POLITECNICO
MILANO 1863

TrackMe project Giorgio Cozza and
Barbara Ferretti

Data4Help and AutomatedSOS

Design Document

Deliverable:	DD
Title:	Data4Help and AutomatedSOS - Design Document
Authors:	Giorgio Cozza, Barbara Ferretti
Version:	1.0
Date:	10-December-2018
Download page:	https://github.com/GiorgioCozza/CozzaFerretti.git
Copyright:	Copyright © 2018, Giorgio Cozza and Barbara Ferretti – All rights reserved

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

As explained in the previous artifact, Data4Help and AutomatedSOS, since handling the same set of data types, but concerning with different purposes, are physically separated applications. The first one represents a privacy-guarateed service to manage third party requests of access to user-provided health data, the second is a service concepted to ensure first assistance whenever dangerous conditions for registered users are detected. The discussion at this point should be focused specifically on the architecture behind these two services, analyzing the way both the applications will collect health data from the monitoring devices, how TrackMe servers will manage such data and more. To this end, since TrackMe wants to not depend strictly on cloud infrastructure providers for commercial and security reasons, it decided to host completely its own server farm. For the moment the system will be centralized, but easily extensible for future needs, according to the growth factor of the number of customers in order to form a private cloud infrastructure built on top of a distributed database system. Unfortunately, the deadline prevents a detailed analysis of this solution. This Design Document, however, will provide an overall definition of the main system components and the relative interactions, an in-depth discussion about the chosen architectural patterns and the complete plan for verification and validation steps. The audience for this artifact is tighter than the one of the RASD due to the more technical language, in particular, it is oriented to all the persons involved in engineering the software-to-be. In any case, in order to facilitate the comprehension of this document a previous RASD consultation is strongly suggested.

1.2 Scope

In this wide software ecosystem there are two common factors

1.3 Definitions, Acronyms, Abbreviations

1.4 Revision History

1.5 Reference Documents

1.6 Document Structure

2 Architectural Design

2.1 Overview

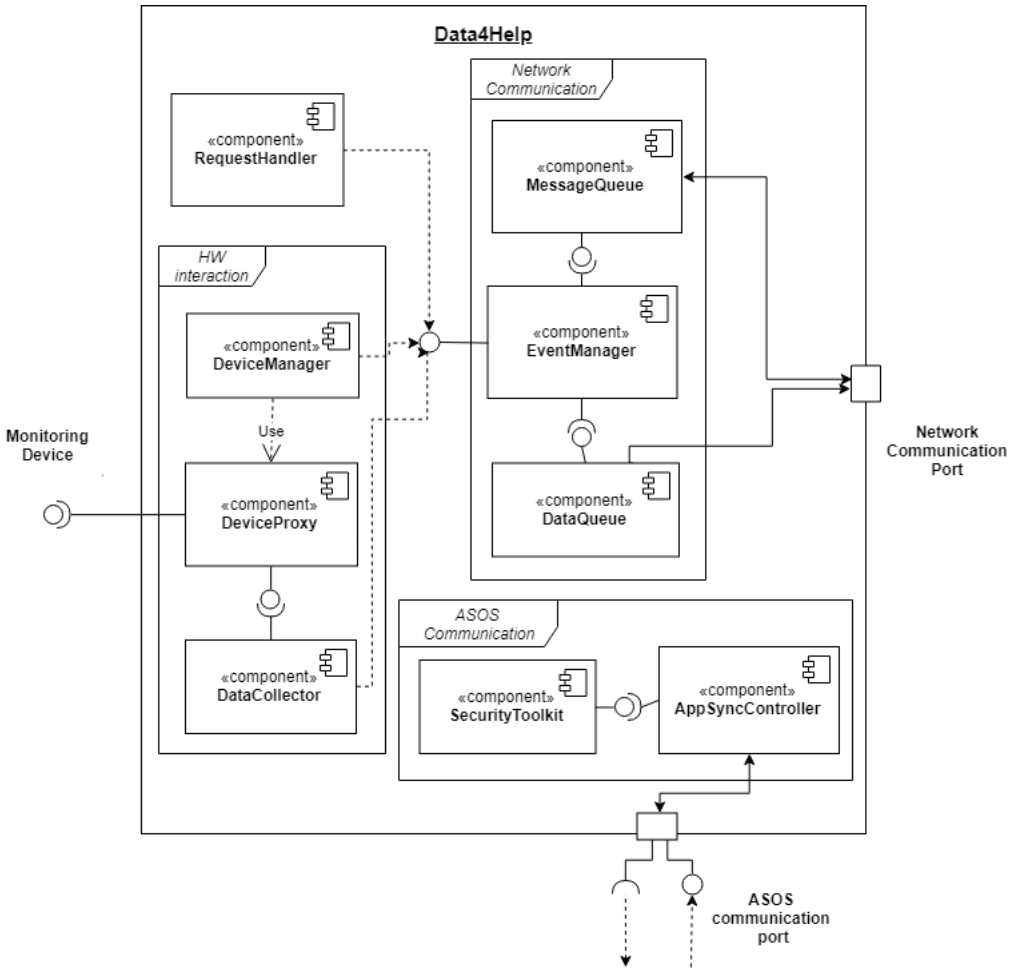
From the client standpoint, there are two applications, each one characterized by a independent architecture. Both ones communicate with TrackMe servers using the client-server paradigm.

2.2 Component View

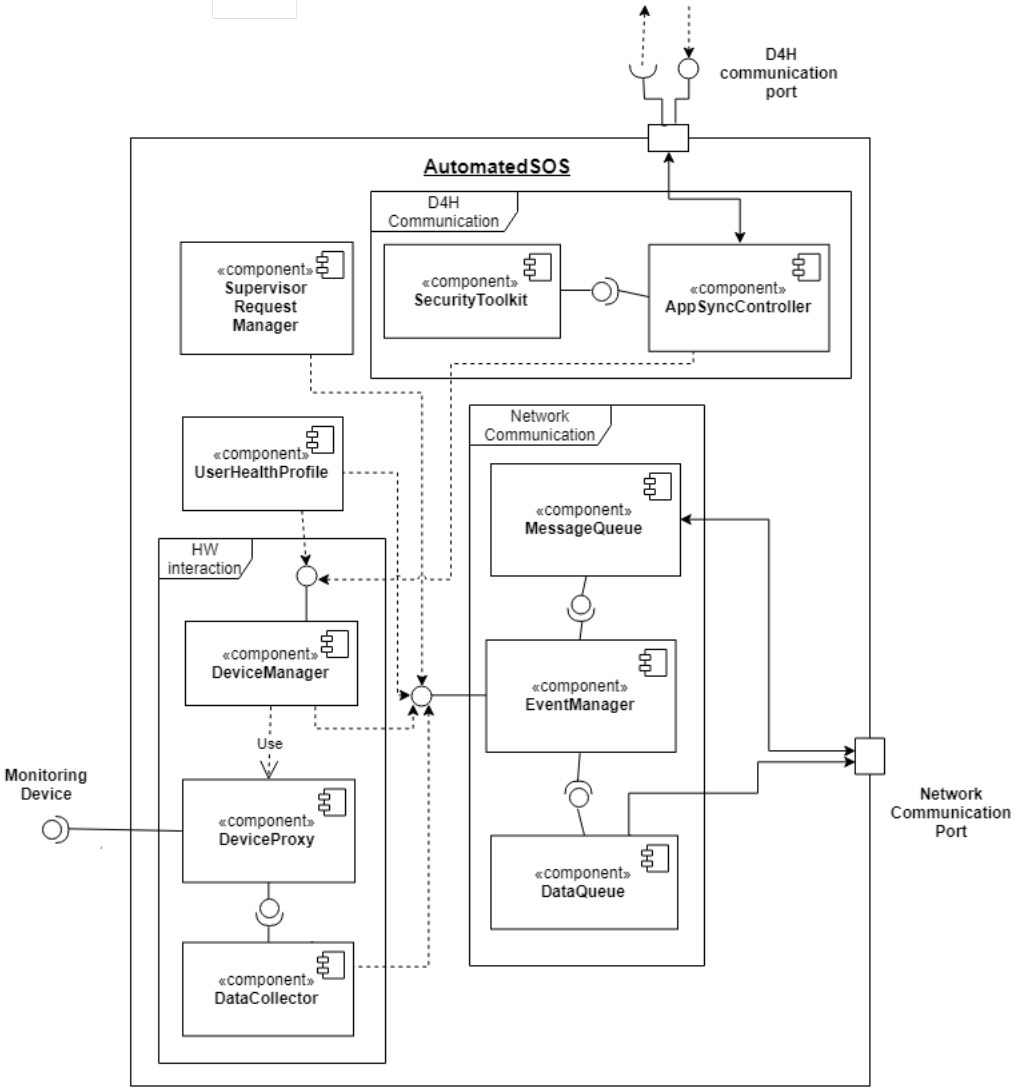
T

2.2.1 Client Side

Starting from the interface to the external sensors, the main problem related to the app separation is due to the fact that if the applications are running at a certain moment both of them could theoretically access the same monitoring device, providing to the server an identical sample. As anticipated in the Overview part, both the applications at clien side implement a personal hardware interface to the connected monitoring devices. This decision seems to introduce redundancy in code since a simple third service running in background could manage the connected devices and dispatch data between the two applications, unfortunately communication among running processes follows strict rules in many mobile OSs, someone only allows message exchanging. This makes more difficult to implement a mechanism of access to the monitoring sensors involving at most three different processes. Furthermore, relying on an HW manager service exposes both the apps to possible internal DOS attacks, since another malicious app could flood of messages the background service preventing the main application to allocate accesses to connected sensors.



It has been decided to create a communication interface between D4H and ASOS, for synchronizing data collection and preventing both the applications to access the same information. This task should be protected in order to avoid other processes to send false messages to one of them. A security toolkit provides encryption mechanism to this end.



2.3 Deployment View

2.4 Runtime View

2.5 Component Interfaces

2.6 Selected Architectural Styles and Patterns

2.7 Other Design Decisions

3 User Interface Design

4 Requirements Traceability

5 Implementation, Integration and Test Plan

6 Effort Spent

References

