HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

ins institute for networked solutions

IoT - Provisioning und Management

Bachelorarbeit FS 2017

Abteilung Informatik
Hochschule für Technik Rapperswil

Autoren: Andreas Stalder, David Meister
Betreuer: Prof. Beat Stettler, Urs Baumann
Gegenleser: Prof. Dr. Olaf Zimmermann
Projektpartner: INS Institute for Networked Solutions
Datum: 30. März 2017

Inhaltsverzeichnis

1. Design

1.1 Systemübersicht

In der folgenden Abbildung ist das System auf hoher Abstraktionsstufe zu sehen. Die Applikation ist über einen Web Browser bedienbar. Auf dem Management Server werden verschiedenstartige IoT Devices verwaltet. Der Management Server kommuniziert über TCP/IP mit den Devices.

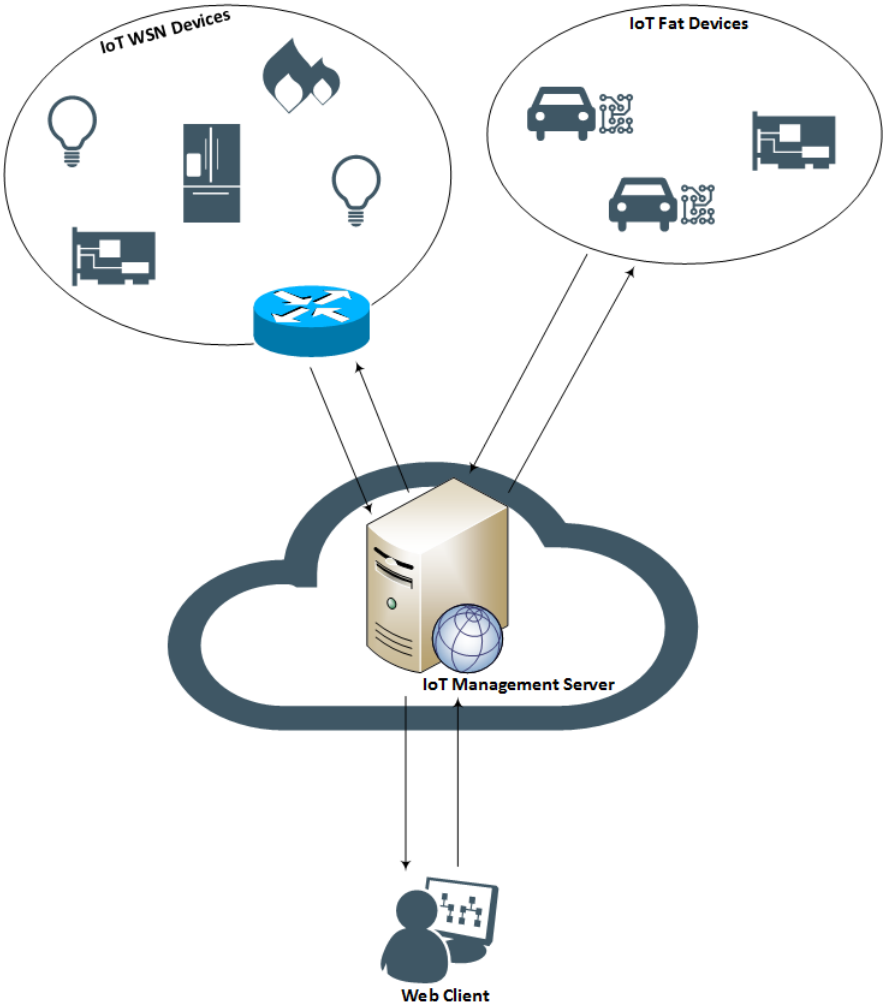

Abbildung 1.1: Systemübersicht

Der User kann seine IoT Devices entweder manuell erfassen, oder über ein Discovery ins System aufnehmen. Sobald ein Device im System ist, können hardware- und konfigurationsspezifische Parameter ausgelesen werden. Gemäss Use Case Analyse sind auch der Austausch von Dateien und das Absetzen von Kommandos vorgesehen.

1.2 Klassenstruktur

1.2.1 Klassendiagramm

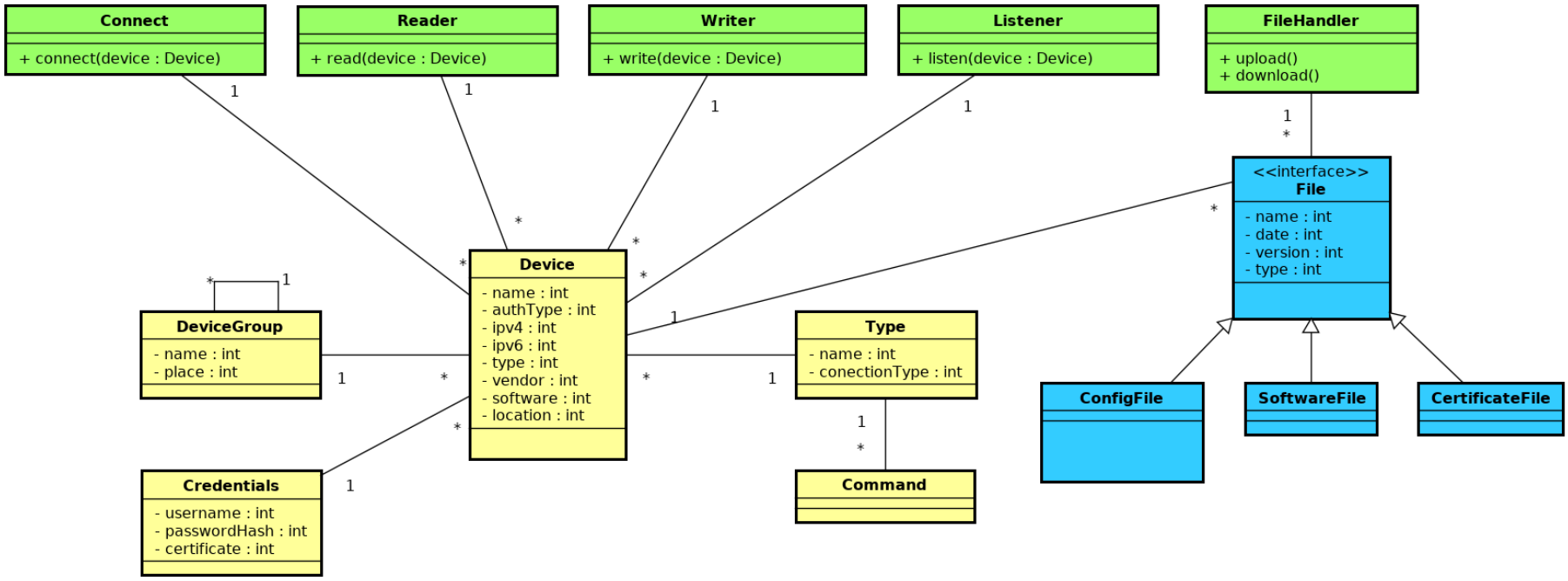

Abbildung 1.2: Klassendiagramm

1.2.2 Klassenbeschreibungen

Connect Die Connect-Klasse ist für die Verbindung zu den Geräten zuständig. Sie behandelt die Authentisierung und stellt die Verbindung den anderen Klassen bereit. Dies ist eine zentrale Klassen, welche bei vielen Tätigkeiten benötigt wird.

Eigenschaft	Beschreibung
connect()	Die Connect Methode, welche die Verbindung zu einem Device aufbaut.

Reader Mit dem Reader werden die Daten von den Devices abgefragt. Je nach Device wird eine andere Implementation der Read-Funktion bereitgestellt, damit alle gewünschten Protokolle unterstützt werden.

Eigenschaft	Beschreibung
read()	Read-Methode, welche die Daten vom gewünschten Device ausliest.

Writer Die Writer-Klasse schreibt die Kommandos und gewünschten Dateien zu einem Device. Diese Klasse wird für das Updaten, Konfig schreiben, so wie andere Parameter verwendet. Je nach Protokoll gibt es einen spezielle Implementation der Write-Klasse.

Eigenschaft	Beschreibung
write()	Die write-Methode schickt die gewünschten Daten zum Device.

Listener Dies ist die Discovery-Klasse. Mit der Listener-Klasse hören wir auf Geräte aus dem Netzwerk und falls welche Vorhanden sind, werden diese in der Datenbank eingefügt. Für die verschiedenen Protokolle gibt es verschiedene Implementationen

Eigenschaft	Beschreibung
listen()	Die Methode nach dem Starten über längere Zeit auf Device-Anfragen und speichert diese.

FileHandler Der FileHandler ist für den Up- und Download zuständig. Er nimmt alle Dateien entgegen und speichert diese auf dem Server ab. Oder er holt eine Datei vom Server und lädt diese herunter, damit man sie mit dem Writer auf ein Device schicken kann.

Eigenschaft	Beschreibung
upload()	Mit dem Upload wird eine Datei vom Filesystem auf das Managementtool geladen.
download()	Durch den download, kann eine Datei vom Managementtool heruntergeladen werden.

Device Device ist eine Datenklasse, welche alle Angaben eines Devices speichert. Jedes Device wird so in ein Objekt gespeichert.

Eigenschaft	Beschreibung
name	Gerätenamen
authType	Authentifikationstyp wie zum Beispiel Passwort oder Zertifikat
ipv4	IPv4-Adresse
ipv6	IPv6-Adresse
type	Gerätetyp
vendor	Gerätehersteller
software	Software
location	Standortangaben des Devices

DeviceGroup Durch die DeviceGroup-Klasse wird das Composite-Pattern umgesetzt. So können die Geräte individuell verschachtelt werden.

Eigenschaft	Beschreibung
name	Device Gruppen Namen
description	Beschreibung der Device Gruppe

Credential Credential-Klasse für die Devices. Durch diese Datenklasse werden alle Usernamen/Passwort kombinationen gehashed abgespeichert. Zusätzlich werden auch die jeweiligen Zertifikate hinterlegt.

Eigenschaft	Beschreibung
username	Benutzername des Devices
password	Devicepassword als Hash
certificate	Zertifikat als Datenblob
hash()	Hashmethode, damit die Passwörter nicht im Klartext gespeichert werden

Type Die Type-Klasse bestimmt die Verbindungsmethode, sowie das benutzte Protokoll. Diese werden pro Device erfasst.

Eigenschaft	Beschreibung
name	Name des Types
connectionType	Typ der Verbindung, wie zum Beispiel Passwort oder Zertifikat
protocoll	Verwendetes Verbindungsprotokoll

Command Mit der Command-Klasse werden alle benötigten Kommandos, wie zum Beispiel SShutdownüsw. erfasst.

Eigenschaft	Beschreibung
command	Device-Kommando

File Das Interface File, bestimmt die Methoden und Variablen, welche die Vererbten Klassen implementieren müssen.

Eigenschaft	Beschreibung
name	Name der abgelegten Datei
date	Datum
Version	Versionsstand der Software oder der Konfigurationsdatei

ConfigFile Die ConfigFile-Klasse ist die Datenklasse für alle Konfigurationsdateien, damit diese in einer Datenbank angepassten Form gespeichert werden können.

Eigenschaft	Beschreibung
-	-

SoftwareFile Diese Datenklasse ist das Objekt für eine Softwaredatei. So kann die Datei in der Datenbank erfasst werden.

Eigenschaft	Beschreibung
-	-

CertificateFile Zertifikatdatei-Datenklasse. Mit dieser Klasse werden Zertifikate in Objekte umgewandelt, damit man sie in der Datenbank abspeichern kann.

Eigenschaft	Beschreibung
-	-

1.3 Logische Architektur

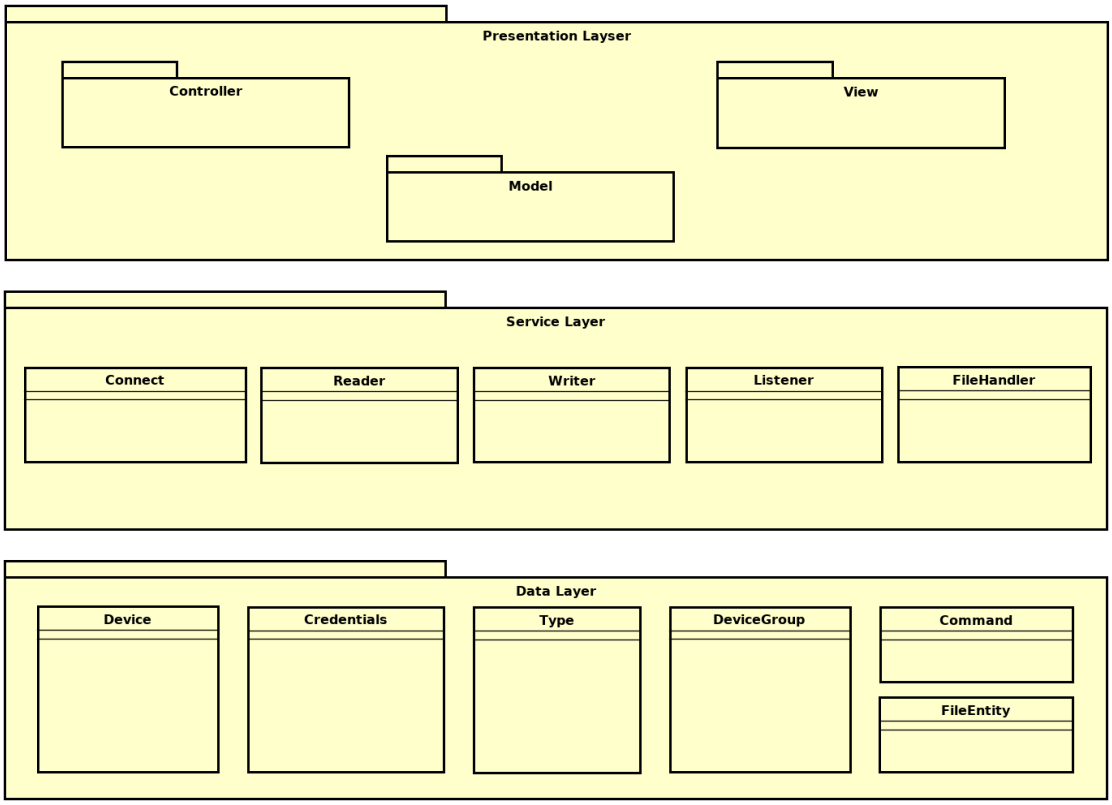

Abbildung 1.3: Logische Architektur

1.3.1 Presentation-Layer

Im Presentation-Layer wird das MVC-Pattern umsetzt. Dazu wird ein Controller, eine View sowie ein Model Package implementiert, welche alle Anfragen bearbeiten und anzeigen.

Packagename	Beschreibung
Controller	Der Controller verwaltet alle Anfragen der View und bearbeitet das Model dementsprechend.
View	Die View bekommt von dem Model die Daten und Renderd dazu die jeweiligen Anzeigen.
Model	Im Model werden alle Daten gehalten, welche die View anzeigt. Nur der Controller hat direkten Zugriff auf diese Daten.

Packagestruktur

Schnittstellen Der Presentation-Layer hat direkten Zugriff auf den Service-Layer. Bis jetzt besteht noch keine weitere Schnittstelle.

1.3.2 Service-Layer

Der Service-Layer beinhaltet alle Backend Klassen, welche für die Verarbeitung der Daten zuständig ist. Hier werden alle Devices erfasst, verbunden und verwaltet.

Klassenname	Beschreibung
Connect	Die Verbindungsklasse des Servicelayer stellt alle Verbindungen zu den Devices auf.
Reader	Alle Daten von den jeweiligen Devices werden von der Reader-Klasse gelesen und an den Data-Layer weitergegeben.
Writer	Der Writer schreibt die gewünschten Daten auf ein Device und aktualisiert die Daten im Data-Layer
Listener	Der Listener horcht auf neue Devices und erfasst diese laufend auf dem Data-Layer
FileHandler	Mit dem FileHandler werden die Dateiobjekte auf dem Data-Layer erstellt und abgespeichert.

Klassenstruktur

Schnittstellen Der Service-Layer hat eine Schnittstelle zum Data-Layer. Durch den Service-Layer werden die Datenobjekte erstellt, bearbeitet und ausgewertet. Der Presentation-Layer muss immer über den Service-Layer, um eine saubere Abtrennung der Schichten zu gewährleisten.

1.3.3 Data-Layer

Der Data-Layer beinhaltet alle Datenobjekte, welche vom laufenden Programm benötigt werden. Diese werden von hier in die Datenbank geschrieben.

Klassenname	Beschreibung
Device	Device ist eine Datenklasse, welche alle Daten von einem Device beinhaltet.
Credentials	Alle Zugriffsdaten der Devices sind in der Credentials-Klasse definiert.
Type	Type beinhaltet die Typendefinition, welche den einzelnen Devices zugeordnet werden.
DeviceGroup	In der Datenklasse DeviceGroup, werden alle Gruppen verwaltet, damit das Composite-Pattern umgesetzt werden kann.
Command	Alle Kommandos werden zentral in der Command-Datenklasse gespeichert.
FileEntitiy	In FileEntitiy wird das Grundgerüst für alle Dateien erstellt, damit diese in einer geeigneten Form in der Datenbank abgespeichert werden können.

Klassenstruktur

Schnittstellen Der Data-Layer hat eine Schnittstelle zu der Datenbank.

1.4 Architekturentscheidungen

In diesem Kapitel werden alle Architekturentscheidungen aufgelistet, welche das Front-End, Back-End, sowie die Datenbank betreffen

1.4.1 Front-End

Themengebiet	Front-End
Problemstellung	Mit welchem Tool soll das Front-End erstellt werden?
Motivation	Diese Entscheidung ist sekundär, da der Fokus nicht auf dem Front-End liegt.
Varianten	• Hardcoded mit Template-Engine • Bootstrap • React.js • Angular.js
Entscheidung	Durch die einfache und schnelle Entwicklungsmöglichkeiten wurde Bootstrap gewählt. Angular.js und React.js benötigen eine zu hohe Einarbeitungszeit und sind daher weniger geeignet. Hardcoding mit einer Template-Enginge ist nicht geeignet, da sie einen sehr hohen Aufwand generieren würden, für eine so einfache Oberfläche.

1.4.2 Back-End

Bereich	Server Back-End
Problemstellung	Was für eine Back-End Technologie soll verwendet werden?
Motivation	Diese Entscheidung ist zentral für die Architektur und Entwicklung des Systems.
Alternativen	• Node.js • Spring Framework • ASP.Net • Play Framework • Ruby on Rails • Django
Entscheidung	Da die grössten Affinitäten bei Java lag, wurde das Spring Framework. Durch die angenehmen Sprachfeatures und dem objektorientierten Ansatz fiel die Entscheidung auf Spring Framework.

1.4.3 Webserver

Bereich	Webserver
Problemstellung	Was für ein Servlet-Container wird verwendet?
Motivation	Diese Entscheidung ist weniger relevant, da der Servlet-Container einfach ersetzt werden kann.
Alternativen	• Jetty • Tomcat • JBoss

Entscheidung	Es wurde Jetty als Servlet-Container gewählt, da er stabil und weit verbreitet ist. Eine einfache Anbindung an das Spring Framework ist auch möglich. Dazu gibt es schon Beispiele auf der Spring Website.

1.4.4 Datenbanktechnologie

Bereich	Datenbanktechnologie
Problemstellung	Wird eine SQL oder eine NoSQL Datenbank verwendet?
Motivation	Die Entscheidung ist weniger wichtig, da beide Varianten möglich wären, ohne grosse Nachteile.
Alternativen	• SQL • NoSQL
Entscheidung	Für dieses Projekt wurde NoSQL als Datenbanktechnologie gewählt. Viele Features von SQL werden bei diesem Projekt nicht benötigt. Auch sind die NoSQL-Datenbanken auf Skalierung ausgelegt, was bei einer solchen Applikation sicher wichtig ist.

1.4.5 Datenbank

Bereich	Datenbank
Problemstellung	Welcher Datenbank soll verwendet werden?
Motivation	Diese Entscheidung ist weniger wichtig, da es viele gute NoSQL Datenbanken gibt. Es muss aber auf den Featureumfang geachtet werden.
Alternativen	• Redis • MongoDB • CouchDB
Entscheidung	tbd

Abbildungsverzeichnis