabilities of team members to deliver work.



SrumBoard

SPRINT: Sprint 1 QUICK FILTERS: Only My Issues Recently Updated

5 Open

SEN-3
As a User I want to login to sensorial framework with defined credentials
1

SEN-4
As a User I want to change password when I forget via email channel
3

SEN-5
As a User I want to change password when I am logged in
1

SEN-6
As a User I want to change email when I am logged in
2

SEN-7
As a User I want to logout
1

1 In Development

SEN-2
As a User I want to register to sensorial framework with specific
6

0 Waiting for QA

0 In QA review

0 Done

Figure 12 Story board - work

Each member of the team continuously updates status of a given user story during sprint which basically enables to monitor and report project status online even before the sprint ends. The velocity of the team can be expressed as multiplication between sum of difficulty of all tasks and their team members and multiply with empirical coefficient based tasks criteria.

5.3. System architecture & Component model

This chapter is dedicated to high level system architecture design and its used component to be ready to dig deeper into each part of the system. We consider client / server architecture [Figure 13] for information distribution where server is defined as a single instance and clients are multiple instances of mobile device application. Client consists from application, User Interface (UI), background service, sensors readers and database. The end users controls and view all information over UI from application module which gathers from server or from locally saved data in database. Once end user connect mobile device into the system the background service provides data from sensors to server and update local database. Server consists of several components as listeners, Application Program Interface (API), core, management, logging and database. The listeners storing sensorial data gathered from mobile devices into database. Those data are consolidated by core module and are ready for distribution if client requests them through application interface.

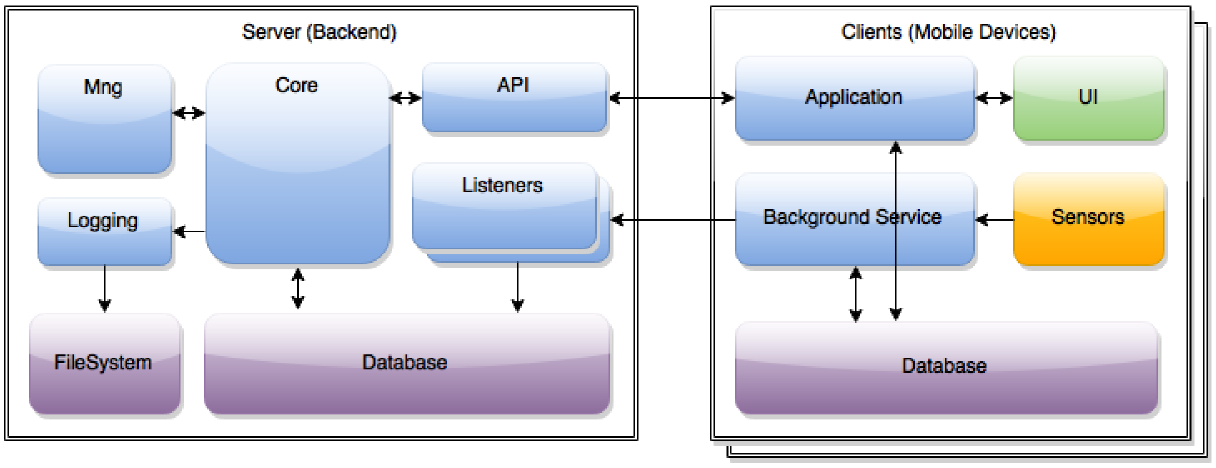


Figure 13 System architecture

5.4. Activity & Flow model

In design and analysis phase of Senzoric framework development process we are now ready to analyze in more detail each user story to provide us necessary insight for implementation. We define process/activity flow in comprehensive form to gather data to define classes and data models later on. For every first activity within the system end user has to have Android mobile device with access to the internet. The first action is about to install application on mobile devices from cloud store and start application. We outline high level of application flow where all user stories can be applied from specific application point of flow in the following [Figure 14].

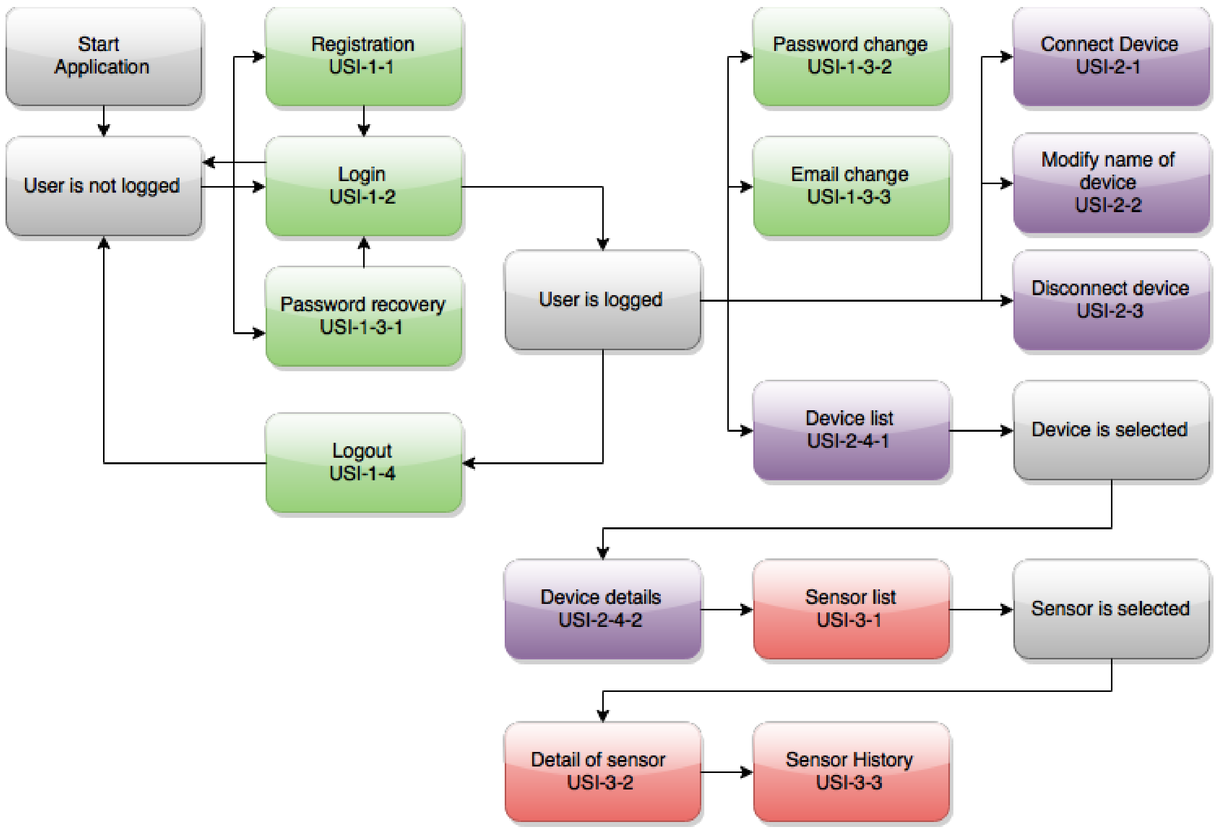


Figure 14 Application flow

At first describe activities after the application has started when user first time launched the Senzoric framework client application on mobile device. For such a start point there are just tree user stories registration of end user (USI-1-1), login already registered end user (USI-1-2) and password recovery for already registered end users via defined email (USI-1-3-1). The end users can choose from these three options by simple tap from menu after start. First we start with registration activity of the end users defined in the following [Figure 15] where we analyze required attributes for successful procedure.

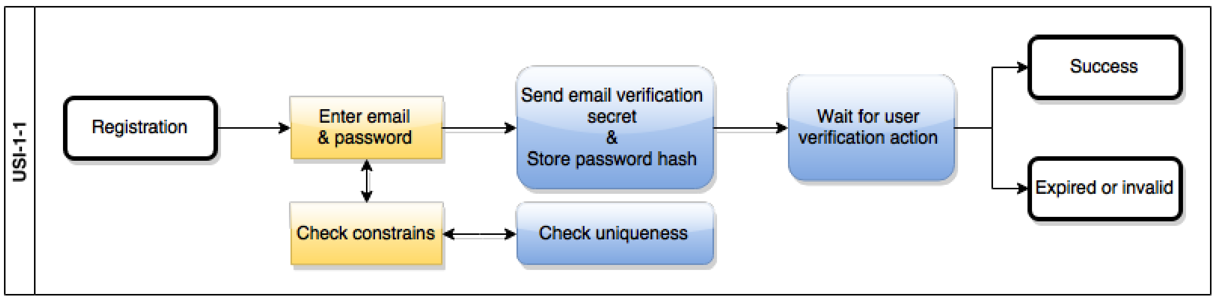


Figure 15 USI-1-1 User registration

During typing or when end user enters email where are constrain checks of validators with check of uniqueness entered email and once all checks passed then system can send verification code via email to the end user. Whenever user has verified the email by secret token included in the email the system can mark down email as verified and registration ends up as a success otherwise after defined period of time or in case of failed verification the system announces the registration as expired or invalid. Users have to confirm validity of email address by confirmation email to be able to login into the system later on. Once user was successfully registered and email address is verified as valid he or she can then use login activity to access the system. We defined in the following [Figure 16] the login activity itself. The credentials consists of email and password where constrains check controls only validity of email string entered and then verifies credentials in secured form with already stored credentials.

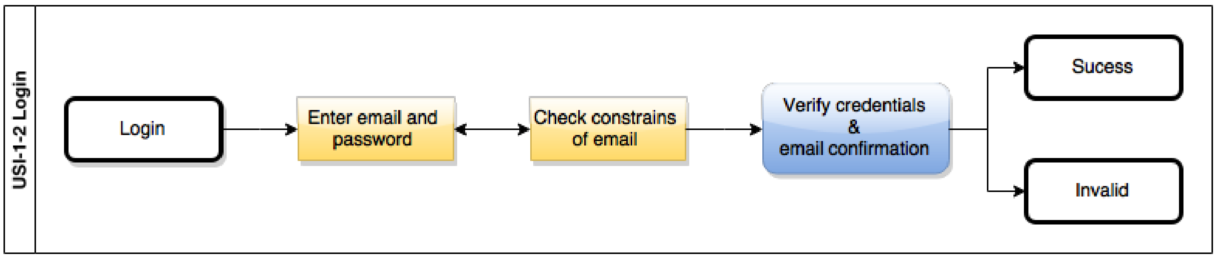

USI-1-2 Login
Login
Enter email and password
Check constrains of email
Verify credentials & email confirmation
Sucess
Invalid

Figure 16 USI-1-2 User login

If user is verified the login process end up with success and is able to proceed to authorized section otherwise if login process has failed the user ends up still in unauthorized section and password recovery procedure is right procedure which should be offered therefore we have to take it into account during application flow design in next chapters. Nevertheless following [Figure 17] defines mentioned password recovery procedure where user enters correct email which is checked if exists within the system or not.

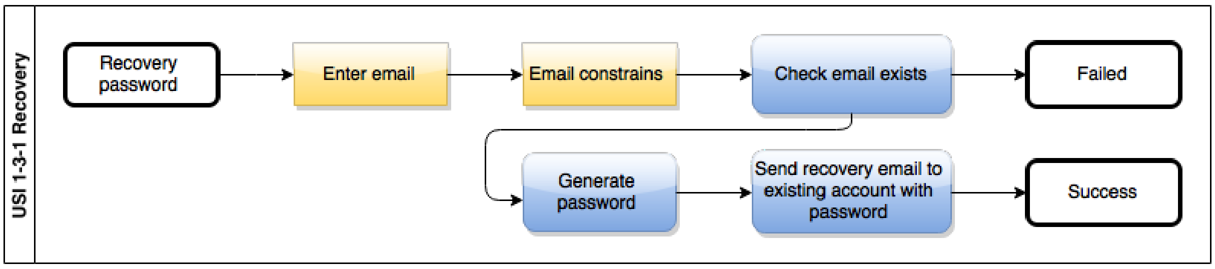


Figure 17 USI 1-3-1 Password recovery

System generates access token for user to be able to change password in the system via recovery email and user after password change can be logged again into the system with correct email and password. This is why we have to verified user's email prior to start any operation. Different scenario is when user is already logged into the system. The authorization was provided and we can provide authorized operation for the existing user's account such as password change designed in following [Figure 18] where from user's state is logged we perform activity password change and system asks to enter new password in dual control mode which means that user has to enter password two times. The system verifies password correctness and generates hash for password string which is transmitted to backend side for update. If everything goes well the procedure ends up with success and password for the user account is correctly changed otherwise in case of any failure user has to start the operation again.

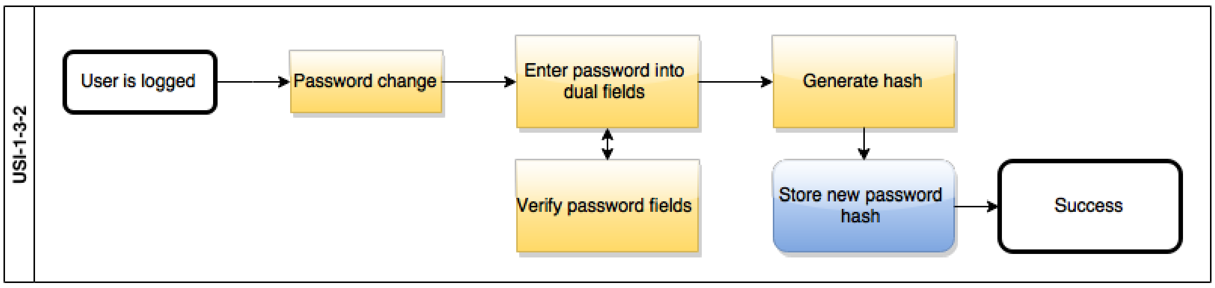


Figure 18 USI-1-3-2 User password change

Next user story is about to change account identification which is email therefore we can basically consider such scenario as security relevant where possibility to change email of user account may leads in case of any kind of break into the system to takeover of existing users accounts. We should be aware of it and this functionality relies on a high level of security and authorization level provided by the system. We define in the following [Figure 19] user email change. We consider user as logged into authorized system section and after requiring email change action user has to provide new email which is also verified based on sending email verification and receiving user verification response. In any case operation failed we consider the email change transaction as invalid and last verified email is being used.

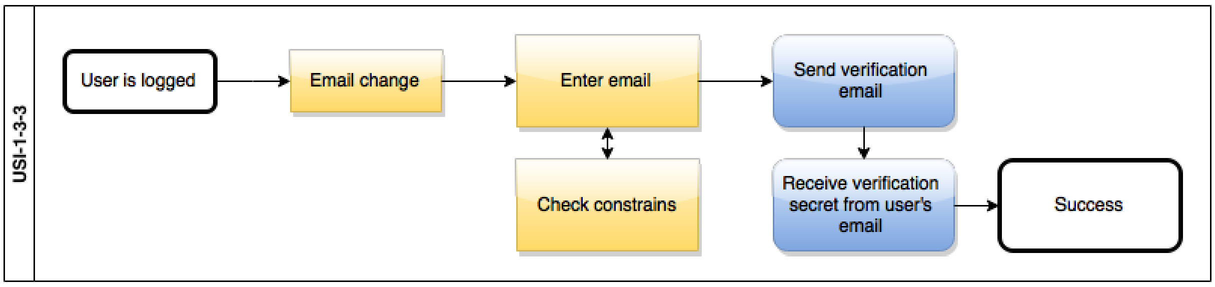


Figure 19 USI-1-3-3 User email change

The last access control operation we consider in our user stories is user logout. Basically once the user requests logout [Figure 20] the system we have to perform security based solution to close any other action till another authorization is provided from the user side. The solution is based on authorization token which provides access to the system for defined operations and once the logout action is requested the current authorization token which is being used has to have revoked permission from the system. Basically we provide OAuth authorization system for dynamic account access which can be modified for the 3rd parties as well as other purposes.

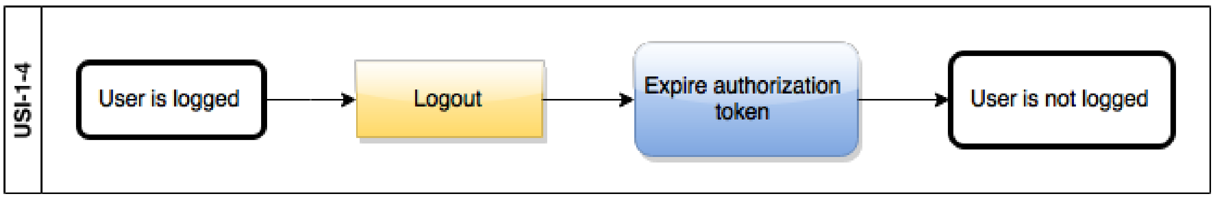


Figure 20 USI-1-4 User logout

That's all done with basic management and now we can go further into design of core functionalities of the system related to the devices. We consider as core device management user stories namely connect (USI-2-1), disconnect (USI-2-2), change name of the device (USI-2-3), view list of devices (USI-2-4-1) and view details of the devices (USI-2-4-2) within sensorial framework. We start with connection of device into the system outlined in the following [Figure 21] where user confirms current mobile device as provider of sensorial data which are going to be collected by sensorial framework.

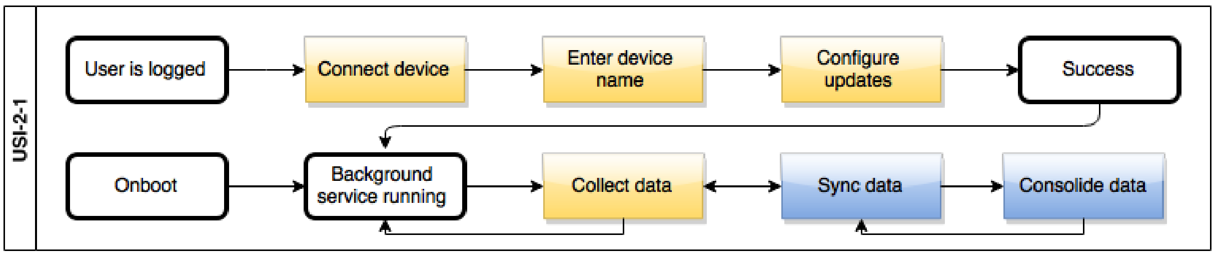


Figure 21 USI-2-1 Device connect

First user has to enter device name as main human readable identification otherwise as default there is proposed device type name. In the next step there is configuration of sensorial data updates how should be post into sensorial framework and be distributed. Once configuration is done we consider device connect as a success and every time after the device is started up there have to be running background service which is posting data to backend to collect and consolidate. In case of off-line mode sensorial data are not sent to the server and they are stored in local storage for later use on upload. Also user has to be able to disconnect device from the system and for such operation we design following activity in [Figure 22].

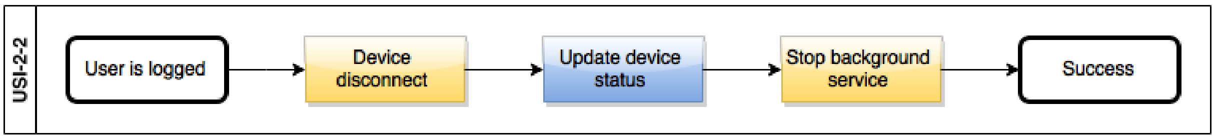


Figure 22 USI-2-2 Device disconnect

Once device is disconnected the background services are stopped after uploading disconnection status into backend system and the background service is not started after boot until the user connects again device into the system. Another user story describes device name change therefore we consider such activity outlined in [Figure 23] as possibility of name change only current device.

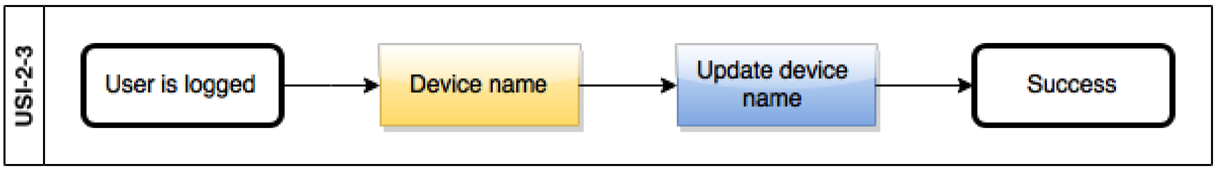


Figure 23 USI-2-3 Device name

After several connected devices we appreciate to have list of them at one place as easy and fast direction to each of them with basic description as a name, type and connection status. We design list of devices associated to user which is currently logged in as next activity visualized in the following [Figure 24] together with edit device name directly from device list story.

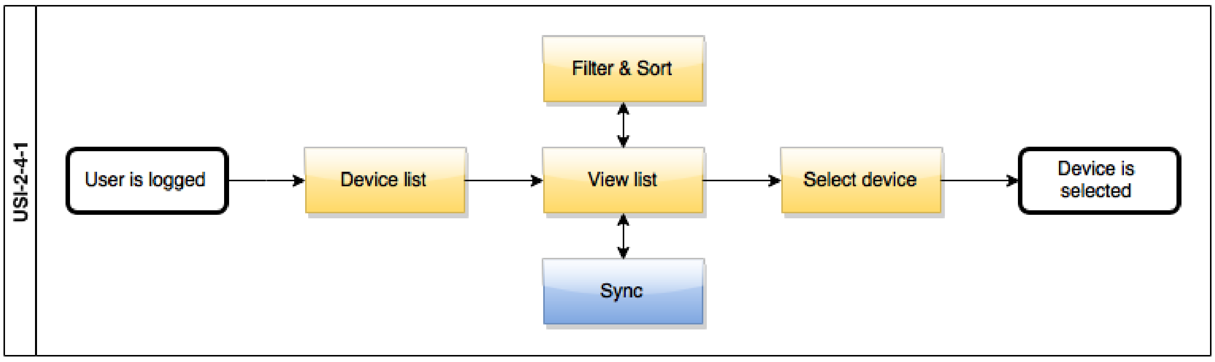


Figure 24 USI-2-4-1 Device list

When the user is logged in there is an option to view devices in the list with fresh data based on synchronization with backend system for remote devices therefore in offline mode only device visible is current device being used itself and in case of online mode all connected devices are visible. We are able to sort and filter devices in device list based on name, device status and time of last update. For device item such action as selection of device we continue into device details activity in the next [Figure 25] where together with sensor list activity is outlined the necessity of synchronization in case of remote devices. We consider having single approach to get data collection from local device or remote devices. So data for local device can be gathered directly over local data collection cache and for remote devices are available upon requests.

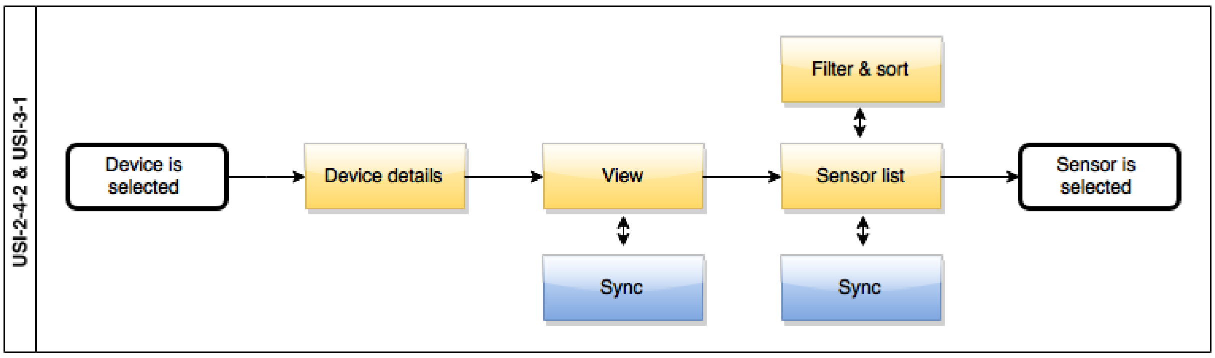


Figure 25 USI-2-4-2 Device details & USI-3-1 Device sensor list

Once the user selects sensor list in device details view the list is provided with possibility to filter and order cross sensor attributes such as name, type and period of collecting. User can also select single sensor to visualize sensor details and its history outlined in the following [Figure 26] as well as previous flow of the data collections are gathered on the local cache which is synchronized with system over the network based on user specific requests.

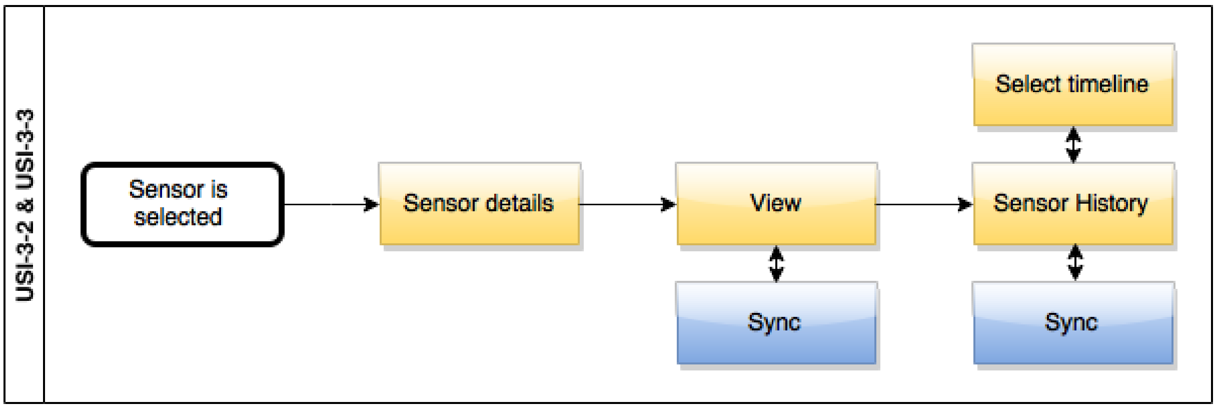


Figure 26 USI-3-2 Sensor details & USI-3-3 Sensor history

Those all previous defined use cases are just core functionality of Senzoric Framework. We consider possibility for third party developers to be able to implement plugins or customized widget views on mobile devices. And now let's start with models based on required use cases in following chapter.

5.5. Class & Data model & State model

This chapter is dedicated to modeling states, classes and data of the system. We consider activity/process flow design as sufficient for basic understanding what and how it should works within the system and from now on we have to be focused on how we are implementing such of functionalities. The state model can be for us the basement agreement about possibilities of system entities how they can behave and core principal how information is distributed internally between entities. We have to define main entities which can have states and later on which are natural binding for others derivate entities in our classes and data model universe. First we describe a few modeling tools which we are using and which we found out to be considered as useful.

5.5.1. Modeling tools

There are plenty of software tools for Unified Modeling Language (UML) based modeling. We highlight just a few of them and point out their advantage and disadvantage regarding to our purposes. For better overview we add small comparison set into the following [Table 13] where main attributes are highlighted as openness of software, latest release date and other features.

Table 13 Model tools for UML

LABEL	LICENSE	LAST RELEASE	GENERATION / REVERSE ENGINEERING
ENTERPRISE ARCHITECT	Commercial	2015-05-13	ActionScript, C, C#, C++, Delphi, Java, PHP, Python, Visual Basic, Visual Basic .NET, DDL, XML Schema, WSDL
MODELIO	GPL open source	2015-02-23 (3.3.1)	Java, C++, C#
PAPYRUS	EPL open source	2014-06-25	None
UML DESIGNER	EPL open source	2015-05-29	None

It is quite tricky to find out the correct modeling tool which fulfills specific project needs as programing languages used for generation code as well as reverse engineering to get models from code and also management tools possible to connect with. We propose at first well known modeling tools Enterprise Architect from Sparx System originated from Australia. It is handy, sophisticated and provides support for variety programing languages but we are not willing to use it because it is commercial. There are others well known modeling tools such as Papyrus and UML Designer which are open sourced and covers main functionalities with UML standard but generation and reverse engineering is missing so we decided to use Modelio which comes as modified eclipse environment standalone and provides Java, C++ a C# reverse and generation software engineering.

5.5.2. State models

We conceive core entities as natural representation of reality such as user, device and sensor. Those are main three entities which are basement for others to interact with and to be connected with. The user entity represents all information related to single user in data collection. The states where user entity can result are defined firstly for authorization purposes as you can see in the following [Figure 27].

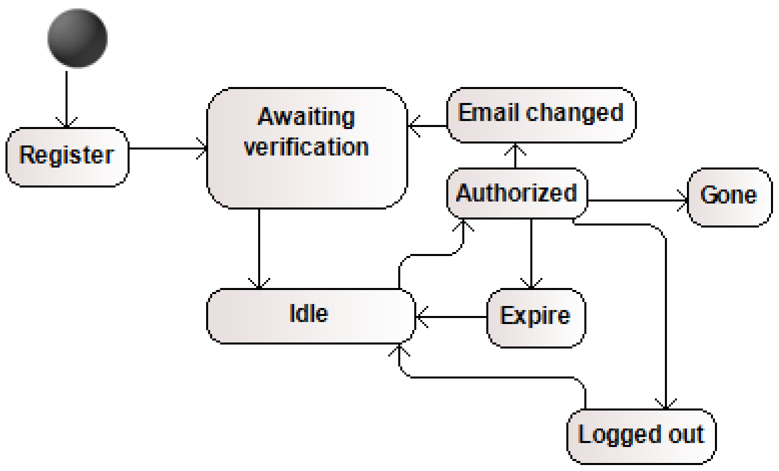


Figure 27 State model - User

Those user authorization states are relevant to influent other parts of the system to behave correspondingly. In the case of *Register* state which is first initial state the user fill out credentials and post them on backend system which waits for verification via user's email. When user verified the email then user's state is idle and system from now on can accept login credentials into the system. After success login the user state became authorized in which authorize users can perform tasks within the system. The user can also change email which requires also email verification therefore once email change is performed the user state became *Awaiting for verification*. This means that when the user is authorized and asks for email change with invalid email, the old email address is still used for authorization for login into the system otherwise we allow dead lock in process flow. From *Authorized* state user can become logged out or expired after specific timeout for security reasons. And lastly when user is asking for closure account we consider such state as state *Gone* and user account is disabled and email address is not used for uniqueness check so the same user with that email address is able to register again but with new account. Next core entity is device where we understand the states as are visualized in the following [Figure 28].

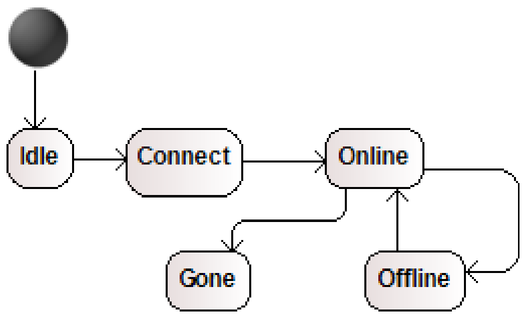


Figure 28 State model – Device

The device initially is in state idle which means that the device only view the system data. Once the user explicitly connects device into the system the device from that point provides data into the system which can be distributed. If devices has network available then can change state to online which means it provides data on background. If the network is not available the device falls down to state offline which identifies no network and no data are currently provided to the system. Lastly the state *Gone* is for excluding the device out of the system by uninstalling application or in case of that the device is lost or is damaged. Another core entity of system is sensor and it state model is outlined in following [Figure 29].

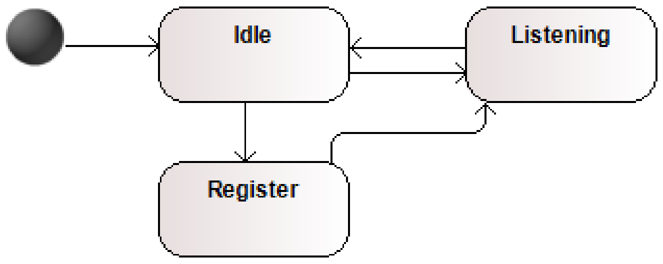


Figure 29 State model – Sensor

The sensor after boot started in *Idle* state where the mobile device enables interface for registration listeners. After sensor listener is registered in state *Register* the sensorial data are received by *Listening* state through inner handler to process incoming data.

5.5.3. Class model & Data model

In design phase the class model is main principal to express how system should look like in implementation phase and is really necessary to straight up minds from class point of view to not to get confused and to get sustainable design proposal. In our case we already declared basic entities important for modeled system such as user, device and sensor. We outlined in the following [Figure 30] the core class infrastructure which shows association in between those entities. Each user can authorize zero, one or more devices which are connected to the system and can be controlled or viewed by this user. Also each device can consist of zero, one or more sensors for gathering sensorial data for distribution. The act of data collecting from sensors we defined as measurement entity which can be considered as fact representation of reality. In diagram are mentioned main functionalities defined in user stories which seems to be sufficient but we assumes that during implementation phase the amount of functions and classes rapidly increases to cover defined goals together with platforms possibilities and other aspects.

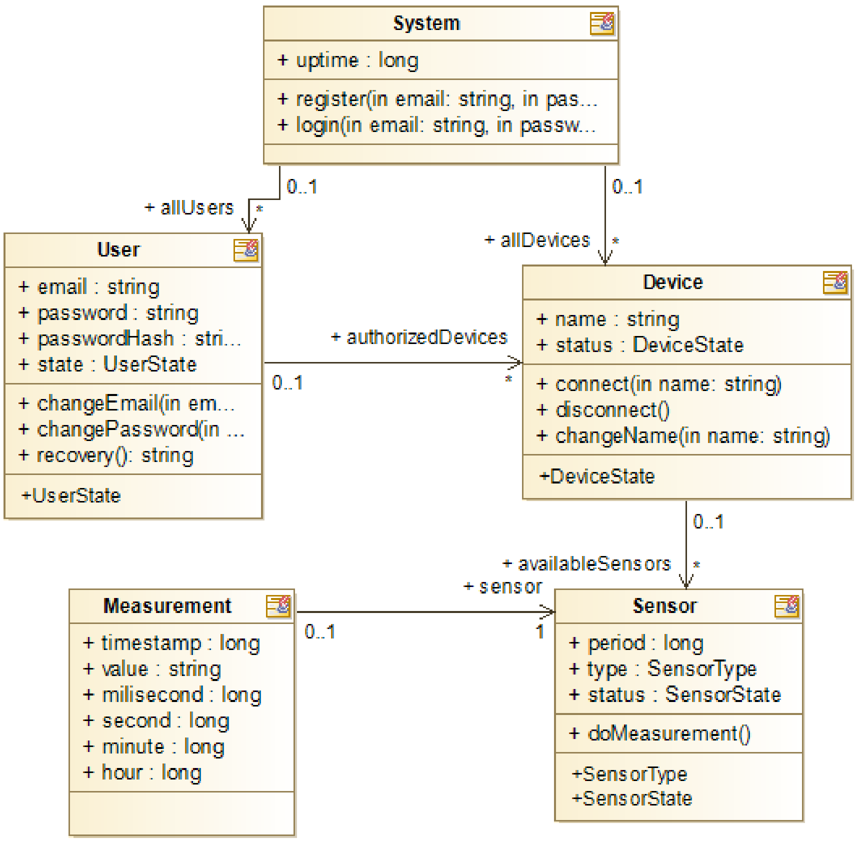
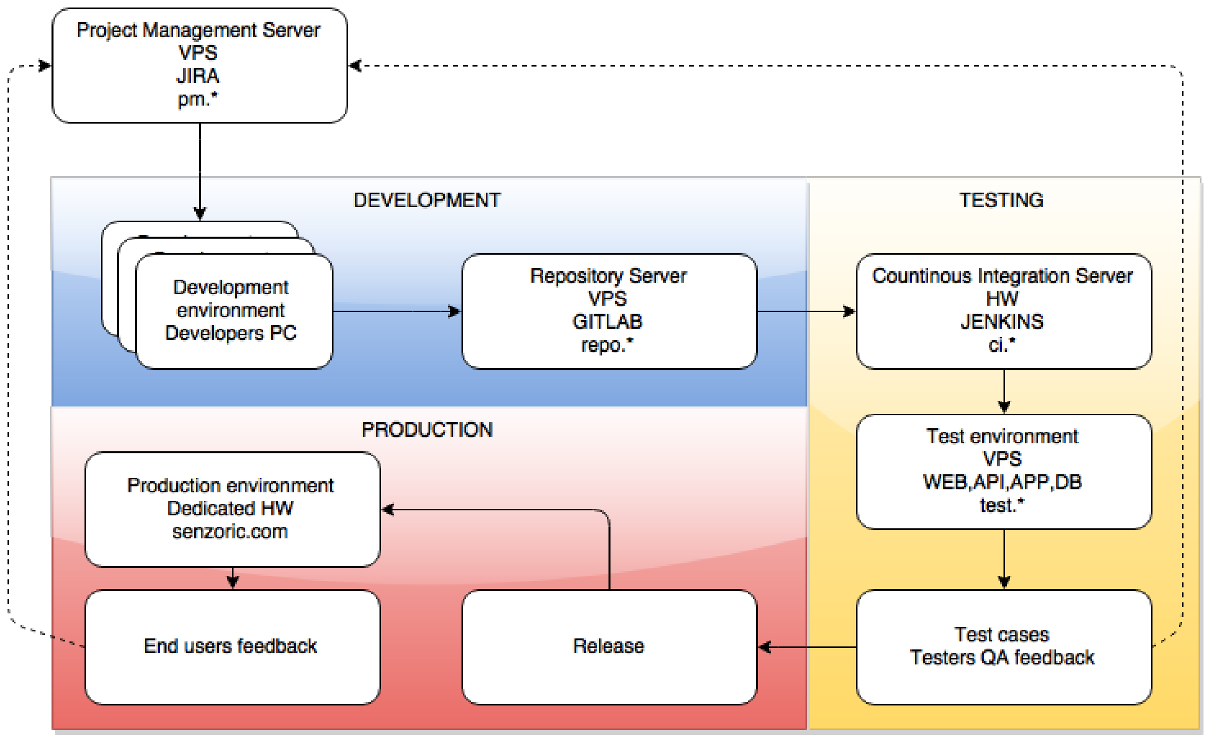


Figure 30 Class model – core system

5.6. Deployment model

The key part of software development is deployment process flow where we decided to use customized agile flow rather than classical once such as water flow. We considering development, testing and production phases being processed on demand with as minimal as possible granularity of source code increment. Based on long term experience our deployment flow outlined in the following [Figure 31] using specific tools and customized process flow where the issue lifecycle is designed to minimalizing delays and providing automation where possible. The flow starts by formulation of task or issues which are materialized by project management software (JIRA) located on dedicated VPS. Each task after formalization is consider to be implemented locally on development computer where implementation is committed and pushed to repository server. We uses as repository server a version control management (GitLab) to be able to handle user management, control and maintain repositories easily. From repository the dedicate VPS such as Continuous Integrational (CI) server with running web application (Jenkins) performs compilation and testing of committed source code in customized test environment.



Project Management Server VPS JIRA pm.*
DEVELOPMENT
Development environment Developers PC
Repository Server VPS GITLAB repo.*
TESTING
Countinous Integration Server HW JENKINS ci.*
Test environment VPS WEB,API,APP,DB test.*
PRODUCTION
Production environment Dedicated HW senzoric.com
End users feedback
Release
Test cases Testers QA feedback

Figure 31 Deployment Process Flow

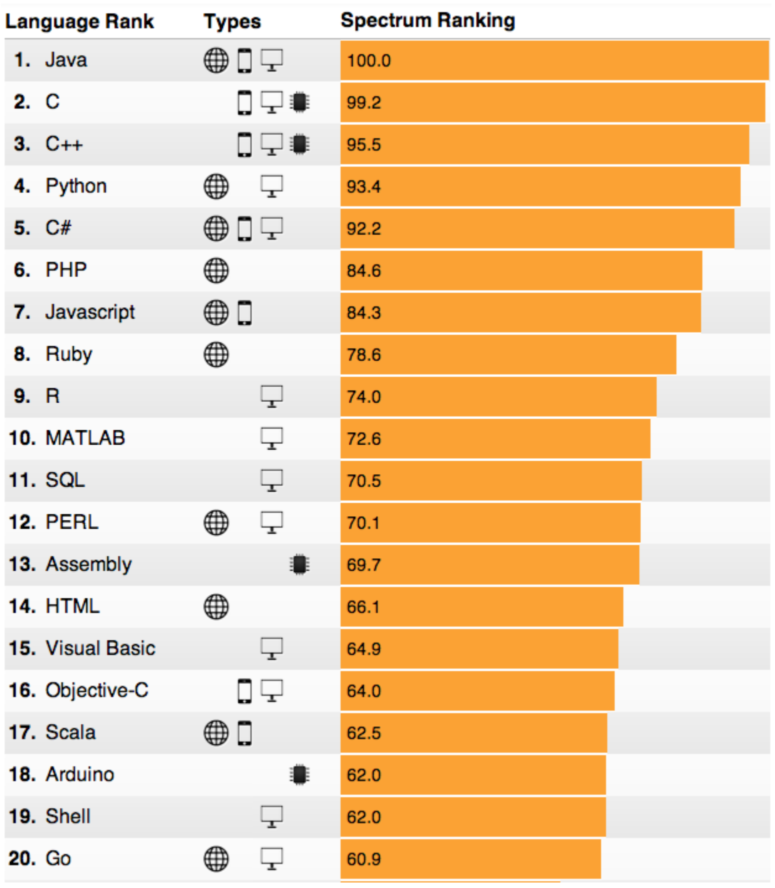
Whenever the builds are processed with error in test or compilation the outcome is announced to developers responsible for source code which causes the error. And if builds on CI are processed correctly without error then Quality Assurance (QA) performing the testing based on user stories manually or by monkey runners. The most valuable testing from QA is the end to end testing where the function is tested across the whole system. If all issues were implemented properly and QA approve their correctness then system parts can be released for production environment where the additional quality is improved by the end users of the system in the form of feedback management.

6. Implementation

In this chapter we go through implementation phase of the system development lifecycle. We describe tools and programing languages which are being used at first and then we produce the source code based on defined class model and criteria from design phase. At the end we review our goals by testing the implementation according defined test cases.

6.1. Programing tools & used frameworks

We consider Java programing language as well as Java community as the most suitable for ours implementation needs. The main reason is of course the front end application for Android mobile device platform where Java is used as programing language. In following [Figure 32] is highlighted the the ranking of programing language in terms of independent open sourced code available within community, ability of usage on different platforms, the size of knowledge base available on the internet and the size of community for co-working.



Language Rank	Types	Spectrum Ranking
1. Java	🌐 📱 🖥	100.0
2. C	📱 🖥 ▪	99.2
3. C++	📱 🖥 ▪	95.5
4. Python	🌐 🖥	93.4
5. C#	🌐 📱 🖥	92.2
6. PHP	🌐	84.6
7. Javascript	🌐 📱	84.3
8. Ruby	🌐	78.6
9. R	🖥	74.0
10. MATLAB	🖥	72.6
11. SQL	🖥	70.5
12. PERL	🌐 🖥	70.1
13. Assembly	▪	69.7
14. HTML	🌐	66.1
15. Visual Basic	🖥	64.9
16. Objective-C	📱 🖥	64.0
17. Scala	🌐 📱	62.5
18. Arduino	▪	62.0
19. Shell	🖥	62.0
20. Go	🌐 🖥	60.9

Figure 32 Programing Language Ranking [36]

There are other advantages speaking for Java programming language which is debugging mode and performance monitoring tools (JMX) of Java Virtual Machine (JVM). Against others interprets Java provides debugging even for remote usage so developers are able to remote debug backend or frontend. That is such significant advantage for developers in case of resolving bugs which are often not able to be seen in source code. Also Java is based on bytecode which provides cross platforms code delivery and developers are not tight to once specific platform of operation system. Therefore those reasons led us to choose Java based application server for backend, client frontend application and development environment tool.

There is plenty of programing Integrated Development Environment (IDE) tools. One of them is open sourced and one of them is commercial and also some of them are both models together. We will use open sourced IDE due to our academic aim and of course the price nevertheless in the following [Table 14] are outlined also commercial for better overview for programmers to get the best choice of programing tools.

Table 14 Integrated Development Environment

NAME	DESCRIPTION
ECLIPSE	License: EPL, originated from IBM VisualAge, the Eclipse foundation was created at 2004, and the IDE is based on workspace and extensible plug-in system written in Java
IDEA	License: Community Edition: Apache License v2.0, Ultimate Edition: proprietary, developed by JetBrains with first release 2001
NETBEANS	License: CDDL, GPL2, started as student project IDE on MatFyz UK in 1996 and after it was bought by Sun Microsystem 1999 and later on by Oracle 2010, IDE with modular based software development

For our development we use Eclipse together with Idea and from developers point of view the Idea benefits with more error prone solution, faster search based on indexed files and better appearance variability. Eclipse still has some advantages in compilation based tasks where each change of source file edited in IDE is automatically being processed for compilations across the whole project therefore the errors or mistakes made by developers are highlighted immediately.

Now we describe implementation of the frontend application which in comparison with backend does not have that much variety across single programing language platform. Frontend is closer to hardware and firmware therefore the framework is kind of glue between developers and framework providers and manufacturers. We support nowadays only Android mobile device platform for its openness and the highest market share around the world. But there are also others different platforms which are good to mentioned in the following [Table 15].

Table 15 Frontend – Client application

NAME	DESCRIPTION
ANDROID	Google, free and open source, 65.4% (2014), build on Linux kernel
IPHONE	Apple, closed source and proprietary, 14.8% (2014), build on open source Darwin core OS
WINDOWS	Microsoft, closed source and proprietary, 2.7% (2014), build on Windows 10 core OS

The Android platform also benefits with open source code availability against IPhone and Windows the only Application Program Interface (API) is provided to developers. Nevertheless in some cases this closed platform still can have benefits in terms of performance of graphical output on mobile device. For instance on IPhone platform the system handles graphical tasks with maximal priority in exclusive mode for processor instead of Android platform where the same tasks are being processed in parallel to other common tasks on mobile device processor. The freedom has to pay for kind of power in usage. As next we describe the backend implementation of application server where the set of well-known application servers based on Java programing language is outlined in the following [Table 16]. In our case we used the Tomcat application server for web sites because of relative excellent performance and simplicity to use for deployment and security.

Table 16 Backend – Web Application Servers

NAME	DESCRIPTION
TOMCAT / APACHE 2.0 FREE	Small download (12MB size), Maven dependency integration, great IDE support, great community
JETTY / APACHE 2.0 EPL	Smallest download (just 8MB size), easy to start, Maven dependency integration, single config file
JBOSS / REDHAT COMMERCIAL	Largest download, slower, good IDE support, excellent administration and monitoring
GLASSFISH / ORACLE COMMERCIAL	Reasonable download size, no straightforward way to start, full JEE and OSGi support

There are also other types of backend written in and using as interpret for instance Javascript (Node) which may result in faster requests delivery but regarding debugging we stay with Java based. Also Jetty server based implementation covers real advantages of simplicity and performance in production environment but on the other hand more robust backend Java application servers such as JBoss or Glassfish are excellent in management tasks. We decide to compromise between performance, management and simplicity and therefore the Tomcat outcome as winner in our criteria

6.2. Architecture of the system

We consider the architecture of the system defined earlier in design part for implementation in terms of used technology of each component. We consider there are two main parts of the system which are frontend and backend but there are also others to be considered as intersection of both parts such as models and communication principals interface encoded in kind of blueprints and also website for public information distribution.

Table 17 Project infrastructure

PROJECT NAME	DESCRIPTION
SENZORIC-BACKEND	The representation of all server side implementation of SF functionalities written in Java and using embedded Tomcat as TCP web server container and using Nette framework for UDP messaging Available on `git@repo.cexbit.com:senzoric/senzoric-backend.git`
SENZORIC-FRONTEND	The representation of client side for Android mobile device written in Java and using Android framework and external libraries for networking and processing Available `git@repo.cexbit.com:senzoric/senzoric-frontend.git`
SENZORIC-COMMONS	The library representation of intersection between backend and frontend where models and blueprints are defined and used in other projects Available `git@repo.cexbit.com:senzoric/senzoric-commons.git`
SENZORIC-WEBSITE	The public web site representation to bundle all necessary information for open source project Senzoric.com Available on `git@repo.cexbit.com:senzoric/senzoric-website.git`

SENZORIC	The representation of container for bundle all project modules at one place. Available on `git@repo.cexbit.com:senzoric/senzoric.git`

Therefore we need to define project infrastructure in combination of versioning source code to handle each part independently defined in [Table 17]. Over time we have been experienced with necessities of developing system parts separately with multiple users in parallel. From code versioning point of view the smaller parts of code are easier to be managed in independent build system and for independent developers which have to cooperate. That is a reason why at first we have to define project infrastructure and code repository settings for proper workflow.

Each project module has its own GIT repository and the shared part `senzoric-commos` is included in `senzoric-backend` and `senzoric-frontend` as their GIT submodule. The shared parts are visualized in the following [Figure 33] and [Figure 34] as yellow components marked as commons. The change of shared components influence both system parts therefore the versioning and releases of share part have to be done in minimal amount with consideration of backporting compatibility.

First we start with frontend architecture where used cases are closer to reach the goals and basically define the backend functionality at the beginning. The frontend consists of User Interface (UI), core application, background service, sensors monitor and database. In the following [Figure 33] we outlined the basic components of client architecture grouped by functionality.

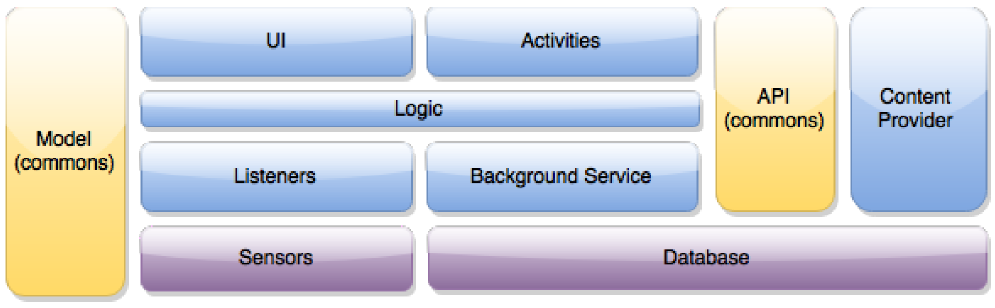


Figure 33 Client Architecture

The UI provides interactions with user and visualizes necessary information such as the collected data from sensors during monitoring and the application control information. Core application communicates with backend and provides data to database and UI. The data itself are gathered by background service which is started after boot and reading by listeners provided sensors data. On the other hand the backend outlined in the following [Figure 34] have to take into account the management of multiple client instances which are going to be served. Such logic is covered in the heart of backend system called core logic application where the message flows for incoming and out coming messages is defined. As entry point for system to provide data fronted client we define the application program interface where clients are able to question desired information. Instead of entry point API we defined the component called listeners which is dedicated to receiving bundles of sensors data from mobile device sensors which are being stored later on into database.

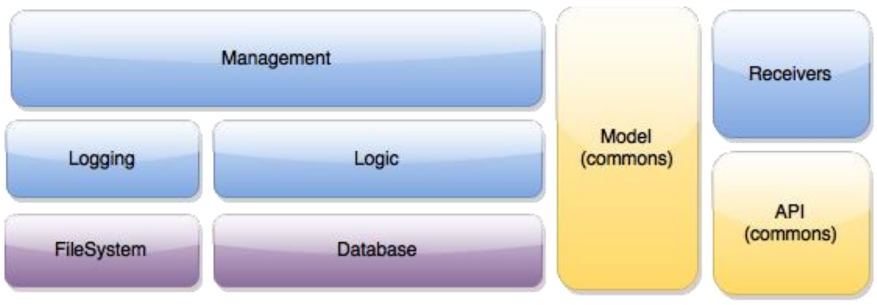


Figure 34 Server Architecture

The architecture both frontend and backend can change during implementation itself based on available external open sourced libraries and capabilities of dedicated environment.

6.3. Frontend implementation

Before we start implementation of the frontend application we need to understand Android framework in more details. The Android framework consists of several layers Linux kernel, Hardware Abstraction Layer (HAL), Libraries, Android runtime, Application framework and finally Applications their selves outlined in the following [Figure 35]. The Linux kernel is well known project which was originally developed in 1991 by Linux Torvalds together with set of GNU tools, utilities and compiler developed by Richard Stallman. Currently Android using Linux kernel version 2.6 to provide preemptive multitasking, low-level core system services for memory, processes, power management, networking and device drivers for hardware. The upper layers such as Android libraries and HAL are written in C/C++ and providing unified access to hardware together with commonly used functionalities available in shared libraries. The Android runtime uses Dalvik Virtual Machine instead of Java Virtual Machine as interpret and Dalvik Executable (DEX) instead of Java bytecode as Android code to be interpreted. The Dalvik format excels with almost 50% smaller memory footprint than standard Java bytecode and Dalvik VM is specially designed to provide sandbox for each application by its own virtual machine instance and is optimized to run efficiently within resource constrains of mobile device. Upper layer such as Application framework is written in Java and provides high level interface to system based services for developers. And finally the application top layer where are some default application located together with our frontend application which are gathering sensorial data from mobile device.

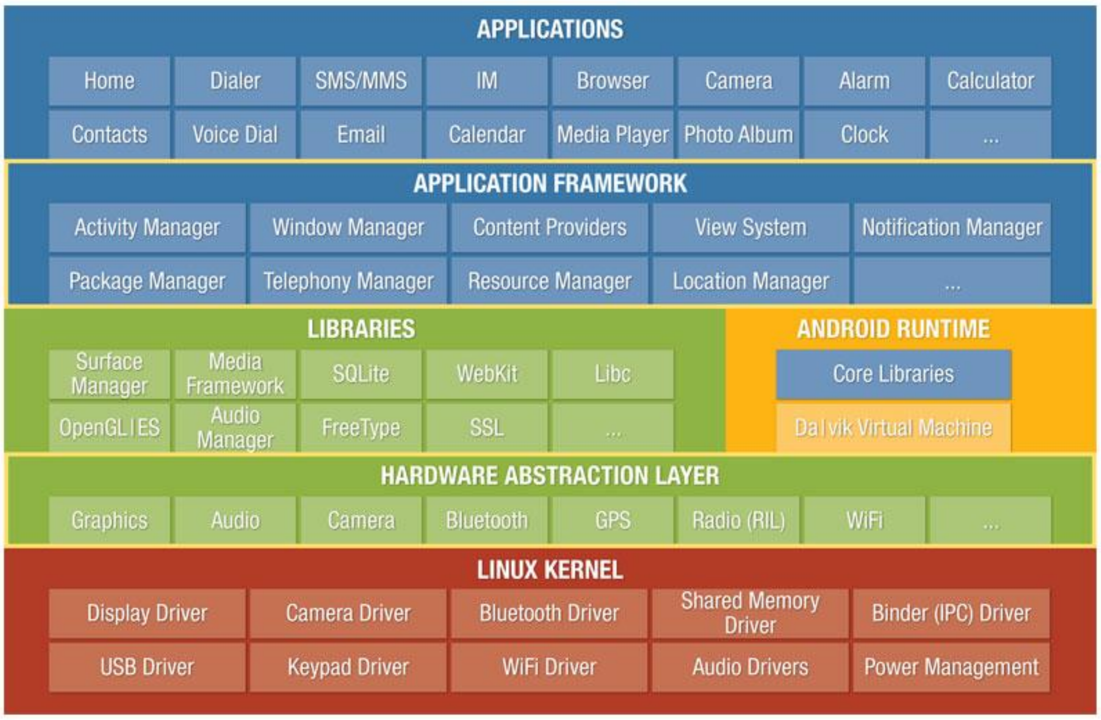
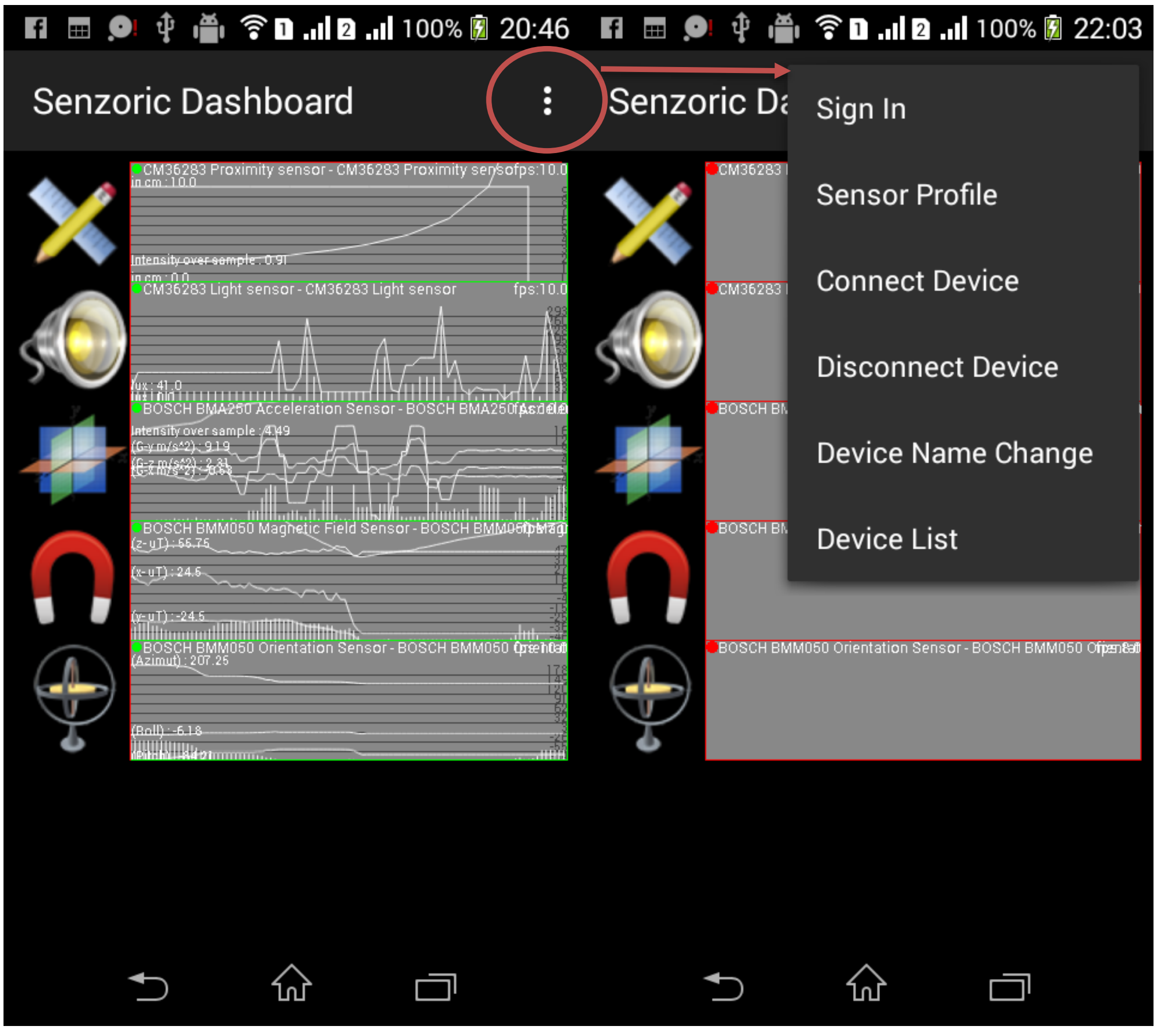


Figure 35 Android framework [37]

We customize proposed system architecture from previous chapters with consideration of Android framework where application layout UI, application logic and application data store are the main parts to be implemented. We start with implementation of UI in the following application layout chapter.

6.3.1. Application Layout

The visualization mechanism on Android platform is based on static definition wrapped into xml file layout or dynamic definition when during runtime we create visual components as instances and adding them into our layout. The fist method help developers for fast and What You See Is What You Get (WYSIWYG) design mode. On the other hand the dynamic definition access advance design mode is based on variables in runtime. We will use the combination of both where basic layout is defined as static and will not change during the application lifecycle and inner visual components which some of them will be generated during the start of the application. So we define two basic layout which are highlighted in the following [Figure 36] where user can see sensor list immediately after startup and can by touching on each graph start monitoring or stop monitoring of sensor. The navigation is handled by menu in right top corner which is also visualized in right screenshot. The users are able to log in into the system, see sensors profile, connect / disconnect device to cloud, see devices which are also connected to system and even change device name. We consider that in common development scenario where the layout is designed by graphics and user experience (UX) the layout may not change but in our case the proposed layout can be optimized and simplified based on future change requests.



100% 20:46
100% 22:03
Senzoric Dashboard
Senzoric Da
Sign In
Sensor Profile
Connect Device
Disconnect Device
Device Name Change
Device List
CM36283 Proximity sensor - CM36283 Proximity sensofps:10.0
CM36283 Light sensor - CM36283 Light sensor fps:10.0
BOSCH BMA250 Acceleration Sensor - BOSCH BMA250
BOSCH BMM050 Magnetic Field Sensor - BOSCH BMM050
BOSCH BMM050 Orientation Sensor - BOSCH BMM050

Figure 36 Client Android Application – Sensor List

This layout dashboard is dynamically generated by list of available sensor from `SensorManager` supplied with each item by `SensorViewGraph` class which is responsible for proper data visualization of cached data in memory of each sensor. This view is refreshed at least 10 fps and we consider that for prototype it is sufficient to able to see in real time what is behind sensor raw data but in further development we consider to provide visualization with GL rendering for better refresh rate. Nevertheless current solution is based on `Canvas` and provides real time data graphing. Sensors data are firstly collected into memory cache by sensor listeners. Then collected data are processed and the intensity is being calculated. Independently view manager whenever the `canvas` can be refreshed after propagating `invalidate` state on view the `canvas` is redraw with current values from cache memory. This process is repeatedly provided and the sensor data animation appears on the screen.

Next we describe Android framework static layout defined by xml file. The layout files has to be located in res/layout/*.xml and looks like following simple example of the sensor profile customization layout called `activity_setup.xml` where you can see defined attributes of sensor which can be customized as sensor type, sampling rate, sampling period, logging rate for local data store and sending rate for posting to cloud data store.

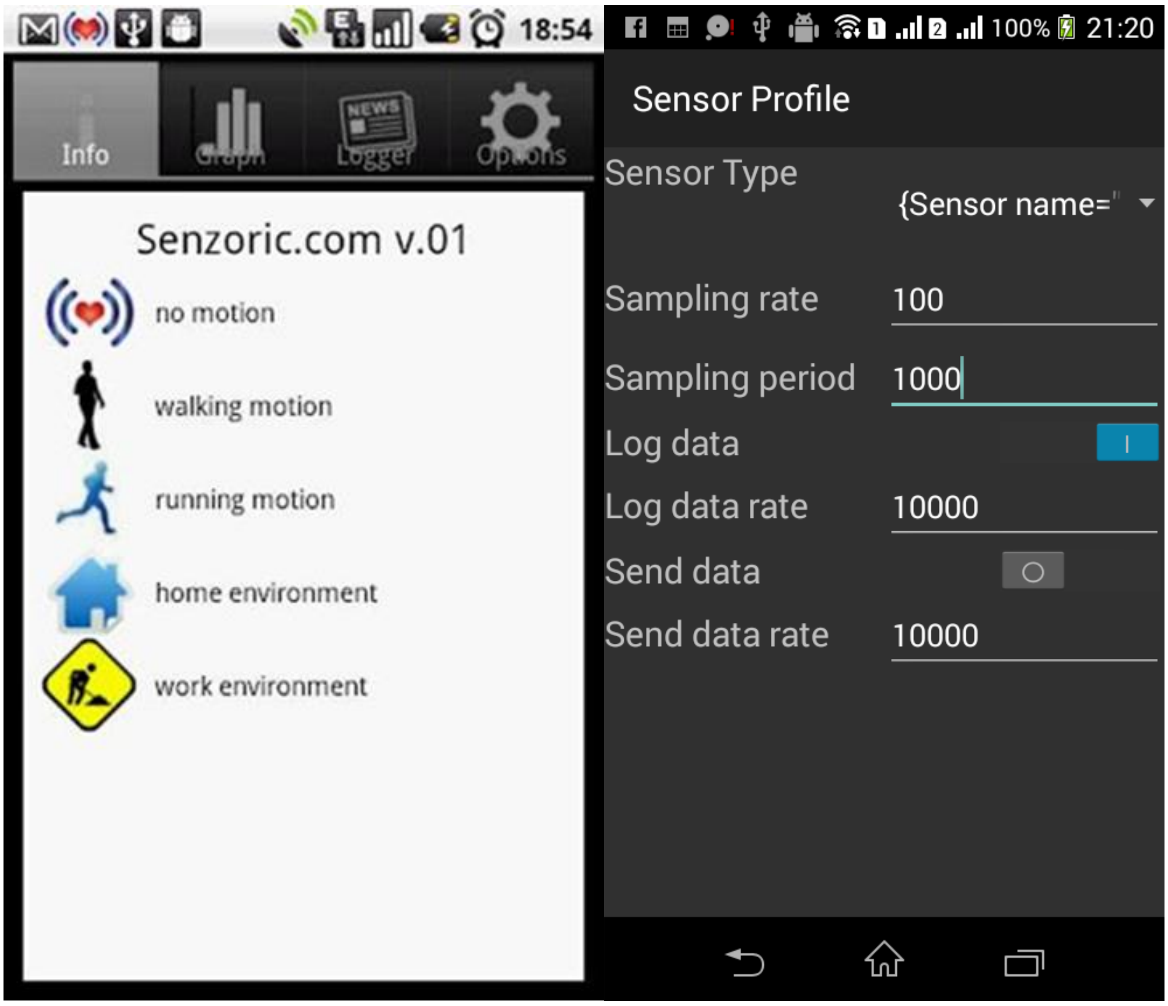
```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent" // define inner width to outer parent
    android:layout_height="wrap_content" // define inner height to outer parent
    >

    <TableRow                            // add button widget to layout
        android:layout_width="match_parent"// define inner width to outer
        android:layout_height="wrap_content"> // maximize height as possible

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="@string/sensor_type"    // define string resource
            android:id="@+id/textView"            // define id of widget
            android:textSize="20dp"               // define size of text
            android:layout_weight="1" />

        <Spinner
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:id="@+id/spinner"
            android:layout_gravity="center_horizontal"
            android:layout_weight="1"
            android:spinnerMode="dropdown" />
    </TableRow> ….                               // others elements
</TableLayout>
```

The activity layout defined in previous code example called `activity_setup.xml` is visualized in the following [Figure 37] on right side. The sensor profile layout of client Android application is dedicated to customization of embedded mobile device sensors. The user can choice by Spinner Android component from scanned sensor list the required embedded sensor. After selection of desired embedded sensor of mobile device user can customized sensor attributes in order to change the sampling rate, sampling period, logging data rate and cloud sending data rate. Those attributes of sensor profile can be set via interactive number entering dialog. The user can customized sampling rate in millisecond just by entering integer value in input form. The entered value is locally stored first and then synchronized with cloud to sensor profile. The values are applied instantly by updating live instance and updating listeners of sensor via `SensorManager` after user exists from activity by back button.



Sensor Type {Sensor name="
Sampling rate 100
Sampling period 1000
Log data
Log data rate 10000
Send data
Send data rate 10000

Figure 37 Client Android Application – Sensor Profile & Info

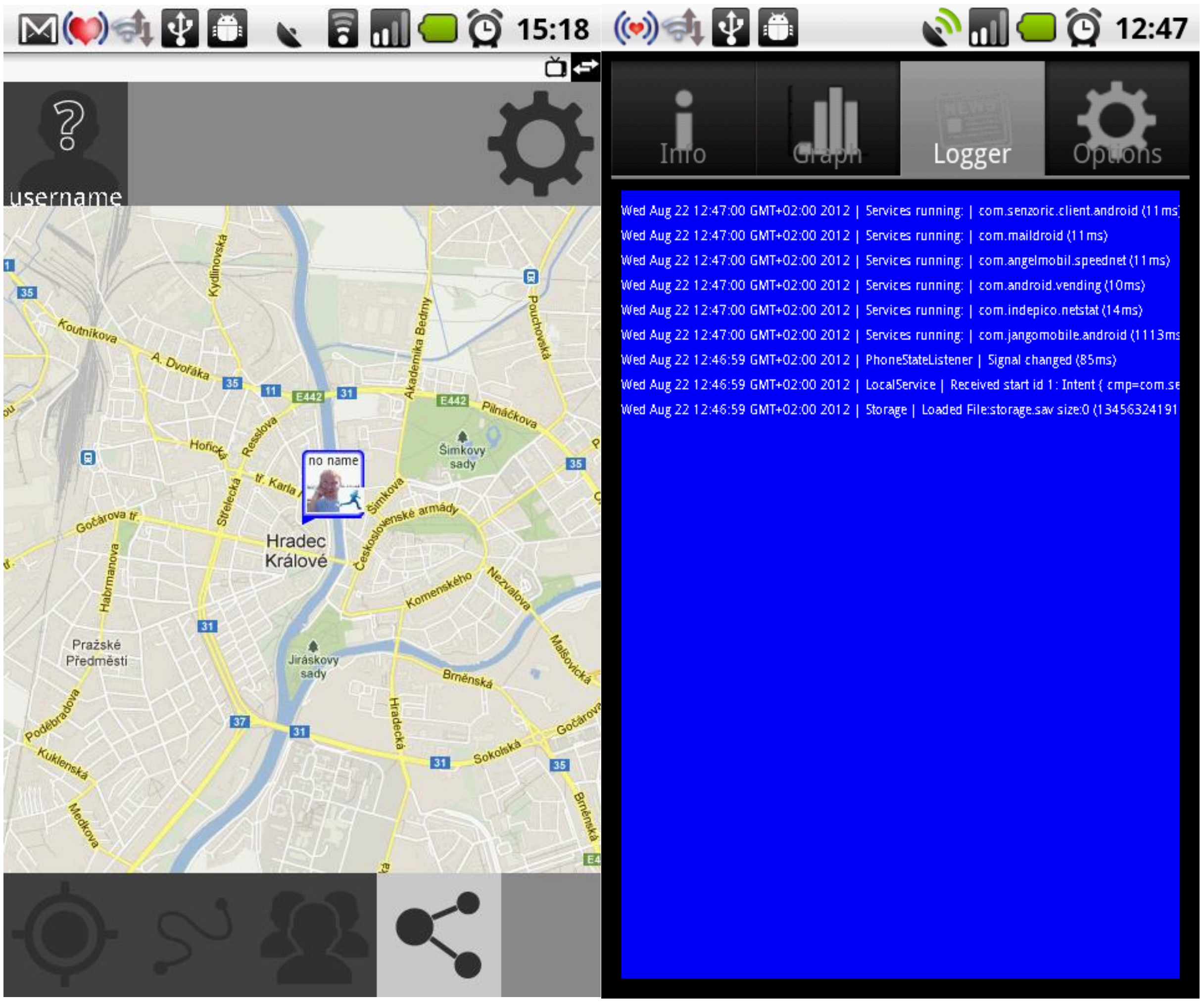
As future extension for sensor profile layout we consider advance UI components where inputted data do not have to be entered in raw form as number but rather as touchable progress bar where exact number value is changed by sliding bar from left to right or opposite.

On the left side of [Figure 37] there are illustrated behavior pattern recognition such as walking, running and no motion. Those behavior recognition is based on monitoring accelerometer it means when intensity of measured data exceed over empirically defined threshold the engine evaluate that the user is walking, running or doing nothing and show up notification on client application as well as storing status locally which is latter on synchronized with cloud. Instead of that the environment recognition using location input data and based on user time spent on specific place the home or work environment is recognized and also the notification is provided on top status bar with local and remote synchronization. Each graphical element used in layout and notification have the same source. We call them in Android framework resources which are located in res/* directory. Regarding statically defined layout in Android applications the identification of each element is provided by `android:id` attribute of View Widget type of XML element where (`@`) is required when you are referencing any resource from XML. The plus sign (`+`) is relevant only for the first time generation the resource into `R.java` where all resources are generated during build and the id represents resource type where all ids type are grouped with unique identification. For referencing resource there are defined rules as follows in Java code and in XML files which are bind for developers together.

```
In XML   :       @|?[<package_name>:][+]<resource_type>/<resource_name>
In Java  :       R.<resource_type>.<resource_name>
```

And for referencing style attributes which are defined in theme we have to use instead of at-symbol (`@`) the question-mark symbol (`?`). Once we have defined layout with elements we can access them in application logic easily and each resource such as images can be easily provided in Java code or XML files with simple reference to single point in resource location.

Another layout visualization which have to be provided in location based services is map layout viewed in the following [Figure 38] on the left side. We providing the localization of measured events base on geo graphical information and Google maps API. We implementing overlay on the top of Google maps and throughout our own visual definition of items in overlay and Google API the map is visualized with overlaid items. The item is composited from personal image, personal name and current activity recognized. The personal image is externalized from social connector such as Facebook, Gplus, Twitter or nonsocial identification such as Gavatar which is based only on valid email. Personal name can be only externalized from social connectors and current user activity is provided by our client application.

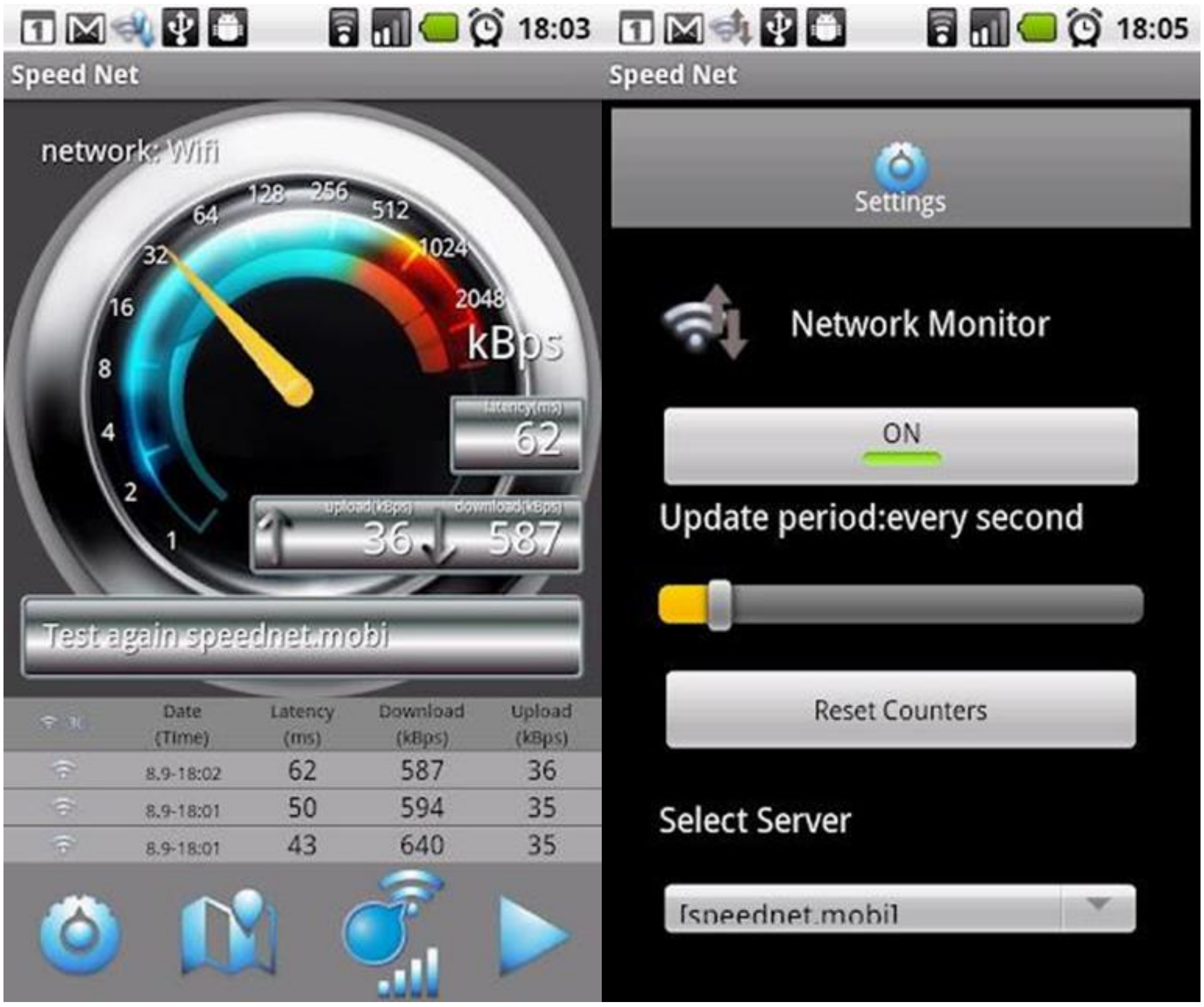


```
Wed Aug 22 12:47:00 GMT+02:00 2012 | Services running: | com.senzoric.client.android (11ms
Wed Aug 22 12:47:00 GMT+02:00 2012 | Services running: | com.maildroid (11ms)
Wed Aug 22 12:47:00 GMT+02:00 2012 | Services running: | com.angelmobil.speednet (11ms)
Wed Aug 22 12:47:00 GMT+02:00 2012 | Services running: | com.android.vending (10ms)
Wed Aug 22 12:47:00 GMT+02:00 2012 | Services running: | com.indepico.netstat (14ms)
Wed Aug 22 12:47:00 GMT+02:00 2012 | Services running: | com.jangomobile.android (1113ms
Wed Aug 22 12:46:59 GMT+02:00 2012 | PhoneStateListener | Signal changed (85ms)
Wed Aug 22 12:46:59 GMT+02:00 2012 | LocalService | Received start id 1: Intent { cmp=com.se
Wed Aug 22 12:46:59 GMT+02:00 2012 | Storage | Loaded File:storage.sav size:0 (13456324191
```

Figure 38 Client Android Application – Map & Logger

For debugging purposes we added `LoggerView` where logging information are outlined based on logging level applied [Figure 38] right side. There are several logging levels which differs from Java and Android which are `ERROR`, `WARN`, `INFO`, `DEBUG`, `VERBOSE`. The log is also stored locally with log rotation enabled due to sparing disk space of mobile device and in case of cash provide relevant information.

We consider that the layout can change in time to provide efficient and comprehensive user interface and also we are planning to add once implemented functionality in already published works [Imp1] where application such as Speed Net Tester [Figure 39] and Wi-Fi Connectivity Test [Figure 40] was proposed.

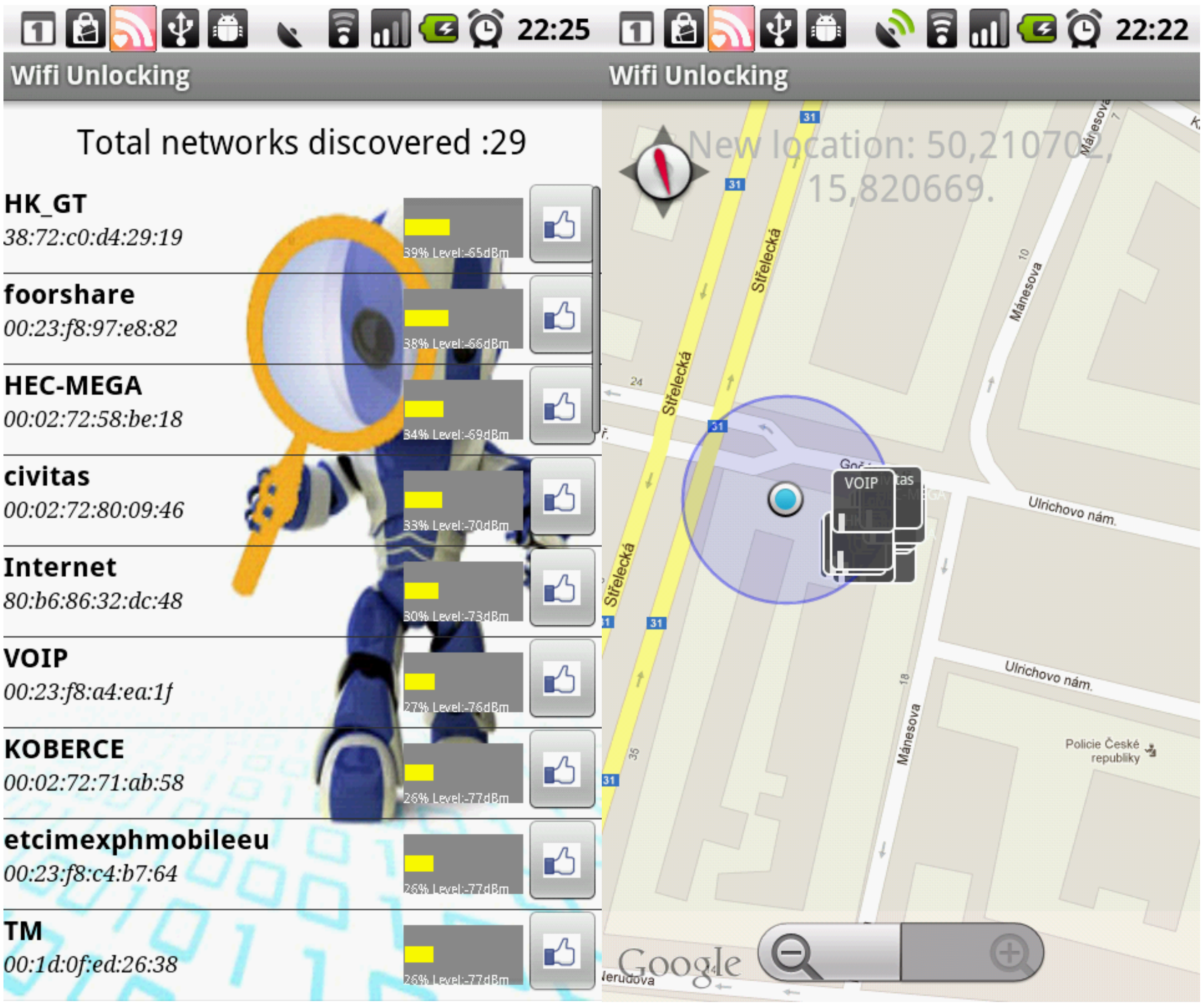


	Date (Time)	Latency (ms)	Download (kBps)	Upload (kBps)
	8.9-18:02	62	587	36
	8.9-18:01	50	594	35
	8.9-18:01	43	640	35

Figure 39 Client Android Application – Speed Net Tester [Imp1]

The Speed Net Tester is dedicated to testing the throughput and Round Trip Time (RTT) of current network connection. The Android client application provides network testing results in real time by speedometer where current value is viewed by pointer on logarithmic scale on left side of [Figure 39]. The test is started by start on the very first right test button in the bottom navigation bar. The result of passed speed measurements are also viewed in list view in the middle of the screen with attribute Network type, Date/Time, Latency in ms, Download speed in kBps and Upload speed in kBps. If location provider is enabled the results can be even viewed on the map view. On the right side of [Figure 39] there is setting of application where periodicity of measurements can be applied by progress bar in the middle of screen. Also reset of global counters is presented to clean out previous statistics of measurements. And the last is backend server connection choicer to choice which server should be measurement mirror provider.

The Wi-Fi Connectivity Tester using scanning of surrounding environment to find out available Wi-Fi access points. The client application scans in intervals active signal and trying to connect to those which are without password authentication. Once the connection is establish between client application and trusted server the access point is considered as free connectivity resource for our sensorial network. To use such open free Wi-Fi based network for connection extremely speeds up throughout of data and increase the battery capacity significantly.


22:25
Wifi Unlocking
Total networks discovered :29

HK_GT
38:72:c0:d4:29:19
39% Level:-65dBm

foorshare
00:23:f8:97:e8:82
38% Level:-66dBm

HEC-MEGA
00:02:72:58:be:18
34% Level:-69dBm

civitas
00:02:72:80:09:46
33% Level:-70dBm

Internet
80:b6:86:32:dc:48
30% Level:-73dBm

VOIP
00:23:f8:a4:ea:1f
27% Level:-76dBm

KOBERCE
00:02:72:71:ab:58
26% Level:-77dBm

etcimexphmobileeu
00:23:f8:c4:b7:64
26% Level:-77dBm

TM
00:1d:0f:ed:26:38
26% Level:-77dBm

22:22
Wifi Unlocking
New location: 50,210702, 15,820669.

Figure 40 Client Android Application – Wi-Fi Connectivity Tester [Imp1]

The reservoir of access point measurements can also serves for in-door localization based on fingerprinting where minimal batter consumption can be reached to extract location with comparable precision to Global Positioning System (GPS).

6.3.2. Application Logic

After we defined application layout we can start to bind view with application logic. In Android framework there is Java class called Activity which is binding together UI and application logic. It is basic fundamental principal in Android framework philosophy of application design to use activity which are in Model, View and Control (MVC) point of view the control. Each activity has to be defined in application manifest file called AndroidManifest.xml which is XML file for high level meta-information about application outlined in the following example.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.senzoric.frontend">

    <application
        android:icon="@mipmap/ic_launcher"        // define icon of app
        android:label="@string/app_name"          // define label of app
        android:theme="@style/AppTheme" >         // define theme of app
        <activity
            android:name=".ActivityMain"          // bind java class for activity
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

The most concerning in the example is definition of how activity is going to be invoked on Android mobile device. In the Android framework intent provides a facility for performing late runtime binding between different applications. Its most use is to launch the activities where it is basically a passive data structure holding an abstract description of an action to be performed. In our example there is defined XML element called `intent-filter.`

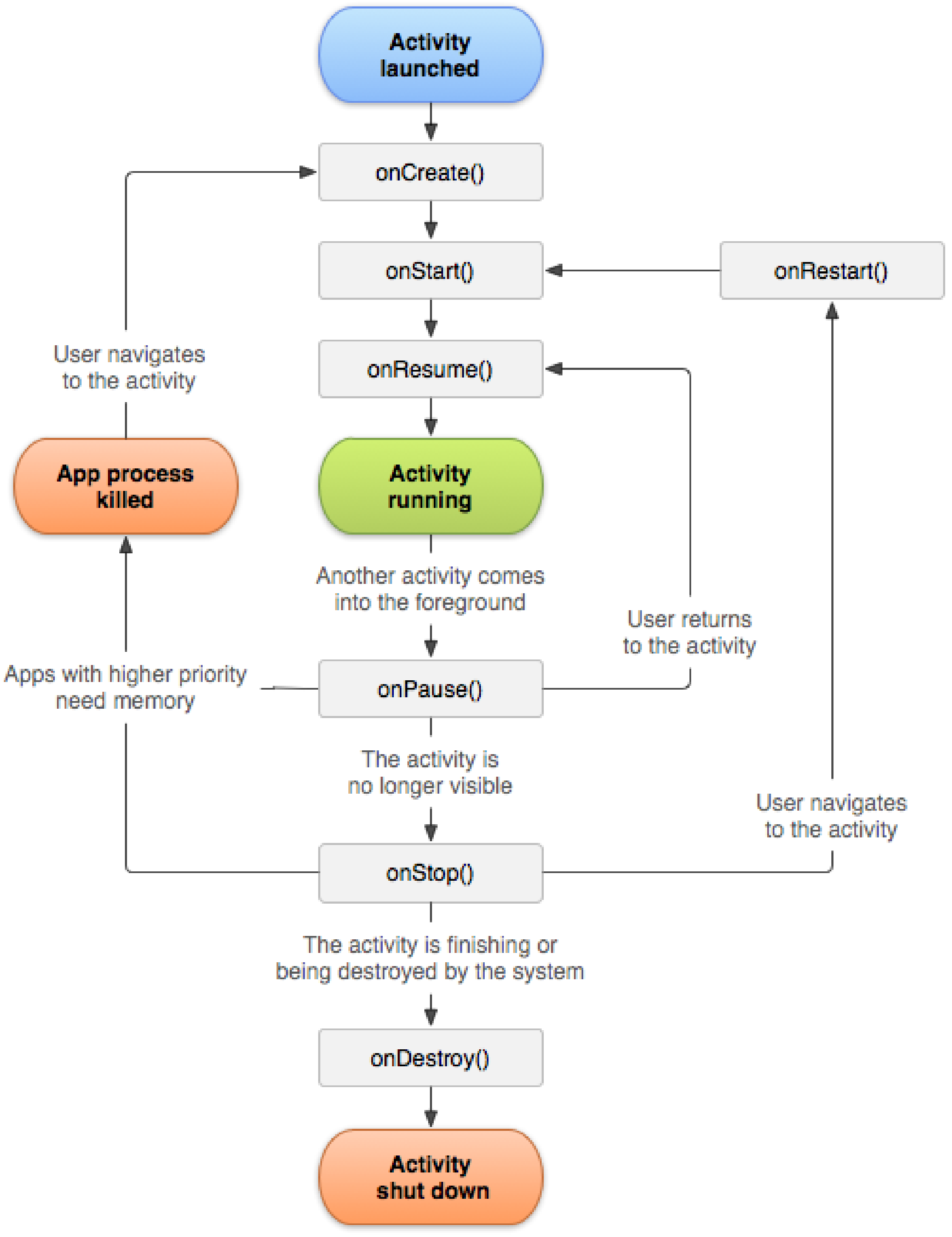

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Figure 41 Android Activity Lifecycle [37]

The intent filter providing kind of fish net to catch correct intent specified by `action` attribute with unique identification in `android:name` and when such intent is broadcasted over Android system the activity is invoked. Once the activity is being launched it goes through defined lifecycle model viewed in the following [Figure 41]. The developers are able to inherit Activity type and override each of `on<EVENT>()` function to react according to application needs.

In our case of activity we use `onCreate()` for static view initialization and binding model/data with background service and `onPause()` for saving persistent data to database or configuration files. The others callbacks such `onRestart()` and `onStart()` are called just before application becomes visible. Then `onResume()` is called just before user can interact with activity and that means the activity is on the top of activity stack. Lastly `onStop()` is called when activity is not visible and `onDestroy()` is called when activity is finished or destroyed by the system. The callbacks `onStop()` and `onDestroy()` may not be called at all due to the system process resource handling.

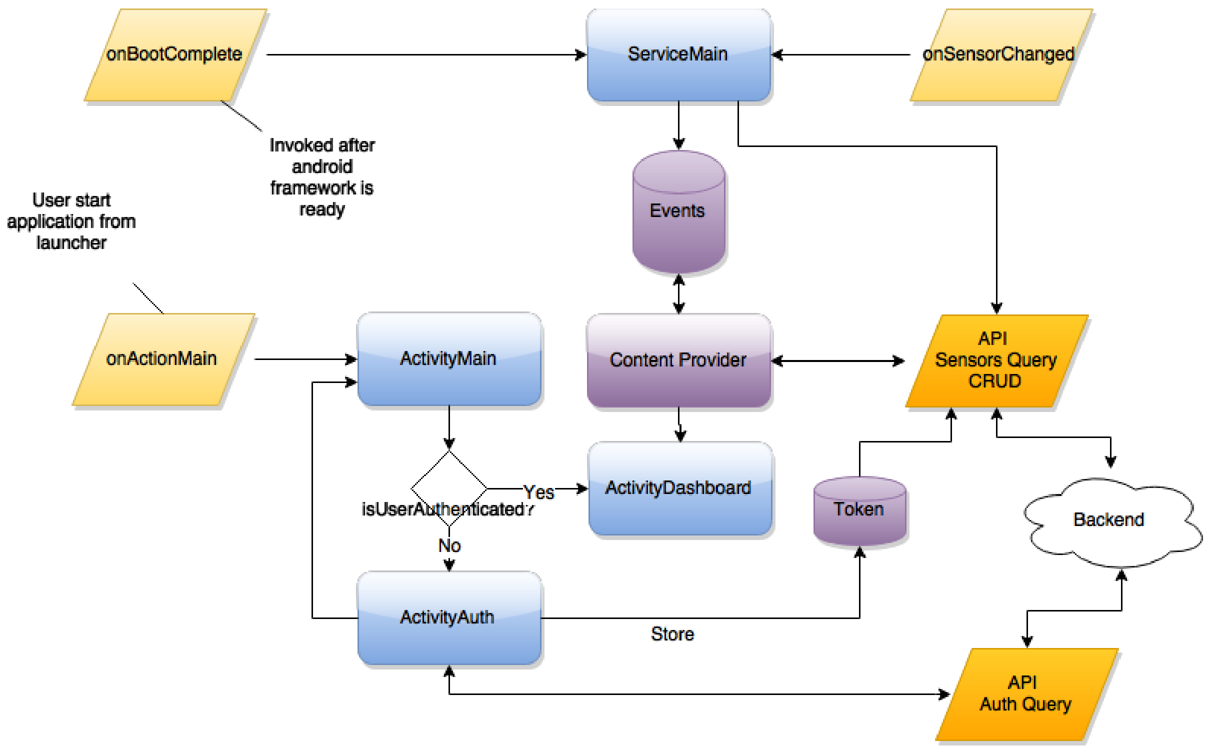


Figure 42 Application Logic

And now when we understand activity lifecycle we can implement global application logic how should be activities coupled together and react on events. As was mentioned we can call activity from another activity via intent or by listening broadcasted intents within system such as `onBootComplete`, `onActionMain` or `onSensorChanged` visualized in the following [Figure 42]. The background service `ServiceMain` which is responsible for gathering data from sensors and storing them in effective way to local database where content provider is able to populate them to be used externally.

We consider user workflow scenario started with user intent by launching application from launcher and jumps to `ActivityMain` which is responsible for static initialization control flow. The control is given to activity `ActivityAuth` where all authentication tasks are performed with backend. Once the valid token is received from backend and stored locally in secure store the control is given again to `ActivityMain` with success result code and `ActivityDashboard` can be started. Now on dashboard operates with Android fragments so there is no extra need for another activity. The `ActivityDashboard` provides all necessary data for UI from local and also from remote datasets with valid token. If token expires over time the control is again given back to `ActivityMain` and authentication proceeds again. We will describe secure storage and authorization procedures in next chapters also with Fragment handling in Dashboard view but at first we will focus on providing content in Android Framework in the next chapter.

6.3.3. Data Storage

Loading and storing data on mobile device is the key part of each mobile application. There are some significant read/write accesses which can be used with consideration to operation speed or data size amount. We are using all of methods which are outlined in the following [Table 18]. Each of them is dedicated to its purpose and benefits application responsiveness and better user experience. We are using in-memory caches for calculation global patterns of sensors data where amount of that data is not possible to store in a real time and also it is waste of space. Therefore we are using cache like system to provide calculated samples of data based on intensity in precise delta timing.

Table 18 Mobile Data Storages

METHOD NAME	DESCRIPTION
IN-MEMORY HASHMAP	Dedicated to cache types of storages without persistency and the high data throughput
DATABASE CRUD	Selective search with persistency and reasonable speed of response
FILE READ/WRITE	Suitable for huge amount of media or raw data types
NETWORK REQUEST/RESPONSE	Scalable storage with needs of internet connection

As next we need database like storage for storing and loading sample patterns and for such reasons there is in Android Framework by default embedded SQLite database engine which provides common SQL commands functionality for mobile application. But there is one difference between commonly known SQL database engine and SQLite engine and that is access controlling which is not provided for SQLite. That means if we want to use access control on Android Framework we have to use kind of wrapper on SQLite database engine and by default there is such solution available and is called Content Provider. In essence it establish read / write permissions for users which are defined in Android manifest file and outlined in the following example.

```
<provider
    android:name=".SensorContentProvider"
    android:authorities="com.senzoric.frontend.provider"
    android:grantUriPermissions="true"    // grant perms via URI
    android:readPermission="com.senzoric.frontend.provider.READ"
    android:writePermission="com.senzoric.frontend.provider.WRITE"
    android:enabled="true"
    android:exported="true" >             // allowed for third parties
</provider>
```

For data access by external application like the 3rd party developers we have to specify `android:exported="true"` and also permission Uniform Resource Identifier (URI) in our case `com.senzoric.frontend.provider` for dynamical and static read and write permission definition where `grantUriPermissions="true"` means dynamical defined by generated intent from external application with correct URI and request. And on the other hand we can also define permissions statically in Android manifest file of client application with explicitly defined read or write permissions as follows.

```
<uses-permission android:name="com.senzoric.frontend.provider.READ">
<uses-permission android:name="com.senzoric.frontend.provider.WRITE">
```

Once we deal with all permissions configuration we can now easily use content provider cross whole Android platform within any application by following URI read and write access.

```
content://com.senzoric.frontend.provider/events
```

6.3.4. Client API

For exchanging request/response type messages between backend and frontend we design Application Programing Interface (API) based on Representational State Transfer (REST) which is the simplest way for Create, Read, Update, Delete (CRUD) information on server with HTTP calls. Each entity has its own unique identifier known as Uniform Resource Identifier (URI) and with basic CRUD operation we are capable to manage entity content remotely. To handle such implementation in our framework easily we are using shared library program code called `senzoric-commons` where classes are shared between backend and frontend code. In shared library are blueprints for class models and API interfaces. For instance following code represents `AuthService` interface to define server and client side API interface implementation with type safe constrains.

```
public interface AuthService {           // shared interface in library
    @POST("/auth/register")              // HTTP mapping on post method and path
    public Token register(@Body User user);

    @GET("/auth/login")                  // HTTP mapping on get method and path
    public Token loginWithEmail(@Query("email") String userName,@Query("password")
String password);

    @GET("/auth/recover")                // HTTP mapping on get method and path
    public Boolean recover(@Query("email") String email);
}
```

On the client side the implementation of `AuthService` is pretty straightforward because we are using external library called `retrofit` which wraps Android `DefaultHttpClient` and `URLConnection` to REST oriented client `RestAdapter` by simple commands as follows

```
static final String apiUrl = "https://api.cexbit.com";
RestAdapter restAdapter = new RestAdapter.Builder().setEndpoint(apiUrl).build();
AuthService authService = restAdapter.create(AuthService.class);
Token token = authService.loginWithEmail(email, password);
```

The `AuthService` is instantiated by `retrofit` library and all methods are from now on available by simple calling of request method invocation anywhere throughout the application. The library has its own thread pool so it can be called even from main foreground activity thread with non-blocking aspects on application. The API request can use HTTP request methods as classical GET, POST, PUT, DELETE, and HEAD where the method input parameters are used as containers for content. The Java annotation identifies specific type of method input parameters (see blueprint of `AuthService` interface). The library covers several Java annotations for method input parameters which specifies how should be HTTP request generation handled in more detail in the following [Table 19].

Table 19 HTTP Request Input Method Annotations

ANNOTATION	DESCRIPTION	USAGE
`@BODY`	Any Java object is serialized and deserialized by `RestAdapter` converter and used for HTTP request content.	`GsonConverter` / JSON `JacksonConverter` / JSON `SimpleXMLConverter` / XML `ProtobufConverter` / Streaming
`@FIELD`	Form type field of HTTP request commonly goes with POST or PUT in query string as content.	Can be also with `@FormUrlEncoded` to be encoded where `key=value` are defined as follows `@Field(“key”) String value`
`@HEADER`	Setting up HTTP header by string value for instance authorization token value.	For using headers as `name: value` we have to use method input parameter defined as follows `@Header(“name”) String value`
`@PATH`	In most cases we use path parameter together with GET method for working with entity identification such as ids.	First we have to declare path location for replacement with `@GET(“/users/{id}”)` by curly brackets and then use method input parameter as follows `@Path(“id”) int value`
`@QUERY`	The queries are often used for parametrization of web content defined as string after question mark character in HTTP request URI parameter.	Usage for query is simple for `?name=value` by method input parameter as `@Query(“name”) String value`

Once the requests are correctly generated on frontend side we have to also do implementation how to handle responses on backend side in the next chapter backend implementation.

6.4. Backend implementation

There are many ways how to implement our backend and we choose the one which is based on the String Framework convenient, easy to use and really rapid for developing application in Java. In essence the Spring Framework is the movement forward from old conventional rules to fresh and reasonable easiness for developers through comprehensive solutions. When we consider web based services throughout Spring Framework for our backend there it come `String-Boot` which combines all classical web containers based on XML configuration files but for us rather than static useless old school approach we prefer Java annotation based configurations in minimalistic form outlined in following code. The whole application is in couple of lines of code where main as usual is entry point to Java application and `@SpringBootApplication` annotation which are joining all following annotations `@Configuration`, `@EnableAutoConfiguration` and `@ComponentScan` into one since they are used commonly together. Basically that means it will search all components in the classpath and inject instances or configure bean where is also defined by annotation for instance `@Autowired`.

```
@SpringBootApplication   // combines component scanning and auto configuration
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);   // run standalone app
    }
}
```

Such web embedded container Java application can be started by single command as follows by Java as jar, by building system such as Maven or by Gradle.

```
java -jar application.jar
mvn spring-boot:run
gradle bootRun
```

Basically now web server is running on default port 8080 with embedded tomcat web server container. Further we describe implementation of main parts of backend system such as core logic, used databases and web service. When we need to change server configuration the most convenient way is one configuration file embedded in application archive which differs by test mode or production mode. By default the Spring framework using `application.properties` file which have to be located in case of test environment in `src/test/resources` directory and for production environment in `src/main/resources` directory. For instance in our backend project the following parameters are related to customize database attributes.

```
db.host=db.senzoric.com
db.port=27017
db.admin.login=backend
db.database=senzoric-test
```

By two configuration files we can easily use one configuration for test database and attributes and for production. The mapping of key/value parameters from properties file to application instance is straightforward. We have to define for each parameter `key=value` the specific Java class which is enabling mapping throughout Java annotations. The Spring framework search for classes which are annotated with `@Configuration` and proceed by default with `application.properties` file to map values into class fields by annotation `@Value("${key}")` and the value is injected directly to field object or primitives.

```
@Configuration                      // auto mapping of config file management
public class AppConfig {
    @Value("${db.host}") private String host;     // map db.host to variable host
    @Value("${db.port}") private int port;        // map …
    @Value("${db.admin.login}") private String login;
    @Value("${db.admin.password}") private String password;
    @Value("${db.database}") private String database;
    public @Bean MongoClient mongoClient() throws Exception {
        return new MongoClient(host, port);       // provide mongo database client
    }
}
```

Everywhere in backend web application we are able to use `AppConfig` class by `@Autowire` annotation and use predefined values for different environments.

6.4.1. Core Logic

The key principal of backend server is about request / response handling throughout Hyper Text Transfer Protocol (HTTP) where the knowledge how HTTP works come a handy. Our implementation is based on Spring framework which brings to developers intention to be focused on their own business logic of application rather than rewrite once invented wheel. For better understanding the backend implementation we outlined in the following [Figure 43] the request / response process flow principal. We using external parts of Spring framework which are pictured with blue color and our internal implementation is colored by yellow.

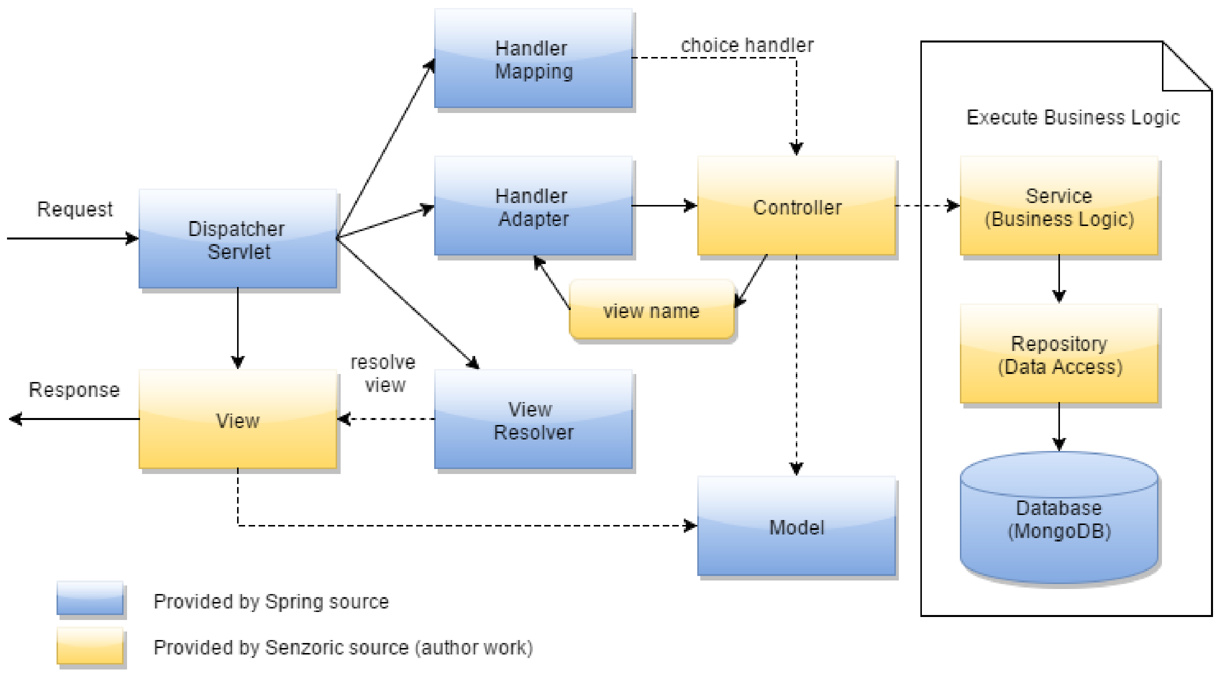


Figure 43 Backend request / response process flow

At first the request is received on server via `DispatcherServlet` which immediately dispatch the task to `HandlerMapping` for selection the appropriate controller by mapping defined in `Controller` and returns selected `Handler` and `Controller` back to `DispatcherServlet`. Then the task is dispatched to `HandlerAdapter` which calls the business logic of `Controller`. Business logic with persistence data layer are being processed and then the processing result is set into `Model` and returns the logical name of view to `HandlerAdapter`. `DispatcherServlet` dispatches the task of resolving the `View` corresponding to the `View` name to `ViewResolver` and returns the `View` mapped to View name. `DispatcherServlet` dispatches the rendering process to returned `View` which renders `Model` data and returns the response. All services are designed to server itself purpose as gathering date from / to database or processing data bundles. The services can be handled independently no matter what they are servicing. The main reason for such architecture is about to distribute request in proper order and way which suits to developers scenarios where the main aspect for implementation is about to design effective way.

6.4.2. RESTfull API

To implement RESTfull web services we have to be quite familiar with Representation State Transfer (REST) HTTP request / response messaging. In most Java API for RESTfull Web Service (JAX-RS) based on Java Specification Requests (JSR-311) framework implementation are used Java annotation on classes which handles HTTP requests without needs to explicitly define configuration in XML based files such as `web.xml` and others. Just to mention some of them which implements JAX-RS such as Jersey, RESTeasy, Restlet, Apache CXF and of course String Framework. We are using the last one mentioned and therefore our implementation relies on the state of art approach for Java based web application with embedded web container where the future development style is about to compose single web application into single web container without needs of multiple web application instances tuning and cross application bugs tracking. The proposed solution is lightweight easy to maintain and scalable based on Model View Control (MVC) fundamentals. For instance following code for `UserController` implements `AuthService` from shared library `senzoric-commons` with Java annotation based HTTP request mapping.

```
@RestController          // Spring rest controller mapping
public class UserController implements AuthService {
   @Autowired
   private UserProvider userProvider;
   @RequestMapping(value=AuthService.URL_REGISTER, method=RequestMethod.POST)
   public Token register(User user) { …
        // register user into system
   }
   @RequestMapping(value=AuthService.URL_LOGIN, method=RequestMethod.GET)
   public Token loginWithEmail(String email, String password) { …
        // login user
   }
   @RequestMapping(value=AuthService.URL_LOGIN, method=RequestMethod.POST)
   public Token loginWithEmailEncryptPassword(String email, String secret) { …
        // login user with secured pass in md5 hash
   }
   @RequestMapping(value=AuthService.URL_RECOVER, method=RequestMethod.GET)
   public Boolean recover(String email) { …
        // recover user via email and secret
   }
}
```

As you can see the controller receives the data for instance in form of JSON which is depended on used data convertor. The requests are receiving the data form on method input values and are being processed by inner implementation which is basically covered by some checks and by some data manipulation operation. Controller distribute data to database or in memory cache where can be delivered over the network to multiple clients. To use Spring Framework `Controller` or `RestController` correctly is the about to define proper request mapping defined by Java annotations. Following several combinations of annotations describe request mapping in more detail.

JAVA ANNOTATION	DESCRIPTION/EXAMPLE
@REQUESTMAPPING	Core request mapping annotation used on class or class method in order to define match between request and processing method. Usage `@RequestMapping("/users")`
@PATHVARIABLE	Path variable mapping where define variable as pattern in given URI as follows `@RequestMapping("/users/{userID}")` the variable itself is mapped on method input variable as follows `findUser(@PathVariable String userID)`
@REQUESTBODY	Maps the whole object as method input parameter as follows `createUser(@RequestBody User user)` the content type JSON and POST method is required in order to use default Jackson converter.
@REQUESTPARAM	Annotation used for request mapping query parameters where `@RequestMapping("/users")` is base of URI and `?userID=<value>` is query string which is mapped as follows `findUser(@RequestParam("userID") int userID)`

Once we understand in more detail the request mapping scenario we are able to implement the whole scale of HTTP request API handlers. After all sets of request handlers are implemented they are implicitly activated by injection dependency of Spring framework and therefore the implemented services should be as much different as possible. We consider to proceed the data in services as independent to main request stream. The most effective way how to implement backend services is about to effective middle layer processing. The less of procedures mounted on request controller is less processing time consuming.

6.4.3. Services

The most of business logic implementation in our concept should be implemented in kind of services which should be independent to data store layer and request handler layer. The services are core functionalities defined within the system to serve user cases to fulfill designed goals. For instance the email service on our backend server as mail client is designed to proceed all email requirements via Simple Mail Transfer Protocol (SMTP) dedicate to remote Mail Transfer Agent (MTA) as mail server with validation based on Sender Policy Forward (SPF) and Domain Keys Identified Mail (DKIM) where the Secure Socket Layer (SSL) or Transport Layer Security (TLS) secured layer should be used for connection between mail client and server since the Simple Authentication and Security Layer (SASL) we using SMTP credentials exchanging in plain text form.

We implemented authentication service two types of public and private entities where public are accessible to all frontends without authorization and private once are accessible only after correct authorization process based on provided credentials or short term tokens.

6.4.4. Data Storage

We decided to use as a main database engine the MongoDB where its qualities were proven over the time in many cases. MongoDB is document based database where documents are similar to JSON objects. The values of fields may include other document, arrays and arrays of documents. It supports dynamic schema definition which can change in real time without data constrains. Provides high performance data persistence where index supports faster queries together with high availability provided via replication facility called replica sets. Also automatic scaling where horizontal scalability is part of core MondoDB functionality with automatic sharding distribution data across a cluster of machines. Once we have database engine installed on server and it runs in separate process listening on default port `27017` we can start to use Java database clients to connect on localhost. We are using for interaction with database engine the database client implementation based on Spring Framework via `spring-data-mongodb` in the following source code is defined interface which coupling data manipulation operations by annotations. The `MongoRepository` class binds together all basic CRUD operation with defined entity `UserEntity` and implements customized once by input parameter and class field identification. Such customization can be also enhanced by Java annotation support Java Persistence API (JPA) outlined in following code of simple entity `UserEntity` where identification `@Id` is mandatory for all entities definition. The `@Id` annotation defines the unique identifier for each entity and can be used for indexed search and joins of entities throughout database. In our `UserEntity` definition we inherited `User` class which is basically kind of blueprinted container for JSON data transportation between frontend and backend and on the backend side we are adding database related fields with Mongo database annotations.

```
public class UserEntity extends User {
    @Id                  // must be included for database entity
    String id;
    public UserEntity() {
    }
    public UserEntity(String email, String password) {
        super(email, password);
    }
    … // getters & setters
}
```

There are other useful annotations which help to drive database mappings for `MappingMongoConverter`. An overview of the annotations is provided below [38]

- `@Id` - applied at the field level to mark the field used for identity purpose.
- `@Document` - applied at the class level to indicate this class is a candidate for mapping to the database. You can specify the name of the collection where the database will be stored.
- `@DBRef` - applied at the field to indicate it is to be stored using a com.mongodb.DBRef.
- `@Indexed` - applied at the field level to describe how to index the field.
- `@CompoundIndex` - applied at the type level to declare Compound Indexes.
- `@GeoSpatialIndexed` - applied at the field level to describe how to geoindex the field.
- `@Transient` - by default all private fields are mapped to the document, this annotation excludes the field where it is applied from being stored in the database.
- `@PersistenceConstructor` - marks a given constructor - even a package protected one - to use when instantiating the object from the database. Constructor arguments are mapped by name to the key values in the retrieved DBObject.
- `@Value` - this annotation is part of the Spring Framework. Within the mapping framework it can be applied to constructor arguments. This lets you use a Spring Expression Language statement to transform a key's value retrieved in the database before it is used to construct a domain object.

Once we define the entity class with appropriate annotations we are ready to define repository access for that type of entity in the following source code we outlined interface `UserRepository` which is extended from `MongoRepository` from Spring Framework and it provides the intelligent database CRUD operation based on naming convention. All methods starts with findBy<attribute> define search in database documents by specific entity where <attribute> is used as search criteria with value entered as input value of method. It can be combined together with logical operators such as `And`, `Or`, `Between`, `LessThan`, `GreaterThan` or `Like`. The outcome would look like for search user via email and last name as `findByEmailAndLastname(String email, String lastname)` that kind of strategy really minimize boilerplate code and increase clarity of code.

```
public interface UserRepository extends MongoRepository<UserEntity, String> {
    public UserEntity findByEmail(String email);  // db search users via email
    public UserEntity save(UserEntity user);      // store user into db
    …
}
```

6.5. Continues Integration

The integral part of the development is Continues Integration (CI). These sets of techniques and tools are enabling developers to integrate their incremental piece of work within teams and allowing to dramatically reducing amount of time wasted for searching defects which can be automatically found by specialized tools. We describe in this chapter commonly used CI tools as well as criteria and advantages for those who are using this proved by time approach. First we start with a bit of history where in early sixties of the last century the IBM came out with a special method for building processes which were triggered multiple times per day. In couple of decades most companies producing software solutions were realizing similar processes based on daily automated building environments and servers. The main reasons for usage of CI techniques and run CI servers are following hints.

- High bugs rate in source code of developers
- To find out single bug is time and resource much consuming
- Creation of new builds is sophisticated process
- New builds with minimal effort can be created daily
- Inconsistence of release versioning leads to automation
- Simplification of repository access and its maintenance

Those are just a few common criteria leads to the necessity of usage Continuous Integration. To demonstrate classical integration server with basic functionality we highlight in the following [Figure

44] of high level CI process flow. At first developers are committing source code to remote shared repository where the code is being downloaded. Once there are no conflicts in the source code and is delivered to repository process flow then can start compilation where all static syntax bugs are resolved immediately. Then there can be provided test cases implementation and those can be automatically tested by framework without human needs. If tests are valid then it goes further deep to code analysis where the knowledge of programing rules is applied on code and it searches for error, warning or even trivial issues which can be resolved. Next step is documentation generation where for distribution purposes and descriptive capabilities to other parties the document type outcome is generated mostly based on code and its annotations. Lastly after all checks are passed then the binaries are distributed to specific location of production.

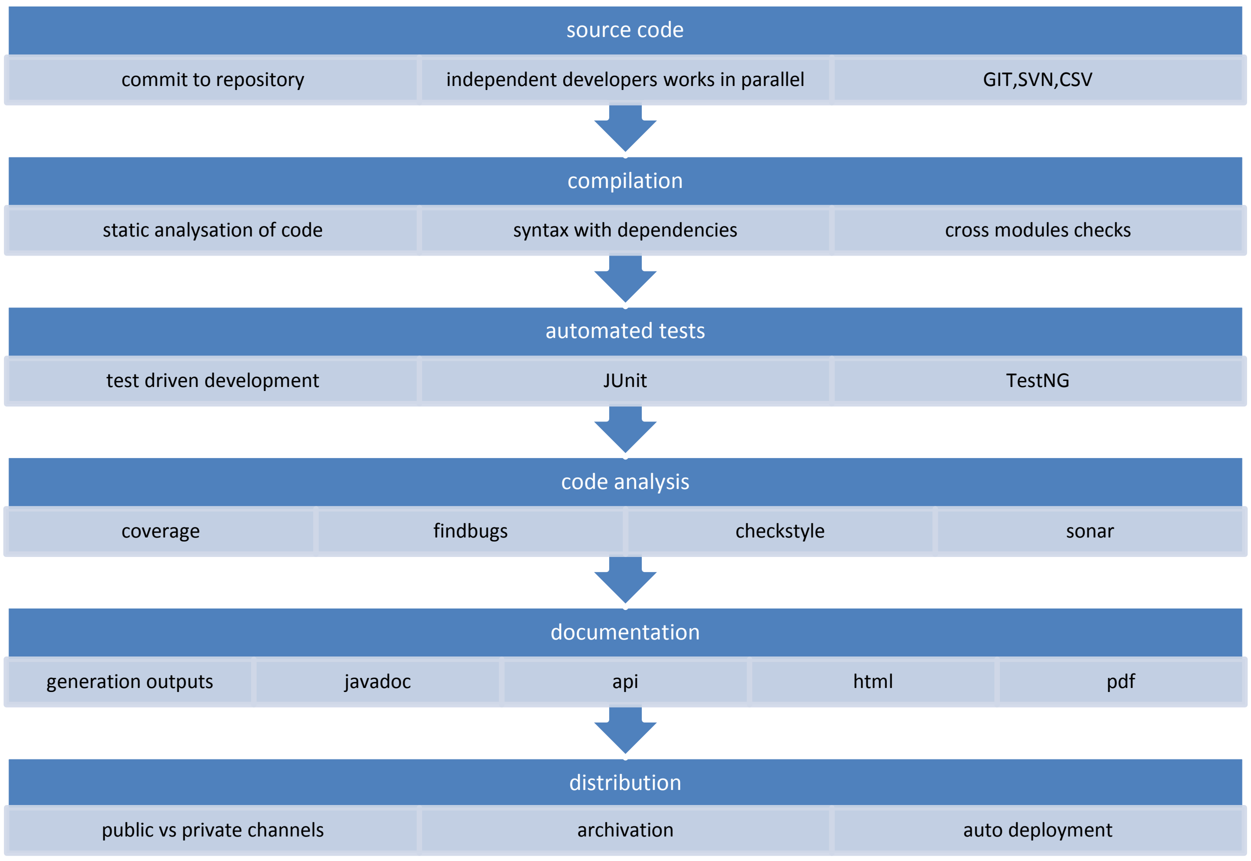


Figure 44 Continuous Integration Workflow

Each sub process of continuous integration process flow is basically provided by different tool which are all together and are invoked from CI server with specific inputs and outputs. There are several well-known CI servers such as Apache Continuum, BuildBot, Hudson, Jenkins or Bamboo. They perform process flow based on customized configuration and run the builds in a certain period of time typically during the night. Some of building tasks can be quite time consuming for instance Android source build have to run around one hour to build all binaries on high performance server and therefore the time management of builds in such cases has to be also considered.

We are using in our development continuous delivery Jenkins which is Java web application with build management system and possibility of many plugins for application. The Jenkins worker has to have access to shared repository which is in our case GIT and in our case on different server so the most convenient way is to create Jenkins account with SSL based authentication. The repository structure is defined previously in [Table 17] and is used by build server as well as by developers. Once the building process is started all source code is cloned to working directory within Jenkins worker and is being compiled. The way how to tell Jenkins what to do with source code is typically project management file `pom.xml` in case of Maven build tool or in our case `gradle.build` of Gradle build tool. Such file defines all necessary information for build locally such as which external resource has to be downloaded, definition of versions, definition of signing binaries for security reasons and how should be binaries or documentation for source code distributed. Following code outlined backend Gradle build definition file where are mandatory information specified.

```
group 'com.senzoric'
version '1.0-SNAPSHOT'
buildscript {
   repositories {
        jcenter()
    }
    dependencies {
      classpath 'org.springframework.boot:spring-boot-gradle-plugin:1.2.5.RELEASE'
    }
}
apply plugin: 'java'
apply plugin: 'idea'
apply plugin: 'spring-boot'
jar {
    baseName = 'senzoric-backend'
    version =  '1.0'
}
repositories {
    jcenter()
}
sourceCompatibility = 1.8
targetCompatibility = 1.8
```

```
dependencies {
    compile 'org.springframework.boot:spring-boot-starter-web'
    compile 'org.springframework.data:spring-data-mongodb'
    compile 'org.slf4j:slf4j-api:+'
    compile 'ch.qos.logback:logback-classic:+'
    compile 'org.codehaus.jackson:jackson-mapper-asl:1.9.11'
    compile project(':senzoric-commons')
    testCompile 'junit:junit'
}
```

The build file defines global repository for downloading as well as which resources and libraries are necessary for our implementation and therefore have to be downloaded locally. There are some advantages in contrast to Maven based build system. Gradle build system is based on Domain Specific Language (DSL) Groovy instead of traditional eXtensible Markup Language (XML) which provides real advantages for developers to use programing techniques directly through build file. And it really excels in multi-project builds because it supports incremental builds by intelligent determining which parts are up to date and which should be re-executed in building tree based on dependencies. There are plenty of plugins which can be used by simple command `apply plugin: ‘java’` and from that point Gradle has defined default structure of project where source, resource, test source and test resource are located in directory structure. For instance source is located in `src/main/java` but it can be easily changed by Groovy command within Java plugin as follows `sourceSets.main.java.srcDirs = ['src/java']` in build file. After build file is properly set we can call Gradle build by simple command from bash `gradle build` if environment PATH variable is correctly setup to Gradle binaries. The outcome from simple project based on Java plugin is as follows.

```
> gradle build
:compileJava
:processResources
:classes
:jar
:assemble
:compileTestJava
:processTestResources
:testClasses
:test
:check
```

```
:build

BUILD SUCCESSFUL
```

All of these steps processed by Gradle build are basement for continuous integration server how should behave in general. Each step can be configured in build file or directly in integration server management console but the good practice leads to single build file tree possible to build on any environment except specific external native builders for specific platform different from Java.

Another requirement for effectiveness of building system for frontends when we are talking about Android platform build variants. That customization of build file based on build variants enables to provide system unique identification of packages for each product flavors and their build variants. In our project we are building frontends with build variants `debug` and `release` which are by default enabled for Android builds and differs only by suffix added to each output Android Package Kit (APK) file. The product flavors are defined as `demo` and `full` where the name of flavor defines subdirectory in `src` directory where only source code or resource files are located and which should override default once in `src/main` directory. Following Gradle build file defines the build variants together with product flavor.

```
android {
    ...
    defaultConfig { ... }
    signingConfigs { ... }
    buildTypes {
        release {
            minifyEnabled false
             proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
         debug {
            debuggable true
        }
    }
    productFlavors {
        demo {
            applicationId "com.senzoric.frontend.demo"
            versionName "1.0-demo"
        }
        full {
            applicationId " com.senzoric.frontend.full"
            versionName "1.0-full"
        }
    }
}
```

There are others relevant topics regarding proper auto build procedures which may be considered such as proguard for code redundancy optimization routines in Android applications. Also testing with automatic tests which are implemented by independent developer who not implement feature itself. And at last all possible code analysis as Sonar, Coverage, FindBugs, CheckStyle, etc.

6.6. Security aspects

We consider really important the implementation of security relevant aspect for Senzoric framework on frontend side, backend side and communication in between to provide safe, robust non leak able environment. First let's talk about frontend side where the core security principals lay on Linux hardening techniques in combination of Android framework security. The Android mobile device after power on give control to secure boot and it gives control to Linux system which starts independent Java Virtual Machines (JVMs) for each application in sandbox. We consider for Senzoric framework on Android platform several security enhancements which are outlined in following list:

- **Application sandbox** - apps are isolated with their code execution and data from other apps. That is provided by Android framework naturally based on Dalvik Virtual Machine.
- **Framework security functionality** – cryptography, file system permission and secure Inter-Process Communication (IPC). We are using local keystore and truststore as repository of certificates and private keys for encryption database sensitive data as well as communication data.
- **Encrypted filesystem** – enables to protect data on lost or stolen devices
- **Application defined permissions** – control system features and user data based on application level. The permission for application defined in Android manifest file we defined to fulfill required sensors, connectivity and user access account needs.
- **Input user data validation –** performing validation of entered data prevents SQL injection to minimalize vulnerabilities.

- **Handling credentials** – user passwords are required only first time of login on specific device since than the short live access token is used and the password is not stored on frontend and on backend only in form of MD5 hash value for verification.

The backend side is little bit more complex due to security features provided by Android platform which are not as default on standalone Linux server. We using for backend Debian Linux distribution running on Virtual Private Server (VPS) in computing center where the security after server boot is supported by VPS provided since the server instance is running. We are connection to VPS via SSL terminal for setup secure environment outlined in following list:

- **Encrypt data communication –** Using SSL for terminal connection, TLS for SMTP email clients and HTTPS for frontend communication.
- **Linux system management** – Uninstalled all unnecessary software to minimize vulnerability. Using chrooted sandbox network service with Tomcat web server. Keeping automatically Linux kernel and software updates to apply security updates as soon as possible.
- **Logging and Auditing –** We using standard Linux logging and auditing system to monitor any suspicious activity on server.
- **Database security –** The access to database is restricted only for localhost. The database server runs on the same machine with the same IP address and connection is based on private / public key SSL layer. User passwords are stored in MD5 hash format to minimize risk of database administrators to compromise users.
- **Security Enhance Linux –** Implementation of Mandatory Access Control (MAC) to enhance security of Linux operation system which is considered in our Senzoric framework used in future.
- **Client web server communication –** We using for communication HTTPS with first class certificate provided by StartSSL certification authority for domain name verification and communication encryption between frontend and backend. Frontend require root certificate of certification authority to be included in trust store to verify backend.

The security of user access control to backend application is based on provided access tokens which are generated for short time period upon correct user credentials. The tokens provide access internally in backend application to user resources which were once approved explicitly by user on implicitly by user action.

6.7. Monitoring

Once we have implemented the system and deployed on production environment we need to monitor the system to be able to analyze performance and scalability of the system. We are using Java Mission Control from Oracle to monitor Java Virtual Machines (JVM) locally or remotely based on Java Management Extensions (JMX). Such remote JMX Tomcat Java instance monitoring provides tuning and easy issue resolving in production environment. To enable JMX for Tomcat web server is about to setup couple of parameters for `CATALINA_OPTS` or `JAVA_OPTS` as follows:

```
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=9010
-Dcom.sun.management.jmxremote.ssl=true
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl.need.client.auth=true
-Djavax.net.ssl.keyStore=<keystore file>
-Djavax.net.ssl.keyStorePassword=<keystore password>
```

After properly parameters configuration the Tomcat web server can start with JMX remote monitoring optionality and we are able to connect remotely via SSL by Java Mission Control (JMC) software [Figure 45]. The monitoring consists of comprehensive layout with possibility to monitor parameters, setup triggers, view memory and dig into threads.

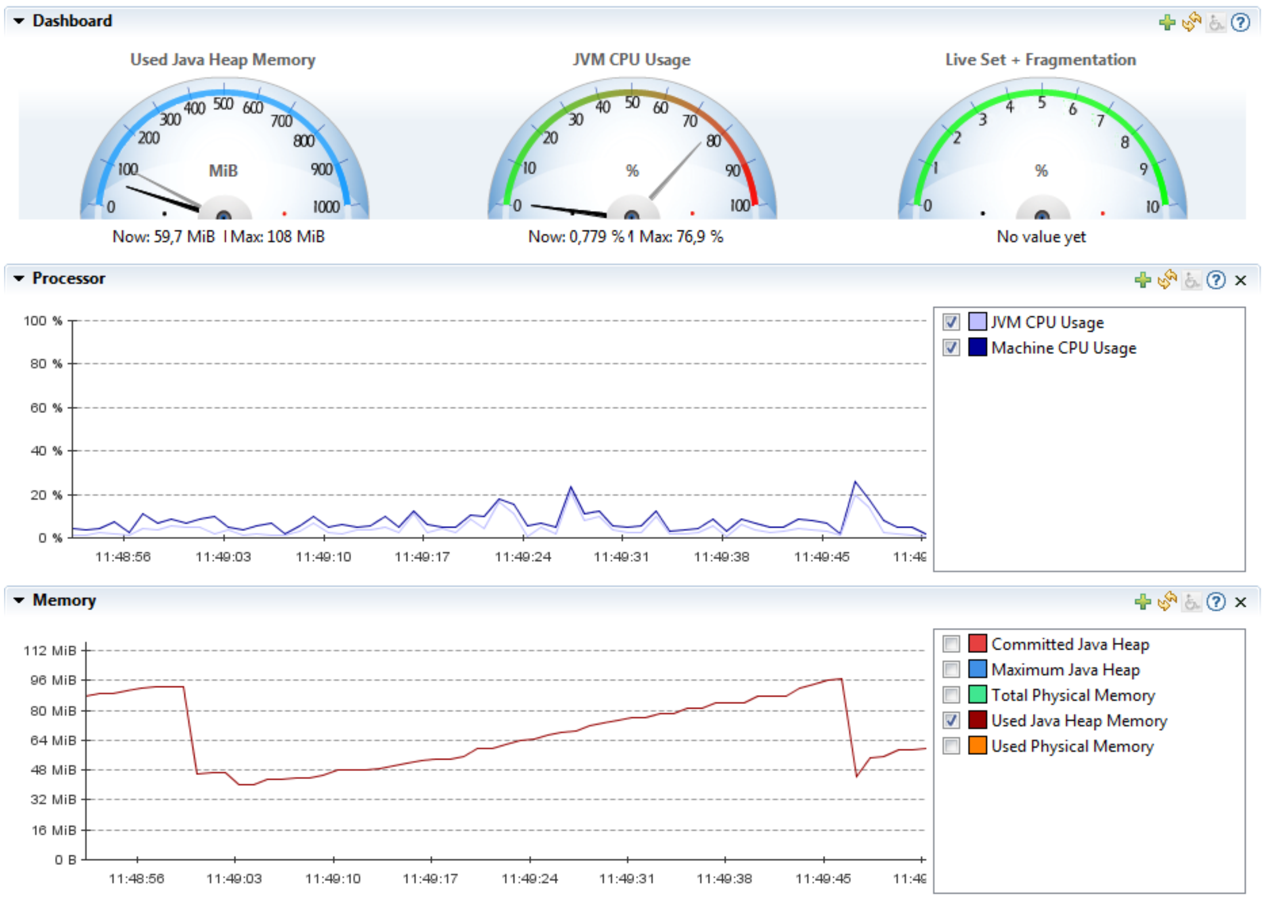


Figure 45 Monitoring JMX with JMC

6.8. Review Board

Based on definition of user stories we are prepared to schedule an activity which leads to provide outcome to working system. Each user story has its own lifecycle outlined in the following stage list

- Design – definition of functionality within system
- Implementation – incremental update of working system
- Test – testing functionality & find bugs
- Verification – consideration of requirements fulfillment
- Deployment & Production – upgrade of working system in production environment

And as such each user story has to be picked up into specific sprint where its lifecycle can start. In our project all tasks are performed by me but in a real world each edge of stages has influence on increasing software quality caused by necessity for externalized knowledge between team members. The communication between team members partially resolves the possible gaps in the user stories definition. Also during stage implementation the developers can rise a lot of technical issues related to the core system functionalities as data model and its application, class model and its application and other technical aspects which are not possible to be seen during the planning stage. We highlighted such additional tasks which have to be done prior to its own implementation of the user story in implementation chapter. Therefore we have to just plan non-technical issues which are basically user stories outlined in the following [Table 20] with attributes of story points, sprints and production review. All records can change in time and we consider table as the final one.

Table 20 User stories review board

IDENTIFIER	NAME	POINTS	FINAL STAGE / REVIEWED
USI-1-1	User registration	2	Test – add build auto testing
USI-1-2	Login	1	Test – add build auto testing
USI-1-3-1	Password recovery	1	Test – add build auto testing
USI-1-3-2	Password change	1	Design – minor
USI-1-3-3	Email change	1	Design – minor
USI-1-4	Logout	1	Implemented / Change – timeout session expiration
USI-2-1	Connect device	5	Design – request change to messaging updates during live data stream
USI-2-2	Modify name of device	1	Design – minor
USI-2-3	Disconnect device	1	Design – together with connect device
USI-2-4-1	Device list	3	Design
USI-2-4-2	Device details	2	Design
USI-3-1	Sensor list	3	Implemented – on client only
USI-3-2	Detail of sensor	2	Implemented – on client only
USI-3-3	Sensor history	5	Design

Currently we consider the implementation of prototype as partly implemented where core functionalities are available. To deploy system on production server we have to finish others use cases. And during implementation we realize another functionalities which conforms to sensing platform such as automatic social media notification feature or auto execution procedures in surrounding area which we will describe in more detail in following chapter of using sensing framework in real environments.

7. Use Cases in Real Environment

We dedicate this chapter to Sensoric Framework based solutions used in real environments. The most significant usage would be in Smart Home where everyone can recognize the advantage of home which sensing surrounding environment. We publish the work Smart Home Point As Sustainable Intelligent House Concept based on Senzoric Framework published [Conf8] and [Conf9] therefore we describe work in more detail in the following Smart Home chapter. There are others areas where SF can be useful in business, public or private environments. We consider main area in business in automation processes which are saving money to companies in business process, in public where aggregation of users may effective declare demand of public services and therefore they can be distributed more effectively, and in private mostly in combination with social media where kind of excitation of users context awareness is desired content to be consumed.

7.1. Smart Home

The idea behind Smart Home Point is easy, comprehensive and social connected UI with their home place. The system is based Sensoric Framework and enables to monitor, control and socialize householders with their facilities, families and friends. The monitoring is covering any sensor which could be connected via Ethernet network such as motion, light intensity, temperature and other ambient sensors as well as voice, video or other signal readers. The controlling enables door locking, heating, air-conditioning, lighting and networking management on spot or remotely from the mobile device or web browser. Mentioned socialization feature ability of the system enables to create new interactive environment within home users who are identified by over their personal mobile devices and could share their own social channels. The cost of chips and sensors will reach ubiquitous level soon and we will interact and provide sensorial data to our environment. The smart environment at home will have for us more attractive background due to human social and personal behavior needs. Our relaxation, comfort, social zones will increase their efficiency thanks to the intelligent sustainable environment. The houses will manage energy from recoverable sources and provide free of charge sustainable ecological live support without waste. The challenge is more than contrived where limited resources are drained in meaningless outcome in terms of human society. Therefore we propose sustainable open source project called Smart Home Point in exampled house schema [Figure 46] where all house management needs are connected in a single UI ready for any customization. Nowadays we consider personal automation as one of the biggest challenges. The personal automation stands in the line just after the industrial and business automation, but requires higher computation power. There are two factors which enable such opportunity as wireless communication and mobile computing. Both in time will become the main stream providers of personal daily bases solution resolvers or advisers in terms of location, society or other contexts. Also the human interaction with the system is considerable as mandatory where visualization and control of human environment functionalities would be as one undivided unit which keeps purpose and sense of information.

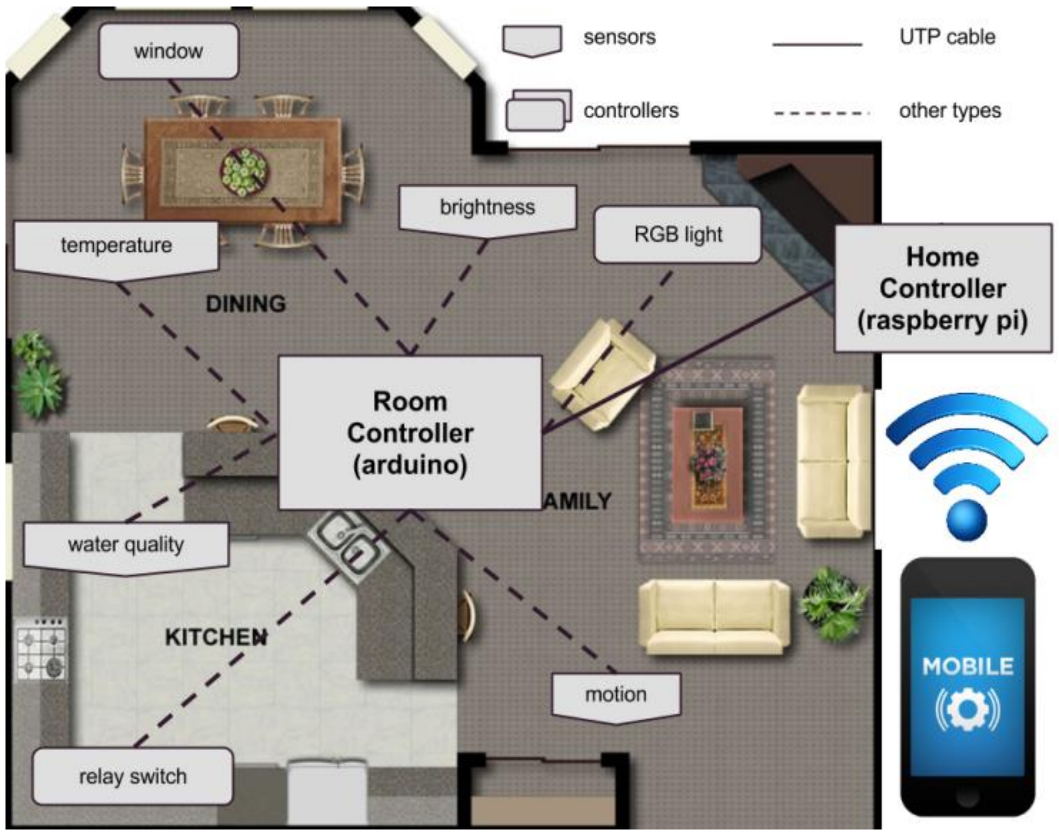


Figure 46 Smart Home Point overview deployment schema

The prototype of system consists of Commander, Controller, Element and UI units. We consider Commander Unit as Linux based minicomputer known Raspberry Pi where nonstop running web server responds for user's mobile device and other UI applications. Also inner part of the system is socket server bind with logic and database which is listening on port 5000 for every event or command messages transported over local network. The communication with web server is provided by Representational State Transfer (REST) as web resource based access. The Application Programing Interface (API) describes main functionalities. We consider also third party access, therefore open based API access is provided and the documentation with mock's objects is available. API conforms to latest consideration of resource based best-practices where all types of resources if are allowed to access are able to reach from top level hierarchy or from relation between resources. The relations are expressed as inner mount point with identification of upper leading resource. The authorization to resource is provided by implementation of Filter class of web server where access privileges are defined in profile type of user and therefore resources are defined as accessible to specific type of profile. For instance Admin or Householder is able to add new user into the system and authorize for him or her to execute commands. We consider implementation on different mobile platforms as Android, iOS and Windows for UI application with simple user friendly component as a part of further discovery. Nevertheless nowadays we just propose open resource based interface which leads to correct implementation of mobile UIs. The practical main resources are User, Element, Controller, Command, Event and House which allow settings and maintaining the system which was at first configured by technical person and provide proper setup of controller and commander unit. The User resource allows plain Create Read Update Delete (CRUD) operation for authorized actor with resource. These are provided to all resources as generic plain implementation. The other operations as login, logout, findUserByKeyword, createWithArray or createWithList are provided for specific functionality where name or detailed description expresses their purpose defined by annotation on entity classes and generated on every request, therefore the API documentation is all the time up to date. Mobile application is designed as a remote controller of the house system. It depends only on connectivity weather mobile device is connected to internet or within local network to home server. We implement simple remote controller where are possibilities of shortcuts keys, menu direction handlers and easy buttons such as hide or show menu on TV screen. The prototype layout in [Figure 47] and [Figure 48] is briefly describe functionality in its layout and keywords.

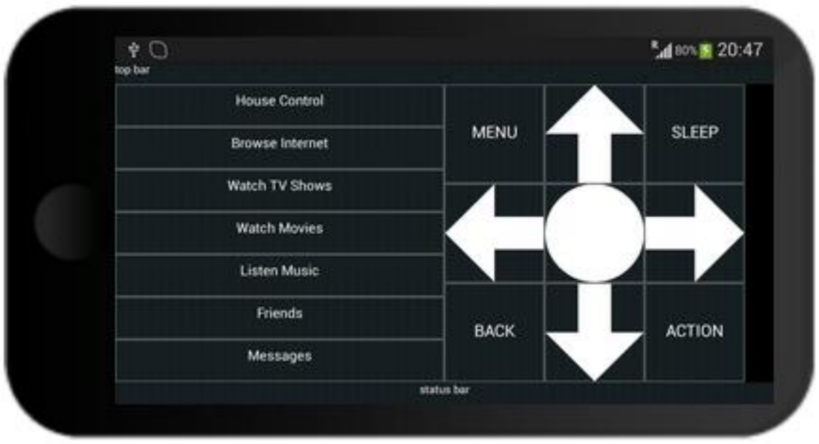


Figure 47 Mobile application prototype layout

Also the home point screen of the system is mandatory in terms of usability. We are considering as authentication mechanism with mobile device time interval tokens which are encoded to QR code.

Therefore you could see on prototype layout and the visualization of smart home point application where is content and house information centralizing also QR codes. The mobile device at first has to scan QR code viewed on TV screen to obtain access token for remote control. When the token is verified over server the mobile application controls the smart home point and QR code disappears. The controlling is based on driving through menus by simple arrows. We are considering also as improvements mouse type cursor handling where only UDP packets are that fast to handle this solutions in terms of ANR (Application Not Responding). The activation selected menu is done by speed button called action and this triggers routines on server side. For instance when the switch all lights are willing to be on the menu all lights on have to be activated. That triggers background process which send UDP packet to all nodes with command to switch lights on. In case of RGB lights the command would be set all values R, G and B to default values.

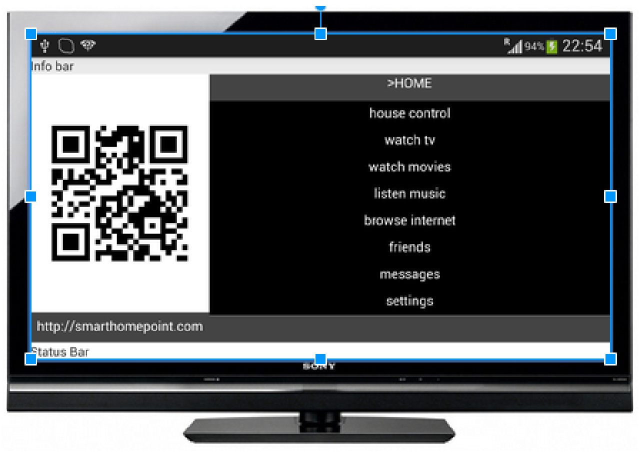


Figure 48 Application prototype TV layout

7.2. Public Transport

The public environment where people can share something in common is for example public transport. They share traveling paths and there is only challenge for public providers of transportation to effectively maintain resources such busses, trains or other vehicles to provide optimal transportation paths and timing to fulfill public demand. The sensorial framework such as Senzoric can assist with creating and updating effective timetable and paths since the distribution of population in cities can change overtime. Normally the public traffic is based on solid timetable and paths given by public traffic designers based on experience and previous state. But think about if such prolonging process could be shortened into on demand process based on real situation power by sensors of personal mobile devices with public traffic sensors. The traffic lights can be managed by current density of needed transportation in demanded way. The busses can increase or decrease intervals on bus stops upon current amount of requests or even there could be dynamic paths of busses driven by destination spots of travelers [Figure 49]. There can help Sensoric Framework to provide sense data of users and with extension in form of travelling application where users specify destination spots it can maintain whole public traffic system with automation.

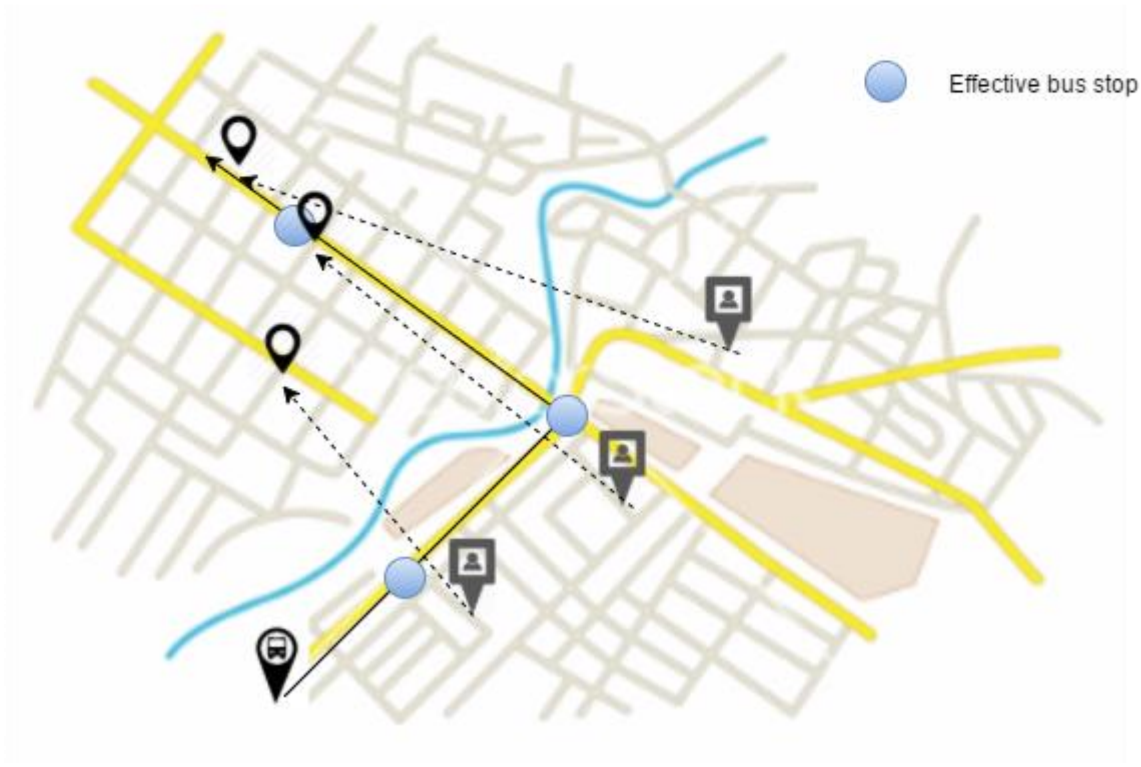


Figure 49 Public transport

The only problem is to balance current state of roads where the bus stops are at solid places and the challenge for smart cities is about to uniform roads to provide stopping places anywhere based on current demand. Such idea can be kind of revolutionary but future public transport will be about automation vehicles without drivers which take you from your positions precisely to your destination.

8. Discussion of Results

In evaluation of achieved results during phases of Senzoric framework realization such as analyzation problem definition area, designing architecture and concept, and also implementation and deployment we consider following points which were fully or partly completed to achieve working application prototype available for multiple Android mobile devices which are capable to gathering online sensorial data from device embedded sensors.

- Analysis of embedded sensors with theirs physical and software capabilities
- Analysis of mobile devices available on market
- State of art regarding software development technique and used tools
- Declaring use cases and process flow for framework
- Proposal of system architecture on frontend and backend
- Implementation of system components in Java
- Design and implementation of database structure and data types
- Implementation of automatic test during build
- Verification of implemented solution by running application in real environment

All those achieved results increase our knowledge in area of discovery sensorial networks embedded in mobile devices. We realize that others relative works based on sensorial framework are completed within teams during couple of years and therefore we consider that our minimalistic prototype is well performed result in terms of programing capacities. We need to join others developers world widely to able to provide real production sensorial framework with rich plugins and visualization widgets on more than one mobile platform. The main our contribution we believe is in state of art development techniques and used tools as well as design and implemented architecture with the secured, effective data transmission and minimal battery consumption during behavior pattern recognition.

We provided real time sensorial framework prototype which is being used by group of voluntaries in real environment where evaluated data gathered by framework showing repetitive users behavior patterns which may be used for further discovery to enable framework triggering smart environment actions for instance automatic house doors opening based on prediction patterns and location sensor bases. There are plenty of real usages in our daily lives where the most challenging is integration within other system to provide complex smart environment behavior.

To deploy Senzoric framework in real mass production environment is profile optimization needed to don’t bother common users to fulfill precise sensorial data gathering in current form but rather than that provide intelligent self-managing profile handling based on previous experience or by wizard handling form where user define desired outcome from framework in combination of available mobile device embedded sensors. Currently we provide just one default profile which can be customized by user and stored in cloud but the customization for not advance users are not realistically to ask for. The common user customization ends up with not optimized data a transfer result which ends up in capturing more mobile device resource than should be desired.

We consider the extended usability of Senzoric framework in open sourced code available on website for third parties to provide more implementation customized sensorial based features. For third parties it unifies embedded sensors access for mobile devices application and backend management system. The developers can implement sensorial triggers to run external procedures in smart environment by behavior or environment patterns recognition. Our implementation provides location monitoring of mobile devices which can be used in many real business cases. For instance traveling agency can monitor travel agent during the work. The shipping company can monitor driver's paths and optimize workload. Or microphone and camera sensors monitoring can be invoked by accelerometer and mobile devices can server as extended node for video monitoring system.

The Senzoric framework was designed with regard to extensibility in future internally or with third parties to provide connection within smart applicants in smart environments. The behavior of users can trigger actions of electronic applicants in surrounding area such as smart walls, smart cars, smart fridge and others. Based on such automatic actions users can literally feel that the environment can sense them. Another huge area of extensions is in auto notification on social media networks where the behavior, environment or context recognition sense users publicly and users can share their common habits as well as they can joining in predictive recognized behavior patterns.

9. Conclusions

The sensorial based frameworks are currently evolving areas in fast changing mobile devices environment. The effective concept for distributive sensorial information creates a challenge concerning the battery consumption, communication, and effective sensorial data gathering. There is group of sensorial applications which partly provides desired functionalities but none of them provides fully capable sensorial framework with open sourced code, with social connectors available, with predictive pattern recognition, with customization of visual components, intelligent profile handling and minimal battery consumption. All of these aspects are evaluated equal in order to help to increase the usability and we considered those aspects as challenge for us to design and implement such framework in real environment.

The goal of dissertation was to create sensorial framework which capture embed sensors data and transform them into comprehensive information which can be share through the cloud. To reach that goal was necessary analyze currently available mobile device sensors, discover the most suitable development tools and techniques, design and implement prototype application, test prototype in real environment and evaluate results. We choice in the beginning agile development approach together with Java and Android based platforms for implementation which latter on was proven as good choice.

The key benefit of the proposed architecture is in scalability and applicability for further location, motion, and environmentally based real-time solutions.

The work proposes effective sensorial information distribution in terms of usability and we consider the future smart environment based on mobile device embedded sensorial networks. Parts of work was published in journals [Scop1], [Imp1] with impact factor 0,805 and conferences [Conf6], [Conf8] and [Conf9].

We stand at the beginning of a new era of sensorial social networks where we will perceive only information which is relevant to us. In other words, we will create, through our activities, our own artificial informational shield or receiver, and through our life style, we will obtain corresponding informational channels.

10. Bibliography

[1] P. Mikulecky, "Remarks on ubiquitous intelligent supportive spaces," in *15th American Conference on Applied Mathematics/International Conference on Computational and Information Science*, Houston, 2009.

[2] P. C. Hii and W. Y. Chung, "A comprehensive ubiquitous healthcare solution on an android (TM) mobile device," *Sensors 11(7),* p. 6799–6815, 2011.

[3] "What is sensor," 2014. [Online]. Available: http://what-is-a-sensor.com/.

[4] B. I. Poids and , Le Systéme International d'Unités (SI), The International System of Units (SI), France: Sèvres, 2006.

[5] A. Thompson and N. Barry, Guide for the Use of the International System of Units (SI), NIST Special Publication 811, 2008.

[6] S. Bobek, G. J. Nalepa and A. Ligęza, "Mobile context-based framework for threat monitoring in urban environment with social threat monitor," in *Multimedia Tools and Applications*, 2013.

[7] H. Efstathiades, G. Pa and P. Theophilos, "Feel the World: A Mobile Framework for Participatory Sensing," in *10th International Conference, MobiWIS 2013, Paphos, Cyprus, August 26-29, 2013.*, 2013.

[8] M. Schirmer and H. Höpfner, "Approaches for reducing the energy consumption of smartphone-based context recognition," in *SenST*, Modeling and Using Context 6967*, 2011.

[9] P. Zhang, W. Jiang and J. Y. Zhu, "MobiSens: A Versatile Mobile Sensing Platform for Real-World Applications," *Mobile Networks and Applications,* pp. 60-80, 2013.

[10] P. Brida, J. Machaj, J. Benikovsky and J. Duha, "An experimental evaluation of AGA algorithm for RSS positioning in GSM networks," *Elektron. Elektrotech 104,* p. 113–118, 2010.

[11] W. Woerndl, A. Manhardt, F. Schulze and V. Prinz, "Logging user activities and sensor data on mobile devices," *Analysis of Social Media and Ubiquitous. Data Lect. Notes. Comput. Sc. 6904,* p. 1–19 , 2011.

[12] L. Lhotska, M. Bursa, M. Huptych, V. Chudacek and J. Havlik, "Standardization and Interoperability: Basic Conditions for Efficient Solutions," *IFMBE Proceedings, vol. 37 ,* p. 1140–1143, 2011.

[13] K. Juszczyszyn, N. T. Nguyen, G. Kolaczek and A. Grzech , "Agent-based approach for distributed intrusion detection system design," *Lect. Notes. Comput. Sc. 3993,* p. 224–231, 2006.

[14] O. Krejcar, "User localization for intelligent crisis management," in *Conference on Artificial Intelligence Applications and Innovation (AIAI)*, Athens, 2006.

[15] S. Huseth and S. Kolavennu, "Wireless networking based control," *Localization in Wireless Sensor Networks,* p. 153–174, 2011.

[16] IDC, "Market share Smartphones OS," 2014. [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

[17] Google, "Android Source," 2015. [Online]. Available: https://source.android.com/index.html.

[18] "Top sales Smartphones," Forbes, 2014. [Online]. Available: http://www.forbes.com/sites/chuckjones/2014/04/03/apples-iphone-5s-still-the-top-selling-smartphone-worldwide/.

[19] DeviceSpecifications, "Comparision," 2015. [Online]. Available: http://www.devicespecifications.com/en/comparison/eaf0277a0.

[20] K. Choros, "Further tests with click, block, and heat maps applied to website evaluations," in *Lect. Notes in Artif. Int. 6923*, 2011.

[21] S. Huseth and S. Kolavennu, "Localization in Wireless Sensor Networks," in *Wireless Networking Based Control*, 2011.

[22] S. Tarkoma, M. Siekkinen, E. Lagerspetz and Y. Xiao, Smartphone Energy Consumption: Modeling and Optimization, Cambridge University Press, 2014.

[23] D. H. Kim, Y. Kim and D. Estrin, "SensLoc: Sensing Everyday Places and Paths using Less Energy," *SenSys'10,* 2010.

[24] W. Brunette, R. Sodt, R. Chaudhri and M. Mayank, "Open data kit sensors: a sensor integration framework for android at the application-level," *MobiSys '12 Proceedings of the 10th international conference on Mobile systems, applications, and services,* pp. 351-364, 2012.

[25] P. Lin, S. Chen and C. Yeh, "Implementation of a smartphone sensing system with social networks: a location-aware mobile application," in *Multimedia Tools and Applications*, 2015.

[26] N. Győrbíró, A. Fábián and G. Hományi, "An activity recognition system for mobile phones," in *Mobile Netw. Appl. 14(1)*, 2009.

[27] W. Woerndl, A. Manhardt, F. Schulze and V. Prinz, "Logging user activities and sensor data on mobile devices," in *Analysis of Social Media and Ubiquitous Data*, 2011.

[28] G. B. Gil, J. M. Molina, A. Berlanga and Gil, G B; A Berlanga, A; Molina, J M; A Berlanga, , "In Contexto: multisensor architecture to obtain people context from smartphones," in *Article ID 758789*, 2012.

[29] D. Honda, N. Sakata and S. Nishida, "Activity recognition for risk management with installed sensor in smart and cell phone.," in *Human-Computer Interaction. Towards Mobile and Intelligent Interaction Environments.*, 2011.

[30] N. E. Klepeis, "The national human activity activity pattern survey," *Journal of Exposure Analysis and Environmental Epidemiology,* p. 231–252, 2001.

[31] L. Lhotska, M. Bursa, M. Huptych, V. Chudacek and J. Havli, "Standardization and Interoperability: Basic Conditions for Efficient Solutions," in *IFMBE Proceedings, vol. 37*, 2011.

[32] T. Choudhury, H. Lu, J. Yang, Z. Liu and N. Lane, "The Jigsaw Continuous Sensing Engine for Mobile Phone Applications," *In: Proceedings of the 8th ACM Conference on SenSys, MobiSys 2010,* p. 71–84, 2010.

[33] N. Aharony, A. Gardner and C. Sumter, "The Funf Open Sensing Framework is an extensible sensing and data processing framework for mobile devices.," [Online]. Available: http://funf.org/about.html.

[34] K. K. Rachuri, C. Mascolo, M. Musolesi and P. J. Rentfrow, "Sociablesense: Exploring the tradeoffs of adaptive sampling and computation offloading for social sensing," *In: Proceedings of the 17th MobiCom,* pp. 71-84, 2011.

[35] "Agile development," [Online]. Available: http://eontek.co/services/software-development/web-application-development/.

[36] IEEE, "Spectrum’s Ranking," 2014. [Online]. Available: http://www.sitepoint.com/best-programming-language-learn-2014-mid-year-update/.

[37] "Android Operation System," [Online]. Available: https://en.wikipedia.org/wiki/Android_(operating_system).

[38] Spring, "Spring Framework - spring.io," [Online]. Available: http://docs.spring.io/spring-data/data-document/docs/current/reference/html/#mapping-usage-annotations.

[39] A. G. Parada, E. Siegert and L. B. de Bri, "Generating Java code from UML Class and Sequence Diagrams," in *Computing System Engineering (SBESC)*, 2011.

[40] M. Behan and K. Ondrej, "Smart Home Point As Sustainable Intelligent House Concept," in *12th IFAC Conference on Programmable Devices and Embedded Systems*, 2013.

11. Appendix a - Author’s publication

Journal papers indexed by ISI WOK database as JCR impact factor

[Imp1] Behan, M., Krejcar, O., Modern Smart Devices based Concept of Sensoric Networks. *EURASIP J ournal on Wireless Communications and Networking*. Vol. 2013, No. 155, DOI 10.1186/1687-1499-2013-155, Received 9 October 2012; Accepted 15 May 2013; Published 6 June 2013. ISSN 1687-1499. (**Impact Factor** (2013 Thomson JCR Science Edition)**: 0,805**)

Journal papers indexed by SCOPUS database

[Scop1] Behan, M., Krejcar, O., The Concept of the Remote Devices Content Management. *Journal of Computer Networks and Communications.* vol. 2012, Article ID 194284, 2012. DOI 10.1155/2012/194284. Received 20 June 2012; Accepted 31. 08. 2012

Conferences indexed by ISI WOK database (CPCI index) and by SCOPUS database

[Conf1] Behan, M., Krejcar, O., Adaptive Graphical User Interface Solution for Modern User Device. In *4th Asian Conference, ACIIDS 2012, Kaohsiung, Taiwan, March 19-21, 2012, Proceedings, Part II.* Intelligent Information and Database Systems, Vol. 7197. pp. 411-420. Springer, Heidelberg. ISBN 978-3-642-28490-8, ISSN 0302-9743, DOI 10.1007/978-3-642-28490-8_43

[Conf2] Behan, M., Krejcar, O., Smart Communication Adviser for Remote Users. In *8th International Conference on Multimedia & Network Information Systems 2012, MISSI 2012, September 19-21, 2012 Wroclaw, Poland.* Advances in Multimedia and Network Information System Technologies, Series Advances in Intelligent and Soft Computing, Vol. 183. pp. 169-178. Springer, Heidelberg. ISBN 978-3-642-32334-8, ISSN 2194-5357, DOI 10.1007/978-3-642-32335-5_16

[Conf3] Behan, M., Krejcar, O., Open Personal Identity as a Service. In *8th International Conference on Multimedia & Network Information Systems 2012, MISSI 2012, September 19-21, 2012 Wroclaw, Poland.* Advances in Multimedia and Network Information System Technologies, Series Advances in Intelligent and Soft Computing, Vol. 183. pp 199-207. Springer, Heidelberg. ISBN 978-3-642-32334-8, ISSN 2194-5357, DOI 10.1007/978-3-642-32335-5_19

[Conf4] Behan, M., Krejcar, O., Concept of the Personal Devices Content Management Using Modular Architecture and Evaluation Based Design. In *Context-Aware Systems and Applications, ICCASA 2013,* Lecture Notes of the Institute for Computer Sciences, Social Informatics and

Telecommunications Engineering, Vol. 109. pp 151-159. Springer, Heidelberg. ISBN 978-3-642-36641-3, ISSN 1867-8211, DOI 10.1007/978-3-642-36642-0_15

[Conf5] Behan, M., Krejcar, O., Mobile Widget Technology as a Solution for Smart User Interaction. In *Third International ICST Conference, AMBI-SYS 2013, Athens, Greece, March 15, 2013.* Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, LNICST Vol. 118. pp. 10p. Springer, Heidelberg.

[Conf6] Behan, M.; Krejcar, O. Open Source Based Concept of Intelligent House. In: *Intelligent Environments (Workshops)*. 2013. p. 330-337. IE 2013. *Athens, Greece*

[Conf7] Behan, M., Krejcar, O., Vision of smart home point solution as sustainable intelligent house concept, In 12th IFAC Conference on Programmable Devices and Embedded Systems, PDeS 2013; Velke Karlovice; Czech Republic; 25 September 2013 through 27 September 2013, IFAC Proceedings Volumes (IFAC-PapersOnline), Volume 12, Issue PART 1, 2013, Pages 383-387, DOI 10.3182/20130925-3-CZ-3023.00057

[Conf8] Behan, M., Krejcar, O., Open IP-based sustainable concept of intelligent house controlled by mobile devices, In IEEE 8th International Symposium on Intelligent Signal Processing (WISP 2013), 16. - 18. September 2013, Funchal, Madeira, pp. 121-125, DOI 10.1109/WISP.2013.6657494

12. Appendix b - Figures

13. Appendix c - Tables