INSIST

Deliverable

D5.1.1
Data Collection

Editor:

Emine Ferraro

ITEA 3 Project 13021

Date: 01.07.2017

Document properties

Security	Confidential
Version	Final
Author	Emine Ferraro, Argedor
Pages	

History of changes

Version	Author, Institution	Changes
0.1	Argedor	Draft Document
0.2	Argedor, Verisun	Finalisation of the document

Abstract

This document describes the collected representative datasets from the INSIST application fields. The datasets will be a reference for development purposes for the INSIST project and for future projects in this field. Additionally this information will complement the proposals for the partners who are involved in standardisation bodies so that the main results could be integrated into new or updated standards

In order to meet the target, datasets should be composed of the following sections:

- The purpose of the dataset
- Context information
- Applications and field tests specific data
- Conclusions

Table of contents

1. Executive Summary

This document describes the collected representative datasets from the INSIST application fields. The datasets will be a reference for development purposes for the INSIST project and for future projects in this field. Additionally, this information will complement the proposals for the partners who are involved in standardisation bodies so that the main results could be integrated into new or updated standards

In order to meet the target, datasets should be composed of the following sections:

- The purpose of the dataset
- Context information
- Applications and field tests specific data
- Conclusions

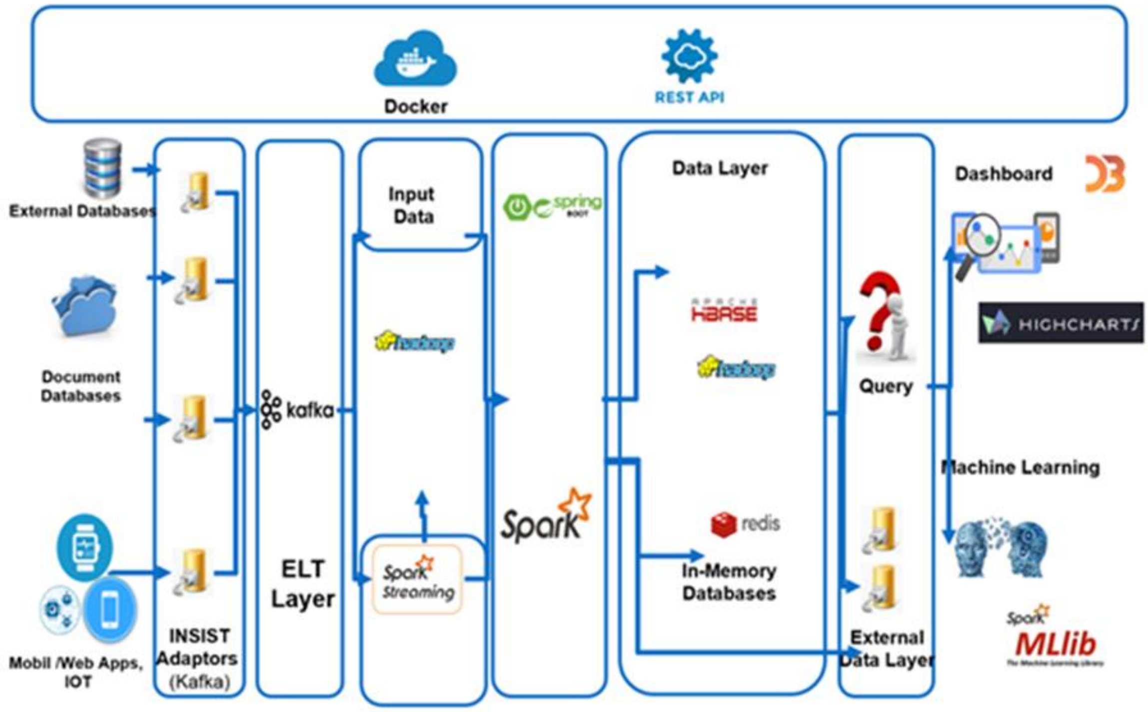
2. Project Outline

The urban spaces are full of stand-alone sensor based installations of different services designed according to their own purpose and requirements. For example, the municipalities provide several services to the citizens: safety and security services of citizens in the streets, traffic management, etc. These services rely mostly on street-implemented infrastructures, such as cameras, sensors, induction loops. In addition, local businesses have their own illumination systems, advertising infrastructure including neon signs, public displays etc. They also monitor customer behavior using various sensor systems. The INSIST project proposes an integration of these sensor-based systems into a wider perspective. The INSIST project aims to develop a smart connected ecosystem for public spaces, where the sensor data provided by the different INSIST sensor systems can be efficiently used for not only the proprietary infrastructure services but also to offer value added services based on data fusion from multiple sensor systems in the business areas of:

- Traffic management
- Advertising and atmosphere
- Business intelligence

3. System Design

The overall architecture of the Common Database Layer shall be defined as in the following Figure.



The Common Database System Requirements of the INSIST Project are;

a. APACHE KAFKA

Kafka is a distributed publish-subscribe messaging system that offers a solution capable of handling all data flow activity and processing these data on a consumer application. Kafka is used for building real-time data pipelines and streaming apps. It is horizontally scalable, fault-tolerant, wicked fast, and runs in production in thousands of companies. Apache Kafka will be used to store the initial point of the streaming data in INSIST project.

b. APACHE SPARK STREAMING

Spark Streaming brings Apache Spark's language-integrated API to stream processing, letting you write streaming jobs the same way you write batch jobs. It supports Java, Scala and Python.

Apache Spark Streaming will be used for processing the streams of data input throughout the INSIST project.

c. HADOOP

An open source software library project administered by the Apache Software Foundation. Apache defines Hadoop as “a framework that allows for the distributed processing of large data sets across clusters of computers using a simple programming model.”

HADOOP Big Data library creates the base storing point(database) of the INSIST project.

d. REDIS

Redis is an open source, in-memory data structure store, used as a database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs and geospatial indexes with radius queries. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.

Redis will be the main in-memory database for storing cached data in INSIST project.

e. SPARK MLIB

MLlib is Spark’s machine learning (ML) library. Its goal is to make practical machine learning scalable and easy. At a high level, it provides tools such as:

- ML Algorithms: common learning algorithms such as classification, regression, clustering, and collaborative filtering
- Featurization: feature extraction, transformation, dimensionality reduction, and selection
- Pipelines: tools for constructing, evaluating, and tuning ML Pipelines
- Persistence: saving and load algorithms, models, and Pipelines
- Utilities: linear algebra, statistics, data handling, etc.

In INSIST project SPARK MLIB will be used to be the base library for machine learning and data mining.

f. REST API

INSIST architecture will be based on RESTful Services.

A RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data. A RESTful API - also referred to as a RESTful web service -- is based on representational state transfer (REST) technology, an architectural style and approach to communications often used in web services development.

g. MICROSERVICES

Each services in the INSIST project will have microservices design patterns.

Microservices is a variant of the service-oriented architecture (SOA) architectural style that structures an application as a collection of loosely coupled services. In a microservices architecture, services should be fine-grained and the protocols should be lightweight.

i.DOCKER

Each microservices developed in ht INSIST project will be packaged inside a container. A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings. We will use Docker, which is an open platform for developers and sysadmins to build, ship, and run distributed applications, whether on laptops, data center VMs, or the cloud.

4. Sample Dataformat

All records should be in JSON Object. When posting and retrieving multiple record using REST API, all records should (and will) be in a JSON Array.

There are some fields that is required regardless of sensor or data type. These are:

Type: Type of the sensor.

Timestamp: The timestamp that the record captured. (//Is this true?)

Coordinate or location (adress): //geojson

Id : Serial number, unique id of the sensor.

Brand : Brand of the sensor.

Model : Model of the sensor.

Example:

[

{“type”:“environmental | barometer?”, ”timestamp”:”formatiburaya”, “coordinate”:”geojsonobjesiburaya”, “id”:”idornegiburaya”, “brand”:”brandornegiburaya”, “model”:”modelornegiburaya”},

{“type”:“environmental | barometer?”, ”timestamp”:”formatiburaya”, “coordinate”:”geojsonobjesiburaya”, “id”:”idornegiburaya”, “brand”:”brandornegiburaya”, “model”:”modelornegiburaya”}

…

]

Sensor Types | (Data Type ?)	**Description | (Sensor Type ?)**
Environmental	
Atmospheric pressure	Barometers
Humidity	Hygrometers
Temperature	Thermometers
Wind_direction	Weather vanes
Wind_speed	Anemometers
Light	
Ambient Light Sensors(Ligh Chromacity)	Chromaticity as a counted array of float values

Light Level Lux	Illuminance level, in lux.
Light Temperature	Color temperature, in degrees Kelvin.
Location	
Dead Reckoning	These sensors first calculate the current location and then update the current location by using motion data.
GPS	Global positioning system sensors.
Location Lookup	Lookup sensors, such as those that provide information based on the user's IP address.
Location Other	Fixed-location sensors, such as those that use preset, user-provided information.
Location Static	Triangulation sensors, such as those that determine current location based on cellular phone tower proximities.
Motion	
Accelerometers 1D	One-axis accelerometers.
Accelerometers 2D	Two-axis accelerometers.
Accelerometers 3D	Three-axis accelerometers
Gyrometers 1D	One-axis gyrometers.
Gyrometers 2D	Two-axis gyrometers
Gyrometers 3D	Three-axis gyrometers.
Motion Detectors	Motion detectors, such as those used in security systems.
Speedometers	Rate-of-motion sensors.
Orientation	
Aggregated Device Orientation	Specifies the current device orientation by returning a Quaternion and, in some cases, a rotation matrix. (The rotation matrix is optional.)

Aggregated Quadrant Orientation	Specifies the current device orientation in degrees.
Aggregated Simple Device Orientation	Specifies the device orientation as an enumeration. (This type specifies the device orientation using one of four general quadrants: 0 degrees, 90-degrees counter clockwise, 180-counter clockwise, and 270-degrees counter clockwise. It also indicates the face-up or face down orientation of the device.)
Compass 1D	One-axis compasses.
Compass 2D	Two-axis compasses.
Compass 3D	Three-axis compasses.
Distance 1D	One-axis distance sensors
Distance 2D	Two-axis distance sensors.
Distance 3D	Three-axis distance sensors.
Inclinometer 1D	One-axis inclinometers.
Inclinometer 2D	Two-axis inclinometers.
Inclinometer 3D	Three-axis inclinometers.
Other Sensors	
Traffic Density - 1	Video Camera Data Type: text, Data Size: Max 200 byte, Write Access Frequency: 5 min
Traffic Density – 2	Social Media Data Type: text, Data Size: Max 200 byte, Write Access Frequency: 60 sec
Traffic Density – 3	Traffic Density Maps Data Type: text, Data Size: Max 200 byte, Write Access Frequency: 10 sec
Traffic Density	Application Data Type: text, Data Size: Max 200 byte, Write Access Frequency: 5 min
Event	Social Media Data Type: text, Data Size: Max 200 byte, Write Access Frequency: 5 min
BusLines Vehicle Location Info	BUS GPS

	Data Type: text , Data Size: Max 200 byte, Write Access Frequency: 1 min
How2Go	Shortest Public Transportation Algorythm Data Type: text , Data Size: Max 5 k byte, Write Access Frequency: 1 min
Buses	Application Data Type: text , Data Size: Max 5 k byte, Write Access Frequency: 1 min
Bus Stops	Application Data Type: text, Data Size: Max 5 k byte, Write Access Frequency: monthly
Search Old	Application Data Type: text, Data Size: Max 5 k byte, Write Access Frequency: 1 min
Search	Application Data Type: text, Data Size: Max 5 k byte, Write Access Frequency: 1 min
Time Table	Application Data Type: text, Data Size: Max 5 k byte, Write Access Frequency: Monthly
Bus Lines Old	Application Data Type: text, Data Size: Max 5 k byte, Write Access Frequency: monthly

Table 1.2.1 [1]

[1] https://msdn.microsoft.com/en-us/library/windows/desktop/dd318969(v=vs.85).aspx