Customer Requirements Specification

(Lastenheft)

(TINF18C, SWE I Praxisprojekt 2018/2019)

Project: *Profinet DCP Client als WEB-Applikation*

Customer: *Rentschler & Ewertz*
Rotebühlplatz 41
70178 Stuttgart

Supplier: Team 2 *(Jannik Schwarz, Sinan Yurttadur, Noah Broß, Marvin Sonntag, Nicholas Breuninger, Rene Scholz)*
Rotebühlplatz 41
70178 Stuttgart

Version	Date	Author	Comment
0.1	23.09.2019	Jannik Schwarz	Created the document
0.2	4.10.2019	Jannik Schwarz	New open questions and ideas for use-cases and functionality
0.3	7.10.2019	Jannik Schwarz	Added the use-cases
0.4	18.10.2019	Jannik Schwarz	Mostly everything was added, fine tuning will still be needed
1.0	21.10.2019	Jannik Schwarz	Added the figures for use-cases and the product environment
1.1	31.10.2019	Jannik Schwarz	Added Product Data

CONTENTS

1. Goal

What is has to do

The application is used to ease the management of Profinet-Devices, by displaying them for the user. All devices connected to the network of the user must be displayed. Because the program is used to help with configuration of the network and finding configuration errors it must display the device descriptor and the IP-address.

Optionally a function to reconfigure the IP-addresses from within the Application would be of great use. With this optional feature, an easy reconfiguration of the entire network would be possible from within this one application.

2. Product Environment

Data structure and environments

The main goal is to create a Profinet-DCP-Client, which is able to scan and display all the Profinet-Devices in the network. Furthermore, it should handle the display and management the IP-addresses and the DNS names of each device.
An additional graphical interface has to be created in Angular. The backend for the DCP-Client shall be programmed in C, C++ or NodeJS with the usage of an open-source-stack.
The communication between front- and backend must be realized with a JSON-based REST-API.

Setup and installation

The installation of the program should be handled easily, preferably by using an installer, which guides the user through the installation process. Options such as the installation path must be-included. Additionally, a manual is required, explaining how to use the program and its limitations.

The operating systems and installation

The Product will be used on Windows and or Linux Systems.
Program will be installed over an installer, which will guide you through everything.

Rough Diagram of the Environment

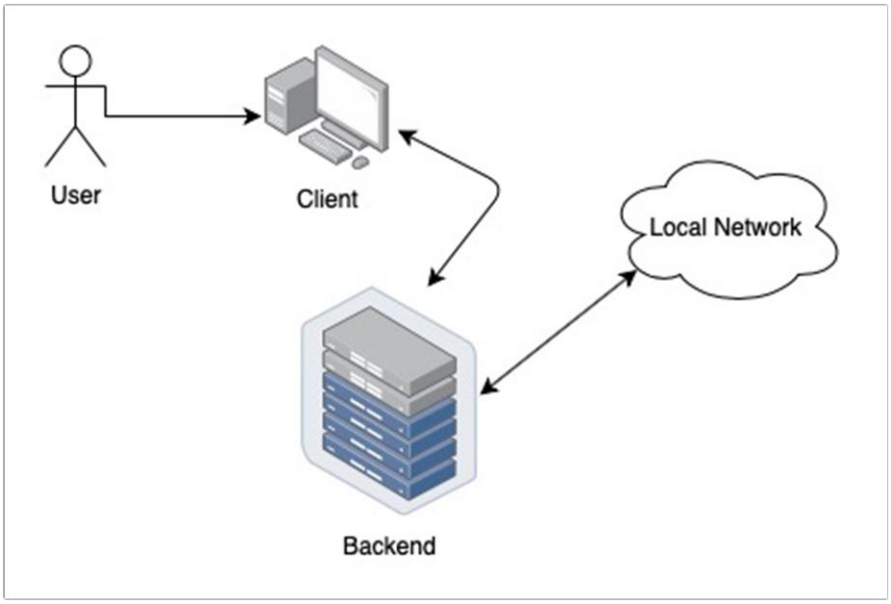


Figure 1 of the environment

3. Product Usage

Using the Application
The Backend is started locally on a computer. After the Backend started it is possible to retrieve data and display it on the frontend, which is going to be accessible through the browser, by browsing for the localhost on a specific port.

3.1. Business Processes

Since this is a Project by students of the DHBW the processes behind this project are to learn about different tools, project management and problem solving. The tools which will be used are GitHub, the DCP-Protocol, how to implement a REST-API, document that API with Swagger and use Postman to test it.

3.2. Use Cases

The use cases of the application are the searching of devices in the local network and returning a list of them to the user (<UC.001>) and also displaying detailed information about a chosen device out of the list (<UC.002>).
Optional use cases are the configuration of devices from within the client (<UC.003>).

3.2.1. <UC.001> Search Devices

Use Cases Objective:	The server handles the frontend request over an API-request. The server will start to listen for all Profinet-devices in the network. Then all found devices will be send back to the frontend in form of a list or array.
System Boundary:	The system is bound to the local network of the server
Precondition:	The connection to the network is required
Postcondition on success:	The backend must find all devices
Beteiligte Nutzer:	Frontend Server All Profinet-devices in the network
Triggering Event:	When the user clicks on “Search Devices”

3.2.2. <UC.002> Display detailed data of Device

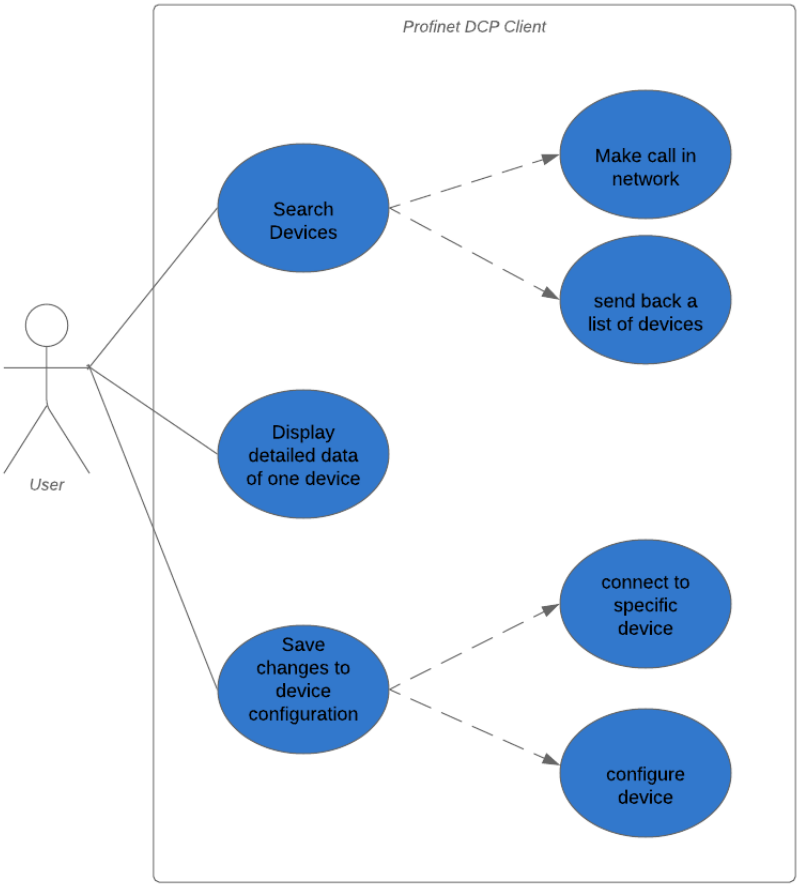
Use Cases Objective:	The frontend displays the information of one of the found devices. Information like the IP-address and name of the device are shown.
System Boundary:	The frontend is limited to the list of devices it received by the server.
Precondition:	A list of devices must have been sent back to the frontend
Beteiligte Nutzer:	Frontend Server

Triggering Event:	Selecting a device out of the list

3.2.3. <UC.003> Save changes to device configuration

Use Cases Objective:	The objective is to save the new configuration of a selected device. Changes like a different IP-address or name are to be used by the device from that point on.
System Boundary:	The operation is limited to the selected device.
Precondition:	A user must have selected a device. A connection to the device is needed.
Postcondition on success:	The device is visible on new IP-address.
Beteiligte Nutzer:	Frontend Backend (Server) A single device in the network
Triggering Event:	When the user saves changes in the configuration of a device

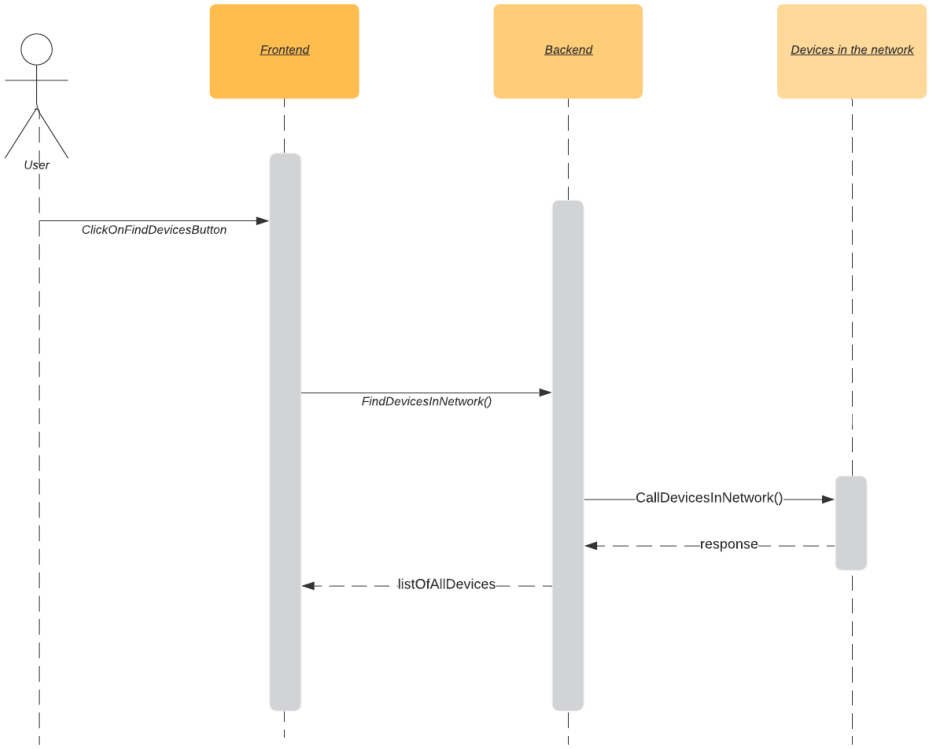
Use-Case-Diagram:



3.3. Features

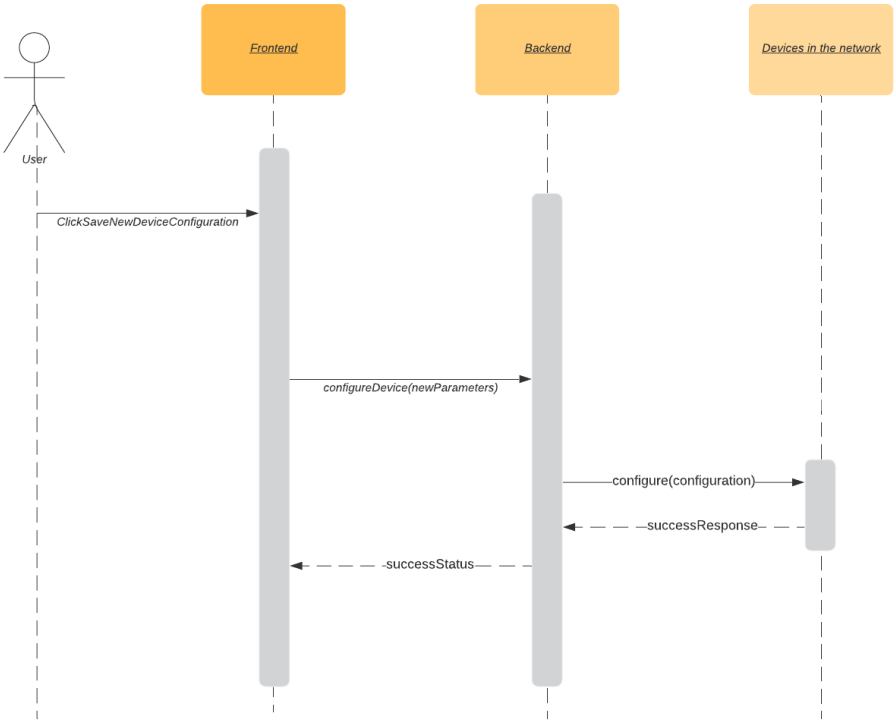
3.3.1. /LF10/ Device discovery

When making a search request for all devices the server must use the DCP-protocol to find the devices in the local network so, the server can return a list of them to the client.



3.3.2. /LF20/ Device configuration

When saving a new configuration of a device the server must reconfigure the selected device and set the new IP-address or name of the device.



4. Product Data

4.1. /LD10/ Data topology

Devices are connected via Ethernet cables.
The Information flow will occur in two directions. Data about the devices will be called from the client. The client will make a call to all Profinet-devices in the local network and will listen to the answers. A list of these answers will be stores in the backend of the application.
The backend will then continue to forward the information gathered to the frontend, where it will be displayed to the user.

4.2. /LD10/ Device data

The data of the device will be everything that can be gathered. It must include the IP-address of the device, the name of the device and, if possible, the port it is connected to.

5. Other Product Characteristics

5.1. /NF20/ Easy use of the application

Intuitive design should be implemented by guiding the user through the application. This should be achieved by making certain options or steps accessible only after certain conditions are met. E.g., you can only click on “Show Details of Device” after you already selected a device. This may be achieved by either hiding these actions or clearly disabling them.

5.2. /NF30/ Reliability

The system should display all connected devices reliably. A scan for devices in the network must return all devices.

5.3. /NF40/ Modularity

The program should be created with modularity in mind to make it easy to continue the project. Future teams may pick up the works and continue developing it. To make it easy on them the project shall oblige the modularity idea to create an easier entry in project.

6. References

[1] https://github.com/Eiwanger/profinet_scanner_prototype/blob/master/SendPacket/ReadMe.txt
[2] https://www.feldbusse.de/Profinet/grundlagen.shtml
[3] https://www.beckhoff.de/default.asp?fieldbus_components/system_profinet.htm?pk_campaign=Ad-Words-AdWordsSearch-Profinet_DE&pk_kwd=%2Bprofinet%20%2Bdevice