Masterarbeit

WEBSHOP TO MACHINE

ausgeführt am

Fachhochschul-Masterstudiengang
Automatisierungstechnik

von
Niclas Podrietschnig
1510322026

betreut und begutachtet von
FH-Prof. Dipl.-Ing. Dieter Lutzmayr

Graz, im Jänner 2017 .....

Unterschrift

EHRENWÖRTLICHE ERKLÄRUNG

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die benutzten Quellen wörtlich zitiert sowie inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

.....

Unterschrift

DANKSAGUNG

In diesem Zuge möchte ich meiner Lebensgefährtin Marie, meiner Familie, sowie meinem Chef, Kolleginnen und Kollegen und meinem Betreuer Herrn FH-Prof. Dipl.-Ing. Dieter Lutzmayr für die moralische Unterstützung und für Rat und Tat danken.

KURZFASSUNG

Die vertikale Integration von Geschäftsprozessen ist ein wesentlicher Bestandteil der Idee Industrie 4.0. Der Datenaustausch innerhalb des Unternehmens kann durch Enterprise Resource Planning (ERP) und Manufacturing Execution Systeme (MES) unterstützt werden. Diese Systeme bieten eine hohe Flexibilität und können eine Vielzahl von Daten verarbeiten, ihre Umsetzung und die spezifische Anpassung sind jedoch mit einem erheblichen wirtschaftlichen Aufwand verbunden. Dies ist vor allem für kleine und kleine Unternehmen eine große Herausforderung.

Ziel dieser Masterarbeit ist es, Konzepte für ein flexibles und sicheres System zu erarbeiten, das einen Datentransfer zwischen verteilten Produktionsanlagen und einem Webshop ermöglicht. Dieses System sollte eine automatische Abwicklung von Aufträgen in einem Online-Shop ermöglichen. Zusätzlich soll eine weitere Website für den Betreiber die Überwachungsdaten der Anlagen zur Kontrolle und Optimierung des Herstellungsprozesses zur Verfügung stellen. Zuerst wurde der Beitrag der Idee, Webanwendungen mit der Produktion zu verknüpfen, in Bezug auf die Industrie 4.0 umrissen. Im theoretischen Teil wurden die Gestaltung von Webanwendungen und die Möglichkeiten einer Kommunikation mit der Steuerung der Produktionsanlagen untersucht. Basierend auf diesen Erkenntnissen wurden mögliche Bedrohungsszenarien skizziert und mögliche Gestaltungsvarianten für die Systemarchitektur entworfen. Nach der Auswertung dieser Konzepte wurde das Projekt als Prototyp realisiert und an einer Produktionsanlage getestet.

Das ausgeführte System sorgt für einen sicheren Datenaustausch mit Web-Applikationen und ist besonders einfach in bestehende Systeme zu integrieren. Darüber hinaus ermöglicht es eine automatische Abwicklung der Aufträge im Online-Shop und eine Datenüberwachung für den Betreiber. Es bietet eine wirtschaftlich attraktive, kleine Alternative zu bestehenden MES- oder ERP-Systemen. Basierend auf diesen Ergebnissen sind weitere Langzeitversuche erforderlich, um die höhere Leistung aufgrund der automatischen Auftragsabarbeitung zu bestätigen und weitere Verbesserungen am Prototypen vorzunehmen.

ABSTRACT

The vertical integration of business processes is an essential part of the idea Industry 4.0. Data exchange within the company can be supported by Enterprise Resource Planning (ERP) and Manufacturing Execution Systems (MES). These systems offer high flexibility and can process a wide range of data, however their implementation and the specific adjustment are associated with a considerable economic expense. This is a serious challenge especially for flexible small businesses.

The aim of this master thesis is to expound concepts for a flexible and safe system which offers a data transfer between distributed production plants and a web shop. This system should enable automatic processing of orders, placed in an online shop. In addition, a further website for the operator should provide monitoring data of the plants to control and optimize the manufacturing process. First of all the contribution of the idea to link web applications with the production, with respect to Industry 4.0 was outlined. In the theoretical part the design of web applications and possibilities for communication with the

control of the production plants were examined. Based on this knowledge, possible threat scenarios were outlined and possible design variants for the system architecture were designed. After the evaluation of these concepts the best one was selected to be implemented as a prototype and tested on a production plant.

The expounded system provides a safe data exchange with web applications and is especially easy to integrate in existing systems. Furthermore it enables an automatic processing of the orders in an online shop and data monitoring for the operator. It provides an economical attractive, small alternative to existing MES or ERP systems. Based on these results further long term tests are necessary to confirm the higher performance due to the automatic processing and to modify the prototype of the system.

INHALTSVERZEICHNIS

1 EINLEITUNG

Eine äußerst wirtschaftliche Variante Waren anzubieten stellen Onlineshops dar. Der Bestellprozess verläuft in den meisten Onlineshops automatisch, jedoch werden die Aufträge zurzeit anlagenseitig größtenteils manuell abgearbeitet. Dadurch tritt ein erhöhter Zeitaufwand auf, um Informationen vom Onlineportal in das Produktionssystem zu transferieren. Des weiterem stellt dieser notwendige Schritt eine sehr heikle Fehlerquelle dar, simple Tippfehler können ein Unternehmen schädigen und Kunden aufgrund von Fehllieferungen verärgern.

Die vertikale Integration von Geschäftsprozessen, sprich die Verbindung der Geschäftsebene mit der Produktion, ist ein wesentlicher Bestandteil der Idee von Industrie 4.0. Das Vernetzen von Anlagen mit dem Office Bereich bietet eine attraktive Möglichkeit, Prozesse zu optimieren und dadurch entsprechend effizienter und wirtschaftlicher zu produzieren. Der notwendige Informationsaustausch im Unternehmen kann durch Enterprise Resource Planning (ERP) und Manufacturing Execution Systemen (MES) unterstützt werden. Diese Systeme bieten eine hohe Flexibilität und können verschiedenste Daten verarbeiten. Jedoch ist deren Implementierung sowie die spezifische Anpassung mit erheblichem wirtschaftlichem Aufwand verbunden. Für sehr flexible Kleinunternehmen stellt dies eine ernstzunehmende Herausforderung dar.

1.1 Aufgabenstellung

Die von Kunden im Webshop getätigten Bestellungen eines individuell konfigurierten Produkts, sind möglichst automatisch an einer örtlich unabhängigen Produktionsanlage abzuarbeiten. Außerdem ist dem Anlagenbetreiber in Form einer Webanwendung eine Möglichkeit zum Monitoring der Produktion zur Verfügung zu stellen. Die Konzeption von möglichen Systemarchitekturen, welche einen Informationsfluss zwischen Produktion sowie den Web basierten Schnittstellen zu Endkunden und Betreibern gewährleisten, ist dabei besonders zu beachten. Anschließend ist ein möglichst sicheres und flexibles System auszuwählen, um die automatische Auftragsabarbeitung, von der Bestellung im Onlineshop bis zur Fertigung des gewünschten individuellen Produkts, sowie die Datenerfassung von Prozessdaten an einer Testanlage zu erproben. Das in Abbildung 1 äußerst grob skizzierte System soll kleine bis mittlere Produktionsanlagen Industrie-4.0-tauglich machen. Eine Datenbank ist demnach das zentrale Element des Systems und beinhaltet sämtliche Informationen, welche von einem Webserver an die Clients zu verteilen sind. Es sollte eine Alternative zu kostenintensiven ERP und MES Systemen darstellen, wobei ein Onlineshop die Schnittstelle zum Endkunden bildet.

1.2 Zielsetzung

Im Verlauf der Masterarbeit ist ein im Sinne von Industrie 4.0 konzipiertes System aufzubauen. Dieses soll zeigen, wie Informationen (Aufträge vom Webshop und Prozessdaten der Produktion) von einem Onlineportal, welches den Webshop und die Monitoring-Daten beinhalt, auf dezentrale Produktionsmaschinen übertragbar sind. Das neu entwickelte System soll die Möglichkeit bieten, dass eine Maschine die Bestellungen aus einem zu Demonstrationszwecken erstellten Webshop automatisiert

übernimmt. Außerdem ist der Prozess mittels Monitoring-Ansicht überwachbar. Dafür wird zunächst der technische Aufbau eines Onlineshops bzw. von Webanwendungen betrachtet, sowie Möglichkeiten zum Datenaustausch mit einer SPS gesteuerten Produktionsmaschine näher analysiert, um anschließend die Entwicklung von Konzepten für eine Verbindung zwischen beiden Komponenten zu ermöglichen. Außerdem wird eine ausgewählte Konzeption für das Produktionssystem auf einer eigens dafür vorgesehen Produktionsanlage getestet.

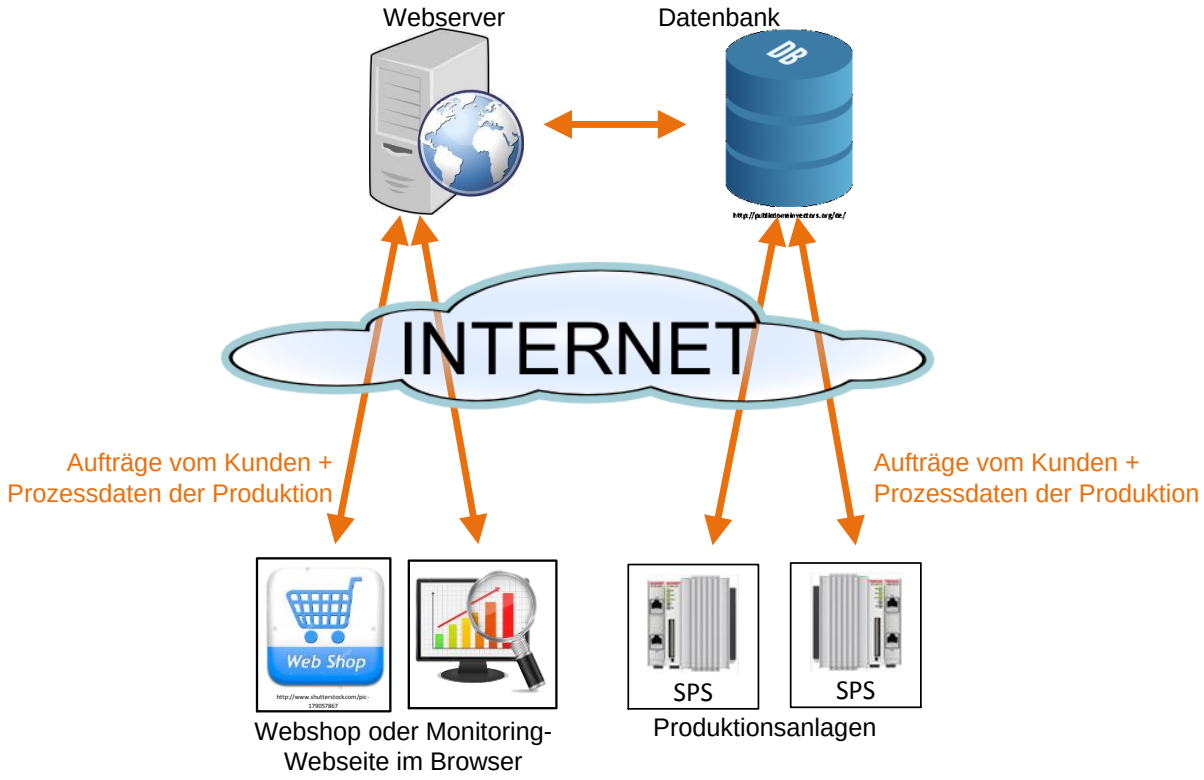


Abbildung 1: Prinzipieller Aufbau, Quelle: Eigene Darstellung.

2 INNOVATIONSGRAD IN HINSICHT AUF INDUSTRIE 4.0

Bevor näher auf den technischen Aufbau eines Onlineshops bzw. von Webanwendungen und auf mögliche Kommunikationsvarianten des zu realisierenden Systems eingegangen wird, ist in diesem Kapitel der Innovationsgrad der vorgeschlagenen Lösung, nämlich einen Webshop mit einer Maschine zu verbinden, sowie eine Webanwendung zum Monitoring von Prozessdaten zu verwenden zu unterstreichen. Damit soll gezeigt werden, dass diese Idee einen Beitrag zur vierten industriellen Revolution leistet und welche Prozesse sie übernehmen soll. Zusätzlich wird die grundsätzliche Funktion des zu konzipierenden Systems hier weiter präzisiert.

Abbildung 2 zeigt eine beispielhafte Darstellung der derzeit bestehenden IT-Systeme in einem produzierenden Unternehmen. Diese sind in die Stränge Produkt, Auftrag, Technologie sowie Fabrik eingeteilt und dienen dazu die Produktion zu ermöglichen bzw. zu unterstützen. Das zu realisierende System sollte durch den Webshop die Konfiguration der Bestellung (Kunden konfigurieren die Rezeptur des Produkts im Webshop), sowie die Auftragsbearbeitung (Aufträge in der zentralen Datenbank ablegen) übernehmen, wobei der Typ des Produkts immer gleich bleibt, nur die Rezeptur ist anders. Das im Zusammenhang mit Industrie 4.0 erwähnte Konzept der Selbstkonfiguration der Produktionsmaschinen wird (noch) nicht verfolgt. Eine Kernfunktionalität eines MES-Systems, die Produktionsplanung (Aufträge an Maschinen verteilen) könnte ebenfalls von einer Webanwendung übernommen werden, jedoch wird derzeit ein Ansatz im Sinne der vierten industriellen Revolution verfolgt. Die Produktionsmaschinen sollten nach dem Pull-Prinzip die Aufträge vom Webshop selbständig (nach Benutzerfreigabe) abarbeiten. Jedoch wird das Monitoring von Prozessdaten als Teil eines herkömmlichen MES-Systems betrachtet.

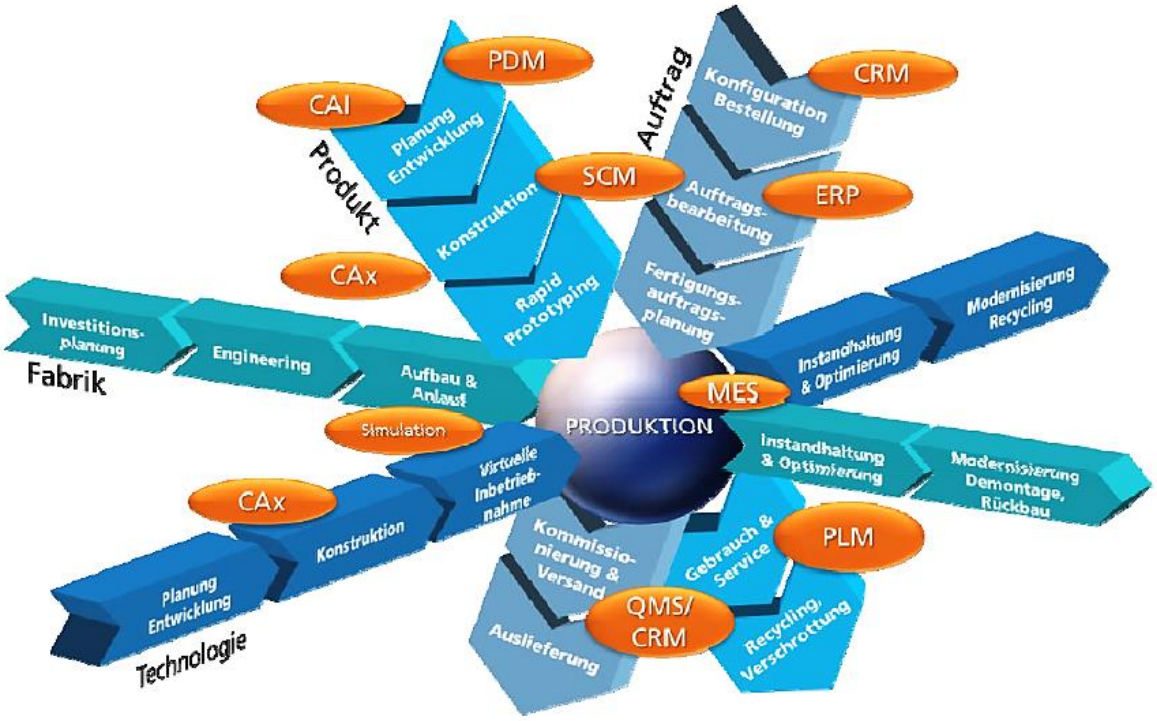


Abbildung 2: Beispielhafte IT-Systeme eines produzierenden Unternehmens, Quelle: Bauernhansel/ten Hompel/Vodel-Heuser (2014), S. 583.

Derzeitige Produktions-IT-Systeme, besonders die entlang der in Abbildung 3 dargestellten klassischen Automatisierungspyramide integrierten Systeme (z.B. ERP- oder MES-Systeme), sind meist umfangreiche Software-Suiten mit eigenen Datenbanken. Für eine horizontale Integration mit durchgängigem Daten und Informationsaustausch über die Grenzen von Unternehmen oder Produktionsstandorten hinweg sind diese Systeme in der Regel nicht vorgesehen. Die horizontale Integration findet, wenn überhaupt, über die ERP-Systeme statt. Aufgrund der aufwändigen Einführung und der aufwändigen Integration dieser bewährten Produktions-IT-Systeme untereinander, gehören diese bei Klein- und Mittelbetrieben nicht unbedingt zur Standardausstattung.[1]

Die automatisierte Fabrik der Industrie 3.0 ist durch das hierarchisch aufgebaute System der Automatisierungspyramide geprägt (siehe Abbildung 3). Das ERP-System tauscht Daten mit dem darunterliegenden MES aus, dieses wiederum mit den darunterliegenden Ebenen bis hin zur Feldebene.[2] „Die Verknüpfung von industrieller Fertigung und Informationstechnologie im Sinne von Industrie 4.0 verspricht dagegen einen wachsenden Grad an Vernetzung und Flexibilität: Vertikal sind Maschinen, Produktionsanlagen und Lagersysteme zunehmend in der Lage, untereinander Informationen auszutauschen, Aktionen in die Wege zu leiten oder – sozusagen die Kür – sich selbst zu steuern. Horizontal lassen sich diese Systeme nun nahtlos zum Beispiel zur Lieferkettensteuerung oder für verbesserte Wertschöpfungsketten in Unternehmen nutzen.“ [3]

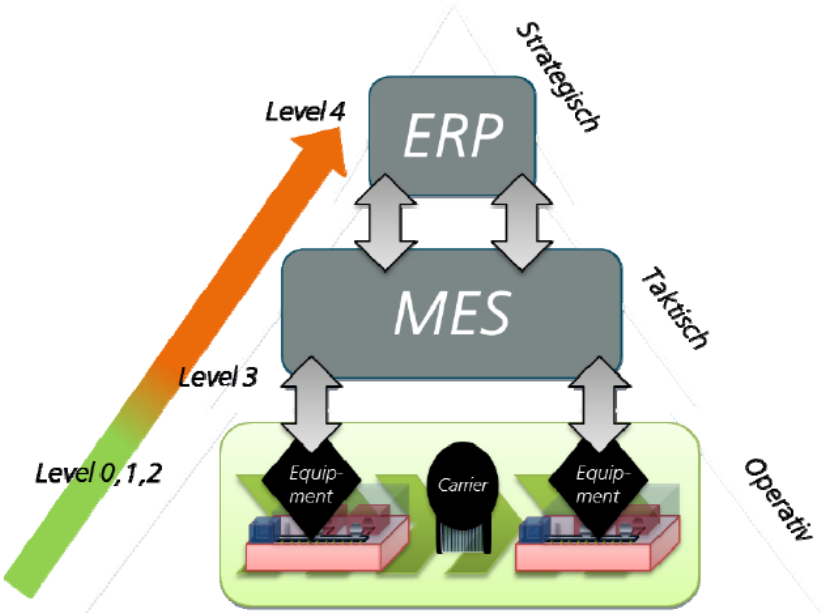


Abbildung 3: Vereinfachte Automatisierungspyramide nach ISA-95, Quelle: Bauernhansel/ten Hompel/Vodel-Heuser (2014), S. 585.

In der Vision von Industrie 4.0 besteht u.a. die Bestrebung Produktionsanlagen und Standorte horizontal, sowie vertikal zu vernetzen. Die bereits in der Aufgabenstellung grob skizzierte Systemarchitektur des im Zuge der Masterarbeit zu realisierenden Systems, basiert auf einer über das Internet erreichbaren zentralen Datenbank. Dieser Datenspeicher soll demnach als Informationsträger für alle drei Teilsysteme

[1] Vgl. Bauernhansel/ten Hompel/Vodel-Heuser (2014), S. 593 ff.

[2] Vgl. Bauernhansel/ten Hompel/Vodel-Heuser (2014), S. 571 ff.

[3] Bauernhansel/ten Hompel/Vodel-Heuser (2014), S. 571.

(Maschinen, Monitoring, und Webshop) konzipiert werden und somit die gewünschte vertikale und horizontale Integration ermöglichen.

In Abbildung 3 wird zusätzlich die Entwicklung der Produktions-IT im Zuge von Industrie 4.0 aufgezeigt. Wobei der Ansatz verfolgt wird, die lokale Produktions-IT-Funktionalität in einer Cloud in Form von Apps bereitzustellen. Im Rahmen der sogenannten Appisierung werden einzelne Funktionen der Systeme in der Cloud in Form von Apps zur Verfügung gestellt. [4]

Die weitestgehend plattformunabhängigen Webanwendung für das Monitoring von Prozessdaten, sowie der Webshop zur Auftragsgenerierung sind demnach als Apps anzusehen. Es könnten auch native Apps (z.B. für Android-Smartphones, oder Anwendungen für Windows-PCs) zur Verwendung kommen, jedoch müsste hier für jede Plattform eine eigene Anwendung implementiert werden. Die bereits in der Aufgabenstellung geforderte, für den Betrieb des Webshops essentielle und über das Internet erreichbare Datenbank dient hierbei als zentraler Datenspeicher. Wobei nur ein, und nicht mehrere verteilte physikalische Server genutzt werden.

[4] Vgl. Bauernhansel/ten Hompel/Vodel-Heuser (2014), S. 593 ff.

3 WEBANWENDUNGEN

Wie bereits in der Aufgabenstellung skizziert, soll der Informationsaustausch mit dem Kunden sowie das Monitoring in Form von Webanwendungen stattfinden. Der Webshop als Kundenschnittstelle wird als datenbankgestützte Webanwendung mit zusätzlichem Bezahldienst angesehen. Zusätzliche Funktionen wie beispielsweise ein Warenkorb oder eine Benutzerdatenerfassung sind meist Teil eines Shopsystems, jedoch sind sie für die weiteren Untersuchungen hier nicht relevant, da in dem zu realisierenden System besonders die Auftragserstellung durch den Webshop von Bedeutung ist. Ebenso sind Monitoring-Daten der Produktionsmaschinen für den Betreiber ersichtlich im Web darzustellen.

Im folgenden Kapitel ist der Aufbau einer Webanwendung, deren Kommunikationsmöglichkeiten mit einem Webserver, sowie verwendete Programmiersprachen und Softwarekomponenten näher darzustellen. Anschließend werden, dies ist vor allem für den Webshop relevant, Funktionsweisen von gängigen Bezahlsystemen vorgestellt. Diese Kapitel soll einen Teil der Basis bieten, um die Entwicklung von Konzepten einer Systemarchitektur für eine Verbindung zwischen den Webanwendungen der zentralen Datenbank und den dezentralen Produktionsmaschinen zu ermöglichen.

3.1 Grundsätzliche Funktionsweise von Webanwendungen

Die für Webanwendungen benötigten statischen und dynamischen Webseiten, sowie weitere Daten für deren Darstellung (z.B. Bilddateien) sind auf einem Server gespeichert. Der Client (z.B. Browser oder .NET-Anwendung) fordert die Inhalte vom Webserver an und zeigt die statische Webseite grafisch an. Statische Webseiten bestehen ausschließlich aus HTML-Code (Hypertext Markup Language), sowie einer darauf aufbauenden Designbeschreibung CSS (Cascading Style Sheets) und den verknüpften Bilddateien. Im Gegensatz dazu kann bei dynamischen Webseiten der Inhalt je nach Bedürfnis des Benutzers generiert werden. Dynamisch generierte Webseiten beinhalten ein serverseitiges Skript, wie PHP (Hypertext Preprocessor), welches den passenden HTML-Code zur Zeit der Anfrage generiert. Viele Aufgaben können jedoch ebenfalls mittels clientseitigen Skripten (z.B. JavaScript) im Webbrowser abgearbeitet werden. Mit Erweiterungen wie AJAX (Asynchronous JavaScript and XML) kann durch asynchronen Datenaustausch, fast die Performance einer Desktop-Anwendung erreicht werden.

3.2 Clientseitige Programmiersprachen

3.2.1 Beschreibung der Webseite mit HTML

Die Basis jeder Webseite bildet HTML als Dokumentenbeschreibungssprache mit einem festgelegten Satz von Anweisungen. Im Wesentlichen sind darin die logischen Strukturen eines Dokuments beschrieben, die vom Browser (Client) interpretiert werden. Da HTML-Dokumente aus reinem Klartext bestehen sind sie plattformunabhängig. Ab Version 4 sind in HTML Schlüsselwörter enthalten, wodurch der Aufruf von Funktionen wie JavaScripts mittels Eventhandler ermöglicht wird. Anhand der ständigen Weiterentwicklung von Beschreibungssprachen wie HTML sind über die Zeit eine Vielzahl von Zwischenversionen und herstellerspezifischen Spezialvarianten wie Beispielsweise XHTML entstanden. Ein wesentlicher Unterschied besteht bei XHTML in der (fast) nicht vorhandenen Fehlertoleranz. Bei

HTML-Seiten führt der Browser alle korrekten bzw. bekannten Anweisungen aus. Fehlerhafte Anweisungen werden einfach ignoriert und als unformatierter Text angezeigt.[5]

3.2.2 Optimierung durch das Document Object Model

Im Document Object Model (DOM) wird eine (X)HTML-Seite nicht als statisch aufgebaute, fertige und nicht unterscheidbare Einheit, sondern als Struktur betrachtet. Die einzelnen Bestandteile dieser Baumstruktur sind Programmen und Skripten dynamisch zugänglich. Das DOM-Konzept veranlasst den Browser die (X)HTML-Seite, einerseits wie eine Textdatei zu lesen, sowie HTML-Anweisungen auszuführen. Zusätzlich werden die im Code enthaltenen von ihm identifizierbaren Elemente anhand ihrer relevanten Eigenschaften, dem Typ, sowie ihrer Position innerhalb der Webseite indizieret. Daraus wird, solange die Webseite im Browser geöffnet ist, im Hauptspeicher des Clients (z.B. PC) eine Baumstruktur generiert. Die Verwaltung erfolgt in Feldern die der Browser logisch gruppiert. Diese Vorgehensweise ermöglicht dem Browser beim Laden einer Webseite eine genaue Kenntnis sämtlicher für ihn relevanten Daten aller eigenständig ansprechbaren Elemente zu erlangen. Die Art und die Verwendung der Elemente kann je nach verwendeten Browser variieren. Die Fehlertoleranz spielt hier eine besondere Rolle. Viele Browser können fehlerhafte Elemente automatisch ergänzen oder weglassen, wodurch die Struktur des DOM-Baums möglicherweise unterschiedlich aufgebaut sein kann. Ein zuverlässiger Zugriff wird hiermit erschwert. Das DOM-Konzept bildet einen Grundbaustein für dynamische Webseiten, denn es ermöglicht, jedes ansprechbare Element (z.B. ein (X)HTML-Tag) mittels Skript auch nach dem Seitenaufbau einzeln zu aktualisieren oder zu verändern. [6]

3.2.3 Die neuere Spezifikation für Dokumentenbeschreibung - HTML5

Die neueste Spezifikation der Dokumentenbeschreibungssprache HTML5 (Oktober 2014) bietet Vorteile in den Bereichen von dynamischen 2D und 3D-Grafiken, sowie der nativen Video- und Audioverarbeitung. Das lokale Speichern von Informationen, das Verlagern von Aufgaben in Hintergrundprozesse und Drag-and-Drop Funktionen werden ebenso ermöglicht. Es bietet außerdem mehrere neue native Spezifikationen von Formulareingaben und stellt neue Benutzereingabeelemente sowie eine Erweiterung der Datums- und Farbkomponenten bereit. Zusätzlich soll HTML5 die Fehlerfreiheit von Webseiten vorantreiben, wobei es nicht so rigide wie XHTML ist. Jedoch ist geplant anhand der daraus folgenden Wohlgeformtheit des HTML-Codes eine konsistente Erweiterung der DOM-Schnittstelle zu gewährleisten. Weitere Schlagwörter des HTML5-Standards sind: Sicherheit, Vereinfachung, Konsistenz, Zugänglichkeit und Universalität. Die HTML-Spezifikation ist in Zukunft als sogenannter Living Standard konzipiert. Ständige Korrekturen und Erweiterungen sind demnach vorgesehen. Wie bereits erwähnt, bietet HTML5 neue strukturierende Elemente wie section, article, nav, aside, header oder footer. Sie ermöglichen eine übersichtlichere Strukturierung der Webseite als die bisher verwendeten div-Elemente oder Tabellen, denn sie bezeichnen die Struktur und die Art ihres

[5] Vgl. Steyer (2014), S. 22 ff.

[6] Vgl. Steyer (2014), S. 26 ff.

Inhalts, zum Beispiel enthält nav eine Navigation oder section einen zusammenhängenden Textabschnitt. [7]

Ein Großteil der in HTML5 enthaltenen Elemente wird derzeit von allen neuesten Versionen der gängigen Browser unterstützt. Erkennt der Browser das Element nicht, wird es entsprechend dem Fehlertoleranz-Prinzip behandelt und wird somit in der DOM-Struktur ergänzt oder ausgelassen.

3.2.4 Clientseitige Funktionen mit JavaScript

Die Skriptsprache wird direkt im Browser ausgeführt, sie bietet unter anderem die Möglichkeit Benutzeraktionen auszuwerten und Inhalte während der Anzeige im Browser zu ändern.[8] JavaScript 1.5 basiert auf dem 1999 von der ECMA ECMA (European Computer Manufactures Association) standardisierten ECMAScript-Edition 3. Die Version 1.5 wird im Web immer noch am häufigsten verwendet, nur wenige Browser unterstützten neuere Formate. Die Anweisungen eines JavaScripts können einfach durch Eintragen des Klartextes in den HTML-Code in eine Webseite eingebunden werden. Der Skriptbereich ist allerdings eindeutig vom restlichen Code zu trennen. Mittels HTML-Steueranweisungen <script> bzw. </script> wird ein Container für die Skriptanweisungen realisiert. Der Browser interpretiert den darin enthaltenen Code als Skript. Außerdem können externe Skripts, die sich in eigenen Dateien am Server befinden, aufgerufen werden. [9]

3.3 Webserver

Wie in Abbildung 4 ersichtlich ist ein Webserver die zentrale Hardware- und Softwarekomponente, um eine Webanwendung zu betreiben. Weitverbreitete Webserver-Softwaresysteme sind beispielsweise der Apache HTTP-Server und der Microsoft Internet Information Server (IIS).

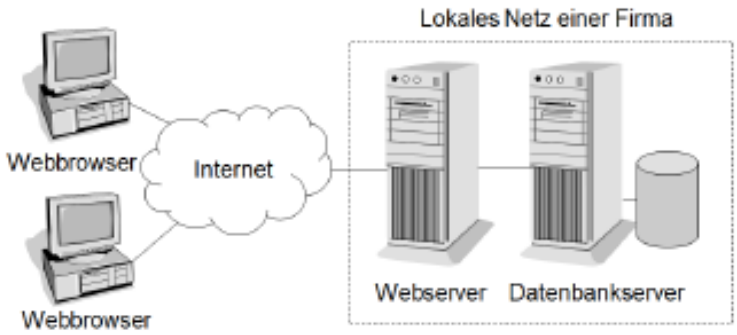


Abbildung 4: Webanwendung: Quelle: Abts (2015): S. 5.

Im Gegensatz zu Microsoft IIS ist Apache ist auf mehreren Betriebssystemen lauffähig, wobei diese teilweise nur bedingt für den dauerhaften Einsatz geeignet sind. Jedoch kann eine Webserver-Anwendung in vielen verschiedenen Programmiersprachen (z.B. Java) implementiert werden. Ein

[7] Vgl. Steyer (2014), S. 26 ff.

[8] Vgl. Stender (2011), S. 4 ff.

[9] Vgl. Steyer (2014), S. 40 f.

zusätzlicher Datenbankserver kann physisch am selben Gerät implementiert sein, oder über ein Netzwerk mit dem Webserver verbunden werden. Die Clients stellen die Verbindung mit dem Webserver über das Internet oder über ein lokales Netzwerk her. Unabhängig von der physischen Konfiguration besteht die Hauptaufgabe eines Webservers in der Bereitstellung von statischen und dynamischen HTML-Dateien welche vom Client (Webbrowser) interpretiert werden.

Derzeit sind mehrere Varianten von Webservern am Markt vertreten:

- Webhosting-Pakete (Shared Virtual Hosting): Hier werden die Websites vieler Kunden auf einem gemeinsamen physikalischen Server betrieben. Wobei alle Kunden-Websites, je nach gewählten Paket, gleich behandelt werden müssen. Außerdem ist die Domain meist im Angebot enthalten. Die Adresse der Webseite sowie die benötigte Namensauflösung (DNS-Service), für eine komfortable weltweite Erreichbarkeit sind demnach nicht vom Kunden zu organisieren. Diese Lösung bietet entsprechend wenig serverseitige Flexibilität, jedoch eine hohe Verfügbarkeit und Wirtschaftlichkeit. (z.B. one.com)
- Rootserver (Dedicated Server): Hierbei wird ein eigener physikalischer Server betrieben, diese Lösung bietet ein Höchstmaß an Leistung und Flexibilität. Jedoch müssen für eine weltweite Erreichbarkeit ein DNS-Server und eine Domain organisiert werden. Der Internetverbindung kommt dabei besondere Bedeutung zu. Aufgrund der nötigen Hardware sind hohe Anschaffungs- und Wartungskosten zu erwarten.
- Dedicated Virtual Hosting: Der Webserver wird auf Betriebssystemebene virtualisiert. Mehrere logisch voneinander abgekapselte Instanzen eines Betriebssystems werden somit auf einer gemeinsamen Hardware betrieben. Beliebte Vertreter dieser Technologie sind Produkte wie Xen oder VMware. Dieses Verfahren kann durch den minimalen Hardwareeinsatz recht günstig vom Webhosting-Betreiber (z.B. Amazon) angeboten werden, bietet aber die Flexibilität eines eigenständigen Servers. [10]

3.3.1 Serverseitige Funktionen mit Hypertext Preprocessor

PHP (PHP: Hypertext Preprocessor), früher als Personal Home Page Tools bezeichnet, ist eine serverseitige Skriptsprache, welche von vielen Webservern standardmäßig unterstützt wird. Das im HTML-Code eigebettete PHP wird vom Webserver an eine PHP-Laufzeit übergeben. Das Ergebnis wird als reines HTML an den Webserver zurückgereicht und anschließend an den Client übertragen. Die objektorientierte Sprache ist frei verwendbar und wird ständig weiterentwickelt. Viele objektorientierte Sprach-Konstrukte wie Überladung oder Exceptions sind möglich. Besonders die Integration von Datenbanken steht im Vordergrund, weshalb PHP viele Datenbanktreiber nativ unterstützt.[11] PHP 5 (ab 2004) ist am weitesten verbreitet, jedoch ist davon auszugehen, dass es von der neuesten Variante PHP 7 (ab Dezember 2015) schrittweise ersetzt wird.

[10] Vgl. Hammer/Bensmann (2011), S. 160 ff.

[11] Vgl. Stender (2011), S. 25 ff.

3.3.2 Node.js als Alternative

Neben PHP kann auch das 2009 veröffentliche Node.js serverseitig Daten verarbeiten. Node.js basiert auf der JavaScript-Laufzeit V8, welche ursprünglich für den Webbrowser Google Chrome entwickelt wurde. Als Programmiersprache kommt demnach, wie sonst bei Browsern üblich, JavaScript zum Einsatz. Der einfache Aufbau der Plattform ermöglicht schnelle und skalierbare Netzwerk-Anwendungen. Node.js verwendet ein ereignisgesteuertes, nicht-blockierendes I/O-Modell, das es leicht und effizient macht. Es ist demnach ideal für datenintensive Echtzeitanwendungen die über verteilte Geräte laufen.[12] Ein sehr häufiger Anwendungsfall für Node.js ist das Erstellen von verschiedensten Arten von Serveranwendungen. Wobei es im Gegensatz zum PHP keine zusätzliche Anwendung benötigt. Beispielsweise läuft eine PHP-Anwendung neben einem Apache-HTTP-Server. Ein mittels Node.js implementierter HTTP-Server kann jedoch beide Funktionen in einer Applikation abbilden.[13] Dessen ungeachtet wird Node.js derzeit von sehr wenigen Webhosting-Anbietern angeboten.

3.3.3 Datenbankanbindung vom Webserver zur Datenbank

Wird eine statische Webseite vom Browser angefordert, sendet der Webserver diese sofort und ohne Änderungen zurück. Die Anforderung einer dynamischen Webseite mit Datenbankanbindung ist in Abbildung 5 veranschaulicht. Fordert der Browser eine dynamische Seite an, wird diese einen Anwendungsserver weitergegeben. Diese Software liest den in der Seite implementierten Code (z.B. PHP), tauscht Daten mit externen Stellen aus (z.B. Datenbankanfrage) und erstellt anschließend den HTML-Code gemäß den Anweisungen. Außerdem wird der am Server ausgeführte Code aus der Webseite entfernt. Der Webserver erhält somit eine statische HTML-Seite, die er dem Browser anschließend zurückschickt. Bei Datenbankabfragen übernimmt ein Datenbanktreiber die Vermittlung zwischen Anwendung (z.B. PHP) und der Datenbank.[14]

Eine Anwendung kann durch die Verwendung der Abfragesprache SQL (Structured Query Language) Daten einer relationalen Datenbank definieren, abfragen und manipulieren. Der Zugriff auf die Datenbank ist mittels Benutzername und Passwort geschützt. Die Anmeldeinformationen sind während des Verbindungsaufbaus bekanntzugeben. Der Sprachstandard SQL ist in Form von ANSI (American National Standards Institute) oder ISO (Internationale Organisation für Normung) standardisierten Dialekten in den meisten Datenbanken implementiert. Aufgrund der festgelegten Syntax von Abfragen kann eine Applikation unabhängig von dem verwendeten Datenbanksystem entwickelt werden. Die meist genutzten Datenbanken sind MySQL, Oracle und PostgreSQL, wobei MySQL als Datenspeicher für Webapplikationen häufig in Verbindung mit PHP und dem Webserver Apache eingesetzt wird. [15]

[12] Vgl. Cantelon/Harter/Holowaychuk/Rajlich (2014): S. 4 ff.

[13] Vgl. Cantelon/Harter/Holowaychuk/Rajlich (2014): S. 12 f.

[14] Vgl. Adobe (2016), Online-Quelle [12.09.2016].

[15] Vgl. Hammer/Bensmann (2011), S. 115.

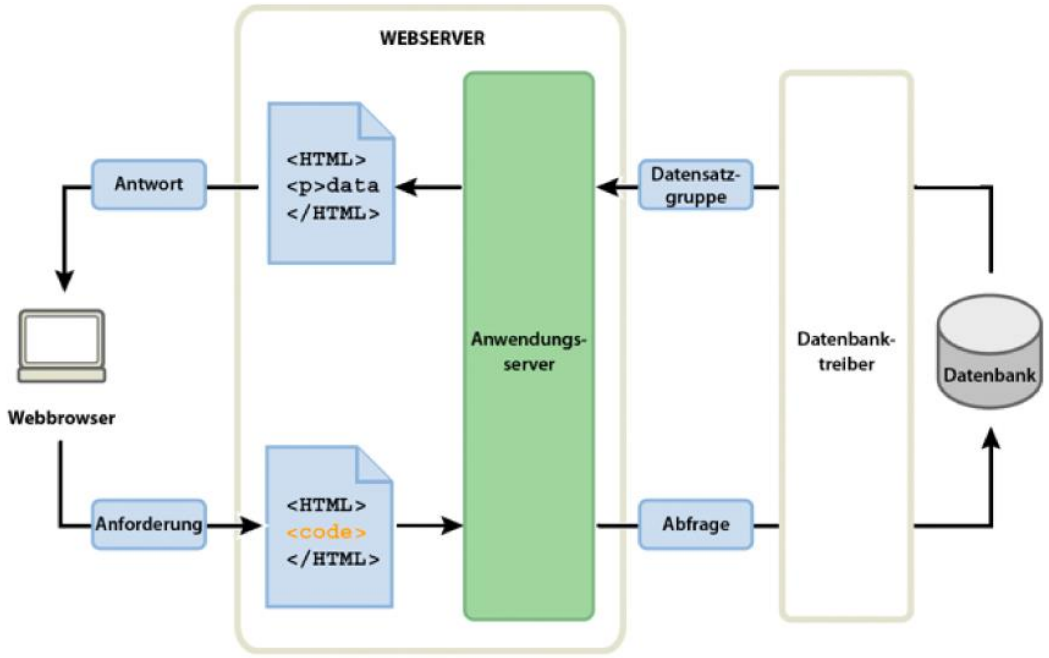


Abbildung 5: Anforderung einer dynamischen Webseite, Quelle: Adobe (2016), Online-Quelle [12.09.2016].

3.4 Varianten zum Datenaustausch zwischen Webserver und Client

Die Kommunikationsgrundlage zum Datenaustausch zwischen den Browser und Webserver bildet das IP- bzw. das TCP/IP- Protokoll. Als Übertragungsprotokoll wird http (Hypertext Transfer Protocol), darauf basierende Protokolle oder bei gesicherten Verbindungen vor allem HTTPS (Hypertext Transfer Protocol Secure) verwendet. [16]

3.4.1 Das Hypertext Transfer Protocol als Grundlage

Das Hypertext Transfer Protocol ist, in der Anwendungsschicht im TCP/IP-Schichtenmodell angesiedelt. Es regelt wie der Webbrowser mit dem Server im World Wide Web kommuniziert und basiert auf TCP/IP. Zum Austausch von Daten muss der Client die URL (Uniform Resource Locator) der gefragten Datei am Webserver kennen. Eine URL besteht im Wesentlichen aus dem verwendeten Protokoll, dem Host-Name oder IP-Adresse des Servers. Optional kann das verwendete Port (z.B. bei HTTP Port 80) und die Resource (z.B. weiterer Dateipfad) angegeben werden, beispielsweise: protocol://server[:port][/resource].[17]

Der Ablauf des Datenaustausches zwischen HTTP-Client und Server erfolgt in folgenden 9 Schritten:

1. Warten des Servers auf eingehende HTTP-Anfragen (Requests).
2. Der Browser (Client) erzeugt eine URL (z.B. Eingabe des Benutzers).
3. Versuch des Clients eine TCP-Verbindung zum Server aufzubauen.
4. Der Verbindungswunsch wird vom Server akzeptiert.

[16] Vgl. Hollosi (2012), S. 231 ff.

[17] Vgl. Abts (2015), S. 162 ff.

5. Senden der Nachricht (HTTP-Anfrage) an den Server und Anfordern der Ressource (mit dem spezifizierten Teil der URL).
6. Verarbeitung der Anfrage am Server (z. B. Ausführung einer Datenbankabfrage und Generierung einer HTML-Seite mit dem Abfrageergebnis).
7. Der Server sendet eine Antwort (HTTP-Response) an den Client. Diese Antwort enthält die angeforderte Ressource oder eine Fehlermeldung.
8. Verarbeitung der Antwort am Client.
9. Schließen die TCP-Verbindung vom Client oder vom Server. [18]

HTTP ist ein zustandsloses Protokoll. Der Server hat somit keine Kenntnis über die vorangegangenen Anfragen der Clients. Bei HTTP/1.0 wird für jede HTTP-Anfrage eine neue TCP-Verbindung aufgebaut. Zudem erfolgt die Bearbeitung am Server unabhängig von vorhergehenden HTTP-Anfragen. Enthält eine angeforderte HTML-Seite beispielsweise mehrere Grafiken, wird jede dieser Bilddateien separat angefordert, wobei der ständige Verbindungsauf- und abbau nicht zu verhindern ist. [19]

Moderne Browser bauen als Abhilfe jedoch nicht nur eine, sondern mehrere parallele, temporäre Verbindungen zu Servern auf. Dadurch wird die Ladezeit der Webseite erheblich reduziert. Der Ladevorgang beim Öffnen der Seite beginnt mit der HTML-Seite, dem CSS- und JavaScript-Dateien. Zuletzt werden mit bis zu acht parallelen Kanälen Bilddateien geladen. Ladezeiten zu optimieren kann anhand vieler verschiedener Maßnahmen geschehen, unter anderem durch das Verwenden eines Caches zur Datenzwischenspeicherung. [20]

Ein HTTP-Request ist in folgende vier Teile gegliedert:

- Kopfzeile: Hier wird die verwendete HTTP-Methode (z.B. GET), der Name der angeforderten Ressource (jedoch ohne Protokoll und Domain-Namen) sowie Protokollversion (z.B. HTTP/1.0) deklariert.
- Optionale Anfrageparameter: Mittels Anfrageparameter werden dem Server zusätzliche Informationen über den Client und seine Anfrage zur Verfügung gestellt. Jeder Parameter setzt sich aus dem Namen, einem Doppelpunkt und dem Wert zusammen und steht in einer eigenen Zeile. Hier können beispielweise unter Host der Rechnername und die Portnummer des Clients, oder mittels accept die am Client akzeptierten Medientypen im MIME-Standard (Multipurpose Internet Mail Extension) bekanntgegeben werden.
- einer Leerzeile (\r\n).
- Nutzdatenteil (optional): Hier befinden sich die in einem Formular eingetragenen Daten, welche bei Anwendung der Methode POST (Daten an Server übertragen) mitgesendet werden. [21]

Eine einfache Form eines HTML-Requests ist in Abbildung 6 dargestellt. Die Methode GET fordert Daten vom Server an, dieser antwortet mit einer HTTP-Antwort (Response).

[18] Vgl. Abts (2015), S. 162 ff.

[19] Vgl. Abts (2015), S. 162 ff.

[20] Vgl. Hollosi (2012), S. 231 ff.

[21] Vgl. Abts (2015), S. 164 ff.

Abbildung 6: HTTP-Transaktion: Quelle: Abts (2015): S. 162.

Die HTTP-Antwort ist folgendermaßen aufgebaut:

- Kopfzeile. Sie beinhaltet die Protokollversion, einen Status-Code und eine optionale Status-Meldung. Der Statuscode 200 steht für OK, 404 für Not Found oder 500 meldet einen Internal Server Error.
- Antwortparameter: Diese Parameter liefern zusätzliche Informationen über den Server und die Antwort. Beispielsweise kann hier der verwendete MIME-Typ (z.B. text/html) oder der Zeitpunkt der Beantwortung (Date) enthalten sein.
- Eine Leerzeile.
- Nutzdaten: Hier sind die angeforderten Nutzdaten der Ressource enthalten, beispielsweise eine HTML-Datei. [22]

3.4.2 Verschlüsselte Übertragung mittels Hypertext Transfer Protocol Secure

Das Hypertext Transfer Protocol Secure gewährleistet eine verschlüsselte Kommunikation mittels Authentifizierung zwischen einem Webbrowser und dem Webserver. HTTPS kombiniert HTTP und das verwendete sichere Übertragungsprotokoll TLS (Transport Layer Security) bzw. SSL (Secure Sockets Layer). Der Webserver muss daher die entsprechenden Softwarekomponenten beinhalten (z.B. TLS/SSL-Element), um den Datenaustausch mittels HTTPS zu ermöglichen. Wie in Abbildung 7 dargestellt, generiert der Inhaber der Webseite einen Public und einen Private Key, wobei der öffentliche Teil des Schlüssels, sowie persönliche Informationen des Betreibers in Form eines Anfragezertifikats zu einem Issuer (Unterzeichner, bzw. Zertifizierungsstelle) übertragen werden. Akzeptiert dieser die gestellte Anfrage, unterzeichnet er indem sein Root-Zertifikat (Stammzertifikat) in das angefragte Zertifikat integriert wird. Der Webbrowser behandelt ein Zertifikat des Servers nur als zuverlässig, wenn er das darin enthaltene Stammzertifikat kennt. Die Root-Zertifikate der wichtigsten Zertifizierungsstellen, auch als CA (Certificate Authority) bezeichnet, sind im Stammspeicher der meisten Browser hinterlegt. Ist das Root-Zertifikat der CA jedoch unbekannt, muss es im Webbrowser manuell als vertrauenswürdig eingestuft werden. [23]

[22] Vgl. Abts (2015), S. 169 f.

[23] Vgl. Heinemann (2014), Online-Quelle [15.09.2016].

Abbildung 7: Zertifikatsvergabe, Quelle: Heinemann (2014), Online-Quelle [15.09.2016].

Der in Abbildung 8 ersichtliche asymmetrische Sitzungsaufbau startet mit einem ungesicherten HTTP-Request an den Webserver. Die Antwort enthält das unterzeichnete Zertifikat des Servers, das vom Browser geprüft wird. Erkennt der Client eine bekannte Signatur (Root-Zeritifikat der CA) innerhalb des erhaltenen Zertifikats, stuft er es als vertrauenswürdig ein. Der temporäre Sitzungsschlüssel (Private-Key) wird mittels des Zertifikates vom Browser generiert und damit verschlüsselt an den Server übertragen. Der Sitzungsschlüssel wird vom Server mithilfe seines Zertifikats entschlüsselt. Webserver und Client besitzen somit den geheimen Sitzungsschlüssel mit dem alle weiteren Nachrichten ver- und entschlüsselt werden (symmetrische Verschlüsselung). [24]

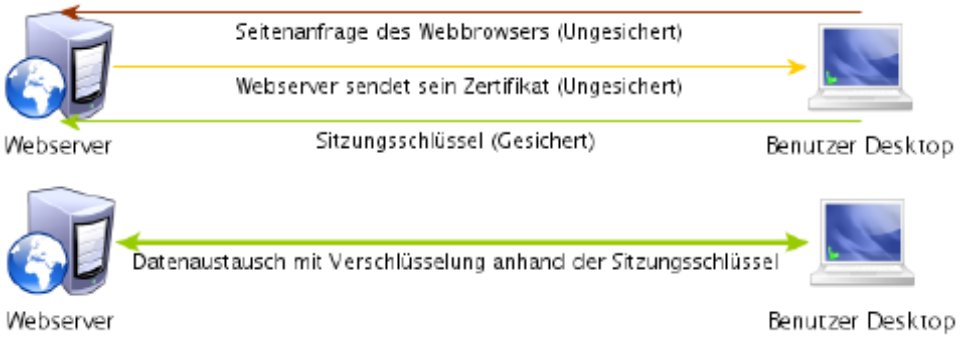


Abbildung 8: HTTPS Verbindungsaufbau, Quelle: Heinemann (2014), Online-Quelle [15.09.2016].

3.4.3 Verbindungsaufbau mittels Websocket

Bei der klassischen HTTP-Verbindung wird jeder HTTP-Request vom Server mit einer HTTP-Response beantwortet, demnach wird die Kommunikation immer vom Client eingeleitet. Wie bereits geschildert besteht jede Anfrage sowie deren Antwort aus Header-Informationen und Nutzdaten, wodurch ein erheblicher Overhead bei jedem Datenaustausch entsteht. Moderne Webanwendungen müssen aber auf eintretende Ereignisse am Server reagieren. Eine vom Server initiierte Übertragung, auch als Server-Push bezeichnet, ist somit unumgänglich. Hierfür könnten mehrere proprietäre Verfahren zur Verwendung kommen. Ein Beispiel dafür ist das Long-Polling-Verfahren. Bei Long-Polling wird ein vom

[24] Vgl. Heinemann (2014), Online-Quelle [15.09.2016].

Client initiierte HTTP-Request erst vom Server beantwortet, wenn eine gewisse Zeitspanne verstrichen ist oder neue Informationen vorliegen. Websocket bietet eine in HTML5-spezifizierte modernere Variante eine vom Server initiierte Verbindung zu implementieren, jedoch mit wesentlich geringeren Overhead als ältere Verfahren wie Long-Polling. Das WebSocket-Protokoll stellt eine Ergänzung zu HTTP dar und basiert ebenso auf TCP. Jedoch ist die Unterstützung von Websocket nicht bei jedem Webserver gegeben. Der Ablauf des Datenaustausches zwischen Client und Server wird in Abbildung 9 veranschaulicht. Die Verbindung wird, wie bei HTTP, mit einer Anfrage des Clients gestartet. Im Gegensatz dazu bleibt die zugrundeliegende TCP/IP-Verbindung auch nach der Übertragung der Daten bestehen. Dadurch können Nachrichten zwischen Server und Client jederzeit und ohne einen ständigen Verbindungsauf- und -abbau ausgetauscht werden, wobei manche HTTP-Header-Informationen entfallen.[25]

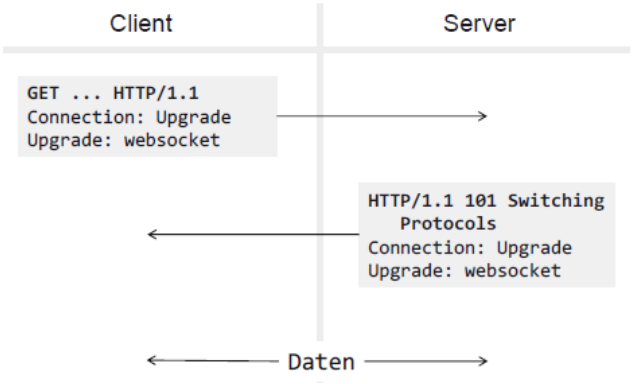


Abbildung 9: Kommunikation mit WebSocket: Quelle: Abts (2015): S. 190.

Das WebSocket-Protokoll besteht demnach aus zwei Phasen, dem Handshake und der Datenverbindung. Wie bereits erwähnt wird der Handshake vom Client eingeleitet, wobei ein Protokollwechsel (Upgrade) am Server beantragt wird. Der Aufbau dieser Nachrichten ist in Abbildung 10 ersichtlich. Die Überprüfung ob beide Seiten diese Kommunikationsweise zulassen erfolgt mittels Challenge-Response-Verfahren. Der Client generiert eine zufällige Zeichenkette (Sec-WebSocket-Key) und überträgt sie mit der Anfragenachricht an den Server. Dieser erzeugt damit den Wert von Sec-WebSocket-Accept, wobei er eine für beide Seiten bekannte Operation anwendet. Der Client führt diese Operation ebenfalls aus und vergleicht das Ergebnis mit dem vom Server empfangenden Wert (WebSocket-Accept). Stimmt das Ergebnis überein, ist eine Verbindung hergestellt. Nun werden die Daten in Frames transportiert, wobei jeder Frame einen maximal 14 Byte langen Header mit wenigen Informationen und die Nutzdaten enthält. Die URL ist jener von HTTP nachempfunden und hat folgenden Aufbau: ws://server[:port][/resource] [26] Ebenso kann eine verschlüsselte Übertragung über HTTPS angestoßen werden, indem die URL mit wss begonnen wird, beispielsweise: wss://server[:port][/resource].

[25] Vgl. Abts (2015), S. 189.

[26] Vgl. Abts (2015), S. 189 f.

```
GET /websocket/echo HTTP/1.1
Host: localhost
Sec-WebSocket-Version: 13
Origin: http://localhost
Sec-WebSocket-Key: CKpmXgNe1Hsp+XBQeDZo/w==
Connection: keep-alive, Upgrade
Upgrade: websocket

                    HTTP/1.1 101 Switching Protocols
                    Server: Apache-Coyote/1.1
                    Upgrade: websocket
                    Connection: upgrade
                    Sec-WebSocket-Accept: QVrIEBDw5cqV1FEBC3VuyLnNmU4=
```

Abbildung 10: WebSocket Handshake: Quelle: Abts (2015): S. 190.

3.4.4 Asynchroner Datenaustausch durch AJAX

Das AJAX-Modell (Asynchronous JavaScript and XML) bietet neue Wege in der Client-Server-Kommunikation von Webanwendungen. Das bereits geschilderte, klassische synchrone Kommunikationsmodell hat jedoch einen gravierenden Nachteil. Nach jeder Eingabe ist so lange zu warten, bis die vom Server angeforderte Seite im Browser neu geladen wurde. Jede Benutzeraktion ist somit mit Wartezeiten verbunden, währenddessen können keine anderen Aktionen ausgeführt werden, dadurch resultiert eine oft mangelhafte Interaktivität von Anwendungen. Jedoch berufen sich teils große Webshops auf dieses Modell, beispielsweise basiert Amazon.de auf der klassischen Client-Server-Schnittstelle. Kunden müssen mehrere Schritte hintereinander erfolgreich abschließen um ein Produkt zu kaufen. Für jeden wird eine neue Seite aufgebaut. Mittels AJAX wird ein asynchrones Nachladen von Nachrichten des Servers ohne einen kompletten Neuaufbau der Seite möglich. Dadurch kann Benutzern die sofortige Reaktion einer Desktopanwendung im Webbrowser geboten werden. Wie in Abbildung 11 dargestellt, basiert das AJAX-Modell auf der Ergänzung des klassischen Modells um eine AJAX-Engine als permanent agierender Vermittler zwischen Client und Server. Der Engine wird beim ersten Seitenaufbau automatisch mitgeladen und soll die Interaktion der Benutzer beobachten, bei Bedarf den Datenaustausch mit dem Server koordinieren (Anfragen schicken und interpretieren) und die Seitendarstellung dementsprechend anpassen. Für Besucher ist diese Vorgehensweise unsichtbar und ermöglicht beispielsweise die sofortige Abfrage von Suchanfragen während des Eintippens (z.B. Google Suggest). Das Skript kann auch ohne Benutzeraktion im Hintergrund ablaufen und Funktionen selbständig aufrufen. [27]

Diese Hintergrundprozesse (Webworker) werden von den meisten gängigen Browsern unterstützt, sind aber im Allgemeinen nur bedingt praxistauglich. Zudem ist die identische Ausführung der Anweisungen von komplexeren Strukturen des Workers derzeit nicht besonders sicher. [28]

AJAX ist jedoch keine eigene Technologie sondern ein Zusammenspiel von etablierten Web-Standards, Techniken und Programmiersprachen. Die AJAX Engine wird im JavaScript-Code aufgerufen. Sie erweitert das JavaScript um zusätzliche Bibliotheken, Methoden und Zugriffsoptionen. HTML und CSS

[27] Vgl. Friedmann (2009), S. 672 ff.

[28] Vgl. Steyer (2014), S. 40 ff.

bilden die standardkonforme Abbildung der Seite. Das Document Object Model (DOM) dient zur dynamischen Anzeige sowie zur Interaktion und Änderung der Seitenstruktur. [29]

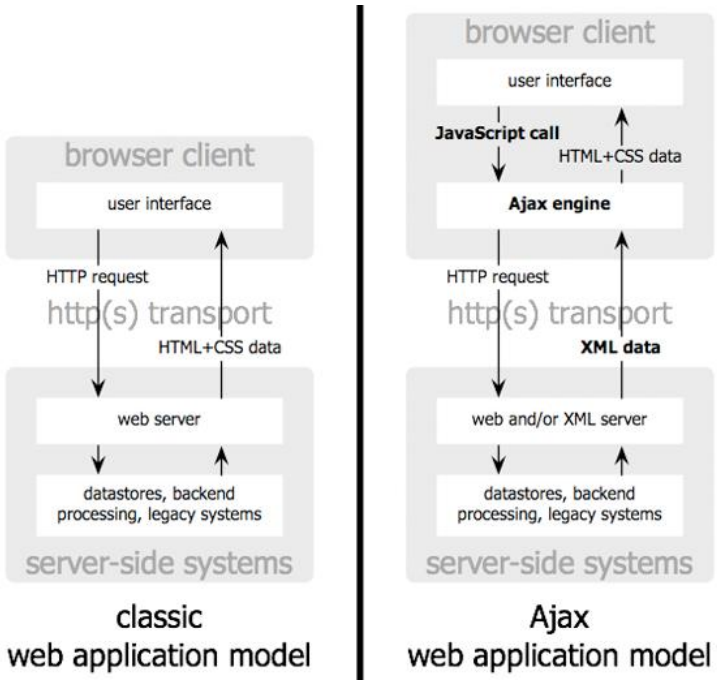


Abbildung 11: web application models, Quelle: Adaptivepath (2005), Online-Quelle [12.09.2016].

Die asynchrone Kommunikation mit dem Server findet mittels XMLHttpRequest-Objekten (XHR) statt. Diese JavaScript- Objekte ermöglichen es HTTP-Anfragen an den Server zu senden und dessen Antwort in Form einer XML-Nachricht zu verarbeiten. Serverseitig kann beispielsweise ein PHP-Skript aufgerufen werden. Über die Attribute und Methoden des XHR-Objekts können andere JavaScript Funktionen auf Serverdaten zugreifen und im Anschluss über DOM-Events die Darstellung der Seite manipulieren. [30] XHR bildet eine Alternative zu den in HTML5 spezifizierten Websockets.

3.5 Nutzung von Bezahldiensten

Bezahldienste sind für jeden Onlineshop essentiell um KundInnen eine komfortable Möglichkeit anzubieten die bestellten Waren zu bezahlen. Außerdem bietet sich die Möglichkeit an, nur Bestellungen anzunehmen die bereits bezahlt sind. Im Folgenden werden mehrere Varianten zur Bezahlung im Onlineshop vorgestellt.

3.5.1 Bezahlung mittels Kreditkarte

Endkunde, Händler, Issuer und Acquirer bilden das vier-Parteien-System im Kartenmarkt für online Bezahlvorgänge. Aufgrund der Beteiligung von Kartenorganisationen (z.B. Visa oder MasterCard) unter deren Lizenzen die Karten durch die Issuer herausgegeben werden, müsste richtigerweise von einem fünf-Parteien-System die Rede sein. Eine Debit- bzw. Kreditkarte wird typischerweise vom Issuer (z.B.

[29] Vgl. Friedmann (2009), S. 672 f.

[30] Vgl. Friedmann (2009), S. 672 f.

Bank oder Bankengruppe) herausgegeben. Mit dieser Karte ist der Endkunde ist er in der Lage, beim Händler elektronisch bargeldlos zu bezahlen. Erfolgt eine Bestellung so übermittelt der Händler dem Karten-Verarbeiter (Acquirer) die Transaktions-Daten. Die Sicherheitsaspekte in dieser Datenübertragung werden in erster Linie vom Karten-Verarbeiter vorgegeben (z.B. PayPal). Außerdem trägt der Acquirer dafür die Verantwortung, dass der Karten-Issuer den Kaufbetrag dem Konto des Endkunden zuschreibt und dem Händler der Kaufpreis gutgeschrieben wird. Der Händler erhält abschließend den Verkaufspreis abzüglich der vom Karten-Verarbeiter vereinnahmten Gebühren, welche unter anderem für den die Kartenorganisation und den Issuer bestimmt sind. [31]

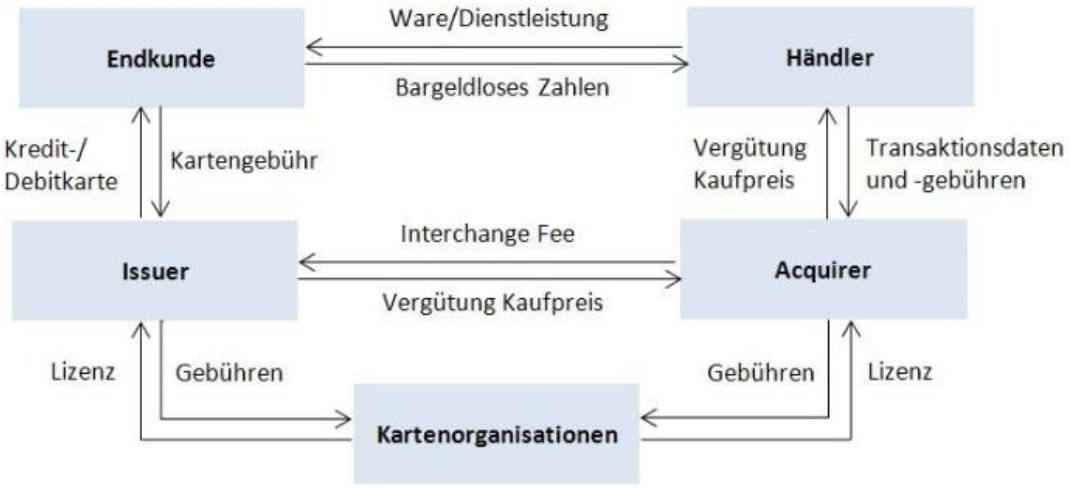


Abbildung 12: Das 4 Parteien-System im Kartenmarkt, Quelle: VEZ (2015), Online-Quelle [03.11.2016].

3.5.2 Bezahlservice SOFORT-Überweisung

Für SOFORT-Überweisung müssen sich Endkunden nicht registrieren und keine Kreditkarte besitzen. Nur der Empfänger der Zahlungen muss ein Konto anlegen, da der Service kostenpflichtig ist. Die Bezahlung erfolgt mit bekannten Online-Banking Daten. Der Dienst verfügt über eine Echtzeitbestätigung des Überweisungsauftrags. Hierfür wird mittels HTTP-POST Methode ein XML-File mit den relevanten Informationen (z.B. Preis und Anmeldeinformationen) verschlüsselt (HTTPS) an die Adresse der am Webserver des Onlineshops implementierten SOFORT-Überweisung-APIs (https://api.sofort.com/api/xml) verschickt. Der weitere Verlauf der Transaktion wird ebenfalls über diese Schnittstelle mitgeteilt, wobei dies für Kunden des Webshops nicht sichtbar ist. Der Bezahlvorgang ist in Abbildung 13 ersichtlich. Sobald eine KundIn bezahlen möchte, wird er/sie auf eine Webseite von SOFORT geleitet, wo die Überweisungsdaten (Empfänger, Betrag, Verwendungszweck) ersichtlich sind. Dort sind die Bankleitzahl und weitere Online-Banking-Daten zur Anmeldung erforderlich. Nach erfolgreicher Anmeldung ist die Überweisung mit einer gültigen TAN zu bestätigen. Nach einer erfolgreich beauftragten Überweisung gelangen Kunden zu einer Zusammenfassung der durchgeführten Transaktion und mittels Klick wieder zurück zum Webshop.[32]

[31] Vgl. VEZ (2015), Online-Quelle [03.11.2016].

[32] Vgl. SOFORT (2016), Online-Quelle [15.09.2016].

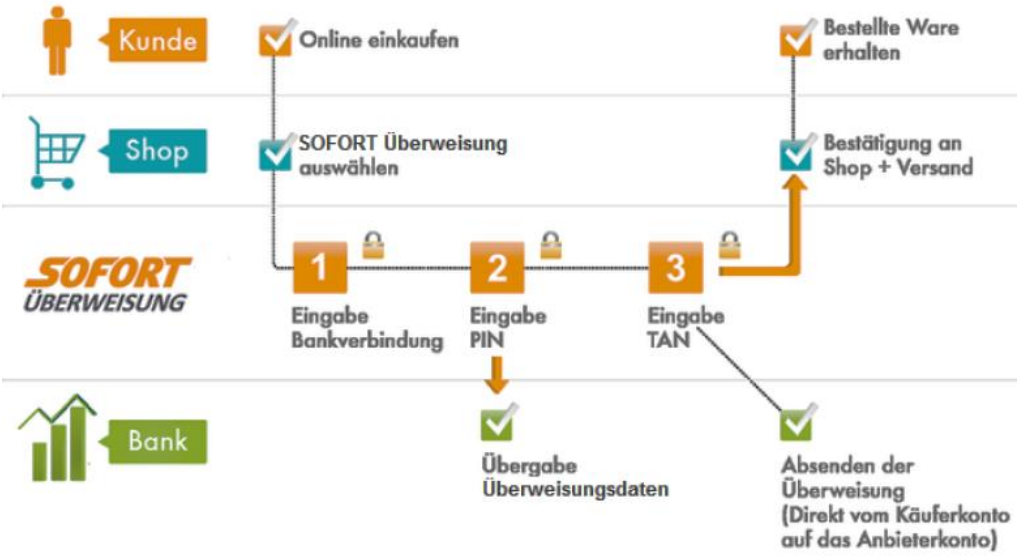


Abbildung 13: Bezahlprozess mit SOFORT, Quelle: SOFORT (2016), Online-Quelle [15.09.2016].

3.5.3 Bezahlservice PayPal

Um diesen kostenpflichtigen Dienst benutzten zu können, müssen der Kunde und der Betreiber des Shops einen PayPal-Account besitzen. Der Bezahlprozess ist in Abbildung 14 ersichtlich. Die Anmeldung erfolgt dabei nur mit dem Benutzernamen und einem Passwort. Kontoinformationen sind nur bei der Registrierung notwendig (z.B. Kreditkarte oder Bankeinzug). PayPal bietet eine Reihe möglicher Implementierungen für Entwickler an. Eine Möglichkeit ist die Generierung eines HTML-Buttons mit verschiedensten Funktionen der bei Betätigung Kunden zu PayPal weiterleitet. Mittels HTTP-POST-Methode werden hier die Daten eines Formulars an den Server von PayPal weitergeleitet. Der Kunde wickelt anschließend die Bezahlung ab und wird bei erfolgreicher Bezahlung an eine (vorher eingestellte) URL weitergeleitet. Ähnlich wie bei SOFORT kann ebenfalls mittels Webservices mit einer API interagiert werden, um ständig über den Status der Transaktion informiert zu sein. Hier wird ebenfalls eine, nach gewissen vom Anbieter festgelegten Formalkriterien gestaltete, XML-Datei mit POST versendet. Der HTTP-Header beinhaltet neben den Bestellinformationen (Preis etc.) die Anmeldeinformationen des Betreibers und eine eindeutige ID (Merchant-account-ID) oder ein eigens von PayPal ausgestelltes Zertifikat. Diese Sicherung garantiert somit die Authentizität des Shop-Betreibers und dessen PayPal-Kontos. Die Responses des Servers informieren über den Status der Transaktion. [33]

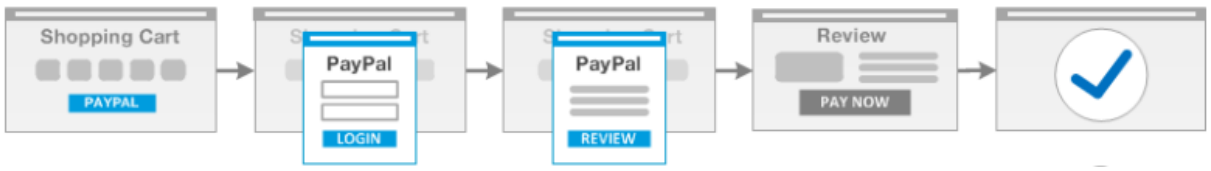


Abbildung 14: Bezahlprozess mit PayPal-Button, Quelle: PayPal (2016), Online-Quelle [15.09.2016].

[33] Vgl. PayPal (2016), Online-Quelle [15.09.2016].

4 KOMMUNIKATIONSMÖGLICHKEITEN MIT DER PRODUKTIONSANLAGE

In diesem Kapitel werden Technologien analysiert, welche eine bidirektionale Verbindung zwischen einer zentralen Datenbank und einer SPS-gesteuerten Produktionsanlage ermöglichen, um anschließend die Entwicklung von Konzepten für eine Verbindung zwischen den beiden Webanwendungen und der Steuerung zu unterstützen und folgend eine geeignete Systemarchitektur zu finden.

4.1 Datenbankanbindung mit herstellerspezifischen Kommunikationstools

Mehrere Hersteller bieten Kommunikationstools an, welche eine direkte Verbindung zwischen der Steuerung und verschiedenen Datenbanksystemen ermöglichen. Jedoch wird zur Erläuterung des Prinzips der TwinCAT Database-Server von Beckhoff näher behandelt.

4.1.1 TwinCAT Database Server

Das nur auf Microsoft Windows lauffähige System besteht aus drei Komponenten:

- TwinCAT Database Server: Dieser Software Service welcher zusammen mit dem Laufzeitsystem der Soft SPS (TwinCAT) gestartet und ebenfalls mit ihm gestoppt wird. Die lizenzpflichtige Anwendung verbindet das TwinCAT-System mit der Datenbank.
- Konfigurator: Der mitgelieferte Konfigurator dient zum Editieren der grundlegenden Datenbankparameter, um eine Kommunikation zu ermöglichen (z.B. Anmeldeinformationen).
- SPS-Bibliothek: Die beinhalteten Funktionsbausteine bieten die Möglichkeit die Datenbankverbindung aufzubauen oder neue Tabellen zu erstellen. Daten können durch Insert-Befehle in beliebige Tabellenstrukturen geschrieben werden. Mithilfe eines Funktionsbausteins wird per Select-Befehl von der Datenbank gelesen. Ebenfalls ist es möglich Datenbankeinträge zu aktualisieren oder zu löschen. [34]

Derzeit unterstützt der Database-Server elf gängige Datenbanken, darunter befinden sich MySQL, Microsoft SQL und Microsoft Access. Die Organisation der Daten im ASCII-Format (z.B. CSV) oder als XML Dateien ist ebenfalls möglich. [35]

4.1.2 Prinzipieller Aufbau

Der Datenbankserver kann auf einem Hutschienen-Industrie-PC neben der SPS-Runtime oder auf einem externen PC ausgeführt werden. Die Datenbank kann sich, wie der Server, ebenfalls auf einem externen PC oder auf der SPS (Hutschienen PC) befinden. Dabei ist das verwendete Betriebssystem (z.B. Windows CE) von besonderer Relevanz, um eine reibungslose Funktion einer lokalen Datenbank zu

[34] Vgl. Beckhoff (2016), Online-Quelle [03.07.2016].

[35] Vgl. Beckhoff (2016), Online-Quelle [03.07.2016].

gewährleisten. Eine Microsoft Access Datenbank ist beispielweise nicht auf Windows CE lauffähig. Eine Kommunikation zwischen den einzelnen Stationen über Netzwerkverbindungen ist möglich. Daraus ergeben sich verschiedenste Einsatzmöglichkeiten und Topologien. Wie in Abbildung 15 ersichtlich, kann ein zentraler Server (TcDatabaseSrv) die Verbindung zur Datenbank (SQL-Server) übernehmen, die restlichen Teilnehmer kommunizieren nur mit dem zentralen TwinCAT Database-Server. Natürlich können sich die Datenbank und der Database-Server am selben Gerät befinden.

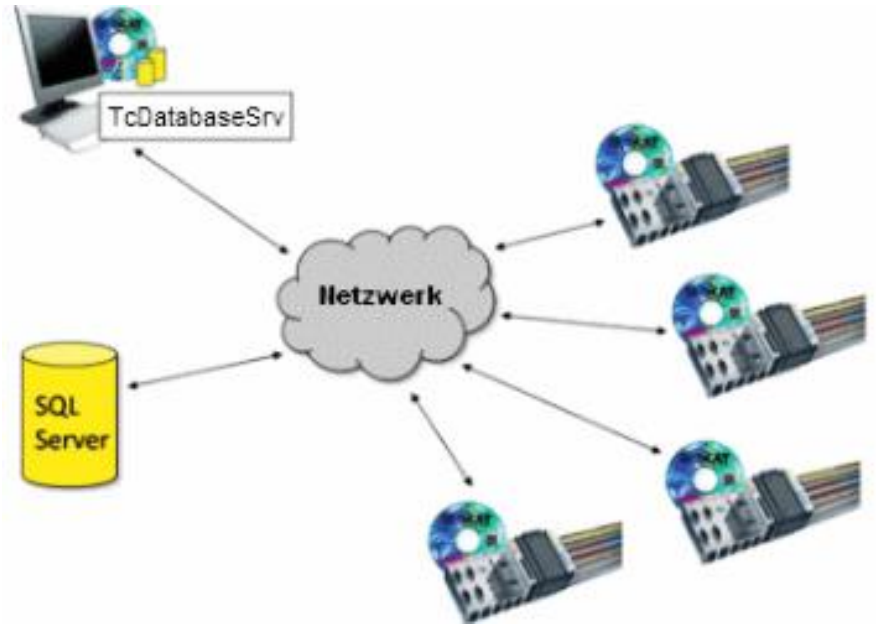


Abbildung 15: Netzwerktopologie mit zentralem Database-Server, Quelle: Beckhoff (2016), Online-Quelle [03.07.2016].

Jedoch bietet die lokale Installation des TwinCAT Database-Servers auf jedem Teilnehmer die Möglichkeit ein verteiltes System aufzubauen. Abbildung 16 zeigt die dezentrale Organisation des Systems. Jede Station kommuniziert über das Netzwerk unabhängig mit einer zentralen Datenbank.

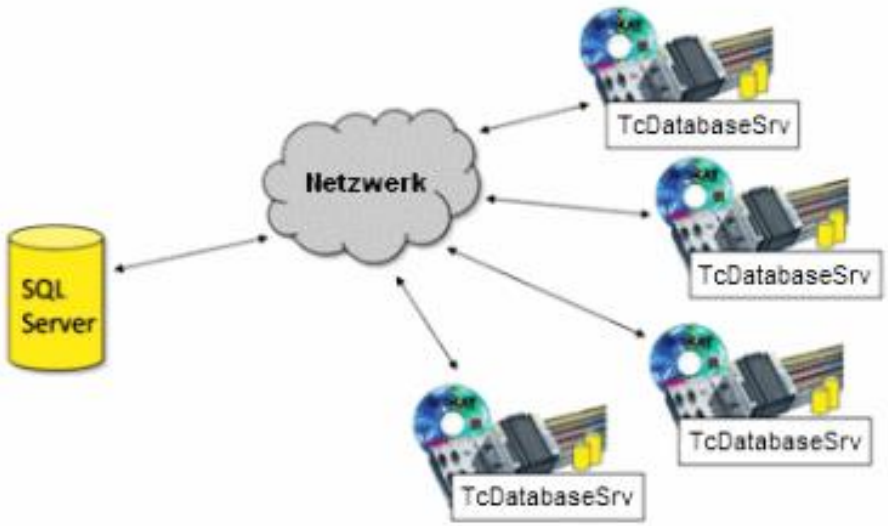


Abbildung 16: Netzwerktopologie mit dezentralen Database-Servern , Quelle: Beckhoff (2016), Online-Quelle [03.07.2016].

4.1.3 Funktionsweise des Tools

Die einzelnen Stationen verwenden zur Kommunikation untereinander oder mit dem Database-Server das von Beckhoff entwickelte Automation Device Specification Protokoll (ADS). Der Nachrichtenaustausch geschieht über die vereinheitlichte ADS-Schnittstelle mithilfe des Message-Routers. Dieser Service verwaltet Nachrichten im System und verteilt sie über Netzwerke hinweg. Je nach Aufgabe ist ein Softwaremodul (Server oder Client) standardmäßig auf den Komponenten vorinstalliert. Die Kommunikation kann, neben anderen Feldbussen, auch über ein Ethernet basierendes Netzwerk mittels TCP/IP abgewickelt werden. Mithilfe der ADS-Schnittstelle kann demnach auf sämtliche Beckhoff Komponenten zugegriffen werden. Beispielweise ermöglicht die Bibliothek TwinCAT.Ads eine Einbindung von SPS Daten in eine .NET Applikation. [36]

Der Datenaustausch zwischen dem TwinCAT Database-Server und der Datenbank erfolgt über den Datenbankspezifischen Treiber. Wie bereits erwähnt unterstützt das System 11 namhafte Datenbanktypen.

Abbildung 17 zeigt die Funktionsweise des TwinCAT Database-Servers. Im Grunde basiert die Kommunikation auf zwei Befehlen:

- DB_TO_ADS: Dieser Befehl prüft die Datenbank (z.B. MySQL) und schreibt Daten per ADS in ein ADS-Gerät (z.B. eine SPS).
- ADS_TO_DB: Prüft das TwinCAT-ADS-Gerät und schreibt Daten in eine Datenbank.

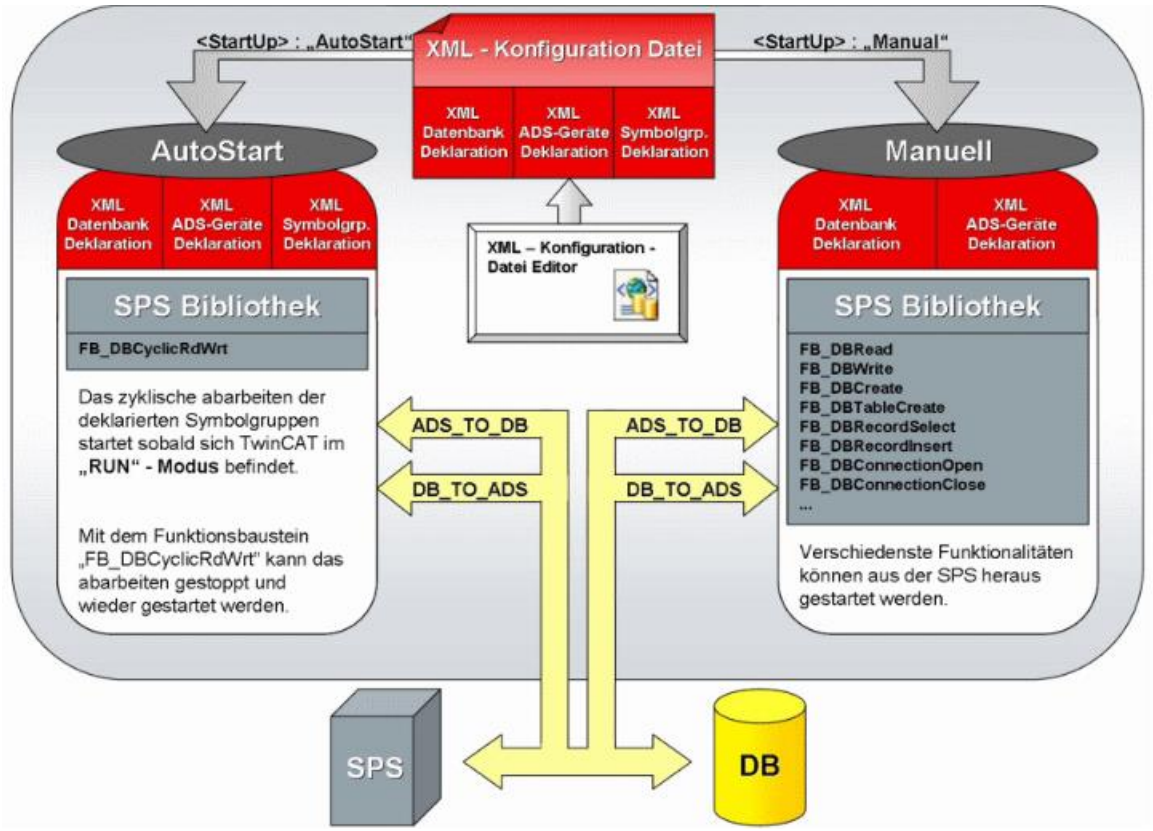


Abbildung 17: Funktionsweise TwinCAT Database Server, Quelle: Beckhoff (2016), Online-Quelle [03.07.2016].

[36] Vgl. Beckhoff (2016), Online-Quelle [03.07.2016].

Die Konfiguration wird mittels XML-Datei durchgeführt. Diese Datei beinhaltet die teilnehmenden ADS-Geräte, die zu verbindenden Datenbanken und die Variablenkonfiguration. Der Database-Server kann anschließend in zwei verschiedenen Modi AutoStart oder Manuell betrieben werden. Im Autostart-Modus wird der Service mit dem Laufzeitsystem gestartet und beginnt sofort mit dem Datenaustausch zwischen Datenbank und Steuerung, wobei einer der beiden Befehle DB_TO_ADS oder ADS_TO_DB automatisch ausgeführt wird. Im Gegensatz dazu wird im Betriebsmodus Manuell der Austausch der Prozessdaten über Funktionsbausteine (Beispiele siehe Abbildung 17) im Laufzeitprogramm angestoßen.

Der Datenbankserver unterstützt die Kopplung von Variablen aus dem SPS-Laufzeitsystem, wobei zwei verschiedene Speicherformate in der Datenbank zu unterscheiden sind.

- Double: In diesem Format können Variablen vom Typ BOOL, LREAL, REAL, INT, DINT, USINT, BYTE, UDINT, DWORD, UINT, WORD und SINT verarbeitet werden.
- Byte: Das Format ist generell für alle Variablendatentypen kompatibel, jedoch ist es insbesondere für Strings und Datenstrukturen geeignet. [37]

4.2 OPC Data Access

Nach dem herstellerspezifischen Kommunikationstool wird nun der herstellerunabhängige OPC-Standard (OLE for Process Control) als mögliche Kommunikationsschnittstelle untersucht. Er ermöglicht eine standardisierte und in der Industrie weit verbreitete Kommunikation zwischen Steuerungen und Anwendungen unterschiedlichster Hersteller. Eine Standardisierung ermöglicht die mittels OPC geschaffene, dem Einzelsystemen übergeordnete, abstrakte Ebene. OPC soll, wie heutzutage verlangt, eine einfache vertikale Integration ermöglichen. Die vom OPC-Server über die Feldbusebene angeforderten Prozessdaten werden den verbundenen Clients im einheitlichen OPC-Format zur Verfügung gestellt und könnten anschließend von einer Anwendung in eine Datenbank gespeichert werden. Die ehemals als OLE (Object Linking and Embedding) bezeichnete Microsoft DCOM Schnittstelle (Distributed Component Object Model) bildet die Kommunikationsgrundlage des OPC-DA (OPC Data Access) Standards.

4.2.1 Optimierung durch OPC

Die Optimierungsmöglichkeiten durch die Verwendung eines einheitlichen Standards sind anhand eines Beispiels, dargestellt in Abbildung 18, einfach zu zeigen. Eine Motorregelung, eine Getrieberegelung und eine Pumpensteuerung sind mit Anwendungen auf der Prozessleitebene zu koppeln (z.B. eine Office-Anwendung, ein Produktionsplanungssystem und eine Anzeige- und Bedienkomponente). Ohne den Einsatz von OPC als standardisierte Schnittstelle sind, wie im linken Teil von Abbildung 18 ersichtlich, in diesem Beispiel 18 Kommunikationstreiber erforderlich. Mithilfe eines OPC-Servers pro Anlage kann die Anzahl an spezifischen Treibern deutlich verringert werden. Applikationen in der Prozessebene müssten nur über eine OPC-Client Schnittstelle verfügen. Eine durch OPC optimierte Topologie, siehe an der

[37] Vgl. Beckhoff (2016), Online-Quelle [03.07.2016].

rechten Seite von Abbildung 18, zeigt die Vereinfachung eines solchen Systems. Außerdem ist eine Erweiterung des Systems um neue Anlageteile wesentlich einfacher durchzuführen.[38]

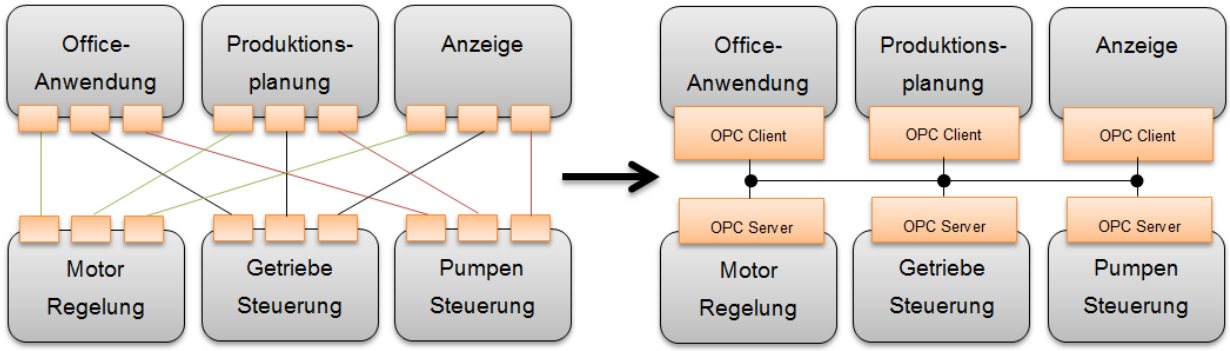


Abbildung 18: Beispiel für OPC, Quelle: Eigene Darstellung.

4.2.2 Architektur des Systems

Eine Implementierung von OPC-DA-Softwarekomponenten ist aufgrund der Bindung zur DCOM-Schnittstelle nur auf Hardware mit einer Microsoft Windows Umgebung möglich. Im OSI 7-Schichtenmodell wird OPC in die letzte Ebene, der Anwendungsschicht, eingeordnet. Das in der Darstellungsschicht angesiedelte DCOM basiert auf der RPC-Technologie (Remote Procedure Call), die den Datenaustausch über ein Netzwerk zwischen Client und Server-Anwendungen ermöglicht.[39]

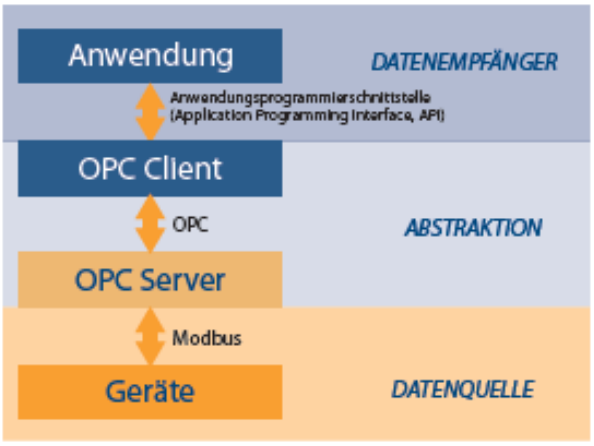


Abbildung 19: OPC Client/Server Architektur, Quelle: MatrikonOPC (2009), Online-Quelle [03.09.2016].

Wie in Abbildung 19 dargestellt, kommuniziert ein OPC-Server einerseits mit den Datenquellen über gerätespezifische native Protokolle (z.B. Modbus) und anderseits auf der Prozessebene über die OPC-Schnittstelle mit dem verbunden OPC-Client (Datenempfänger). Die Abstraktion geschieht indem die Komponenten über ihre nativen Protokolle und Schnittstellen ausschließlich mit einem der beiden OPC Komponenten korrespondieren. Diese Vorgehensweise ermöglicht Daten vom Gerät zur Anwendung zu übertragen, ohne den aufwendigen direkten Kontakt. Hingegen bestimmt die Qualität und Effizienz der Übersetzung vom OPC-Format in das Nativformat und umgekehrt, die OPC-Server bzw. Client-

[38] Vgl. Mustermann (2006), Online-Quelle [03.09.2016].

[39] Vgl. Microsoft Technet (2014), Online-Quelle [07.04.2014].

Implementierung des jeweiligen Herstellers. Der OPC-Server agiert in der Client–Server-Architektur als Slave und der OPC-Client als Master, der den bidirektionalen Datenaustausch anstößt.[40]

4.2.3 OPC Data Access Spezifizierung

Die Data Access Spezifikation definiert den Kommunikationsablauf zwischen einer Datenquelle (Server) und einer Applikation zur Datenverarbeitung (Client), sowie das Datenformat und das Datenmodell. Der am weitesten verbreitete Standard ist OPC-DA 2.05, welcher bereits im Jahre 1998 definiert wurde. OPC-Classic enthält neben Data Access auch Standards für Alarmmeldungen (Alarms and Conditions) und historische Daten (Historical Access). Neuere Spezifikationen, wie Beispielsweise OPC-DA 3.0 aus dem Jahr 2003, sind hingegen selten in Systemen implementiert.[41]

4.2.4 Organisation der Daten am OPC-DA-Server

Die Organisation der Daten erfolgt, ersichtlich in Abbildung 20, mithilfe einer Objekt- und einer Namespacehierarchie. Durch die Objekthierarchie kann jeder Client verschiedene Typen von Objekten selbst anlegen, welche seine Sicht auf den Prozess ausdrücken. Als Top Level Objekt dient das OPCServer-Objekt, das in der Hierarchie darunterliegende Objekte verwaltet. Außerdem beinhaltet es neben mehreren Statusinformationen des Servers Methoden zum Verbinden oder Trennen der Verbindung zum Client. OPCItem-Objekte beinhalten die eigentlichen Prozessdaten der Feldebene und weitere Statusinformationen die sogenannten Properties. OPCItem-Objekte beinhalten laut Spezifikation 2.0 keine Interfaces und Methoden. Durch die Verwendung von OPCGroup-Objekten können OPCItems flexibel organisiert werden. [42]

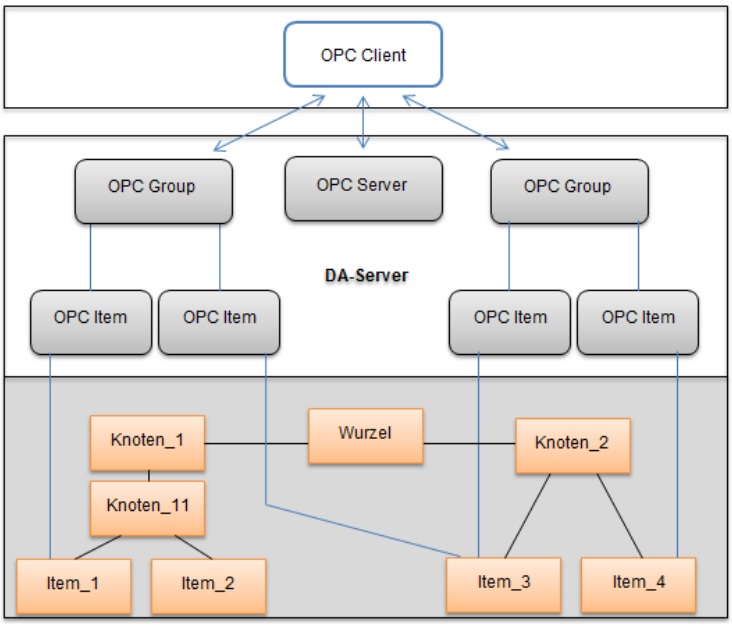


Abbildung 20: Data Access Server mit Objekt und Namenshierarchie, Quelle: Eigene Darstellung.

[40] Vgl. MatrikonOPC (2009), Online-Quelle [03.09.2016].

[41] Vgl. Lange (2014), S. 39 ff.

[42] Vgl. Lange (2014), S. 39 ff.

Zur Steigerung der Effizienz werden beim Aufruf eines OPCGroup Objekts alle verknüpften OPCItems angesprochen. Jedoch sind einzelne Schreib- oder Lesezugriffe auf ein OPCItem im Modell nicht ausgeschlossen. Das OPCItem IO-Interface erweitert ab Spezifikation OPC DA 3.0 das Objektmodell. Es ermöglicht direkten Zugriff auf OPCItem-Objekte ohne eine Objekthierarchie anzulegen. Die Namespacehierarchie strukturiert die Items am OPC-Server mithilfe von Knoten. Die sogenannte Wurzel stellt den OPC-Server dar.[43] Die Erstellung des Namespaces am Server kann automatisch über ein Konfigurationsprogramm erfolgen oder über einen Import aus einer anderen Anwendung. Die Struktur und die Variablennamen der OPC-Items können somit vom SPS-Programm übernommen werden.

4.2.5 Verbindungsaufbau zwischen Client und Server

Das OPCServer-Objekt bietet Methoden den Namespace des Servers zu browsen, um anhand der erhaltenen Bezeichner (Variablennamen) die OPC-Objekte anzulegen. Die fully qualified ItemID eines OPCItems entspricht demnach immer dem Namen eines im Namespace vorhandenen Items. Außerdem wird ein ClientHandle, der Datentyp (RequestedDatatype) sowie der Zustand (AktiveState) übergeben. Beim Erzeugen von OPCServer und OPCGroup Objekten ist die Namensvergabe durch den Client nicht zwingend an den Namespace anzupassen. Der Server legt beim Erzeugen von OPCGroup und Server-Objekten automatisch den symbolischen Objektnamen an. [44]

4.2.6 Kommunikationsmodell

Die Werte (Values) eines OPCItems können vom Data Access Client synchron oder asynchron, gelesen und geschrieben werden. Außerdem besteht die Möglichkeit, Items nur bei Wertänderung zu aktualisieren. Der Client ruft bei synchronen Zugriffen die gewünschte Methode auf und wartet anschließend auf den Rückgabewert des Servers. Dies ist zum schnellen Erfassen von Daten hervorragend geeignet, verlangt aber sofortigen Zugriff, da sonst die Performance durch das Warten auf die Rückmeldung beeinflusst wird. Im Gegensatz dazu erfolgt der asynchrone Zugriff des Data Access Clients über eine logische Warteschlange am Server, welche die Anfragen gereiht abarbeitet. Der Client bekommt sofort eine Übermittlungsbestätigung der Nachricht. Der angefragte Prozesswert (Value) folgt nach Abarbeitung aller vorherigen Aufgaben in der Warteschlange. Diese Form des Datenaustausches bietet wesentliche Vorteile, bei nicht abschätzbaren oder langen Verbindungszeiten zum Server. Synchrone sowie asynchrone lesende Zugriffe auf die Werte der OPCItems können direkt auf das Gerät (z.B. Steuerung) oder über einen Zwischenspeicher (Cache) am Server erfolgen. Die jeweilige Aktualisierungsrate der Items im Cache wird im OPCGroup-Objekt festgelegt. Schreibende Zugriffe sind ebenfalls synchron oder asynchron ausführbar, erfolgen hingegen immer direkt auf das Device. Wie bereits im Datenmodell geschildert, kann ein OPCGroup-Objekt mit mehreren OPCItems verknüpft werden. Die Adressierung erfolgt über Handles. Beim gleichzeitigen Lese- oder Schreibzugriff von Objekten muss demnach eine hohe Anzahl an Handles übertragen werden. Dies verringert wiederum die Performance beim Auswerten der Item-Objekte. Als Abhilfe enthält das OPCGroup-Objekt eine Refresh-

[43] Vgl. Lange (2014), S. 39 ff.

[44] Vgl. Lange (2014), S. 39 ff.

Methode, womit alle verknüpften Items auf einmal angesprochen werden. Neben synchronen und asynchronen Zugriffen, bietet die Variante Items nur bei Wertänderung zu aktualisieren, erhebliches Potential den Datenverkehr zu minimieren. Speziell bei analogen Messwerten, beispielsweise einer Temperatur ist ein zyklisches Abfragen des Prozesswertes nicht unbedingt zielführend, da der Prozesswert lange Zeit konstant bleibt. Bei dieser Vorgehensweise liest der Server die Werte der OPCItem-Objekte zyklisch, nach der definierten UpdateRate und wertet sie mit den vom OPC-Client übertragenen Parametern PercentDeadband und ActiveState aus. Der Wert wird demnach nur zum Client gesendet, wenn sich der Status (ActiveState) des Items ändert oder der neu eingelesene Wert außerhalb des errechneten Deadband befindet. Anhand der Totbandfunktion kann eine Hysterese festgelegt werden, um somit zur zusätzlichen Optimierung des Datenverkehrs beizutragen.[45]

4.2.7 Datenformat von OPC-DA

Das in Abbildung 21 dargestellte OPC-DA Datenformat eines Items besteht aus dem Wert (Prozesswert), dem Zeitstempel und einer Statusinformation. Der Parameter Wert ist aufgrund der Kompatibilität zu allen Datentypen als DCOM-VARIANT abgebildet. Eine Konvertierung zur weiteren Verarbeitung kann server- oder clientseitig geschehen. Der Zeitstempel, dargestellt im Windows Datentyp UTC (Universal Time Coordinated), wird bei jedem Zugriff auf das Item am Server generiert. Die letzten 16 Bits beinhalten die Qualität, den Status und den Limit Parameter. Zwei Bits codieren die Qualität der Daten. Hier kann zwischen den Zuständen „good“ (Wert gültig), „bad“ (Wert ungültig) und „uncertain“ (Wert unbestimmt) unterschieden werden. Besteht eine Verbindung zum Device wird der zum Client übertragene Wert als „good“ gekennzeichnet. Die vier Statusbits bieten nähere Informationen zur Qualität, beispielsweise „not connected“, kombiniert mit „bad“ als Qualität. Mit Limit kann ein Drahtbruch eines Sensors erfasst werden. Die weiteren 8 Bit stehen zur freien Verfügung. [46]

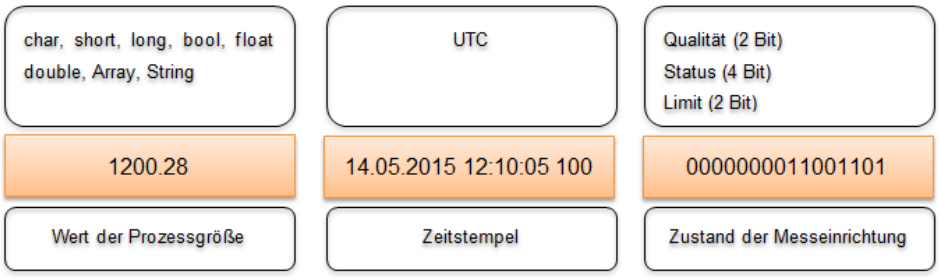


Abbildung 21: OPC-DA Datenformat, Quelle: Eigene Darstellung.

4.3 OPC Unified Architecture

Sämtliche Systeme im industriellen Umfeld verwenden OPC zum vertikalen und horizontalen Datenaustausch. Stetig wachsende Anforderungen in Hinsicht auf sichere, sowie plattform und hersteller-unabhängige Kommunikation, führten zur Weiterentwicklung des OPC-Standards. Mit OPC-UA wurde im Jahr 2007 ein Standard spezifiziert, welcher eine skalierbare und plattformunabhängige Lösung anbietet.

[45] Vgl. Lange (2014), S. 39 ff.

[46] Vgl. Lange (2014), S. 39 ff.

Die serviceorientierte Architektur (SOA) kombiniert die Vorteile von Webservices und integrierter Security mit einem einheitlichen Datenmodell. [47]

4.3.1 Vorteile gegenüber OPC Classic

Wie bereits im Kapitel OPC Data Access erörtert, basieren die Vorgänger des OPC-UA-Standards auf dem von Microsoft entwickelten DCOM-Model. Außer dem OPC XML Data Access Standard sind alle durch die OPC Foundation spezifizierten Architekturen an eine Windows-Umgebung gebunden. Zusätzlich kann DCOM nicht über Firewalls hinweg kommunizieren. Neben dem sehr aufwendigen Schutzmechanismus, ist der Aufbau von örtlich verteilen Systemen über das Internet nur mittels VPN-Verbindungen (Virtual Private Network) realisierbar. Außerdem wird das seit der Einführung der .NET Umgebung abgekündigte DCOM nicht mehr weiterentwickelt. Die Verwendung von Webservices und XML (Extensible Markup Language), die auch als Schnittstellen der .NET Umgebung fungieren, ermöglicht die geforderte örtlich- und plattformunabhängige Kommunikationsgrundlage. [48]

4.3.2 OPC Unified Architecture Spezifizierung

Die Spezifikation aus dem Jahr 2006 stellt eine wichtige Informationsquelle zu OPC-UA dar und ist öffentlich zugänglich. Der auch als IEC Normenreihe (IEC 62541) verfügbare Standard umfasst derzeit 13 OPC-UA-Spezifikationen. Ersichtlich in Abbildung 22, sind sie in drei Gruppen unterteilt:

- Basis-Spezifikationen (Core Specification Parts): Diese enthalten Basiskonzepte (Concepts) der Technologie und des Sicherheitsmodells (Security Model). Außerdem findet hier eine abstrakte Beschreibung des OPC-UA Metamodells (Adress Space Model) und der OPC-UA Dienste (Services) statt. Das konkrete UA-Informationsmodell mit dessen Modellierungsregeln (Information Modell), die Abbildung auf Protokollebene (Service Mapping) und das Konzept der Profile zu Skalierung der Funktionalität (Profiles) sind hier ebenfalls beschrieben.
- Zugriffsmodelle (Access Type Specification Parts): Sie stellen Erweiterungen des Informationsmodells dar und sind OPC-Classic nachempfunden. Sie bieten typische Zugriffe auf Daten (Data Access), Alarme und Meldungen (A&C), historische Daten (Historical Access) sowie auf Programme.
- Dienstfunktionen (Utility Type Specification Parts): Discovery bietet Konzepte zum Finden von OPC-UA Komponenten und Zugangspunkten im Netzwerk. Darüber hinaus werden in Aggregates die Aggregatfunktionen sowie Berechnungen zur Verarbeitung von historischen Informationen beschrieben. [49]

[47] Vgl. OPC Foundation (2015), Online-Quelle [07.09.2016].

[48] Vgl. Lange (2014), S. 95 ff.

[49] Vgl. OPC Foundation (2015), Online-Quelle [07.09.2016].

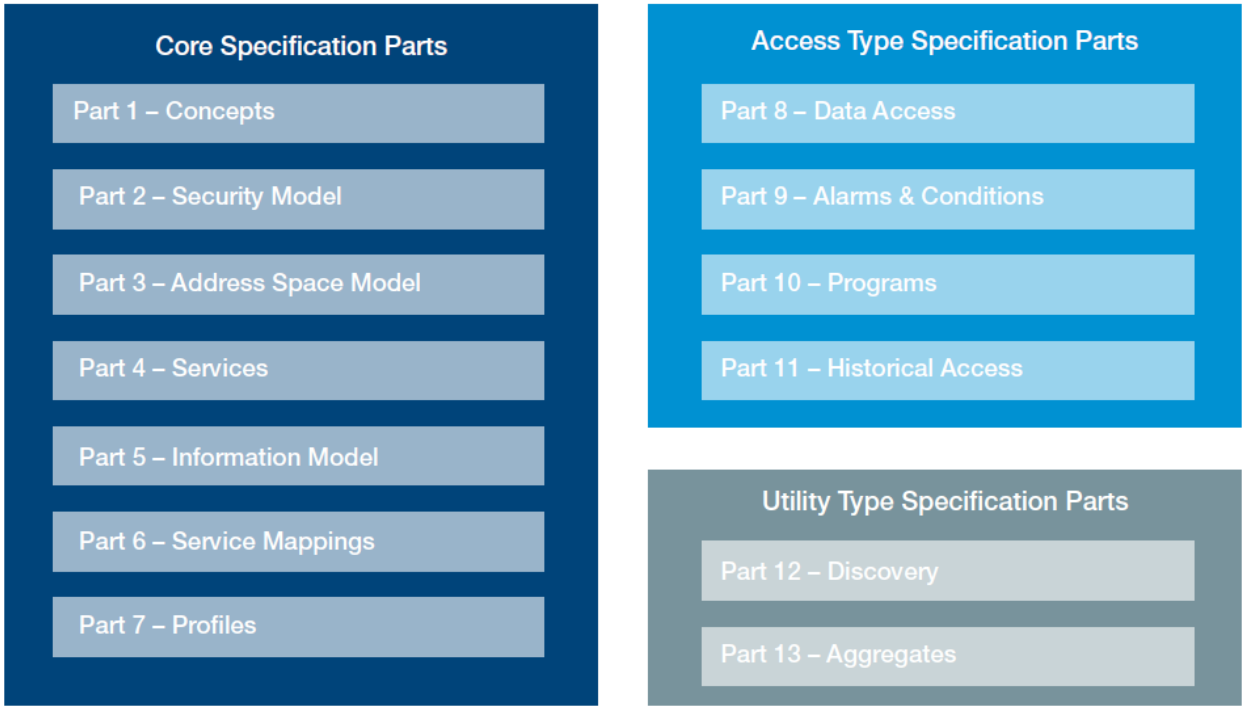


Abbildung 22: IEC62541 - OPC-UA Spezifikationen, Quelle: OPC-Foundation (2015), Online-Quelle [07.09.2016].

4.3.3 Architektur des Systems

OPC-UA beruht auf einer Server-Client-Architektur, wobei eine Applikation gleichzeitig Client- und Server-Komponenten enthalten kann. Die Plattformunabhängigkeit des Standards ermöglicht die Integration des Servers direkt auf einer Steuerung oder einem Sensor (Embedded OPC UA Server). Somit kann das Gesamtsystem dezentralisiert werden, das heißt, viele kleinere Server anstatt einem zentralen PC. Wie bereits geschildert kann der Datenaustausch über Firewalls hinweg und örtlich getrennt über das Internet erfolgen. [50]

Die große Flexibilität von OPC-UA ist auf die modulare Verwendung der in Abbildung 23 ersichtlichen Grundbausteine, zurückzuführen. Die Komplexität von Clients und Servern wird durch deren Implementierung in der Applikation selbst bestimmt. Durch eine bereitgestellte OPC-UA-API kann auf den darunterliegenden Kommunikations-Stack (z.B. Java oder .NET) zugegriffen werden. Dieser Besteht aus folgenden Komponenten, wobei nicht jede implementiert sein muss:

- Transport: Jeder OPC-UA-Server oder Client kann Mechanismen zum Datenaustausch wählen. Je nach Anforderung sind mehrere Transport Protokolle verfügbar.
- Metamodell (Daten bzw. Adressmodell): Es spezifiziert Regeln und Grundbausteine als Grundlage des Informationsmodells, beinhaltet sind verschiedene Einstiegsknoten und Basistypen. Das Metamodell beschreibt ausschließlich in welcher Weise Clients auf Informationen am Server zugreifen. Wie diese Informationen am Server organisiert sind ist jedoch nicht spezifiziert.

[50] Vgl. Portmann (2007), Online-Quelle [07.09.2016].

- Base Services: Mithilfe der Base Services wird die Schnittstelle zwischen Server und Client realisiert. Der anwendungsseitige Datenaustausch findet mittels Service Request oder Response statt.
- Informationsmodelle: Sie sind schichtenweise aufgebaut. Jeder höherwertige Typ basiert auf im Metamodell beschriebene Basisregeln. Neben den in Abbildung 23 dargestellten typischen Zugriffmodellen (z.B. Data Access) bietet OPC-UA die Möglichkeit komplexere Informationsmodelle (z.B. Collaboration Modelle) zu verwenden. Da alle Modelle auf gemeinsamen Regeln basieren, muss ein Client nur die Basisregeln kennen um beliebig komplexe Informationsmodelle bearbeiten zu können. Zusammenhänge sind in diesem Fall nicht ersichtlich, jedoch sind Datenvariablen editierbar und der Adressraum ist ersichtlich. [51]

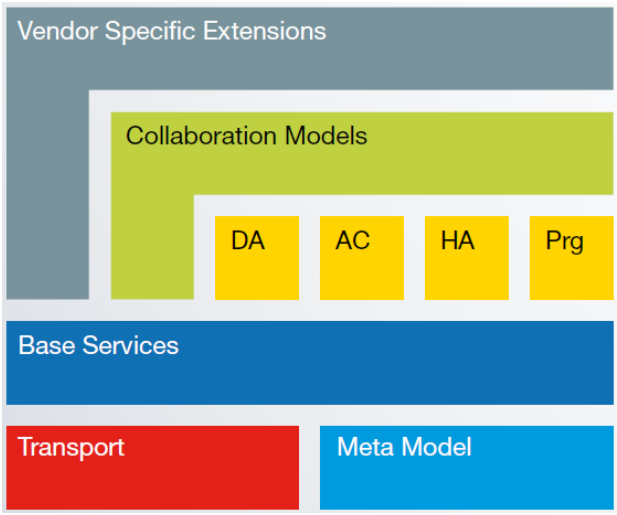


Abbildung 23: Schichtenmodell von OPC-UA, Quelle: OPC Foundation (2015), Online-Quelle [07.09.2016].

4.3.4 Transportschicht

OPC-UA ist eine rein serviceorientierte Architektur (SOA) und ist, wie bereit geschildert, in mehrere logische Ebenen unterteilt. Die protokollunabhängigen Dienste (Base Services) sind abstrakte Methodenbeschreibungen, sie stellen die Grundlage der OPC-UA-Funktionalität bereit. Die darunter liegende Transport-Schicht serialisert bzw. deserialisiert die Methoden und setzt sie somit in das gewünschte Protokoll um. Anhand des stattfindenden Protokoll-Bindings ist die Applikation strikt von der Kommunikation getrennt. Protokolle können wechseln oder nebeneinander verwendet werden, ohne die Applikation zu beeinflussen. Die Adressierung (URL) des Servers bestimmt das verwendete Protokoll, beispielsweise wird als URL nun opc.tcp://MyServer:4840 für das Binärprotokoll anstatt https://MyServer für ein Webservice verwendet. [52]

Derzeit sind ein Binary (UA-Binary) und ein Webservice (UA-XML) Protokoll sowie eine Kombination aus beiden, das Hybrid-Profil spezifiziert. Wie in Abbildung 24 ersichtlich, beruhen alle Protokolle auf TCP/IP, angesiedelt in der Transportschicht des OSI-Schichtenmodells. Die Kommunikation kann mittels UA-

[51] Vgl. OPC Foundation (2015), Online-Quelle [07.09.2016].

[52] Vgl. Ascolap (2010), Online-Quelle [07.09.2016].

Binary direkt über TCP (UA-TCP) oder mittels Webservices erfolgen. UA-Binary ist ein performantes, hoch optimiertes und binärkodiertes TCP-Protokoll. Das Binärprotokoll muss im Gegensatz zu UA-XML bei jedem UA-Stack implementiert sein. Zur Verbindung wird ausschließlich der TCP-Port 4840 verwendet. Der geringe Overhead und die hohe Geschwindigkeit machen diese Kommunikationsvariante vor allem für Embedded-Geräte interessant. UA-Binary ist derzeit das am häufigsten verwendete Protokoll. Die auf Text (XML) basierende und firewall-freundliche Variante mittels UA XML wird wesentlich seltener eingesetzt. Dies ermöglicht jedoch, ohne zusätzliche Hilfsmittel wie VPN, eine Vermittlung über das Internet. Die XML-Nachricht wird, gezeigt in Abbildung 24, mittels SOAP (Simple Object Access Protocol) und HTTP (Hypertext Transfer Protocol) übertragen. Auch eine zusätzlich gesicherte Vermittlung mit HTTPS ist möglich. Die Kommunikation über Webservices ist aufgrund der großen Overheads wesentlich langsamer. Im Gegensatz dazu ist das Hybrid-Profil wesentlich performanter. Die unverschlüsselte binär kodierte Nachricht wird hier über einen verschlüsselten Transportkanal (HTTPS) versendet. Dies vereint die firewall-freundlichkeit von HTTPS mit dem geringeren Overhead des Binary-Protokolls. [53]

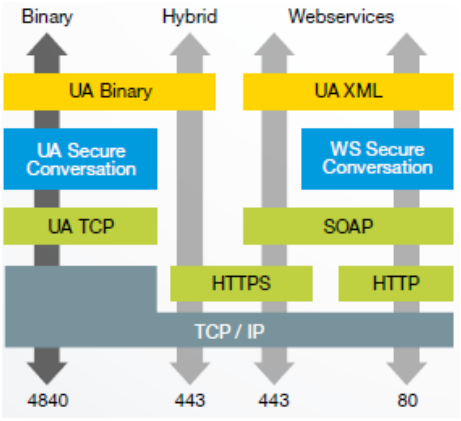


Abbildung 24: OPC-UA Transport Profile, Quelle: Frauenhofer Institut (2013), Online-Quelle [07.09.2016].

4.3.5 Metamodell von OPC-UA

Das aus OPC Classic bekannte hierarchische Datenmodell wird im OPC-UA-Metamodell um ein komplexes Objektmodell inklusive Typsystem und einem Regelwerk zum Verknüpfen von Objekten (References) erweitert. Diese Hilfsmittel ermöglichen eine Beschreibung bzw. Abbildung sämtlicher System-, Geräte- oder Funktionsinformationen am UA-Server. Nodes (Konten) und References bilden die Grundlage des Basismodells. Je nach Anforderung kann aus einer der acht folgenden, in Abbindung 16 dargestellten, Node-Klassen gewählt werden. Alle Knotenklassen sind von der BaseNode-Klasse abgeleitet und besitzen deren Attribute, beispielhaft die einen Knoten eindeutig identifizierende NodeID. Jede Node-Klasse wird dann durch weitere Attribute erweitert. Wiederum kann hier zwischen Properties (Eigenschaften) und DataVariables unterschieden werden. Die Klasse Variable dient zum Repräsentieren von Prozessdaten, wie Mess- oder Sollwerten. Der eigentliche Prozesswert ist in der DataVariable Value

[53] Vgl. Ascolap (2010), Online-Quelle [07.09.2016].

enthalten. Properties können weitere Eigenschaften von DataVariables darstellen, beispielsweise den Bereich einer Messung. [54]

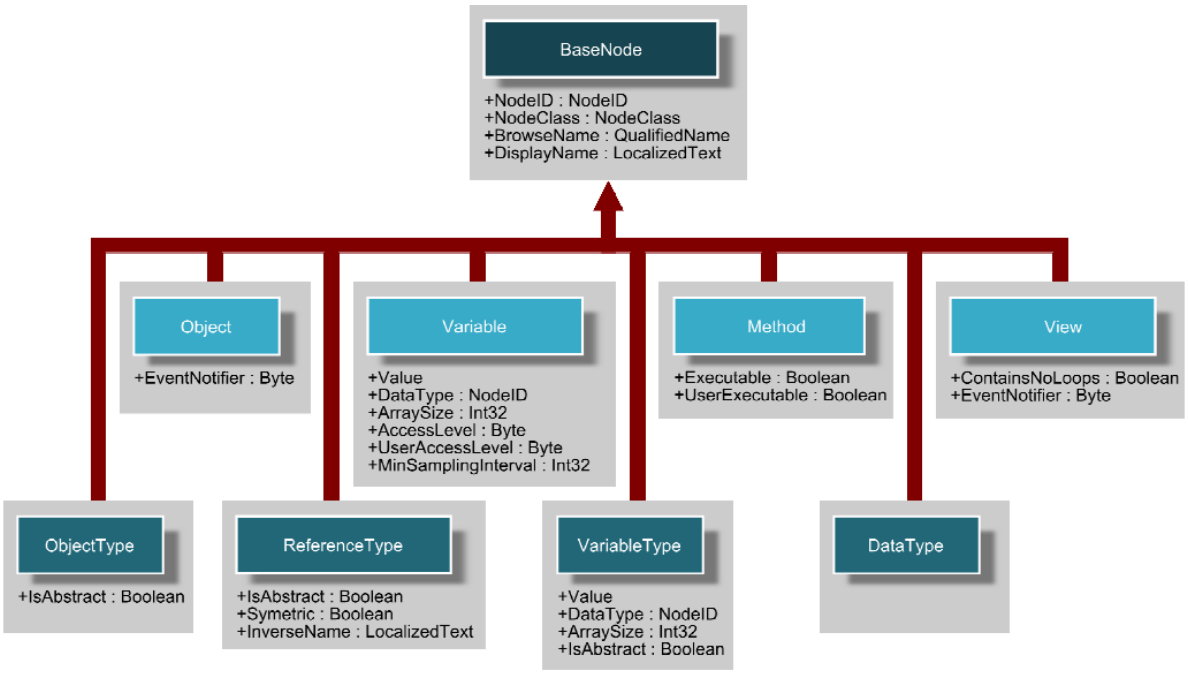


Abbildung 25: OPC UA Datenmodell, Quelle: Ascolap (2010), Online-Quelle [07.09.2015].

Zum Abbilden realer Objekte, wie Softwareobjekten oder Antrieben, kann ein Objekt der Klasse Object zur Verwendung kommen. Objects beinhalten auch eine Event Notification, welche einen Client über eine Änderung einer Variablen automatisch informieren kann. Services ermöglichen es dem Client die vom Server angebotenen Funktionen von Method-Objekten abzurufen. Deren Properties beschreiben die erforderlichen Benutzerrechte, sowie Ein- und Ausgabeargumente der Methode. Aus Sicht des Clients können mithilfe von Views Nodes im Adressraum des UA-Servers zusätzlich organisiert und eventuell ausgeblendet werden, wie Objekte der Klasse Objects beinhalten sie zum Auslösen von Events eine Event Notification. Die im Adressraum verwendeten Typen sind durch die Node-Typ-Klassen Objekt Type, ReverenceType, VariableType und DataType beschrieben. Eine Instanz der Klasse Variables oder Objects verweist demnach über eine Referenz zu dem Node des entsprechenden Typs (z.B. VariableType). Generell kann zwischen zwei Arten von Typen unterschieden werden. Einfache Objekttypen legen eine bestimmte Formatierung des jeweiligen Objektes fest. Im Gegensatz dazu definieren komplexe Objekttypen die Struktur von Nodes (Methoden, Objekte und Variablen), welche unterhalb des Objekttyps liegen. Abbildung 26 zeigt beispielhaft einen zusammengesetzten Objekttyp (MotorTyp), welcher ein weiteres Objekt (Drehzahlmessung) mit der zugewiesenen Variable Umdrehungen und eine weitere Variable Status, sowie die Methoden Start und Stopp enthält. Mittels References wird die Verknüpfung der Knoten spezifiziert. Sie sind eindeutig durch Quelle und Ziel, deren Richtung und den verwendeten Referenztyp identifiziert. Referenzen besitzen keinerlei Attribute und können unidirektionale oder bidirektionale Verknüpfungen darstellen. Jedoch spielt im Gegensatz zu den

[54] Vgl. Lange (2014), S. 95 ff.

symmetrischen Referenzen bei asymmetrischen Referenzen die Richtung der Verknüpfung eine wesentliche Bedeutung. [55]

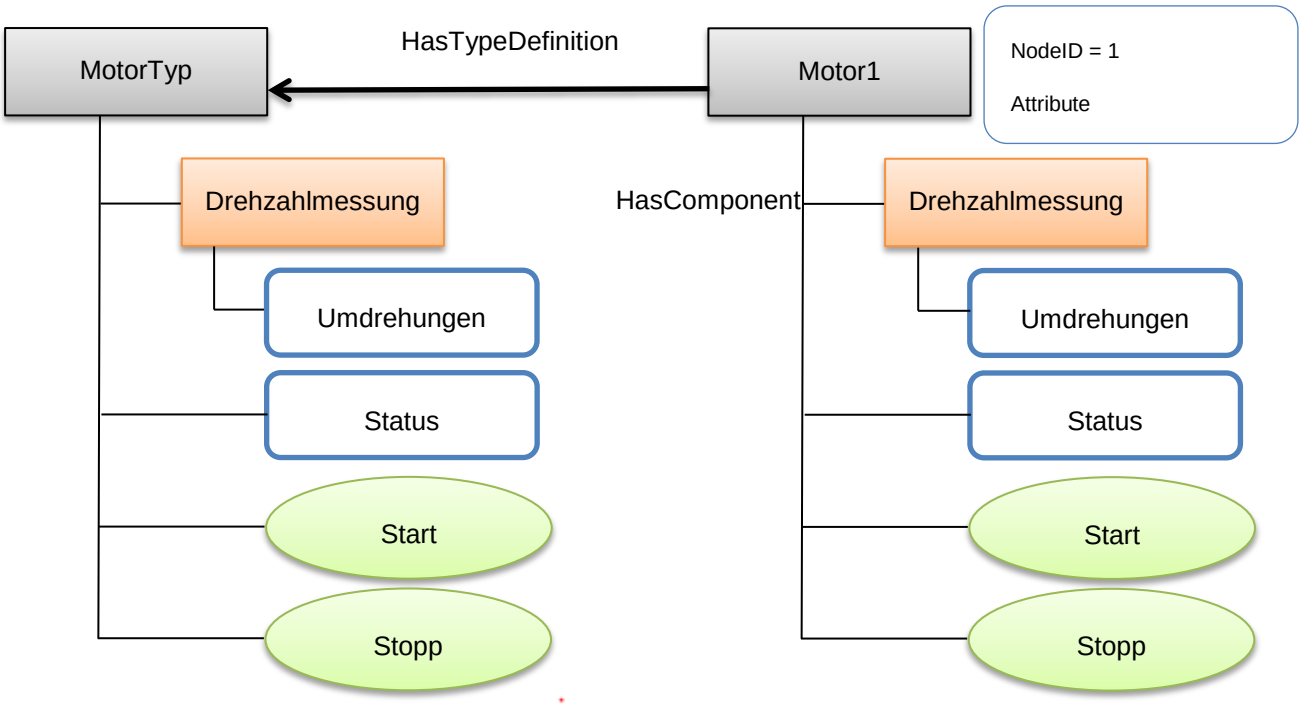


Abbildung 26: Komplexer Datentyp mit Objekt, Quelle: Eigene Darstellung.

4.3.6 OPC-UA-Services

OPC-UA definiert Services (Dienste) um Variablen zu lesen oder zu beschreiben, sich für Events oder Wertänderungen anzumelden oder durch den Namensraum zu navigieren. Die Dienste werden in Service-Sets logisch gruppiert. Daten zwischen Clients und Servern werden anhand von Service-Request und -Response ausgetauscht. Wie bereits erwähnt können hier je nach Anforderung und unabhängig vom Service mehrere Protokolle zur Verwendung kommen. Insgesamt stellt OPC-UA neun Base Service Sets zur Verfügung, welche eine Vielzahl von Funktionen beinhalten. Es sind nicht alle Dienste standardmäßig am Server implementiert, jedoch kann abgefragt werden welche Services sie unterstützen. [56]

- Session Service Set: Hiermit kann eine Verbindung zu einer Applikation auf und abgebaut werden.
- NodeManagement Service Set: Diese Services ermöglichen clientseitiges Hinzufügen, Ändern oder Löschen von Nodes im Adressraum des Servers.
- SecureChannel Service Set: Die Sicherheitskonfiguration des Servers kann über diese Dienste abfragt werden. Zusätzlich ermöglichen sie es einen sicheren Kommunikationskanal zu schaffen.
- View Service Set: Dient zum Browsen des Adressraums um dessen Struktur zu erkunden. Ein Client kann durch die Hierarchie navigieren und Verweisen zwischen Knoten folgen.

[55] Vgl. Lange (2014), S. 95 ff.

[56] Vgl. OPC Foundation (2015), Online-Quelle [07.09.2016].

- Query Service Set: Mithilfe der Query Services können Nodes im Adressraum des Servers nach bestimmten Filterkriterien ausgewählt werden.
- Attribute Service Set: Diese Services ermöglichen dem Client Werte (Attribute) zu lesen oder zu schrieben.
- Method Service Set: Es dient zum Aufrufen von Methoden eines Nodes.
- MonitoredItem Service Set: Dieser Service ermöglicht mittels Einstellungen Attribute aus dem Adressraum für einen Client auf Wertänderungen überwachen zu lassen.
- Subscription Service Set: Dient zum Editieren von MonitoredItems-Mitteilungen.[57]

4.3.7 Informationsmodelle

Diese Modelle sind schichtenweise aufgebaut und basieren auf dem im Metamodell spezifizierten Basisregeln und Basisservices. OPC-UA beschreibt für allgemein gültige Informationen (z.B. Automatisierungsdaten oder Alarme) bereits vier vorgefertigte, dem OPC-Classic-Standard nachempfundene Informationsmodelle. Allerdings können auch eigene komplexe Modelle erstellt werden, um die bereits spezifizierten Informationsmodelle noch zusätzlich zu spezialisieren. Clients, die für allgemeine Modelle vorgesehen sind, können daher in einem gewissen Umfang auch spezialisierte Modelle bearbeiten. Das in OPC-UA Data Access Modell (DA) dient zur Abbildung von Automatisierungsdaten und beinhaltet die Definition von diskreten und analogen Variablen, Quality Codes und Engineering Units. Die Datenquellen sind meist direkt im Feld beheimatet. Mittels dem Informationsmodells Alarms and Conditions (AC) wird definiert, wie Zustände und dessen Alarme und Dialoge gehandhabt werden. Eine Änderung der Zustandsvariable löst ein Event am Server aus. Der Client kann sich über Services für diese Events anmelden und die Begleitwerte des Eventreports (z.B. Quittierverhalten, Meldungstext) auswählen. Das Modell Historical Access (HA) ermöglicht einen Zugriff auf historische Events und Variablenwerte. Ein Client kann diese Daten schreiben, lesen oder ändern. Das Speicherformat dieser Daten am Server ist dafür nicht relevant, sie könnten sich in unterschiedlichsten Arten von Datenbanken befinden. Das vierte Modell Programs bietet Möglichkeiten zum Abarbeiten von komplexen Aufgaben (Programmen), beispielsweise der Zustandsautomat eines Batch Prozesses. Zustandsübergänge im Programm lösen demnach Meldungen an den Client aus. [58]

4.3.8 Security der Spezifizierung

Die Sicherheit ist eine elementare Anforderung und wurde daher in die OPC-UA-Architektur integriert. Die Mechanismen sind vergleichbar mit dem Secure Channel Konzept des World Wide Web Consortium (W3C), welches auf einer detaillierten Analyse der Bedrohungen basiert. Die OPC-UA Security beinhaltet die Authentifizierung von Servern und Clients, die Vertraulichkeit und Integrität des Datenaustausches sowie die Prüfbarkeit von Funktionsprofilen. Die von den meisten webfähigen Plattformen bereitgestellte Sicherheitsinfrastruktur wird somit zusätzlich ergänzt. Ersichtlich in Abbildung 27, basiert sie auf den drei Ebenen: User-, Application- und Transport-Security. Mechanismen der OPC-UA User Level Security

[57] Vgl. OPC Foundation (2015), Online-Quelle [07.09.2016].

[58] Vgl. OPC Foundation (2015), Online-Quelle [07.09.2016].

werden nur einmalig beim Verbindungsaufbau durchlaufen. Zur Identifikation des Benutzers übermittelt der Client ein verschlüsseltes Security Token an den Server. Damit wird der Benutzer authentifiziert und ist damit autorisiert auf Objekte am Server zuzugreifen. Die Application Level Security ist ebenfalls Teil des Sitzungsaufbaus. Sie umfasst den Austausch digital signierter Zertifikate. Diese Software-Zertifikate identifizieren die implementierten OPC-UA Profile und die Client- und Server-Software. Sie beschreiben die am Server eingesetzten Informationsmodelle. Die Transport Level Security kann zusätzlich zu der User und App Level Security eingesetzt werden. Die Verschlüsselung des Transport Levels sichert die Unversehrtheit der übertragenen Nachrichten und verhindert deren Offenlegung an Dritte. Alle drei Sicherheitsmechanismen sind in den von der OPC-Foundation bereitgestellten Softwarepaketen, den OPC-UA Stacks (APIs) realisiert. [59]

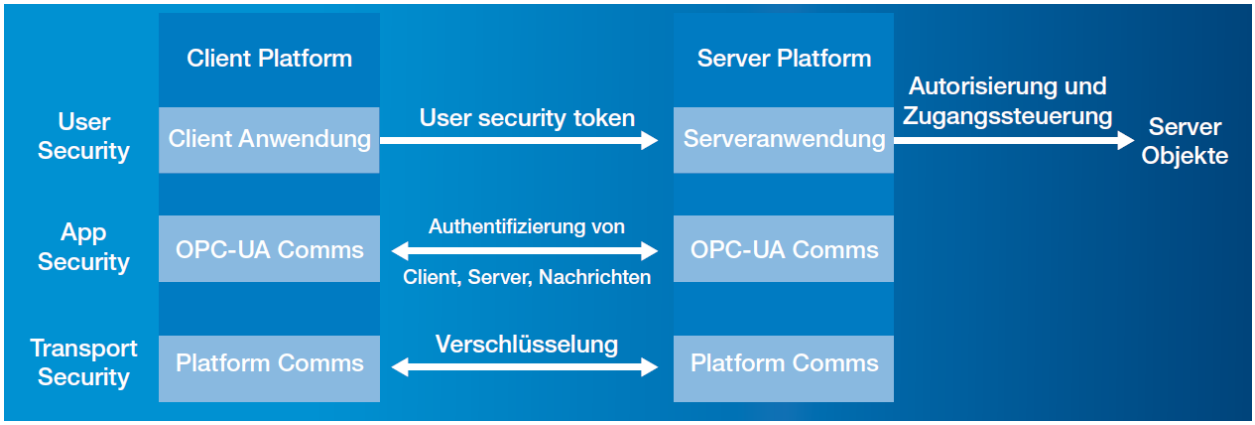


Abbildung 27: Skalierbares Sicherheitskonzept von OPC-UA, Quelle: OPC Foundation (2015), Online-Quelle [07.09.2016].

Zum Schutz der Kommunikation mittels OPC-UA sind laut dem Deutschen Bundesamt für Sicherheit in der Informationstechnik folgende Einstellungen von zentraler Bedeutung:

- SecurityMode: Eine Authentifizierung auf Applikationsebene ist empfohlen. Es sollte zumindest der securityMode Sign (Signieren von Nachrichten) oder SignAndEncrypt (Signieren und Verschlüsseln von Nachrichten) zur Verwendung kommen. Der securityMode 'None' bietet hingegen keinerlei Schutz. securityMode 'SignAndEncrypt' bietet die Möglichkeit besonders vertrauliche Daten über die Integrität hinaus zu schützen.
- Wahl der kryptographischen Algorithmen: Die securityPolicy Basic256Sha256 ist zu bevorzugen. Schwächere securityPolicies verwenden teilweise veraltete Algorithmen und sind dadurch zu vermeiden.
- Benutzerauthentifizierung: Die Anmeldung an den Server sollte nicht mit der Kennung anonymous erfolgen. Dieses Benutzerkonto bietet keinerlei Schutz, da nicht nachvollzogen werden kann wer Daten oder Konfigurationen am Server verändert hat. Zusätzlich könnte ein Angreifer dieses ungesicherte Standardkonto verwenden. Sind die Rechte des anonymous nicht begrenzt, kann ein möglicher Angreifer problemlos Daten am Server bearbeiten. [60]

[59] Vgl. OPC Foundation (2015), Online-Quelle [07.09.2016].

[60] Vgl. Bundesamt für Sicherheit in der Informationstechnik (2016), Online-Quelle [07.09.2016].

5 LÖSUNGSANSÄTZE

Dieser Abschnitt behandelt die Ausarbeitung von Konzepten für eine praktische Umsetzung des Systems und der damit verbundenen Kommunikation zwischen mehreren Produktionsanlagen mit den Webanwendungen (Onlineshop und Monitoring-Ansicht) bzw. der zentralen Datenbank. Die Anforderungen an die Applikation werden anhand der erlangten Kenntnisse des theoretischen Teils der Arbeit zusätzlich präzisiert. Außerdem sind mögliche Bedrohungen aufzuzeigen, um grundsätzliche Sicherheitsanforderungen und zu verwendende Protokolle der Konzepte zu formulieren. Die abschließende Bewertung der Konzepte erfolgt nach den in Kapitel 5.1 festgelegten Kriterien.

5.1 Anforderungen an das System

Im Folgenden wird das grundsätzliche Funktionsprinzip des zu entwerfenden Produktionssystems näher spezifiziert und die Anforderungen an das System anhand der im theoretischen Teil erhaltenen Erkenntnisse präzisiert.

Wie bereits in der Aufgabenstellung skizziert, soll der Informationsaustausch mit dem Kunden mittels eines Webshops stattfinden, wobei die Webshop-Webanwendung die Erstellung von Aufträgen für die automatische Weiterverarbeitung durch die Produktionsmaschinen fokussiert. Zusätzliche Funktionen wie beispielsweise ein Warenkorb oder eine Benutzerdatenerfassung sind meist Teil eines Shopsystems, jedoch sind sie für die weitere Untersuchung nicht relevant.

Der Startimpuls zur Produktion erfolgt vom Kunden, indem er die Bestellung (neuer Auftrag in die Datenbank) im Webshop tätigt, oder durch die Maschine, welche gespeicherte Aufträge nacheinander abarbeitet. Diese Logik sollte idealerweise nicht zentral aufgebaut sein, da wie bereits in Kapitel 2 erwähnt, der Ansatz verfolgt wird, dass sich die Maschine selbstständig die Aufträge von der Webdatenbank holt, um als Alternative zu einem MES-System aufzutreten. Demnach ist eine Speicherung der Bestellungen in einer Datenbank unumgänglich. Ein Datensatz einer abzuarbeitenden Bestellung besteht aus der vom Kunden gewählten Produktkonfiguration der gewünschten Menge und Adressdaten für den Versand. Die Übergabe der Bestelldaten (Aufträge) an die Produktion sowie vom Webbrowser in die Datenbank sollte gesichert erfolgen, da sie sensible Daten beinhalten. Außerdem könnten von Dritten verfälschte Angaben einen hohen wirtschaftlichen Schaden provozieren. Diese Sicherheitsaspekte sind jedoch im nachfolgenden Kapitel näher ausgeführt. Die anlagenseitige Abarbeitung der Aufträge durch die Anlage sollte nach dem Datenaustausch automatisch erfolgen. Wie bereits in Kapitel 3 gezeigt, bieten sich zur Speicherung von Daten (Eingaben der Kunden) in der Datenbank serverseitige Skriptsprachen wie PHP oder Node.js an. Eine weitere Anforderung besteht darin, Monitoring-Daten der Produktionsmaschinen für den Betreiber ersichtlich im Web darzustellen. Diese Webseite sollte in der Datenbank ersichtliche Monitoring-Daten sichtbar darstellen, beispielweise in Form von Tabellen und Charts, wobei hier besonders historische Werte zu fokussieren sind.

Neben der Aufzeichnung von relevanten Fertigungs- bzw. Maschinendaten in der zentralen Datenbank ist eine Protokollierung von Betriebsstörungen und Ausfällen von Vorteil. Die Visualisierung dieser Daten sollte auf einer nur für den Betreiber der Anlage ersichtlichen Webseite erfolgen, dies stellt eine

besondere Herausforderung dar. Wie bereits in Kapitel 4.3 erwähnt, wird der Datenaustausch zwischen Client (Webbrowser) und Webserver meist über den Client initiiert. Die Aktualisierung bei Fehlzuständen (Fehlermeldungen) sollte jedoch ohne zusätzliche Benutzeraktion ersichtlich sein. Für diese Art des Datenaustausches sind Technologien wie AJAX oder Websocket besonders geeignet, wobei wie in Kapitel 4.3.3 gezeigt, Websocket Teil der HTML5-Spezifizierung ist und (noch) nicht von jedem Webserver unterstützt wird.

Des Weiteren sollte das zu konzipierende System leicht in bestehende Anlagen und auch in bestehende Webshop-Systeme integrierbar sein und einen komfortablen Wechsel bzw. Tausch der Steuerung ermöglichen. Eine von der Steuerungsplattform unabhängige Lösung ist zu bevorzugen. Das System sollte besonders leicht erweiterbar sein, um die Möglichkeit zu bieten mehrere örtlich verteilte Produktionsanlagen parallel zu betrieben. Außerdem sollte es wirtschaftlich und hoch verfügbar sein. Besonders ein Ausfall des Webservers (Onlineshops) ist zu vermeiden, denn auch bei einem kurzzeitigen Ausfall einer Produktionsmaschine könnten Kunden trotzdem weiter bestellen. Demnach lassen sich einige Eckpunkte festhalten:

- Gesicherte, bidirektionale Verbindung zwischen Produktion und Datenbank am Webserver (Aufträge lesen, Prozessdaten in Datenbank schreiben)
- Wirtschaftlichkeit, möglichst schlanke Architektur.
- Verteilte Systemarchitektur ermöglichen
- Leicht erweiter- und in bestehende Systeme -integrierbar
- Unabhängig von der Steuerungsplattform (SPS-seitig)
- Keine zentrale Logik, die Maschine holt sich die Aufträge
- Clientseitige Aktualisierung ermöglichen (Fehlermeldungen auf Betreiber Webseite)
- Gesicherte, bidirektionale Verbindung zwischen Webserver und Client

5.2 Bedrohungsszenarien

Zunächst werden anhand der in den vorangehenden Kapiteln erlangten Erkenntnisse mögliche Bedrohungen durch Dritte aufgezeigt und Lösungsvorschläge dafür präzisiert. Diese Vorschläge werden anschließend in den Konzeptionen für eine mögliche Systemarchitektur berücksichtigt, um die Sicherheit des Systems zu gewährleisten. Abbildung 28 zeigt den grundsätzlichen Systemaufbau und mögliche Bedrohungen von außen. Wie bereits in der Aufgabenstellung gefordert, basiert das System auf einer zentralen Datenbank als Informationsspeicher für die Webseiten (Onlineshop und Monitoring) sowie den Steuerungen der Produktion. Innere Bedrohungen wie beispielsweise unberechtigten Zugriff auf die Produktionsmaschinen vom lokalen Netzwerk (SPS-Programm verändern mit PC im LAN) oder mutwillige Veränderung der Webseiten am Webserver (Angreifer loggt sich am Webserver-PC oder in der Web-Plattform ein) werden hier nicht berücksichtigt. Diese Szenarien sind vor allem durch organisatorische Maßnahmen, wie regelmäßigen Passwortwechsel oder Zugangsbeschränkungen für Mitarbeiter zu verhindern. Ebenfalls könnte, wie in Kapitel 4.3.8 beschrieben, der interne Datenaustausch mittels OPC-UA verschlüsselt erfolgen. Jedoch ist dies in diesem Fall aufgrund der geringen räumlichen Ausdehnung und der schweren Zugänglichkeit der Produktionsanlagen nicht unbedingt notwendig. Ein Verändern der abgelegten Skripts und Webseiten am Webserver, sowie von Datenbankeinträgen oder von

Programmcode der Produktion (z.B. SPS-Code) durch unberechtigten Zugriff von außen wäre besonders kritisch und könnte das System beeinträchtigen. Denn über eingeschleuste Schadsoftware auf einem Rechner kann dieser - ohne Sicherungsmaßnahmen - von außen kontrolliert werden. Kapitel 3.3 zeigt, dass die Webserveranwendung auf vielen verschiedenen physikalischen Rechnern lauffähig ist. Besonders bei der Implementierung mittels eigenen physikalischem Rechner (Rootserver) ist auf Sicherungsmaßnahmen des Webservers und der Datenbank besonders zu achten, denn hier übernimmt dies kein externer Anbieter.

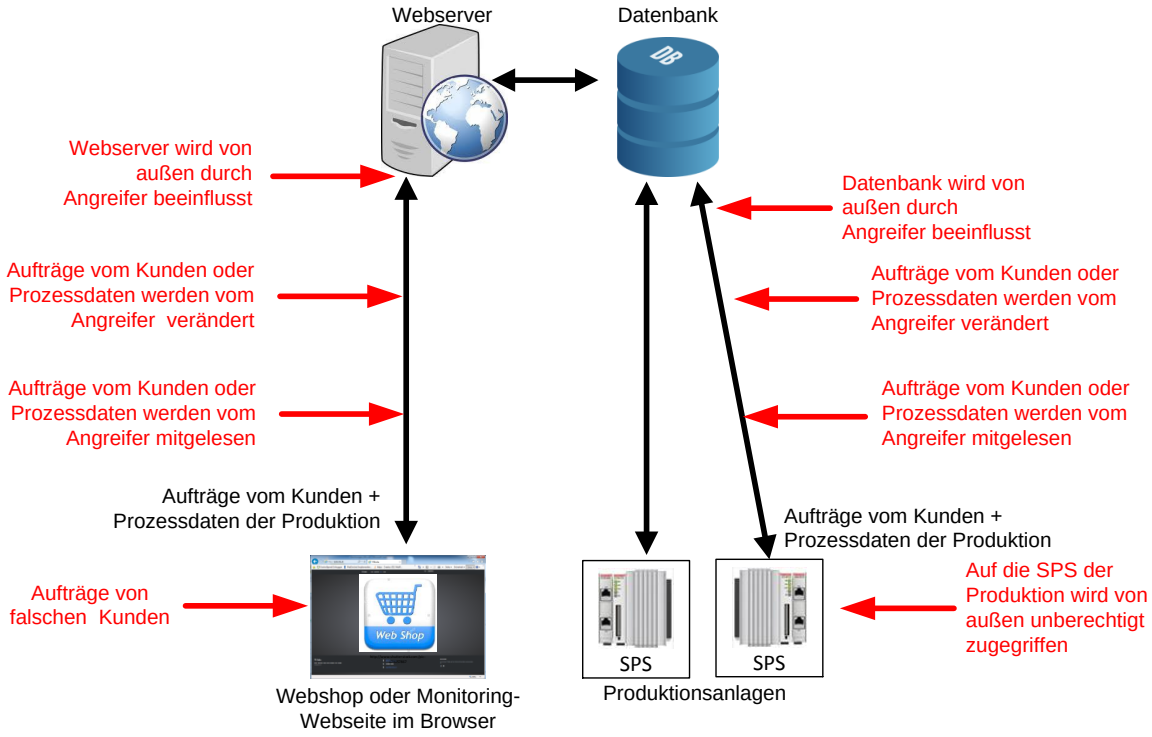


Abbildung 28: Bedrohungsszenario, Quelle: Eigene Darstellung.

Bei Verwendung eines Rootservers ist unbedingt eine zusätzliche externe Firewall zu installieren, um das interne Netzwerk vor nicht freigegebenen Netzwerksverkehr von außen (vom Internet) zu bewahren. Jedoch bietet eine Sperrung der Ports oder Paketfiltertechnologien von handelsüblichen externen Firewalls keinen Schutz vor Netzwerkverbindungen von Schadsoftware am Rechner nach außen (vom Rootserver zum Angreifer). Als Abhilfe ist ein Virenschutz in Kombination mit der Personal-Firewall des PCs als zusätzlicher Schutz ratsam, um Netzwerksverbindungen unerwünschter Dienste zu vermeiden und den PC zu schützen. Ebenso ist der Datenverkehr der Produktionsanlage mit dem Internet zu überwachen und das dahinterliegende Netzwerk zu schützen. Wird die Internetverbindung mittels mobiler Daten aufgebaut, so bietet ein Firewall-Dienst des Mobilfunkanbieters einen gewissen Schutz. Jedoch sind diese nur bedingt vom Endkunden parametrierbar und sie werden auch nicht von allen Mobilfunkanbietern zur Verfügung gestellt. Mehrere Hersteller bieten für Industrieanlagen konzipierte UMTS/HSPA-Mobilfunkrouter an, welche zusätzlich zur Gateway-Funktion eine externe Firewall beinhalten (z.B. Conel Ur5i).

Neben dem Überwachen des ein- und ausgehenden Datenverkehres mittels Firewalls zwischen dem Internet und der Produktionsanlage bzw. dem Webserver zur Vermeidung von unberechtigten Zugriff

durch Schadsoftware könnten Angreifer auch die Kommunikation beider mitlesen und somit sensible Daten abgreifen oder verändern. Die Übertragung von Aufträgen vom im Browser aufgerufenen Webshop zur Datenbank, oder der mit Prozessdaten gefüllten HTML-Seite zum Client ist demnach zu sichern. Wie bereits in Kapitel 3.3 gezeigt, kann die Datenbank auch auf einem externen Rechner ausgeführt werden. Um nicht noch zusätzlich den Datenaustausch zwischen Webserver und Datenbank sichern zu müssen, besteht die Notwendigkeit die Datenbank auf derselben Hardwareplattform wie den Webserver zu betreiben.

Das Mitlesen oder Verändern von Nachrichten zwischen Client (Webbrowser) und dem Webserver kann im globalen Netzwerk dem Internet grundsätzlich nicht verhindert werden. Jeder könnte sich in eine Leitung einschleusen und den Netzwerksverkehr mitlesen. Jedoch bietet die Verschlüsselung der Inhalte in diesem Fall eine Abhilfe. Damit wird der abgefangene Text für den Angreifer nicht interpretierbar übertragen (als Hash). Eine Manipulation der Zeichen ist nicht möglich, da sie bei der Entschlüsselung erkannt wird. Dies kann grundsätzlich mittels asymmetrischer oder symmetrischer Verschlüsselung erfolgen. Eine rein symmetrisch verschlüsselte Verbindung kann allein aus organisatorischen Gründen für die beiden Webanwendungen ausgeschlossen werden, da die Bestrebung besteht vielen Benutzern eine sichere Verbindung zu ermöglichen. Die Verteilung von privaten Schlüsseln an potenzielle Kunden für die Verschlüsselung wäre, vor allem für einen leicht erreichbaren Webshop nicht zielführend. Wie bereits in Kapitel 3.4 gezeigt, bietet hier die Verwendung von HTTPS als Übertragungsprotokoll eine von allen Browsern nativ unterstützte Variante zur verschlüsselten Kommunikation zwischen dem am Client im Webbrowser aufgerufenen Webanwendungen (Webshop oder Monitoring) und dem Webserver, wobei hier eine hybride Verschlüsselung zur Verwendung kommt. Der Session-Aufbau erfolgt asymmetrisch, die weitere Verbindung, beispielsweise die Übertragung von Parametern per POST-Methode oder der Seitenaufruf ist symmetrisch verschlüsselt. HTTPS ist als Standard im Web anzusehen, um Daten verschlüsselt auszutauschen und garantiert durch die Verwendung von glaubwürdigen Zertifikaten die Authentizität des Webservers bzw. des Betreibers. Die Identität der Clients (Betreiber oder Kunden) wird hier jedoch nicht festgestellt, es müsste eine zusätzlich Authentizitätsprüfung stattfinden.

Ein weiteres Szenario wäre eine mutwillige Bestellung von Waren durch einen Angreifer im Webshop oder die Nutzung der Monitoring-Ansicht von Unautorisierten. Zusätzlich ist die Authentizität der Kunden bzw. der Betreiber zu erheben. Die meistgebräuchliche Variante dafür ist die klassische Registrierung mittels Benutzername und Passwort. Die Art des Eingabeformulars auf der Webseite (z.B. durch PHP-Skript) und die Passwortspeicherung in der Datenbank sind besonders anfällig. Sie bieten Angreifern die Möglichkeit auf sensible Daten zuzugreifen, diese aus der Datenbank auszulesen oder zu löschen. (z.B. mittels SQL-Injektion in einem Eingabeformular). Moderne serverseitige Skriptsprachen bieten mehrere Methoden zur Sicherung an. Außerdem können die in Kapitel 3.5 aufgezeigten Funktionen von Bezahldiensten dazu genutzt werden (z.B. Bestellung ist bezahlt), um die Echtheit einer Bestellung im Webshop sicherzustellen. Aufträge werden beispielsweise nur weiterhin bearbeitet, wenn sie als bezahlt markiert sind.

Neben der Sicherung der Verbindung zwischen den beiden im Browser aufrufbaren Webanwendungen und dem Webserver (und der Datenbank) ist für das Gesamtsystem besonders die Sicherung der Datenströme von der Datenbank bis zur Produktion relevant. Wird die Datenbank und die

Webserveranwendung auf einem Rootserver oder auf einer gehosteten virtuellen Maschine ausgeführt, könnten neben einer HTTPS-Verbindung ebenfalls VPN-Verbindungen zum Einsatz kommen, um die lokalen Netzwerke der Produktion über das Internet mit dem Server zu verbinden. Die VPN-Verbindung garantiert ebenfalls die bei HTTPS nicht nativ gewährleistete Authentizität der Produktionsanlage bzw. des zentralen Servers. Zum Verbindungsaufbau ist eine Eingabe von Anmeldeinformationen (Benutzername und Passwort) notwendig. Andere Varianten wie beispielsweise eine Standleitung sind allein aus wirtschaftlichen Gründen auszuschließen. Bei Verwendung eines verschlüsselten VPN-Tunnels können die darin eingeschlossenen Datenpakete gegen Manipulationen und das unbefugte Mithören gesichert werden. Zum Beispiel kann mittels L2TP/IPsec (Layer 2 Tunneling Protocol/IP-Secure) als Tunneling-Protokoll eine Verbindung (auf Layer 2 im OSI-Schichtenmodell) zwischen zwei Netzwerken aufgebaut werden, wobei die native Unterstützung von IPSec in den meisten Betriebssystemen bereits integriert ist. Die Verschlüsselung gilt, im Gegensatz zu älteren VPN-Protokollen wie PPTP (Point-to-Point Tunneling Protocol) als sicher. Generell kann zwischen reinen Softwarelösungen (VPN-Server und Clientsoftware auf den zu verbinden PCs) und Lösungen mit speziell dafür konzipierter Hardware sogenannte VPN-Routern (mit darauf installiertem VPN-Software und einer Clientsoftware auf dem zu verbindenden PC) unterschieden werden. VPN-Router für die Industrie sind meist als kompakte Hardware mit zusätzlichen Funktionen (z.B. dynamisches DNS, UMTS-Router und Firewall) erhältlich und können somit ohne weitere zusätzliche Komponenten das lokale Netzwerk der Maschine mit dem Zielrechner (Webserver-PC) verbinden. Darüber hinaus bieten Hersteller für die M2M-Verbindung konzipierte VPN-Verwaltungssysteme mit zentralem VPN-Server an (z.B. DigiCluster von BellEquip). Diese meist cloudbasierten Lösungen ermöglichen die Verwaltung von vielen VPN-Verbindungen und erleichtern die Konfiguration der Router, da die sonst notwendige eindeutige Adressierung im globalen Netzwerk ohne dynamische IP-Dienste oder fixer IP-Adresse erfolgen kann. Die Art der Verschlüsselung hängt aber vom Anbieter verwendeten Tunneling-Protokoll der VPN-Lösung ab. Hier könnte das im Vergleich zu IPSec modernere OpenVPN zur Verwendung kommen, dieses wird mittels TLS bzw. SSL-Zertifikaten verschlüsselt. Die Entscheidung welche Variante der VPN-Verbindung zu bevorzugen ist, wird daher vor allem von wirtschaftlichen Faktoren geprägt. Sind mehrere VPN-Verbindungen mit vielen dezentralen Stationen vorgesehen, ist ein Verwaltungssystem sicherlich zu bevorzugen, da hier die sonst aufwendige Adressierung der Stationen entfällt.

5.3 Konzepte

Die Erstellung von Konzepten wird in diesem Abschnitt besonders fokussiert. Anschließend ist ein möglichst sicheres und flexibles System auszuwählen, um die automatische Auftragsabarbeitung, von der Bestellung im Onlineshop bis zur Fertigung des gewünschten individuellen Produkts, sowie die Datenerfassung von Prozessdaten an einer Testanlage zu erproben.

Im Folgenden werden Konzepte vorgestellt, welche einen Informationsfluss zwischen Produktion und den web-basierten Schnittstellen zu Endkunden und Betreibern gewährleisten, um mehrere Möglichkeiten zu aufzuzeigen wie das System aufgebaut sein könnte. Alle vier Konzepte ermöglichen eine bidirektionale Kommunikation zwischen den SPS gesteuerten Produktionsanlagen und einer zentralen Datenbank am Webserver. Die Datenbank dient wiederum als Informationsgrundlage für die Webanwendungen.

5.3.1 Implementierung mit Datenbankkommunikationstool und Rootserver

Dieses in Abbildung 29 ersichtliche Konzept basiert auf der Verwendung eines herstellerspezifischen Kommunikationstools zum Datenaustausch zwischen SPS und Datenbank, sowie einer lokalen Installation des Webservers auf einem Rootserver (eigener lokaler PC) mit Internetverbindung.

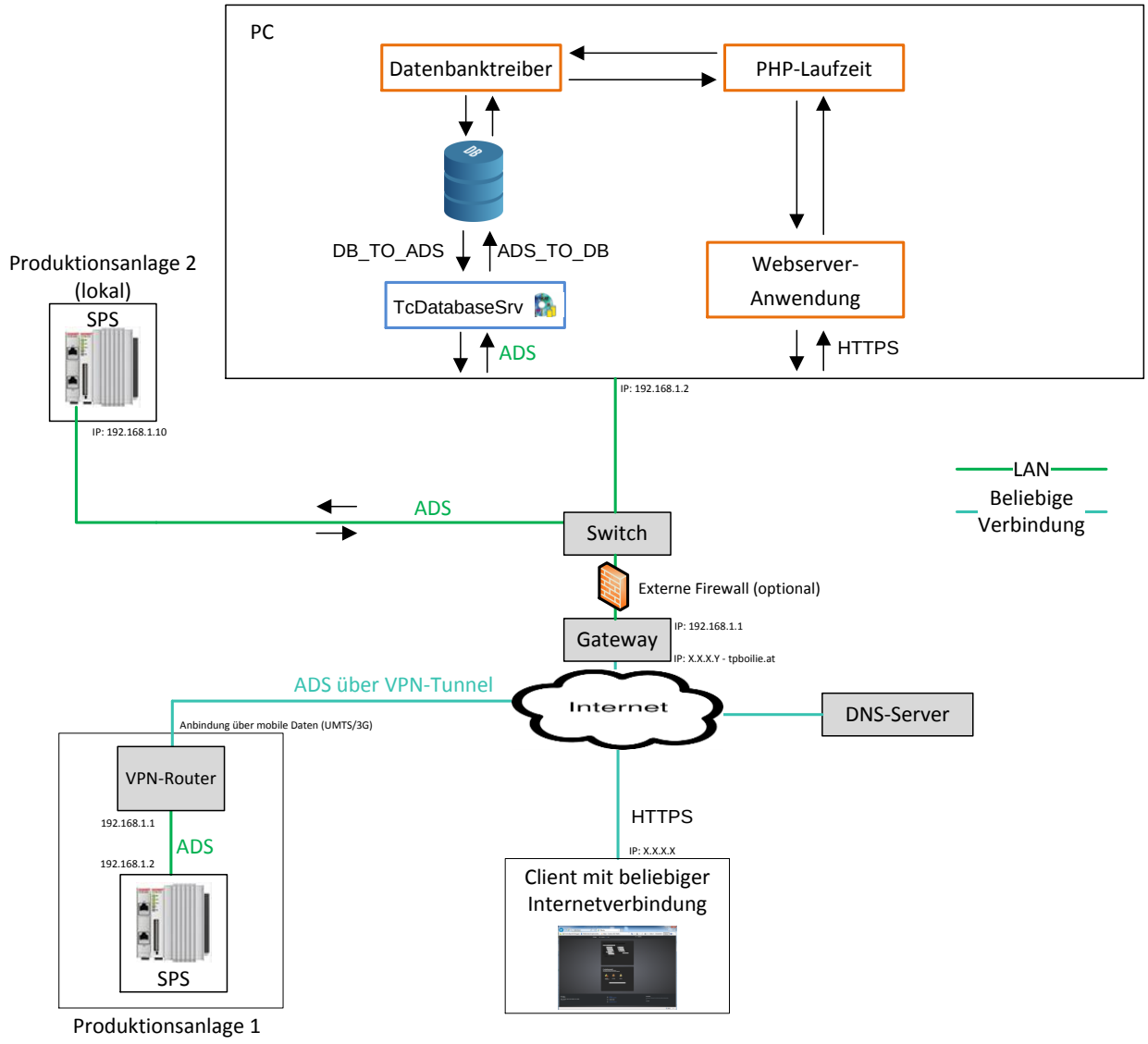


Abbildung 29: Netzwerkschema des Konzepts Datenbankkommunikationstool, Quelle: Eigene Darstellung.

Die Datenbank, in welcher beispielsweise Aufträge und Prozessdaten gespeichert sind, bildet das zentrale Element des Systems. In diesem Fall wird als Schnittstelle zwischen der Datenbank und der Produktionsanlage der TwinCAT Database-Server von Beckhoff verwendet. Wie bereits in Kapitel 4.1 geschildert, ermöglicht diese Technologie verschiedenste Netzwerktopologien. Die Datenbank-Server-Software (in Abbildung 29 als TcDatabaseSrv bezeichnet) kann zentral beispielweise auf einem PC oder dezentral auf jeder Steuerung installiert sein. In dieser Konzeption wird jedoch nur ein zentraler Database-Server verwendet, welcher auf einem lokalen PC zusammen mit dem Webserver installiert ist. Die Kommunikation zwischen Database-Server und den Steuerungen im lokalen Netzwerk würde über das herstellerspezifische ADS-Protokoll geschehen. Die SPS der Produktionsanlage wäre über ein LAN-Netzwerk (z.B. per Ethernet) mit dem PC verbunden. Für eine örtlich verteilte Architektur ist die

Kommunikation über das Internet unerlässlich. Die entfernte Station (z.B. Produktionsanlage 1) müsste über einen VPN-Tunnel mit dem lokalen Netzwerk des zentralen PCs verbunden werden. Wie bereits in Kapitel 5.2 aufgezeigt, sind mehrere Varianten von VPN-Systemen denkbar, jedoch wird hier ein VPN-Verwaltungssystem bevorzugt. Dies erfordert zusätzlich zum Cloud-Dienst (VPN-Server), wiederum einen Mobilfunkrouter mit VPN-Funktion (z.B. Ur5i des Herstellers Conel) sowie eine VPN-Client-Software (z.B. OpenVPN) am Ziel-PC, die eine weitere (virtuelle) Ethernet-Schnittstelle aufbaut. Der VPN-Router muss bei Verwendung eines Verwaltungssystems auch nicht eindeutig im Internet adressierbar sein (keine fixe IP, oder DNS-Dienst). Durch den Aufruf von speziellen Funktionsbausteinen wird der Datenaustauch zwischen der SPS der Produktionsanlage und der Datenbank über den zentralen Database-Server angestoßen. Erhält der Database-Server beispielsweise den Befehl DB_TO_ADS von einer Produktionsanlage, liest dieser den gewünschten Inhalt (z.B. Auftragsdaten) aus der Datenbank aus und überträgt die Daten an die SPS. Die Webserversoftware (z.B. Apache) ist in diesem Konzept auf einem lokalen PC installiert. Für den leichten Zugang der bereitgestellten Webseiten am PC erscheint eine Domain sinnvoll, dazu muss zur Namensauflösung der IP-Adresse des lokalen Gateways ein externer DNS-Server bzw. ein DNS-Service inklusive Domain (z.B. DynDNS) verwendet werden. Das Hosten eines eigenen DNS-Servers wäre hier aufgrund des dafür nötigen Aufwandes nicht zielführend. Beim Besuch der Webseite würden die HTTP-Anfragen der Clients über das Gateway (nach Konfiguration) an den Webserver (am PC) weitergeleitet werden. Außerdem könnte, wie in Kapitel 3.3.2 beschrieben, ein SSL-Zertifikat angefordert werden, um die geforderte gesicherte Kommunikation mittels HTTPS zu ermöglichen. Als weitere Sicherheitsmaßnahme gegen einen ungewünschten Zugriff von außen wird zusätzlich zur Firewall des PCs eine externe Firewall angedacht. Eine zusätzliche PHP-Laufzeit-Software (inkl. Datenbanktreiber) ermöglicht das Ausführen von PHP-Skripts. Sie können, wie bereits in Kapitel 3.2 gezeigt, bei Aufruf oder durch Events (z.B. Button-Click) Daten aus der Datenbank auslesen und dem Client im Webbrowser in Form einer statischen HTML-Seite anzeigen.

5.3.2 OPC-DA mit Wrapper und Dedicated Virtual Hosting

Im Gegensatz zum ersten Konzept sind die Softwarekomponenten nicht auf einem Rootserver, sondern wie bereits in Kapitel 3.2 erörtert, auf einem auf Betriebssystemebene virtualisierten Rechners eines externen Anbieters installiert. In dieser Lösung werden der Aufbau sowie die Topologie der gesamten Hardware von einem externen Anbieter übernommen. Beispielsweise bietet Amazon mehrere Cloud Computing-Varianten bezüglich Rechenleistung, Speicherplatz und Betriebssystem für das Dedicated Virtual Hosting an (z.B. Amazon EC2). Die tatsächliche Hardware, auf der die virtuelle Maschine ausgeführt wird, ist für den Benutzer dieser Dienste jedoch nicht ersichtlich. Es kann nur zwischen den verwendeten Rechenzentren (z.B. Serverfarm in Frankfurt oder in Irland) ausgewählt werden. Jedoch sind diese Zentren in der Regel sehr gut geschützt, wobei nicht nur der Schutz vor unberechtigten Zugriff, sondern auch die Verfügbarkeit der Rechner im Vordergrund steht. Eine Internetverbindung zu den Server-Instanzen ist allein aufgrund der Zugänglichkeit für die Benutzer auf die virtuellen Rechner unerlässlich und wird vom Anbieter zur Verfügung gestellt. Unter Anderem kann bei Amazon ausgewählt werden ob der virtuelle Rechner nur für den Benutzter privat zugänglich oder wie für den Betrieb einer Webserver-Anwendung notwendig, ebenfalls für andere über das Internet erreichbar ist.

Das zentrale Element des Systems ist wiederum eine beliebige Datenbank (z.B. Microsoft SQL), welche sämtliche Informationen beinhaltet und einerseits von den Webanwendungen (Monitoring-Ansicht oder Webshop) und andererseits von den Produktionsanlagen (Aufträge, Prozessdaten) genutzt wird. Wie in Abbildung 30 dargestellt, wird in dieser Konzeption der Datenaustauch zwischen der zentralen Datenbank am virtuellen Server und den Produktionsanlagen von einer Wrapper-Software in Kombination mit einem OPC-DA-Server übernommen.

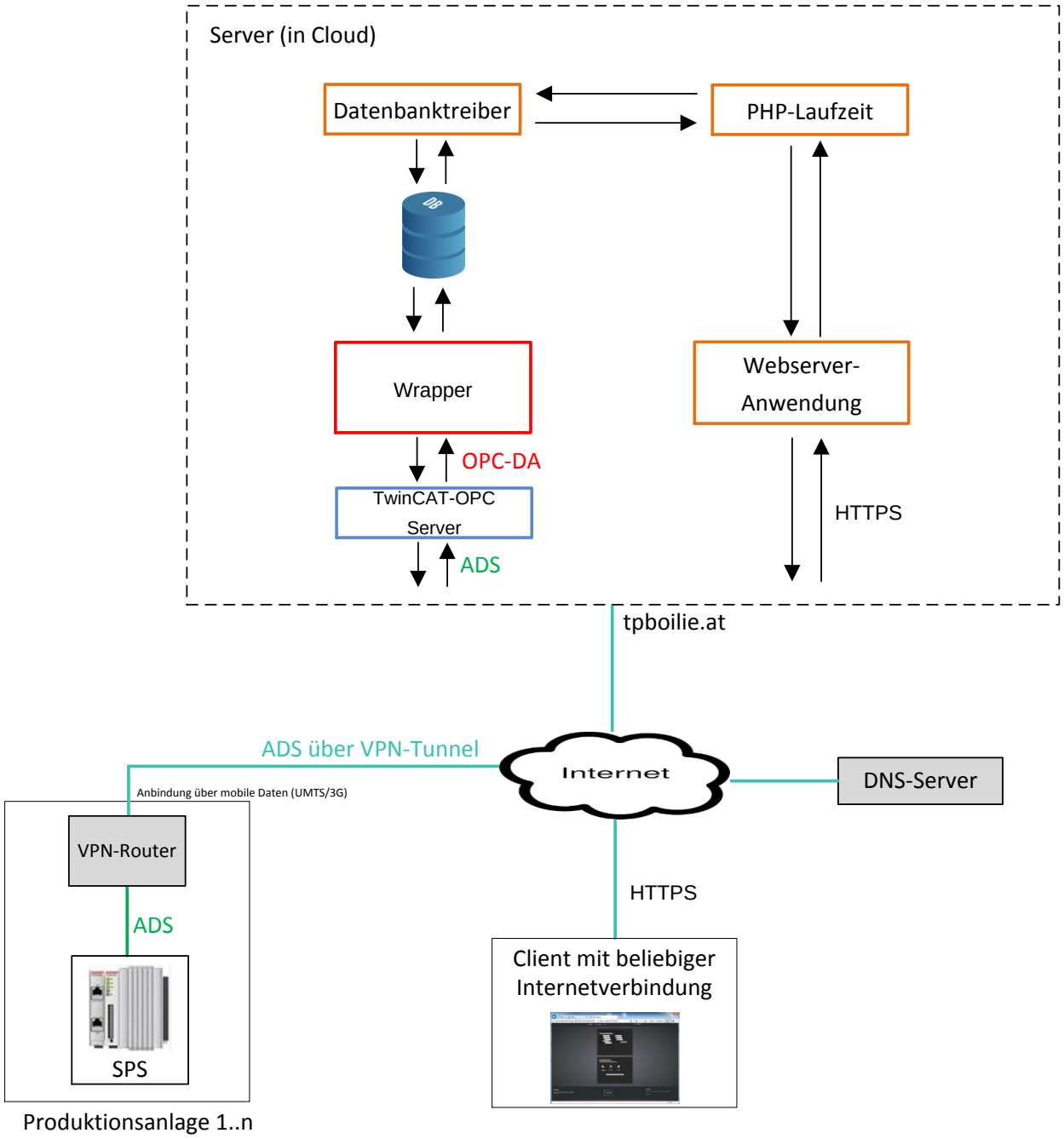


Abbildung 30: Netzwerkschema des Konzepts OPC-DA mit Wrapper, Quelle: Eigene Darstellung.

Prozessdaten der Produktionsanlage (z.B. Monitoring-Werte) werden vom OPC-Server im standardisierten OPC-Format zur Verfügung gestellt, von der Wrapper-Anwendung gelesen (OPC-Client) und abschließend in die Datenbank gespeichert. Außerdem können OPC-Nodes mit aus der Datenbank ausgelesenen Werten beschrieben werden (z.B. Aufträge). Die beispielsweise in .NET implementierte

Wrapper-Anwendung beinhaltet eine OPC-Client-API und eine Datenbankschnittstelle, um Werte aus der Datenbank zu lesen oder zu schreiben. Die OPC-Client-Komponente der Wrapper-Anwendung dient zum Austausch von Prozessdaten mit einem oder mehreren OPC-Servern. Wie in Kapitel 4.2 erläutert, basiert die Kommunikation des OPC-DA-Standards auf der COM/DCOM-Schnittstelle von Microsoft und verlangt eine Windows-Plattform. Der OPC-Server nutzt zum Datenaustausch mit der Steuerung wiederum ein herstellerspezifisches Protokoll. Außerdem unterstützt OPC-DA die Übertragung von String-Variablen (siehe 4.2.7), dies ist vor allem für die Auftragsabarbeitung (Adressen der Kunden) erforderlich. Ebenso könnte der Timestamp für die Aufzeichnungen verwendet werden Die Wrapper-Applikation müsste die Daten aller Produktionsanlagen abfragen. Die Fülle an Daten könnte mittels OPC-Server und Group-Objekten ordentlich strukturiert werden. In diesem Konzept kommt eine Beckhoff-SPS zur Verwendung, die mittels ADS mit dem OPC-Server (TWINCAT-OPC) kommuniziert. Die Verbindung zwischen der Produktion und dem am virtuellen Server installierten OPC-Server erfolgt über das Internet mittels einer VPN-Verbindung, die eine virtuelle Ethernet-Schnittstelle am Server zur Verfügung stellt. Hier könnte wie im ersten Konzept, eine VPN-Client-Software am virtuellen Server sowie ein VPN-Router für jede Produktionsanlage eingesetzt werden. Die Webserver-Anwendung zur Bereitstellung der Webanwendungen wird ebenfalls auf dem auf der Betriebssystemebene virtualisierten Rechner ausgeführt. Die notwendige Namensauflösung der gewählten Domain der Webseite (z.B. tbboilie.at) zur IP-Adresse des virtuellen Rechners erfolgt über einen (externen) DNS-Server, es könnte ebenfalls der DNS-Service des Hosting-Anbieters genutzt werden. Beispielsweise bietet Amazon den DNS-Service Amazon Route 53 an. Wie bereits in der vorherigen Konzeption, würden die Clients mittels HTTP bzw. gesichert über HTTPS mit dem Webserver kommunizieren. Eine zusätzliche PHP-Laufzeit-Software (inkl. Datenbanktreiber) ermöglicht das Ausführen von PHP-Scripts. Sie können (siehe Kapitel 4.2) bei Aufruf oder durch Events Daten aus der Datenbank auslesen und dem Client im Webbrowser in Form einer statischen HTML-Seite anzeigen.

5.3.3 OPC-UA und Node.js

Die in Abbildung 31 gezeigte Konzeption basiert auf die Verwendung von OPC-UA zur Kommunikation mit den Produktionsanlagen und einem in Node.js implementierten Webserver. Die OPC-Fundation stellt OPC-Stacks in mehreren Programmiersprachen bereit, darunter ein in Node.js (JavaScript) implementierter OPC-Client. Dieser Client könnte direkt in eine Node.js-Anwendung eingebunden werden und den Datenaustausch mit den Produktionsanlagen ermöglichen, wobei die Produktionsmaschinen mit einem eigens erstellten OPC-UA-Informationsmodell moduliert werden könnten. Wie bereits in Kapitel 3.2.2 erwähnt, kann eine Node.js-Applikation ebenso die Aufgaben eines HTTP-Servers übernehmen. Die Verwendung von HTTPS als Kommunikationsprotokoll mit den Web-Clients (Browser) ist ebenfalls möglich. Der Webserver müsste jedoch zertifiziert werden, oder ein selbst signiertes Zertifikat zur Verwendung kommen. Der Zugriff auf die zentrale Datenbank könnte ebenfalls mittels der in Node.js implementierten Anwendung erfolgen. Aufgrund der geringen Verbreitung von Node.js-Unterstützung bei Shared-Virtual-Hosting-Anbietern wird auf das schon in den vorrangehenden Konzeptionen geschilderte Dedicated Virtual Hosting zurückgegriffen. Die JavaScript-Umgebung ist somit auf einem auf Betriebssystemebene virtualisierten Rechner realisiert. Die Hardware-Komponente dieses Rechners wird somit einem externen Anbieter übergeben. Zur Namensauflösung des Webservers für eine Erreichbarkeit

der Webseite mittels URL ist die Inanspruchnahme eines DNS-Services unabdingbar. Dies könnte wiederrum über die Verwendung eines Dienstes eines externen Anbieters erfolgen.

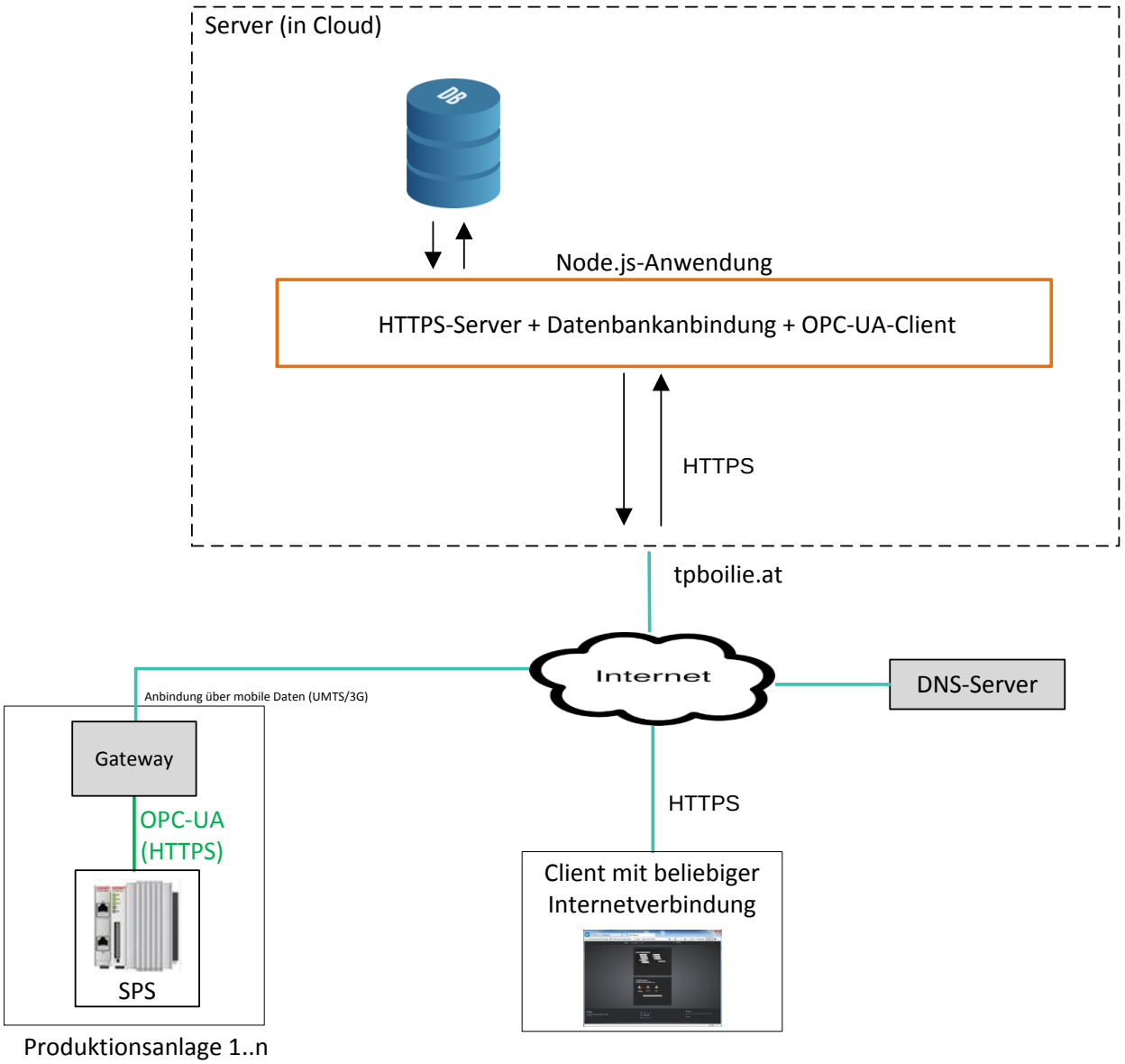


Abbildung 31: Netzwerkschema des Konzepts OPC-UA am Webserver, Quelle: Eigene Darstellung.

Wie in Kapitel 4.3 erörtert, stellt der OPC-UA-Standard mehrere Kommunikationsprotokolle zur Verfügung. Neben dem UA-Binary für lokale Netzwerke könnte mit dem Hybridprotokoll oder mittels Webservices über das Internet hinweg kommuniziert werden. Unterstützt der OPC-Server eines dieser beiden Protokolle, ist eine VPN-Verbindung nicht zwingend notwendig und ein einfacheres Gateway könnte verwendet werden. Beispielsweise kann die Internetverbindung durch ein handelsübliches UMTS-Gateway mit integrierter Firewall-Funktion hergestellt werden. Jedoch ist in diesem Fall eine eindeutige Adressierung des OPC-UA-Servers im globalen Netzwerk erforderlich. Es könnte auch ein Gateway mit Unterstützung von DNS-Services (z.B. DynDNS) zum Einsatz kommen. Ebenfalls wäre der Bezug einer statischen IP-Adresse vom Internetprovider eine mögliche Option.

Das lokale Netzwerk der Produktionsanlage könnte auch über einen VPN-Tunnel mit dem virtuellen Server verbunden werden. Die externe Firewall würde den Schutz vor unberechtigtem Zugriff auf das

lokale Netzwerk der Produktionsmaschine erhöhen. Der OPC-UA Server ist in diesem Konzept direkt auf der Steuerung implementiert. Unterstützt ein SPS-Hersteller diese Vorgehensweise nicht, müsste die jeweilige Produktionsanlage um einen weiteren PC erweitert werden, auf dem der OPC-UA-Server ausgeführt wird.

5.3.4 OPC-UA mit Wrapper und Shared Virtual Hosting

In Kapitel 4.2. schon behandelt, stellen Webhosting-Pakete (Shared Virtual Hosting) eine sehr wirtschaftliche Variante zum Veröffentlichen einer Website dar. In dieser Konzeption wird der komplette Aufbau der Webseite gehostet. Das Paket beinhaltet demnach den benötigten Webspace, sprich den Speicherplatz für die Dateien (z.B.HTML-Seiten, Bilder) und Skripte, sowie einer eigenen Domain (z.B. tpboillie.at). Meist ist der Zugriff auf eine MySQL-Datenbank im Angebot inbegriffen. Damit entfallen die Anschaffung der Hardware und Softwarekomponenten des Webservers, sowie der DNS-Service. Webhosting-Anbieter unterbinden in der Regel jede Art von direktem Zugriff auf ihre Hardware. Als Benutzer (Webseitenbetreiber) ist nur der eigene Webspace über ein vom Anbieter zur Verfügung gestelltes Konfigurations-Webportal ersichtlich. Die Unterstützung von serverseitigen Skriptsprachen des Webservers ist vom Hosting-Anbieter bestimmt. Eine Installation von zusätzlicher Software ist nicht möglich.

Der Zugriff auf die gehostete Datenbank erfolgt, dargestellt in Abbildung 32, ausschließlich über den Aufruf von Skripten durch den Webserver. Wie in Kapitel 3.2.3 näher behandelt, könnte ein PHP-Skript zur Verwendung kommen, welches von einer Client-Anwendung aufgerufen wird. Andere Zugriffsarten auf die Datenbank werden in der Regel vom Hosting-Anbieter aus Sicherheitsgründen unterbunden. Aufgrund dieser Beschränkung muss die Kopplung mit der Produktionsanlage über den besagten Aufruf von Skripts geschehen. Dies kann, wie bei anderen Webclients über HTTP- oder HTTPS-Request erfolgen. In der in Abbildung 32 gezeigten Konzeption wird für diese Kopplung eine Wrapper-Anwendung verwendet. Diese Applikation bildet das Bindeglied zwischen der Produktionsanlage und der Datenbank am gehosteten Server. Die beispielsweise in .NET implementierte Wrapper-Anwendung beinhaltet eine OPC-UA-Client-API und einen Webclient. Prozessdaten der Produktionsanlage (z.B. Monitoring-Daten) werden von einem OPC-UA Server zur Verfügung gestellt, von der Wrapper-Anwendung gelesen (OPC-UA-Client) und an den Webserver versendet. Das aufgerufene PHP-Skript speichert die Daten in der Datenbank ab. Zum Auslesen von Daten aus der Datenbank (z.B. Aufträge) wird von der Wrapper-Anwendung wiederum ein PHP-Skript am Webserver aufgerufen. Jedoch liest die Wrapper-Applikation die am Webserver vom PHP-Skript ausgelesenen Datenbankinhalte und überträgt sie an den OPC-UA-Server. Jedoch kann, im Gegensatz zu den vorangehenden Konzepten, auf ein aufwändiges VPN-Gateway zur Internetverbindung der Produktionsanlage verzichtet werden, da zur Kommunikation ohnehin das internetfähige HTTPS-Protokoll verwendet wird. Als zusätzliche Sicherheit für das lokale Netzwerk ist nach dem Gateway eine weitere externe Firewall zu installieren (siehe Kapitel 5.2).

Die Wrapper-Applikation wird in diesem Fall auf einem handelsüblichen PC ausgeführt, dieser kann lokal mit einer beliebigen Steuerung verbunden werden. Außerdem könnte der zusätzliche PC lokale Visualisierungs- oder Steuerungsaufgaben (Soft-SPS mit TwinCAT) erfüllen. Zur Aufbereitung der Prozessdaten der SPS wird in diesem Konzept auf einen OPC-UA-Server zurückgegriffen. Wie bereits in Kapitel 4.3 behandelt, ermöglicht OPC-UA aufgrund seiner Plattformunabhängigkeit einen Server direkt

auf der Steuerung zu betreiben. Da der PC (mit Wrapper-Applikation) sowie der OPC-UA-Server (auf der SPS) in einem lokalen Netzwerk miteinander verbunden sind, kann das performante OPC-UA-Binary-Protokoll genutzt werden.

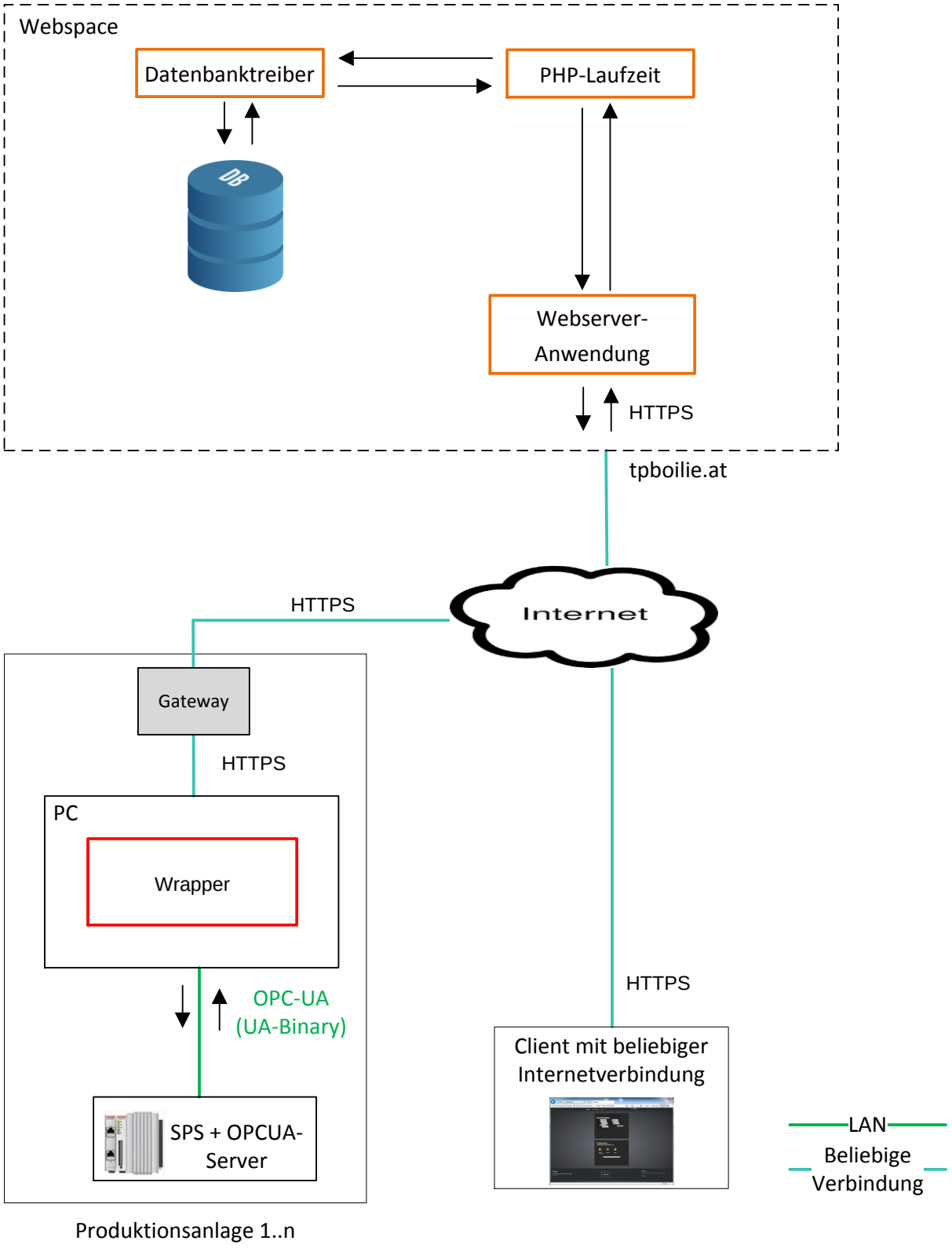


Abbildung 32: Netzwerkschema des Konzepts OPC-UA mit Wrapper, Quelle: Eigene Darstellung.

Die Implementierung des Servers direkt auf der Steuerung ist jedoch noch nicht bei jedem SPS-Hersteller möglich, deshalb könnte alternativ ein OPC-UA-Server zusammen mit der Wrapper-Applikation am lokalen PC ausgeführt werden. Der Datenaustausch mit der Steuerung erfolgt in diesem Fall mittels herstellerspezifischer Protokolle. Die Schnittstelle zur Wrapper-Anwendung beeinflusst die Lokalität des OPC-UA Servers allerdings nicht. Der Informationsaustausch mit den Kunden bzw. den Betreibern würde, wie bereits in den vorherigen Konzeptionen gezeigt, über eine dynamische Webseite geschehen, welche die Daten aus der Datenbank aufbereitet. Clients kommunizieren mittels HTTP bzw. gesichert über HTTPS mit dem Webserver. Die PHP-Laufzeit-Software (inkl. Datenbanktreiber) ist bereits am Webserver vorinstalliert.

5.4 Analyse der Konzepte

Im folgenden Abschnitt werden die ausgearbeiteten Konzepte anhand der in Kapitel 5.1 festgelegten Anforderungen analysiert, sowie Vor und Nachteile der Lösungsvorschläge aufgezeigt, welche die Grundlage zur Entscheidung bilden sollten. Zusätzlich wird die Realisierbarkeit des Systems als zusätzliches Bewertungskriterium hinzugefügt

Alle vier erarbeiteten Konzepte ermöglichen die geforderte verteilte Systemarchitektur und den laut Kapitel 5.2 geforderten sicheren Datenaustausch zwischen den Produktionsstandorten und der zentralen Datenbank über das Internet. Zusätzlich bietet jedes Konzept einen Webserver an, somit können beispielsweise, nach Aufruf der Webseite mit der Shop-Webanwendung Daten in die Datenbank gespeichert werden. Die Verbindung zwischen Client (Browser) und Webserver erfolgt laut den Ergebnissen aus Kapitel 5.2 bei jedem Konzept verschlüsselt, wobei die Kommunikation mittels HTTPS oder verschlüsselter VPN-Verbindung als weitgehend sicher angesehen wird. Jedoch unterscheiden sich die konzipierten Systeme erheblich in der Ausführung und der Flexibilität. Die in Kapitel 5.1 geforderte Verfügbarkeit des Webservers kann mithilfe jedes Konzepts erreicht werden, der Aufwand dafür ist aber unterschiedlich hoch.

5.4.1 Implementierung mit Datenbankkommunikationstool und Rootserver

Das in der ersten Konzeption verwendete Datenbank Kommunikationstool (TwinCAT Database-Server) kann ausschließlich zum Datenaustausch mit Beckhoff-Steuerungssystemen verwendet werden. Auch Systeme anderer Hersteller sind spezifisch auf das proprietäre Protokoll der Steuerungen abgestimmt. Eine mögliche Erweiterung des Systems auf unterschiedliche Steuerungen ist nur bedingt möglich. Je nach Steuerungstype der Produktionsanlagen müsste ein passendes herstellerspezifisches Datenbankkommunikationstool ergänzt werden (z.B. von B&R). Es bietet es eine Lösung mit sehr wenigen Schnittstellen und stellt ein fast schüsselfertiges System dar. Der Webserver ist bei dieser Variante selbst zu betreiben. Eine Einbindung von bestehenden Webshops in das System ist aufgrund der Bindung an ein Windows Betriebssystem nur bedingt möglich, da viele Webserver nicht auf Windows ausgeführt werden können. Die Verbindung der Produktionsanlagen mit dem Server erfolgt, wie bei zwei anderen Konzeptionen mittels VPN-Verbindung. Durch die Verwendung eines (kostenpflichtigen) Verwaltungsdienstes entfällt die (ebenfalls kostenpflichtige) Adressierung der Stationen. Dennoch müsste für jede weitere Anlage ein VPN-Client am Rootserver ausgeführt werden. Außerdem zeigt Kapitel 5.2

den hohen zusätzlichen Hardwareaufwand, aufgrund der Sicherungsanforderungen (z.B. Firewall) für den in dieser Konzeption verwendeten Rootserver. Zusätzlich ist für den reibungslosen Betrieb der Webseiten eine hohe Verfügbarkeit des Servers und der Internetverbindung unbedingt notwendig. Dies erfordert wiederum Investitionen in robuste Hardware und Infrastruktur (z.B. gesicherte Stromversorgung des Servers). Des Weiteren ist zur Sicherstellung des Betriebs eine regelmäßige Wartung des Servers unerlässlich. Die Verwendung eines Rootservers bietet demnach die höchste Flexibilität, jedoch muss die Verfügbarkeit und Sicherheit gegeben sein. Dies ist mit hohem wirtschaftlichem Aufwand verbunden.

Vorteile:

- Schlüsselfertiges System zur Datenbankanbindung der Produktionsanlagen
- Hohe Realisierbarkeit der Datenbankanbindung
- Hohe Flexibilität des Rootservers

Nachteile:

- Hohe Anschaffungskosten (Infrastruktur und Hardware)
- Ständige Wartung des Rootservers
- Windowsplattform am Webserver notwendig
- Von der Steuerungsplattform abhängig
- VPN-Dienst notwendig
- Mäßig integrierbar

5.4.2 OPC-DA mit Wrapper und Dedicated Virtual Hosting

Im Gegensatz dazu wird in den weiteren Konzepten ein externer Anbieter mit den Sicherheitsaspekten (Sicherung gegen unberechtigtem Zugriff) und mit der Sicherstellung der Verfügbarkeit beauftragt. Die in Serverfarmen organisierte Hardware ist hier besonders hoch verfügbar. Beispielsweise garantiert Amazon eine Verfügbarkeit von 99,95 %. Die Auslagerung des physikalischen Webservers bietet keine Flexibilität in Hinsicht auf die Hardware, jedoch bietet es sämtliche Freiheiten auf Betriebssystem-Ebene. Ebenso entfällt der Wartungsaufwand für die Hardware. Der Webserver müsste, wie im vorrangehenden Konzept jedoch selbst betrieben und gewartet werden. Außerdem sind zusätzliche DNS-Dienste zum Betriebt einer Domain notwendig. Aufgrund der Nutzung von OPC-DA zum Datenaustausch mit der Produktion ist am virtuellen PC eine Windowsplattform unabdingbar. Eine Einbindung von bestehenden Webshops in das System ist aufgrund dieser Bindung nur bedingt möglich. OPC-DA ist ein etablierter Standard zur Aufbereitung von Prozessdaten und wird von jedem SPS-Hersteller durch einen eigenen OPC-Server unterstützt. Zur Erweiterung des Systems müsste nur ein weiterer OPC-Server installiert werden. Außerdem könnte der OPC-DA-Client der Wrapper Software auf einen bestehenden OPC-DA-Server einer Anlage über einen VPN-Tunnel zugreifen. Hier kann der Datenverkehr, durch das Aktualisieren von Daten nur bei Wertänderung zusätzlich verringert werden (siehe Kapitel 4.2.6). Auf welche Weise die OPC-Server auch immer verbunden mit dem zentralen Server sind, die Schnittstelle zur Wrapper-Anwendung ist immer standardisiert.

Vorteile:

- leichte Integrierbarkeit in bestehende Anlagen ohne Webshop (hohe Verbreitung von OPC-DA)
- Standardisierte Software-Komponenten (OPC-DA Client und Server)
- Geringe Anschaffungs- und Wartungskosten (virtueller Server)

Nachteile:

- Webserver muss selbst betrieben werden
- Windowsplattform am Webserver notwendig
- Mäßig in bestehende Shops integrierbar

5.4.3 OPC-UA und Node.js

Dieses Konzept basiert ebenfalls auf die Verwendung eines Dedicated Virtual Hosting-Servers, aufgrund der geringen Verbreitung von Node.js unter Webhosting-Anbietern. Jedoch ist diese Umgebung nicht an ein Windows-Betriebssystem gebunden. Hier bildet im Gegensatz zu den vorrangehenden Konzeptionen, OPC-UA die Kommunikationsbasis mit der Produktion. Dies ermöglicht zwei grundlegende Kommunikationsvarianten mit dem in Node.js implementierten OPC-UA-Client. Wird das OPC-UA-Hybrid-Protokoll verwendet, so könnte auf ein aufwändigeres Gateway (VPN-Router) bzw. einem VPN-Dienst verzichtet werden. Jedoch werden die internetfähigen Protokolle nicht von jedem OPC-UA-Server unterstützt und zusätzlich müsste ein DNS-Dienst (oder fixe IP) für jede Produktionsanlage eingerichtet werden, um sie langfristig im weltweiten Netzwerk zu erreichen. Andererseits bietet OPC-UA viele bewährte Funktionen und einige Neuerungen gegenüber dem OPC-Classic-Standard. Ebenfalls kann auch hier der Datenverkehr optimiert werden indem die Event Notification eines Node-Objekts genutzt wird (siehe Kapitel 4.3.5). Des Weiteren bietet OPC-UA Informations- und Metamodelle an, um die Produktionsanlage digital mithilfe komplexer Objekte nachzubilden und leicht zu erweitern (z.B. neue Produktionsanlage). Der Verbindungsauf und -abbau sowie weitere Funktionen würden über vom Server angebotene Services geschehen, wobei hier wiederum nicht alle Services von jedem Server zur Verfügung gestellt werden. Die Verbreitung von OPC-UA ist noch nicht mit der des bewährten OPC-Classic zu vergleichen, jedoch bietet schon fast jeder Hersteller einen geeigneten Server an. Wie bereits erwähnt, müsste das System um einen weiteren lokalen PC mit OPC-Server erweitert werden, wenn der OPC-UA-Server nicht direkt auf der Steuerung ausführbar ist. Eine weitere Variante wäre ein am virtuellen Server realisierter OPC-UA-Server. Jedoch ist hier wiederum eine VPN-Verbindung zur Produktion erforderlich, da das Kommunikationsprotokoll zwischen Server und SPS alleine nicht internetfähig ist. Zusätzlich zur Implementierung der OPC-UA Clients müsste in diesem Konzept ebenfalls ein HTTPS-Server mit Node.js erstellt werden. Obwohl hier die Integration in bestehende Shopsysteme grundsätzlich möglich ist, wäre die Applikation auf die OPC-UA-Komponente zum Datenaustausch mit der zentralen Datenbank und den Produktionsanlagen begrenzt. Außerdem könnte der Datenverkehr durch die umfangreichen Sicherheitsmechanismen (siehe 4.3.8) zusätzlich zum sicheren Kommunikationsprotokoll HTTPS gesichert werden. Vor allem ist eine in Kapitel 5.2 geforderte zusätzliche Authentifizierung der Produktionsanlagen (OPC-Server) bzw. des OPC-UA-Clients (Node.js) möglich.

Vorteile:

- Integrierbarkeit in bestehende Anlagen ohne Webshop (mäßige Verbreitung von OPC-UA)
- Standardisierte Software-Komponenten (OPC-UA Client und Server)
- Geringe Anschaffungs- und Wartungskosten (virtueller Server)
- VPN-Verbindung ist nicht zwingend notwendig

Nachteile:

- Webserver muss selbst betrieben werden
- Node.js am Webserver notwendig
- Mäßig in bestehende Shops integrierbar

5.4.4 OPC-UA mit Wrapper und Shared Virtual Hosting

Dieses Konzept bietet als einziges die Möglichkeit eine Datenbank auf einem Shared Virtual Hosting-Server anzusprechen. Hier ist nämlich eine Installation zusätzlicher Software oder der direkte Zugriff auf die Web-Datenbank nicht möglich. Der Datenaustausch geschieht hier mittels einer Wrapper-Anwendung, welche PHP-Skripts am Webserver aufruft. Diese Vorgehensweise macht das System sehr leicht in bestehende Shopsysteme integrierbar, denn es müsste nur ein zusätzliches PHP-Skript auf dem Webserver gespeichert werden, wobei PHP sehr weit verbreitet ist und von fast jeder Webserveranwendung unterstützt wird. Ist dies nicht der Fall, müsste die Programmiersprache des Skripts angepasst werden. Durch die Nutzung eines Webspaces (Shared Virtual Hosting) entfallen die sonst notwenigen DNS-Dienste zum Erhalt und Betrieb einer Domain. Ebenfalls sind keine weiteren Dienste (z.B. VPN-System, oder DNS) für die Adressierung der Produktionsstandorte von Nöten. Die Verfügbarkeit und die Sicherheit vor unberechtigtem Zugriff ist ebenfalls sehr hoch, jedoch muss dem Anbieter in dieser Hinsicht vertraut werden, bzw. der Zugang vertraglich geregelt sein. Die zu entwerfende Wrapper-Anwendung am lokalen PC, beispielsweise in C# implementiert, bindet das Koppelglied zwischen der SPS der Maschine und der Web-Datenbank. Hier kommt eine standardisierte OPC-UA-Client-API als Kommunikationsschnittstelle mit dem auf der SPS implementierten OPC-UA-Server zur Verwendung. Es könnte - je nach verwendeter SPS - auch eine andere Schnittstelle genutzt werden, beispielsweise OPC-DA für bestehende ältere Anlagen. Die Wrapper-Applikation müsste angepasst werden, wobei mit OPC-UA (und evtl. OPC-DA) ein sehr weiter Bereich abgedeckt werden kann. Außerdem ist die Integration in bestehende Produktionsanlagen vergleichbar einfach, denn es bedarf nur der Installation der Wrapper-Anwendung auf einem (evtl. besehenden) PC mit gesicherter Internetverbindung. Die Authentizität der Wrapper-Anwendung müsste (siehe Kapitel 5.2) im PHP-Skript abgefragt werden, denn die PHP-Skripts (z.B. Aufträge lesen, Prozessdaten in die Datenbank speichern) sind für jeden im globalen Netzwerk aufrufbar, wenn die URL bekannt ist.

Vorteile:

- Integrierbarkeit in bestehende Anlagen mit und ohne Webshop (mäßige Verbreitung von OPC-UA, Datenaustausch über PHP-Skripts)
- Standardisierte Software-Komponenten (OPC-UA Client und Server)
- Geringe Anschaffungs- und Wartungskosten (Webspace)

- Webserver wird von externen Anbietern zur Verfügung gestellt (Shared Virtual Hosting)

Nachteile:

- Zusätzlicher Hardwareaufwand (PC) in der Produktionsanlage

5.5 Entscheidung

In diesem Kapitel werden die vier Konzepte miteinander verglichen und abschließend mittels Punktevergabe bewertet. Die Vergabe ist in Tabelle 1 veranschaulicht und dient zur Auswahl eines geeigneten Systems.

Die Sicherheit der Verbindung zwischen der zentralen Datenbank und den Produktionsanlagen (Gateway mit Firewall) ist bei allen Konzeptionen grundsätzlich gegeben, jedoch werden hier aufgrund des Webservers Punkte abgezogen. Der im ersten Konzept verwendete Rootserver bietet eine sehr hohe Flexibilität, jedoch ist der Betreiber für sämtliche Sicherheitskonfigurationen verantwortlich und muss durch regelmäßige Wartung (Updates) die Sicherheit gewährleisten. Diese Aspekte werden in den drei anderen Konzeptionen von externen Anbietern übernommen, wobei bei dem Hosting-Anbieter in dieser Hinsicht volles Vertrauen geschenkt werden muss. Demnach erhalten alle Konzepte 30 von 35 Punkten.

Wirtschaftlich betrachtet bietet die Verwendung eines Webhosting-Pakets (Shared Virtual Hosting) einige Vorteile gegenüber anderen Varianten zum Betreiben der Webseiten, denn es entfallen sämtliche Anschaffungskosten für Hard- und Software. Zusätzlich wird die Wartung vom Hosting-Anbieter übernommen und die Domain ist ebenfalls im Angebot enthalten. Besonders für kleinere Webseiten ist dies sicherlich die günstigste Variante. Die Bereitstellung eines Root-Severs ist jedoch mit hohen Anschaffungs- und Wartungskosten verbunden und wird demnach in Tabelle 1 mit den wenigsten Punkten bewertet. Ein gemieteter virtueller Server (Dedicated Virtual Hosting) verursacht monatliche Gebühren, aber es entfallen Anschaffungskosten für die Hardware des Webservers. Die Software (Webserver-Anwendung) und die DNS-Dienste für eine Domain sind ebenfalls erforderlich.

Anlagenseitig wird bei zwei Konzepten ein Gateway mit eingebautem VPN-Router und Firewall zur Internetverbindung genutzt, wobei bei der Verwendung einer Wrapper-Anwendung auf einem PC (siehe 5.3.3) oder über das Hybridprotokoll vom OPC-UA eine Verbindung mittels HTTPS erfolgt. Hier wird die VPN-Funktion nicht benötigt, somit kann ein wirtschaftlicheres Gateway genutzt werden. Ebenso entfallen bei Nutzung der Wrapper-Applikation die Lizenzkosten für eine eindeutige Adressierung der Produktionsanlagen. Der zusätzliche Rechner im Produktionsnetzwerk (Konzept 4) kann als ohnehin notwendiges Visualisierungssystem zur Verwendung kommen, ist aber bei den anderen Konzepten nicht unbedingt notwendig.

Eine weitere Anforderung ist die Integrierbarkeit des Systems in bestehende Anlagen. Hier sticht besonders das Konzept aus Kapitel 5.3.4 heraus. Es ermöglicht als einziges die Kommunikation mit einer Datenbank auf einem mittels Shared Virtual Hosting betrieben Webserver. Diese Variante kann aber im Vergleich zu den anderen Konzeptionen auch mit jeder anderen Art von Server kommunizieren, da hier wie bei einem klassischen Webclients PHP-Skripts (Webseiten) aufgerufen werden. Hier ist keine Installation einer zusätzlichen Software (z.B. Node.js oder eine Wrapper-Anwendung) am Webserver notwendig, es müssten nur Skripts am Webserver gespeichert werden. Ebenso ist die Integration in die

Produktionsanlage vergleichbar einfach. Deshalb wird hier die maximale Punktzahl vergeben. Das erste Konzept (Kommunikationstool) ist jedoch als schlecht in bestehende Anlagen integrierbar anzusehen, denn es erfordert einerseits einen eigenen Windows Webserver (mäßige Verbreitung) und andererseits ist es auf einen SPS-Hersteller begrenzt (bzw. eigenes System für andere SPS) und wird deshalb in der Integrierbarkeit und in der Unabhängigkeit mit wenigen Punkten bewertet. Die beiden weiteren Konzepte sind aufgrund der Verwendung eins OPC-Standards relativ gut zur Integration geeignet, jedoch ist OPC-DA weiter verbreitet und wird von jedem SPS-Hersteller angeboten. OPC-DA erfordert aber einen Windows-Rechner als Webserver, was wiederum die Integration erschwert und somit diese Vorteile gegenüber OPC-UA (und Node.js) ausgleicht (gleiche Punktzahl in der Realisierbarkeit). Jedoch wird bei dem Konzept mit OPC-DA ein leichter Vorteil in der Unabhängigkeit vom SPS-Hersteller gesehen.

Betreffend Realisierbarkeit (max. 10 Punkte) bietet das erste Konzept (Kommunikationstool und Rootserver) ein bis zur Webdatenbank fast schlüsselfertiges System an, jedoch mindert der zu implementierende Rootserver die Realisierbarkeit des Systems (es müsste alles selbst gemacht werden). Ebenso mindert der Betrieb und die Konfiguration eines eigenen Webservers die Umsetzbarkeit der Konzepte mit virtuellem Server (minus 2 Punkte). Die Nutzung eines Webspaces ist als relativ komfortabel anzusehen, da hiermit relativ schnell und performant eine Webseite online gestellt werden kann. Die Verwendung von standardisierten Softwareschnittstellen (OPC-APIs) erleichtert die Implementierung einer möglichen Wrapper-Anwendung, welche im ersten Konzept nicht notwendig wäre. Daher wird den Konzepten mit Wrapper (bzw. Node.js) nur ein Punkt abgezogen.

	Datenbank Kommuni-kationstool mit Rootserver	OPC-DA mit Wrapper und Dedicated Virtual Hosting	OPC-UA mit Node.js	OPC-UA mit Wrapper und Shared Virtual Hosting
Sicherheit (35 Punkte)	30	30	30	30
Wirtschaftlichkeit (25 Punkte)	10	18	18	15
Leicht erweiter- und integrierbar (20 Punkte)	5	15	15	20
Unabhängig von der Steuerungsplattform (10 Punkte)	1	9	8	8
Realisierbarkeit (10 Punkte)	8	7	7	9
Summe (100 Punkte)	**54**	**79**	**78**	**82**

Tabelle 1: Bewertung der Konzepte, Quelle: Eigene Darstellung.

Die Tabelle 1 zeigt demnach, dass das Konzept mit dem gehosteten Webspace und der lokalen Wrapper-Anwendung (siehe 5.3.4 und siehe 5.4.4) am besten für das System geeignet ist. Besonders aufgrund der leichten Integrierbarkeit und der hohen Realisierbarkeit wird diese Systemarchitektur bevorzugt. Zusätzlich werden alle in Kapitel 5.2 aufgezeigten Sicherheitsanforderungen ausreichend erfüllt und es ermöglicht als einziges die Verwendung eines gemieteten Webspaces. Der dafür zusätzlich benötigte zusätzliche PC könnte sicherlich für weitere Aufgaben in der Produktion eingesetzt werden.

6 AUFBAU UND UMSETZUNG

Nach der Auswahl einer geeigneten Konzeption für das System und der genaueren Spezifikation der Anforderungen wird im Zuge dieses Kapitels die grafische Benutzeroberfläche der erstellten Anwendungen vorgestellt. Zusätzlich soll die Implementierung der Wrapper-Anwendung OPC-UA-Clients in C# und der Aufbau der beiden Webanwendungen anhand von kurzen Auszügen erläutert werden. Die Vorstellung der Versuchsanlage und ein abschließender Test der in der Aufgabenstellung geforderten und in Kapitel 5 präzisierten Funktionen sind ebenfalls Teil dieses Abschnittes.

6.1 Versuchsanlage

Die Maschine soll eine hohe Flexibilität im Angebot der Produkte ermöglichen und zusätzlich zeigen, wie durch vertikale Integration eine wirtschaftliche und flexible Produktion ermöglicht werden kann. Die gewählte Versuchsanlage dient zur Produktion von runden, gekochten Teigkugeln, in Anglerkreisen Boilies genannt (siehe Abbildung 33). Mithilfe des im vorherigen Kapitel ausgewählten Konzepts sollte es möglich sein, dass die Angelköder vom Kunden individuell konfigurierbar sind und nach Bestellung von der Versuchsanlage weitestgehend autonom gefertigt werden. Die Bestellung und Konfiguration (Rezepturerstellung) des Produkts durch den Endkunden erfolgt durch eine demonstrative Webanwendung (Webshop). In Zukunft könnten neben freikonfigurierbaren auch vorkonfigurierte Produkte erhältlich sein. Wie bereits in den Anforderungen an das System festgehalten, wird nur nach Kundenauftrag produziert. Ein Auftrag besteht aus der Zusammensetzung, sprich der von Kunden gewählten Rezeptur, der gewünschten Menge (z.B. 10 kg) und verschiedenen Adressdaten für den Versand. Die Maschine ist für die Fertigung von minimalen Losgrößen bis ca. 0,5 kg ausgelegt. Der Startimpuls zur Produktion erfolgt nach getätigter Bestellung oder durch die Maschine, welche gespeicherte Aufträge nacheinander abarbeitet. Dies könnte auftreten indem die Nachfrage durch die Abnehmer höher ist als die Zykluszeit der Fertigungsprozesse. KundInnen können aus ca. zehn Zutaten wählen, was sehr viele individuelle Mischungen ermöglicht.

Abbildung 33: Boilies, Quelle: Eigene Darstellung.

Abbildung 34 veranschaulicht zum besseren Verständnis den grundlegenden Fertigungsprozess der Versuchsanlage. Die Produktion von 1 kg Boilies dauert mit der in Abbildung 35 und 36 ersichtlichen Produktionsmaschine ca. 15 Minuten. Das Dosieren und Mischen der einzelnen Zutaten erfolgt automatisch.

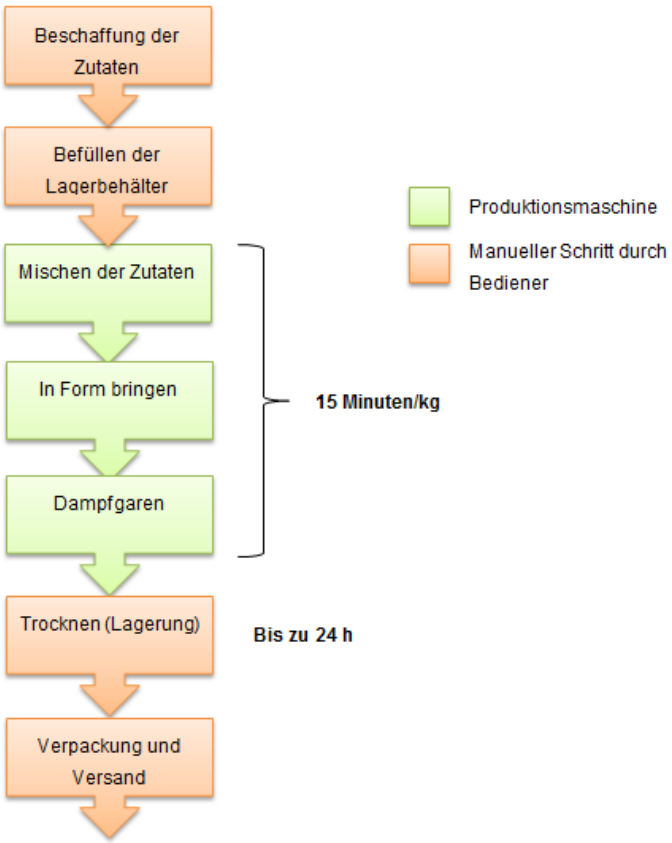


Abbildung 34: Fertigungsprozess der Boilies, Quelle: Eigene Darstellung.

Die Zutaten werden dafür durch motorbetriebe Schnecken in einem Mischbehälter gefördert. Dieser Behälter ist auf drei Wägezellen gelagert. Mit Hilfe der Gewichtsmessung kann die Dosierung auf bis zu 5 g genau erfolgen. Nach der erfolgten Dosierung der Zutaten wird anschließend der Teig trocken vermischt und mit Wasser befeuchtet. Das darauf folgende in Form bringen des Teiges, sowie das Dampfgaren wird ebenfalls automatisch von der Maschine ausgeführt. Die Masse wird durch die Extruder-Öffnung des Mischbehälters gepresst und somit verdichtet. Mit der in Abbildung 35 ersichtlichen Trennvorrichtung wird das Ausgangsmaterial in einen sog. Boilie-Roller befördert, dieses Gerät formt den Teig kugelförmig. Die Teigkugeln werden anschließend in der Kochschnecke bei ca. 85 °C schonend dampfgegart und durch eine Spirale weiterbefördert. Die Dampferzeugung erfolgt mit einem handelsüblichen Dampferzeuger, wobei auf die Innentemperatur der Schnecke geregelt wird (Zweipunktregelung). Die abschließende Trocknung der gekochten Teigkugeln dauert bis zu 24 h. Der Versand und die Verpackung erfolgt im jetzigen Ausbaustadium noch manuell.

Die Steuerung der Anlage wird von einer Beckhoff CX9020 übernommen. Diese SPS bietet eine DVI-Schnittstelle an, dadurch kann die Prozessvisualisierung einfach auf einem angeschlossenen Bildschirm angezeigt werden. Deshalb wird als Benutzerschnittstelle die steuerungsinterne Visualisierung genutzt. Die Bedienung erfolgt hier jedoch mittels Bildschirmtastatur und Maus (siehe Abbildung 36). Die Benutzeroberfläche an der Anlage ermöglicht die Bedienung einzelner Komponenten (Handbetrieb), das Starten und Stoppen von Prozessschritten und stellt darüber hinaus eine grundlegende Rezepturverwaltung zur Verfügung. Zusätzlich besteht die Möglichkeit weitere anlagenspezifische Einstellungen vorzunehmen, beispielsweise können hier Korrekturfaktoren für die Dosierung oder Überwachungszeiten für den Prozess editiert werden.

Abbildung 35: Versuchsanlage, Quelle: Eigene Darstellung.

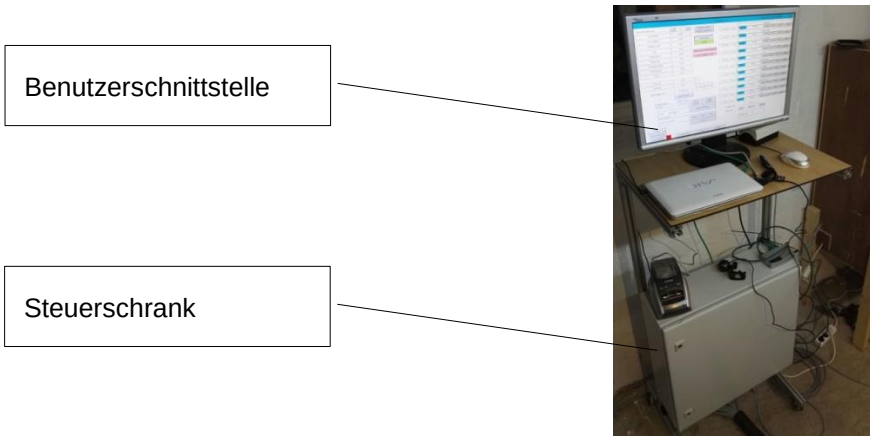


Abbildung 36: Steuerung der Versuchsanlage, Quelle: Eigene Darstellung.

6.2 Webanwendungen

Nach der kurzen Vorstellung der Versuchsanlage wird im folgenden Kapitel der grundsätzliche Aufbau und die Funktionsweise der beiden implementierten Webanwendungen (Webshop und Monitoring-Ansicht) erläutert. Diese Webseiten werden wie bereits in Kapitel 5 näher behandelt, auf einem gemieteten Webspace gespeichert. Aus diesem Grund wird zunächst kurz die Erstellung und Oberfläche des Webspaces vorgestellt.

6.2.1 Aufbau des Webspaces

Die Webanwendungen sollten nach dem für das Produktionssystem in Kapitel 5.5 ausgewählten Konzepts auf einem Webspace ausgeführt werden. Viele Betreiber bieten Pakete mit beinhalteter Domain und Webspace an. Für die Demonstration wird ein Angebot von one.com genutzt. Vor allem aus wirtschaftlicher Sicht ist dieser zu bevorzugen, da das erste Jahr des Services gratis angeboten wird. Auch die weitere Benutzung ist sehr kostengünstig, es fallen für das kleinste Packet - eine Domain, eine Datenbank und 15 GB Speicherplatz - nur jährliche Kosten von rund 10 € an. Zusätzlich bietet one.com auch weitere, für den sicheren Betrieb des Systems relevante Funktionen an. Die Zertifizierung des Webservers und die Unterstützung von SSL sind gegeben. Die in Kapitel 5.2 geforderte Verwendung von HTTPS ist daher ebenfalls möglich. Als serverseitige Skriptsprache kommt hier PHP zum Einsatz, andere Skriptsprachen werden vom bereitgestellten Server nicht unterstützt. Als Datenspeicher stellt one.com eine MySQL-Datenbank zur Verfügung.

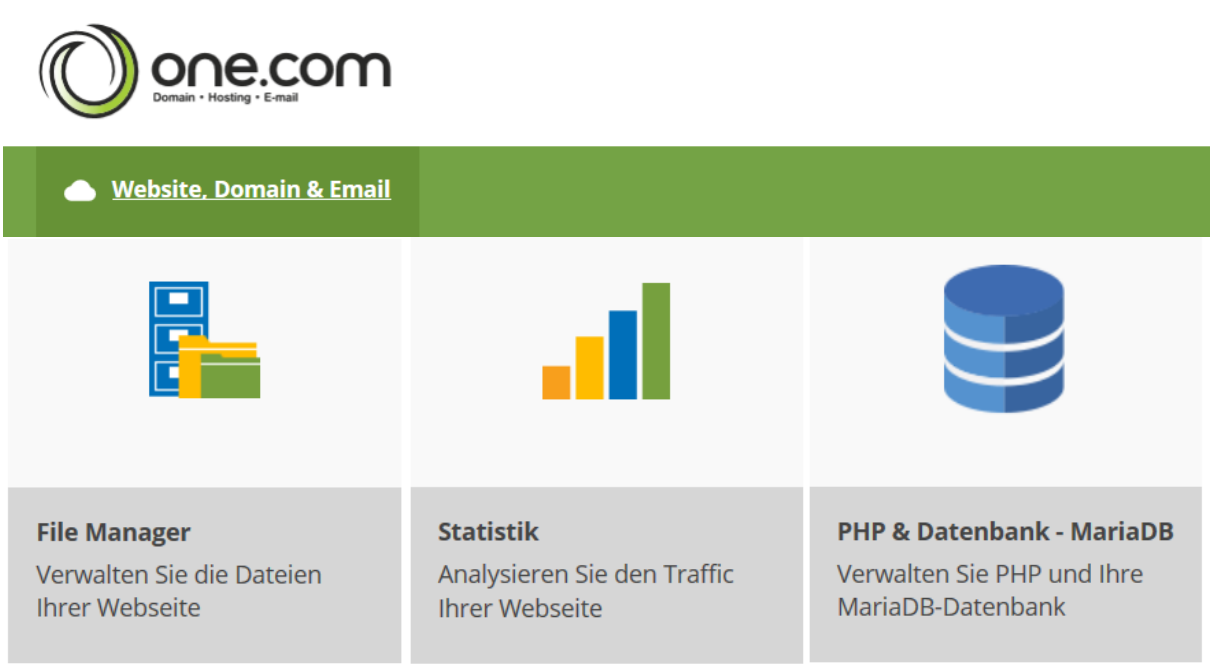


Abbildung 37: Control Panel von Webhosting-Anbieter, Quelle: Eigene Darstellung.

Für den Anwender ist der Webserver nicht direkt sichtbar, bereitgestellte Funktionalitäten sind über ein Webportal nach erfolgreicher Authentifizierung zugänglich. Abbildung 37 zeigt die Weboberfläche des Webspaces. Hier kann beispielsweise die Besucherstatistik verfolgt oder auf die Webdatenbank mit der phpMyAdmin-Oberfläche (ersichtlich in Abbildung 38) zugegriffen werden. Die Verwaltung der Webseiten

und auch die Bearbeitung (online) erfolgt mit dem sog. File-Manager. Die anzuzeigenden Webseiten oder PHP-Skripts können einfach hochgeladen werden und sind sofort über die URL erreichbar (Domain und Filename). Als serverseitige Skriptsprache ist neben dem bewerten PHP 5.4 auch die neueste Version PHP 7 verfügbar.

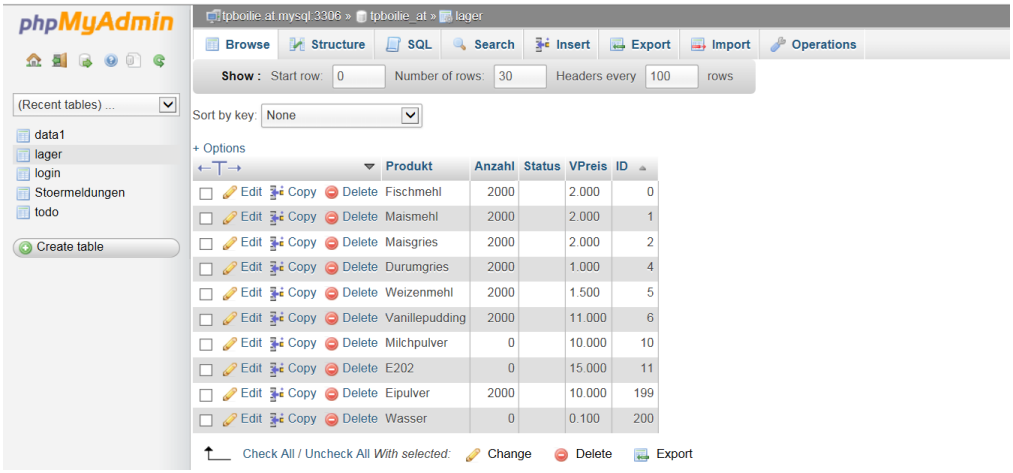


Abbildung 38: phpMyAdmin als Konfigurationsansicht der Datenbank, Quelle: Eigene Darstellung.

6.2.2 Demonstrativer Webshop

Der in Abbildung 42 ersichtliche demonstrative Webshop dient in erster Linie zur Konfiguration und Bestellung des Produkts. Kunden können hier die Mischung ihres Boilies zusammenstellen. Ein Bezahlservice (siehe Kapitel 3.5) wurde derzeit noch nicht implementiert. Dennoch wird hier gezeigt, wie die Daten des Kunden von der clientseitig aufgerufenen Webanwendung in die zentrale Datenbank gespeichert werden.

6.2.2.1 Login-Oberfläche und Benutzeranmeldung

Je nach eingeloggten Benutzern sind verschiedene Registrierkarten und Funktionen ersichtlich. Bevor Waren bestellt (in die Datenbank als Bestellung gespeichert), die Monitoring-Ansicht oder die Lagerverwaltung benutzt werden können, muss nach der Registrierung ein Login erfolgen.

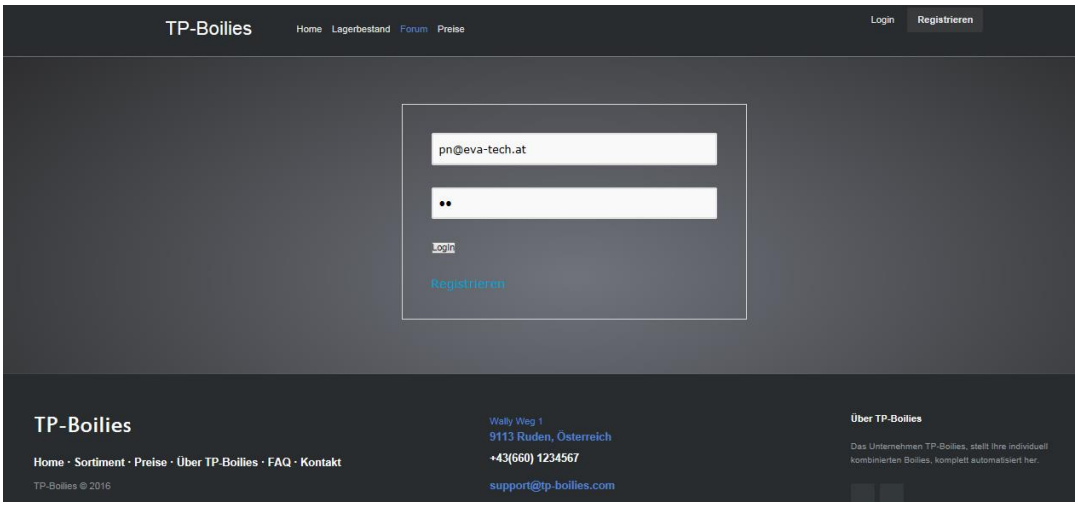


Abbildung 39: Login-Seite, Quelle: Eigene Darstellung.

Beispielsweise wird erst nach erfolgreicher Anmeldung das PHP-Skript nach Betätigung des Bestellen-Buttons vollständig ausgeführt. Ist kein Benutzer angemeldet wird durch die Ausführung des in Abbildung 40 ersichtlichen Codes wieder die Login-Seite (Siehe Abbildung 39) aufgerufen. Außerdem kann die Login-Seite mittels Button im Header-Bereich der Webseiten erreicht werden.

Laut den Anforderungen (Kapitel 5.1) und den Sicherheitsbedenken in Kapitel 5.2 sind eine Benutzeranmeldung sowie die Verwendung einer verschlüsselten Verbindung zum Webserver notwendig. Die Webseiten im Webspace sind jedoch alle als HTTP und HTTPS (nach Aktivierung) aufrufbar.

Abbildung 40 zeigt die Weiterleitung auf die HTTPS-Seite um sicherzustellen, dass nur die gesicherte Seite aufrufbar ist. Hierfür wird mit PHP abgefragt ob der Server HTTPS unterstützt. Ist die Abfrage positiv wird der angefragte Link (REQUEST_URI) mit https:// erweitert, mit dem Befehl header aufgerufen und das Skript beendet. Dieser Codeausschnitt befindet sich neben dem Login auf allen aufrufbaren Webseiten.

```php
<?php
// Weiterleiten auf HTTPS - mit PHP
if (! isset($_SERVER['HTTPS']) or $_SERVER['HTTPS'] == 'off' ) {
    $redirect_url = "https://" . $_SERVER['HTTP_HOST'] . $_SERVER['REQUEST_URI'];
    header("Location: $redirect_url");
    exit();
}
?>
```

Abbildung 40: Weiterleitung auf HTTPS-Seite, Quelle: Eigene Darstellung.

Die Registrierung eines Benutzers erfolgt ebenfalls mit einem HTML-Formular, welches mithilfe eines PHP-Skripts die Anmeldeinformationen in der Datenbank ablegt. Passwörter werden nicht im Klartext in die Datenbank gespeichert sondern als Hash-Code. Dies kann durch die Verwendung der PHP-Methode password_hash automatisch geschehen. Diese Funktion generiert einen 60 Zeichen langen Hash aus dem Passwort und fügt dem Klartext noch ein weiteres Zufallszeichen (salt) hinzu, um eine Entschlüsselung durch Dritte noch zusätzlich zu erschweren (Beispielweise durch Passwortversuche mit bekannten Hash-Codes durch leistungsstarke Rechner). Auch Betreiber können somit die Passwörter der Benutzer in der Datenbank nur als Hash betrachten. Benutzername und Email-Adresse des Benutzers sind als Klartext hinterlegt. Der Vergleich des in der Datenbank hinterlegten Hash-Codes mit der Eingabe im Zuge der Anmeldung erfolgt im Login-Skript (Oberfläche siehe Abbildung 39) mittels der password_verify-Funktion. War die Anmeldung erfolgreich werden Benutzer auf die Hauptseite des Webshops weitergeleitet.

Zusätzlich zur bereits erwähnten Weiterleitung an die HTTPS-Adresse wird bei Aufruf aller mittels Authentifizierung geschützter Webseiten (z.B. Monitoring) wieder mit PHP abgefragt ob ein Benutzer angemeldet ist. Hiermit können Funktionen (z.B. Ausführung von Skriptteilen) oder der Zugriff auf die Seite skaliert werden. Nur gewisse Benutzer erhalten somit Zugriff. Diese Abfrage mit beispielhafter Weiterleitung an die Login-Seite wenn kein Benutzer erfasst ist, wird in Abbildung 41 dargestellt. Außerdem wird mit der Methode session_start eine Browsersitzung (Session) gestartet bzw. reaktiviert. Dadurch ist es möglich webseitenübergreifend Variablen für jeden Client extra zu speichern und auszutauschen. Beispielsweise enthält das $_SESSION Objekt die user Variable welche serverseitig zur Laufzeit jederzeit verändert werden kann (Wechsel des Benutzers im Login). Zusätzlich speichert der

Browser automatisch die eingegeben Passwörter. Besonders hervorzuheben ist, dass die Session vor einer Ausgabe von HTML-Code gestartet werden muss.

```php
<?php
// Session starten
if(session_status()!=PHP_SESSION_ACTIVE) session_start();
// Abfrage ob user angemeldet
if(isset($_SESSION['user'])=="")
{
    header("Location: login.php");exit;
}
?>
```

Abbildung 41: Weiterleitung auf Login, Quelle: Eigene Darstellung.

6.2.2.2 Benutzeroberfläche für die Produktkonfiguration und Auftragsgenerierung

Die in Abbildung 42 ersichtliche Home-Webseite, erreichbar unter der URL https://tpboilie.at/TP/index.php ist für alle Benutzer ersichtlich und dient der Rezepturgestaltung. Erst nach erfolgreichen Login (bzw. Registrierung) ist erst der volle Funktionsumfang der Anwendung abrufbar. Der Seiteninhalt (Zutaten und Preise) wird ebenfalls mit einem PHP-Skript vom Server bei Seitenaufruf generiert. Im Hintergrund sind die Zutaten (Name, Preis etc.) in einer Datenbanktabelle gespeichert. Bei jeder Aktualisierung der Seite (z.B. F5-Taste) werden die aktuell verfügbaren Zutaten von der Datenbank gelesen und mit PHP-Ausgabe in einem HTML-Formular angezeigt. Die Ausgabe der Daten von der Datenbank sind jedoch Teil des nächsten Kapitels.

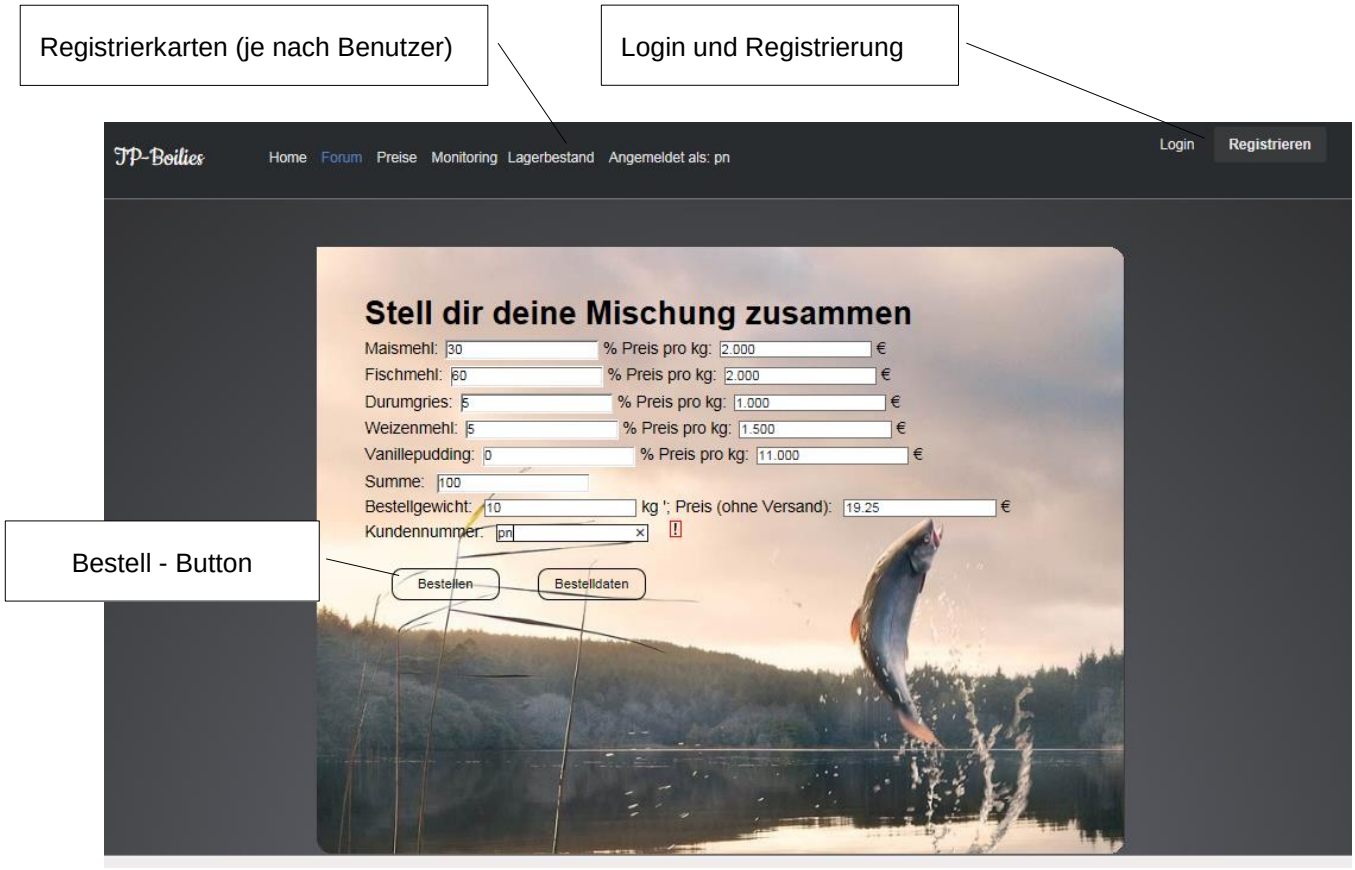


Abbildung 42: Webshop-Anwendung, Quelle: Eigene Darstellung.

Die Eingaben der gewünschten Zusammensetzung (in Prozent) werden einerseits am Client mit einem JavaScript errechnet (Kontrollsumme) und anderseits erfolgt eine zusätzliche Prüfung der nach Button-Click mittels HTTP-POST (siehe Kapitel 3.4.1) gesendeten Daten im aufgerufenen PHP-Skript. Zusätzlich sind alle Eingabefelder des HTML-Formulars als numeric definiert und erlauben somit nur Zahleneingaben. Die Übertragung des Formulars erfolgt verschlüsselt mittels HTTPS. Besonderes Augenmerk wird hier neben der Entfernung von Sonderzeichen (Möglichkeit einer Injektion durch Codeeingabe im Eingabefeld) und auf die Plausibilität der Rezeptur (Kontrollsumme) sowie der Gewichtsangabe gelegt. Der Preis der Ware wird clientseitig von JavaScript errechnet und ebenfalls geprüft. Ausschließlich Bestellungen mit einer Kontrollsumme von 100 und einer Bestellmenge zwischen 0,5 und 100 kg werden in die Datenbank als Auftrag hinterlegt. Andernfalls wird vom aufgerufenen PHP-Skript eine Fehlerbeschreibung ausgegeben und die Ausführung vor der Speicherung der Formulardaten in die Datenbank abgebrochen. Außerdem werden die Werte vor der Speicherung der Daten im PHP mittels mysqli_real_escape_string von möglichen Sonderzeichen befreit.

Der Preis des gewünschten Produkts wird in einem JavaScript zur Laufzeit errechnet und ist somit den Kunden jederzeit bekannt. Zur Vermeidung unerwünschter Eingaben werden wie bereits erwähnt, die Eingabedaten der Mischungsverhältnisse (in Prozent) korrigiert. Hier sind nur numerische Werte zwischen 0 und 100 erlaubt. Andere Eingaben werden durch JavaScript nach Eingabe gelöscht. Zur Vermeidung von Preismanipulationen sind sämtliche Preisangaben (Einzelpreis und Summe) in HTML als readonly definiert.

Abbildung 43 zeigt die Speicherung der Bestelldaten in die Datenbank im aufgerufenen PHP-Skript nach Betätigung des Bestellen-Buttons.

```
//Auftragsdaten in Datenbank speichern
//mit Datenbank verbinden
$link = mysqli_connect('localhost', 'my_user', 'my_password', 'my_db');

// Abfrage ob Verbindung und Anmeldung nicht erfolgreich war
if($link === false)
{
    die("Datenbankfehler:" . mysqli_connect_error());
    exit();
}
// Datenbank auswählen
mysqli_select_db($link,"");
//Weitere Daten vom Formular mappen
$Bezahlt = 1; // Testbetrieb mit Bezahlung = 1
$Gewicht = mysqli_real_escape_string($link, $_POST['Gew']);
$Kunde = mysqli_real_escape_string($link, $_POST['KundenID']);
$Preis = mysqli_real_escape_string($link, $_POST['Preis']);
$Status = "ToDo";
// SQL-Befehl zum Anlegen eines Datensatzes im Table todo (Aufträge)
$sql = "INSERT INTO `todo` (`ID`, `Zutat1`, `MV1`, `Zutat2`, `MV2`, `Zutat3`, `MV3`, `Zutat4`, `MV4`,
    `Zutat5`, `MV5`,`Zutat6`, `MV6`,`Zutat7`, `MV7`,`Zutat8`, `MV8`,`Zutat9`, `MV9`,`Zutat10`, `MV10`
    ,`Zutat11`, `MV11`, `Bezahlt`, `Gewicht`, `Kunde`, `Preis`, `Status`)
VALUES ('','$produkt[0]', '$anzahl[0]', '$produkt[1]','$anzahl[1]','$produkt[2]', '$anzahl[2]','$produkt[3]'
    ,'$anzahl[3]','$produkt[4]','$anzahl[4]','$produkt[5]','$anzahl[5]','$produkt[6]','$anzahl[6]'
    ,'$produkt[7]','$anzahl[7]','$produkt[8]','$anzahl[8]','$produkt[9]','$anzahl[9]','$produkt[10]'
    ,'$anzahl[10]','$Bezahlt','$Gewicht','$Kunde','$Preis','$Status')";
// Abfrage ob Zugriff erfolgreich war
if(mysqli_query($link, $sql)) echo "Erfolgreich bestellt.";
else echo "Datenbankfehler: $sql. " . mysqli_error($link);
// Verbindung schließen
mysqli_close($link);
```

Abbildung 43: Speichern von Aufträgen in die Datenbank, Quelle: Eigene Darstellung.

Die neuere mysqli-Funktion (alte Funktion mysql) zur Verbindung mit der Mysql-Datenbank wird hier im prozeduralen Stil verwendet. Ebenso könnte eine objektorientierte Schreibweise zur Verwendung kommen. Mit der in Abbildung 43 gezeigten Funktion mysqli_connect wird die Datenbankanmeldung und Verbindung ausgeführt, dazu ist wie bereits in Kapitel 3.3.3 geschildert eine Anmeldung mit Benutzername und festgelegtem Passwort erforderlich. Die in Abbildung 43 ersichtlichen Zugangsdaten sind abgeändert und dienen nur zur Information (z.B. my_password). Zusätzlich sind die Datenbanklokalität (z.B. localhost) und der Name der Datenbank anzugeben. War die Verbindung nicht erfolgreich, wird in weiterer Folge eine Fehlermeldung ausgegeben und die Ausführung des Skripts beendet. Optional könnte mittels mysqli_select_db nach der erfolgreichen Verbindung zur Laufzeit eine weitere Datenbank verbunden werden. Nach der weiteren Behandlung der vom Formular der Konfigurationsseite (Zusammenstellung der Rezeptur) mittels HTTP-POST an das Skript gesendeten Daten wird anschließend ein String mit dem SQL-Befehl (z.B. insert), den Namen der Einträge und den übertragenden Daten (Zusammensetzung und Kundendaten) zusammengesetzt. Mit der PHP-Funktion msqli_query wird die Anweisung schließlich ausgeführt. Außerdem erfolgt hier eine Überprüfung des Erfolges der Ausführung. Im Falle eines Fehlers wird auch hier der Benutzer durch die Ausgabe einer Meldung benachrichtigt. Abschließend wird die Verbindung mit der Datenbank geschlossen.

Eine Bezahlfunktion wurde noch nicht implementiert, jedoch könnte die in Kapitel 3.5 gezeigte Interaktion mit dem externen Anbieter hier stattfinden. Wäre die Bezahlung erfolgreich könnte die Variable $Bezahlt auf eins gesetzt werden. Die Kundendaten setzen sich derzeit nur aus einer Kundennummer zusammen, jedoch könnten ohne großen Aufwand weitere von Kunden im Formular eingegebene Daten übertragen werden. Eine weitere denkbare Variante ist die Nutzung der vorhin erwähnten Session-Variablen für eine Benutzerabfrage (siehe Abbildung 41). Ist der Benutzer nicht angemeldet (Session-Variable ist leer) wird das Skript beispielsweise abgebrochen und wieder auf die Home-Seite (Konfiguration) weitergeleitet. Sind Kunden erfolgreich angemeldet (siehe Login) wird der Benutzername in eine Session-Variable gespeichert. Diese kann abgefragt werden und weitere unter dem Benutzernamen in der Datenbank hinterlegte Kontaktinformationen könnten ausgelesen und in die Auftragsdaten integriert werden. Für die weitere Bearbeitung des in die Datenbank gespeicherten Datensatzes wird ein Eintrag mit dem derzeitigen Status des in diesem Skript erstellten Auftrags hinzugefügt.

6.2.3 Webanwendung für das Anlagen-Monitoring

Wie bereits in der Aufgabenstellung und den Anforderungen an die Applikation spezifiziert ist neben der Shop-Webanwendung eine Ansicht für ein Anlagen-Monitoring zu implementieren. Hier werden die Prozessdaten der Versuchs-Produktionsanlage nur nach Anmeldung mit einem gewissen Benutzernamen (siehe Kapitel 6.2.2.1) dargestellt. Im Gegensatz zur Shop-Anwendung liegt hier der Fokus auf das Lesen von Daten aus der zentralen Datenbank und deren Darstellung. Kapitel 3.4 zeigt mehrere Varianten für den Datenaustausch mit dem Webserver. Der klassische Datenaustausch wird hier durch den Client (Webbrowser) bei Seitenaufbau angestoßen. Er sendet hierbei nur HTTP-Requests (z.B. POST-Methode) bis die Seite vollständig geladen ist. Wird der Inhalt einer Datenbanktabelle mittels Aufruf eines PHP-Skripts (siehe Kapitel 3.3.3) als HTML-Code zurückgegeben, werden Änderungen des Datenbankinhalts nach dem einmaligen Seitenaufbau nicht wahrgenommen. Es handelt sich somit um eine Momentaufnahme. Sollten Daten bei ständig geöffneter Webseite möglichst aktuell sein,

beispielsweise für die Darstellung von Störmeldungen, ist die manuelle Aktualisierung (F5) keine komfortable Option. Mittels AJAX oder Websocket (siehe Kapitel 3.4) kann der Datenaustausch auch ohne ständigen Neuaufbau der Seite erfolgen. Websocket werden jedoch noch nicht von jedem Webserver bzw. Hosting-Anbieter unterstützt. Die Kommunikation mittels Websocket ist beispielsweise beim gewählten Hosting-Anbieter one.com nicht möglich. Deshalb wird zum Nachladen des Seiteninhalts (Störmeldungen) auf AJAX zurückgegriffen. Der Datenstrom wird wie alle erstellten Webseiten mittels HTTPS geschützt. Die gesicherte Übertragung ist auch bei AJAX möglich da mittels JavaScript aufgerufene HTTP-Requests zur Verwendung kommen.

6.2.3.1 Benutzeroberfläche

Vor der Erörterung des Codes der mit JavaScript und PHP implementierten Webanwendung wird zunächst die Oberfläche näher dargestellt. Die in Abbildung 44 ersichtliche Benutzeroberfläche beinhaltet die Anzeige von Störmeldungen, einen editierbare Trend-Ansicht und eine zusätzliche Tabelle welche die im Diagramm selektierten Werte anzeigt. Im Diagramm sind die zwei Kurven Temp1 (Temperaturmessung) und cw1 (Gewichtsmessung)ersichtlich.

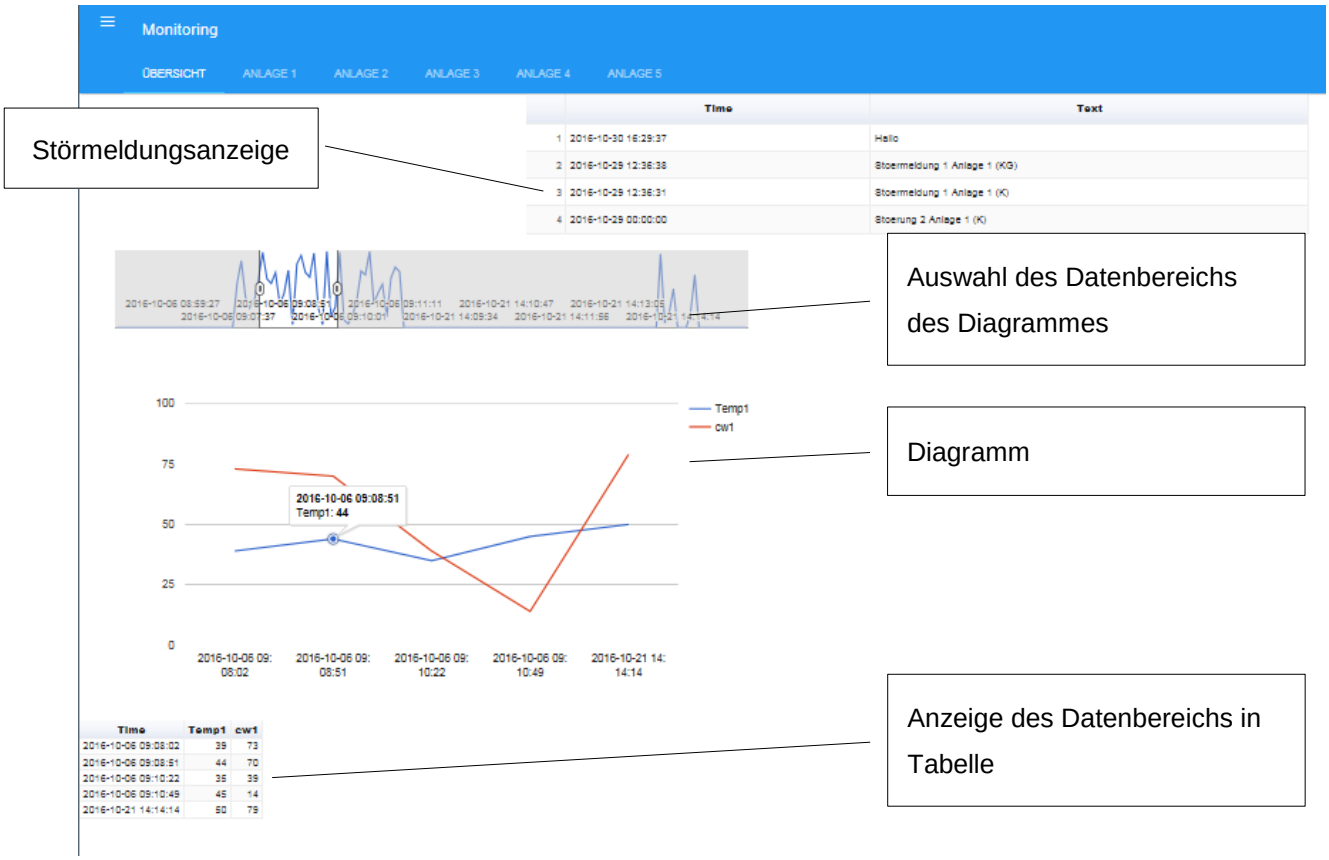


Abbildung 44: Gesamtbild der Monitoring-Ansicht, Quelle: Eigene Darstellung.

Im Gegensatz zur Webshop-Webanwendung wurde hier auf Elemente der Google-Chart-API gesetzt. Besonders für die Visualisierung von Daten bieten diese vom Design vorgefertigten Elemente eine für die Implementierung relativ komfortable und optisch ansprechende Lösung dar. Darüber hinaus bieten die Google-Charts umfangreiche Funktionalitäten an. Beispielsweise kann der Datenbereich des Diagramms (siehe Abbildung 44) mit bereits im Element integrierten JavaScript-Funktionen über die beiden Cursoren des darüber liegenden kleineren Diagramms flexibel verstellt werden. Zusätzlich sind einzelne

Datenreihen ausblendbar und bei Maus-Click wird der aktuelle Wert der Datenreihe in einem kleinem Popup-Fenster dargestellt.

6.2.3.2 Asynchrone Aktualisierung der Störmeldungsanzeige

Die demonstrativen Störmeldungen werden in einer Google-Chart Tabelle (siehe Abbildung 45) dargestellt. Wie bereits erwähnt wird die Tabelle asynchron mittels AJAX mit Daten aus der zentralen Datenbank befüllt. Das Kürzel K im angezeigten Meldetext steht für Kommend (Fehler steht an). Nach der Behebung des Fehlers wird an den Meldetext ein KG (Kommen-Gegangen) zugefügt. Die Quittierung von Meldungen auf der Weboberfläche ist nicht vorgesehen Eine Änderung der Einträge wird somit unmittelbar visualisiert. Aufgrund der Verwendung von AJAX wird der Datenaustausch immer vom Client initiiert. In Gegensatz dazu könnte mittels Websocket, siehe Kapitel 3.4.3 auch nach Verbindungsaufbau ein vom Server angestoßener Datenaustausch erfolgen, beispielsweise bei Wertänderungen.

	Time	Text
1	2016-11-10 18:46:57	Stoerung 2 Anlage 1 (KG)
2	2016-10-29 19:00:00	Stoerung 2 Anlage 1 (K)
3	2016-10-29 12:36:38	Stoermeldung 1 Anlage 1 (KG)
4	2016-10-29 12:36:31	Stoermeldung 1 Anlage 1 (K)

Abbildung 45: Störmeldungstabelle, Quelle: Eigene Darstellung.

Zur Sicherstellung einer unmittelbaren Darstellung von neuen Störmeldungen wird das Auslesen des Datenbankinhalts (Meldungen) zyklisch angestoßen und mit AJAX asynchron nachgeladen. Der Inhalt der Tabelle wird demnach ca. alle 2000 ms aktualisiert. Abbildung 46 zeigt die JavaScript-Funktion welche eine Funktion (updateM) in zyklischen Abständen aufruft. Die zyklische Abarbeitung wird erst nach dem abgeschlossenen Seitenaufbau aufgerufen und gestartet. Dies wird über in der JavaScript-Bibliothek Jquery enthaltenen Funktion $(document).ready abgefragt. Die Google-Chart-API wird ebenfalls asynchron geladen und ist deshalb nicht sofort verfügbar. Deshalb wird mit setOnLoadCallback festgelegt welche Funktion nach dem erfolgreichen Ladevorgang der Google-API ausgeführt wird. Mit setInterval wird anschließend die zyklische Ausführung mit der auszuführenden Funktion und der Zeitdauer in Millisekunden parametriert und gestartet.

```
$(document).ready(function() {
  google.setOnLoadCallback(updateM);
  var refreshId = setInterval(function() {
  updateM();
  }, 2000);
});
```

Abbildung 46: Zyklisches Aufrufen einer Funktion mit JavaScript, Quelle: Eigene Darstellung.

Die zyklisch aufgerufene JavaScript-Funktion updateM beinhaltet nun den eigentlichen Datenaustausch zwischen der Anzeige im Browser und der zentralen Datenbank. Dazu wird bereits in Kapitel 3.4.4 geschilderte XMLHttpRequest verwendet. Die Implementierung ist in Abbildung 47 ersichtlich. Der Request ruft ein PHP-Skript (siehe Abbildung 48) am Webserver auf welches die Daten aus der Datenbank herausgelesen und im einheitlichen JSON-Format herausgibt. Zum Aufrufen des

XMLHttpRequests wird wiederum die $.ajax-Funktion aus der JavaScript-Jquery-Bibliothek verwendet. Außerdem bietet Jquery die Möglichkeit die mittels AJAX (XMLHttpRequest) asynchron geladenen Daten direkt an HTML-Elemente zu koppeln. Dies ist in diesem Fall jedoch nicht zielführend, da nur eine einzelne Zeile (z.B. Ausgabefeld) angesprochen werden kann. Dargestellt in Abbildung 47 wird der verwendeten $.ajax-Funktion als Parameter der Typ des Requests (z.B. POST oder GET), die URL der aufzurufenden Seite (bzw. Skripts) und der erwartete Datentyp der Rückgabewerte angegeben. Zusätzlich könnten Daten an das aufgerufene Skript übertragen werden (z.B. die Datenbankabfrage). Dies ist aber nicht unbedingt zu empfehlen da diese Variablen für alle Clients sichtbar sind. Man könnte so den Namen des Tables bzw. die ganze SQL-Abfrage in der Datenbank leicht mit Debug-Funktionen in Browsern herauslesen. Der Parameter async wird auf true gesetzt um eine asynchrone Abarbeitung des Requests durch den Browser zu genehmigen. Die Benutzeroberfläche wird somit nicht durch die Ausführung blockiert.

```
function updateM() {
//Request
var jsonData = $.ajax({
        type: "POST",
        url: "update.php",
        dataType: "json",
        async: false,
        success: function(jsonData)
            {   // Google-Chart: Daten zuweisen und neu zeichen
                var data = new google.visualization.arrayToDataTable(jsonData);
                var table = new google.visualization.Table(document.getElementById('table_str'));
                table.draw(data, {showRowNumber: true, width: '100%', height: '90%'});
            }
        }).responseText;
    }
```

Abbildung 47: Tabelle asynchron aktualisieren, Quelle: Eigene Darstellung.

War der Request erfolgreich (success) werden die empfangen Daten (jsonData) nun mit der Funktion array.ToDataTable in das Objekt data gespeichert. Es enthält den vom aufgerufenen PHP-Skript (update.php) ausgegebenen Text, der als Array mit zwei Spalten interpretiert wird. Die Verbindung zur HTML-Seite ist über das table-Objekt gewährleistet, welches mit einem HTML-Objekt (table_str) verknüpft wird. Durch Aufrufen der draw-Methode des table-Objekts mit den aktuellen Daten (data) als Inhalt der Tabelle und weiteren optionalen Darstellungsparametern wird die Tabelle neu gezeichnet. Abbildung 48 zeigt das PHP-Skript update.php, welches vom Request aufgerufen wird. Dieses kurze Skript liest Daten aus der Datenbank und gibt sie in JSON-Format aus.

```
$link = mysqli_connect("localhost", "my_user", "my_password","my_db");
$query = "SELECT * FROM Stoermeldungen ORDER BY Time DESC LIMIT 100";
$exec = mysqli_query($link,$query);
$data[] = array('Time','Text');
while($row = mysqli_fetch_array($exec))
{
    $data[] = array($row['Time'],$row['Text']);
}
echo json_encode($data);
mysqli_close($link);
```

Abbildung 48: Daten aus Datenbank auslesen und im JSON-Format ausgeben, Quelle: Eigene Darstellung.

Die Verbindung zur Datenbank wurde bereits in den vorangehenden Kapitel (6.2.2.2) nähe behandelt, wobei hier auf eine zusätzliche Fehlerbehandlung weitestgehend verzichtet wurde, da bei Fehlern die Google-Chart-Tabelle nur eine Fehlermeldung (Falsches Datenformat) ausgibt oder nicht gezeichnet wird. Hier kommt wiederum die in Kapitel 3.2 erwähnte spezifische Fehlertoleranz der Browser zur Geltung.

Die Datenbankabfrage (in die PHP-Variable $query gespeichert) gibt die letzten 100 Fehlermeldungen nach der Zeit geordnet aus (aktuellster Eintrag zuerst). Die aus der Datenbank ausgelesen Daten ($row) werden in die Variable $data geschrieben, wobei nur die Spalten Time und Text von Relevanz sind. Die Spaltenüberschriften welche in der in Abbildung 45 ersichtlichen Tabelle der Monitoring-Ansicht angezeigt werden sind in vorhinein zu definieren und somit die ersten Einträge des $data Arrays. Schlussendlich wird die Arrayvariable in das JSON-Format konvertiert (json_encode) und ausgegeben. Im Gegensatz zur asynchronen Aktualisierung der Tabelle zur Anzeige der Störmeldungen wird das in Abbildung 49 ersichtliche Diagramm nur einmalig beim Aufbau der Webseite mit Daten befüllt. Wie bereits im Zuge der Störmeldungstabelle geschildert wird hier das JavaScript-Element mit der Funktion array.ToDataTable mit den anzuzeigenden Daten in Form eines Arrays befüllt. Jedoch muss hier nicht zwingend auf ein externes PHP-Skript zurückgegriffen werden. Der PHP-Code bestehend aus Datenbankverbindung, SQL-Abfrage und der Ausgabe im geeigneten Format kann innerhalb der JavaScript-Funktion erfolgen. Das Chart besteht aus zwei miteinander verknüpften Elementen, der Trend-Control (oben) und dem eigentlichem Diagramm (unten) mit den beiden Kurven Temperatur (temp1) und der Gewichtsmessung (cw1). Über den Cursor der Trend-Control kann der angezeigte Bereich dynamisch verstellt werden. Eine ständige Aktualisierung der Daten ist hier nicht zielführend da die Oberfläche des Elements bei jedem draw-Befehl neu gezeichnet wird und die temporären Einstellungen verworfen werden. Außerdem ist diese Anzeige für die Visualisierung von historischen Daten gedacht.

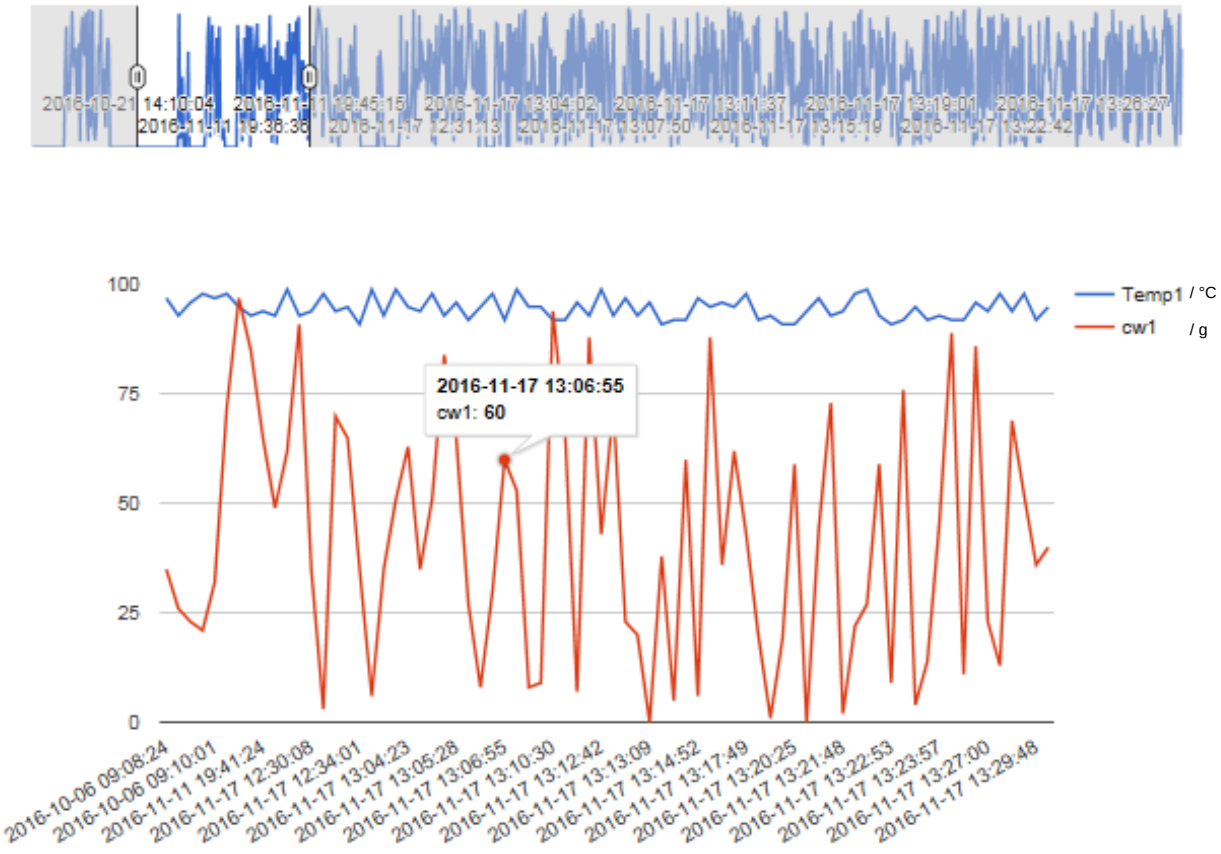


Abbildung 49: Diagramm zur Prozessdatendarstellung, Quelle: Eigene Darstellung.

6.3 Datenkopplung zwischen Produktion und Webdatenbank

Nach der Vorstellung der implementierten Webanwendungen zur Darstellung der gespeicherten Prozessdaten sowie der Auftragsgenerierung (Ablegen der Rezepturen in der Datenbank) wird nun die eigentliche Datenkopplung zwischen der Wrapper-Applikation, der Steuerung der Produktionsanlage und der zentralen Datenbank näher behandelt. Außerdem wird die entwickelte Wrapper-Applikation vorgestellt, kurz gezeigt wie die Aufträge auf der SPS verarbeitet und Variablen über OPC-UA angesprochen werden. Abbildung 50 zeigt die prinzipielle Funktionsweise des Datenaustausches zwischen der Wrapper-Applikation, der Produktionsanlage und der zentralen Datenbank am Webspace. Wie bereits in der Konzeption (siehe Kapitel 5.3.4 und 5.4.4) erwähnt basiert dieser Datenaustausch auf dem Aufruf von PHP-Skripten am Webserver mit der HTTP-POST-Methode. Die Wrapper-Applikation übernimmt dabei das Mapping zwischen den OPC-Nodes (SPS-Variablen) und den Daten aus der Datenbank. Hier sind jedoch zwei Einsatzfälle zu unterscheiden. Einerseits sind Prozessdaten im Zuge des Monitorings zyklisch vom OPC-UA Server zu lesen und in die Datenbank zu übertragen. Andererseits werden Auftragsdaten zyklisch von der Datenbank abgefragt und im Falle eines neuen Auftrages an die SPS-übertragen und der Ablauf (Produzieren der Boilies) gestartet.

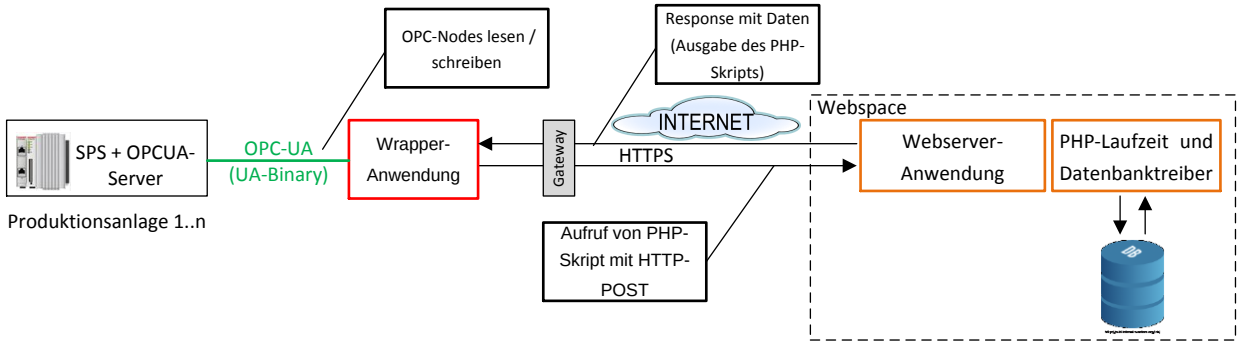


Abbildung 50: Prinzipielle Funktionsweise des Datenaustauschs, Quelle: Eigene Darstellung.

6.3.1 Benutzeroberfläche der Wrapper-Applikation

Bevor näher auf den Datenaustausch zwischen der Produktion und der Web-Datenbank eingegangen wird, müssen zunächst die grundsätzliche Konfiguration und Funktionsweise sowie die Benutzeroberfläche der implementierten Wrapper-Applikation von Interesse sein. Diese Anwendung dient vor allem dem Mapping der gelesenen Auftragsdaten mit den OPC-UA-Nodes und der Weitergabe der gelesenen Prozessdaten. Für den Betreib des Produktionssystems ist diese Anwendung nur einmalig zu konfigurieren und freizugeben. Prozessdaten werden im Betrieb zyklisch in die Datenbank gespeichert. Die Auftragsabarbeitung wird ebenfalls von der Produktionsmaschine angestoßen (genauer Ablauf siehe Kapitel 6.4).

Die Wrapper-Anwendung ist in C# realisiert, besonders wegen der Verfügbarkeit einer OPC-UA- Client-API die den .NET-Kommunikations-Stack beinhaltet, darauf wird im Kapitel 6.3.1.3 näher eingegangen. Als Projektierungstool kommt Visual Studio 2010 zum Einsatz.

Abbildung 51 zeigt das Ausgangsbild der Windows-Form Anwendung. Hier erfolgt die grundsätzliche Bedienung und Konfiguration der Applikation. Weitere Fenster beispielsweise für die Monitoring-Ansicht oder für grundsätzliche Einstellungen sind hier ebenfalls aufrufbar. Der Namespace des verbundenen OPC-Servers ist in einer Browse Control ersichtlich. Hier kann, wie bereits in Kapitel 4.3 erläutert, mittels der vom Server zur Verfügung gestellten Browse-Methode der Adressraum des Servers und alle darin beinhalteten Modelle und Nodes beobachtet werden. Die URL des Servers ist einerseits in einer Konfigurationsdatei oder in der Textbox im Kopfbereich editierbar. Der aktuelle Status der OPC-Verbindung ist in der Fußzeile des Fensters ersichtlich. Außerdem wird hier der von der Datenbank ausgelesene Auftrag in einer DatagridView (Auftragsdaten) dargestellt, welcher nur nach Freigabe (Button From Web) an den OPC-UA-Server und in weiterer Folge auf die Steuerung weitergegeben wird. Mit diesem Button ist demnach die automatische Auftragsabarbeitung (Form Web) zu Starten bzw. zu unterbrechen. Die OPC-Nodes welche zur Interaktion mit der Produktionsanlage notwendig sind, werden ebenfalls in einer separaten Liste (Control) dargestellt. Zusätzlich wird der aktuelle Status der Produktionsanlage, bzw. der aktive Schritt des Produktionsablaufs als Text angezeigt. Durch Betätigung der Stopptaste kann der aktuell bearbeitete Auftrag beendet und ein neuer gestartet werden. Diese Funktionalität (From Web und Stopp) sind ebenfalls auf der lokalen Visualisierung, welche bereits in Kapitel 6.1 kurz vorgestellt wurde verfügbar. Daher kann die Bedienung einerseits am PC, an welchem die Wrapper-Applikation ausgeführt wird, oder an der Benutzerschnittstelle der SPS (siehe Abbildung 36) erfolgen.

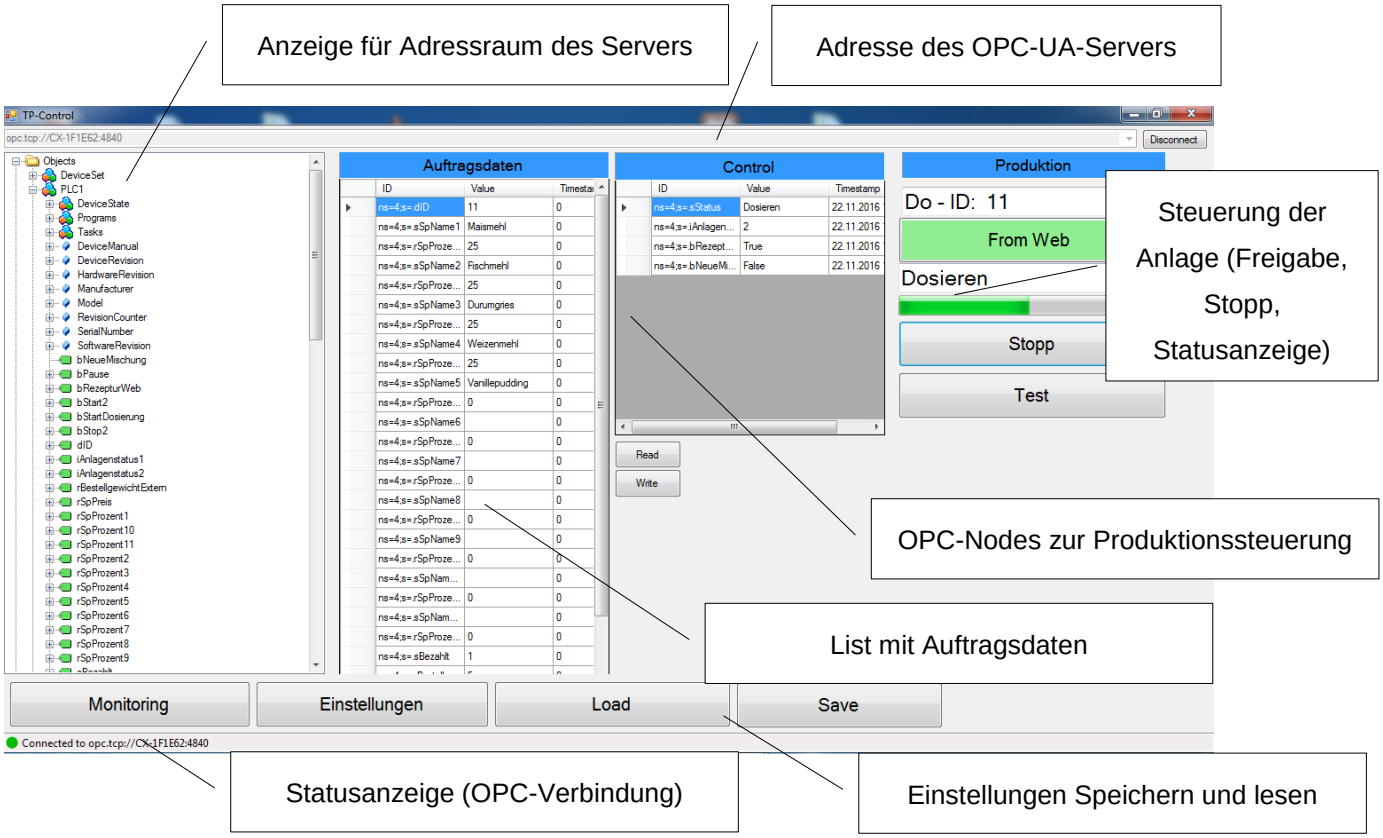


Abbildung 51: Grundbild der Wrapper-Applikation, Quelle: Eigene Darstellung.

Neben den aktuellen Auftragsdaten (Spalte Values) ist ebenfalls das Mapping der von der Web-Datenbank gelesen Daten mit den OPC-Nodes in der Liste Auftragsdaten ersichtlich. Die Reihenfolge der Variablen ist jedoch durch die Formatierung der Ausgabe der aufgerufenen PHP-Skripts bzw. durch den Rückgabewert der POST-Methode vorgegeben. Diese Einstellung muss jedoch nur einmalig erfolgen. Der Datenaustausch und das Mapping sind jedoch in den weiteren Kapiteln näher angeführt. Eine Änderung der Reihenfolge oder der ID des OPC-UA-Nodes ist einerseits direkt in der Liste (wenn keine zyklische Aktualisierung der Auftragsdaten erfolgt) oder in der zugrundeliegenden Textdatei möglich. Zur Sicherstellung der Flexibilität dieser Anwendung sind die Konfigurationen der ersichtlichen Listen (z.B. Aufträge, Monitoring) in jeweils einer eigenen Textdatei hinterlegt. Die Verwendung der Textdateien als Datenspeicher ermöglicht einerseits eine zur Laufzeit editierbare und remanente (nach Neustart der Applikation verfügbare) Datenquelle für die Einstellungen. Sie müssen demnach nicht fix im Programmcode festgelegt werden, sind leicht zu ändern und auch übertragbar. In diesem Fall wurde eine gewöhnliche Textdatei als Datenspeicher gewählt, auch eine Variante mit XML-Files wäre denkbar. CSV-Dateien (mit deutscher Windows-Spracheinstellung) scheiden jedoch als Datenspeicher aus, da die OPC-UA-Nodes Semikolons beinhalten.

Bei Betätigung des Load-Buttons (siehe Abbildung 51) oder bei jedem Start der Applikation wird der Inhalt der Textdatei in die Liste geschrieben. Wird ein Listeneintrag während der Laufzeit in den Listen (in einer DatagridView) geändert dient der nebenstehende Button Save zum Speichern der Daten in die zuvor erwähnte Konfigurationsdatei.

Die globalen Einstellungen sind in einem weiteren Fenster editierbar. Abbildung 52 zeigt die Einstellungen der Wrapper-Applikation. Hier werden die Pfade der Konfigurationsdateien, die Adresse des OPC-UA-Servers sowie die URLs der aufzurufenden PHP-Files am Webserver festgehalten. Wie bereits erwähnt sind auch diese Einstellungen im Fenster editierbar und können zusätzlich mithilfe der Buttons Load und Save von der Textdatei (z.B. INI-File) gelesen bzw. in diese geschrieben werden. Beim erstmaligen Ausführen der Applikation muss der Pfad des dieser Liste zugrundeliegenden Textfile (INI-File) möglicherweise angepasst werden. Dies kann im Textfeld neben der DatagridView geschehen.

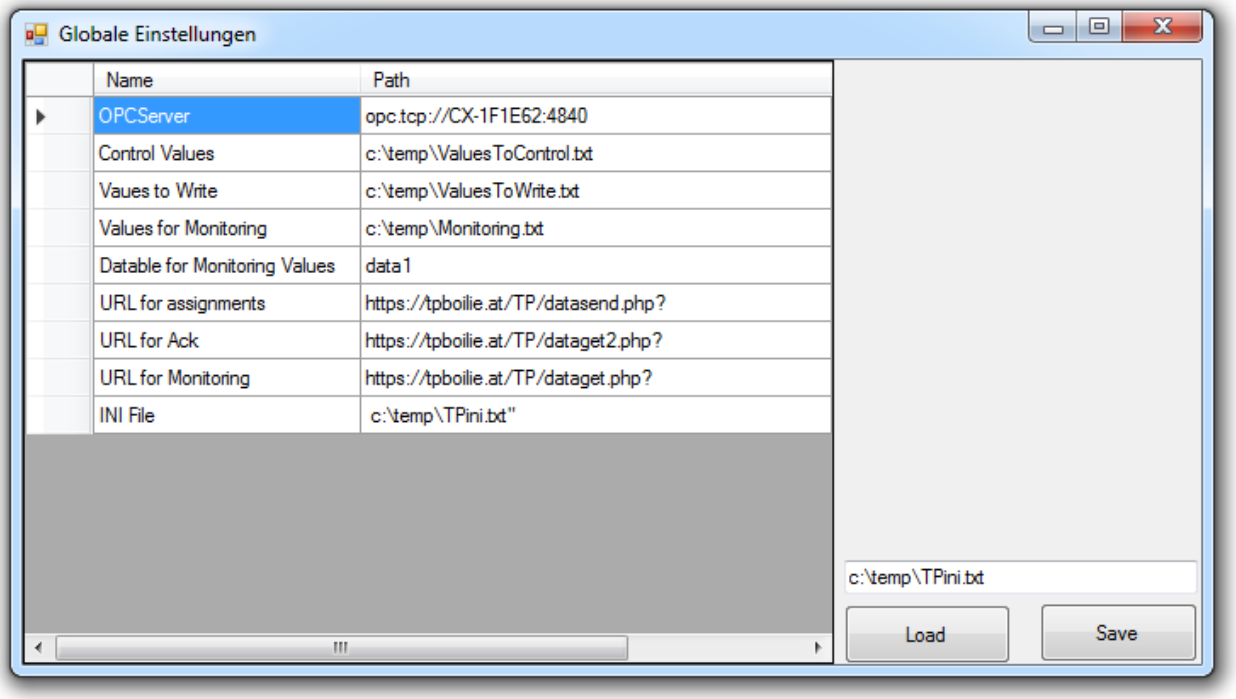


Abbildung 52: Einstellungsfenster der Wrapper-Applikation, Quelle: Eigene Darstellung.

Das Mapping der Prozessdaten, sprich die Zuweisung der ID der OPC-UA-Nodes von Variablen der SPS auf die an die Webdatenbank zu übertragenden Werten findet in einem eigenem Fenster (siehe Abbildung 53) statt. Die vom OPC-Server gelesenen Prozessdaten werden nach der Freigabe durch den Monitoring Button zyklisch mittels HTTP-POST-Methode übertragen, wobei aus Gründen der Benutzerfreundlichkeit diese Freigabe beim Starten des Fensters bereits gegeben ist und somit die Übertragung sofort beginnt. Die Alarme sind ebenfalls in einer eigenen Liste dargestellt, wobei hier die Spalten um die Einträge Zustand und Alarmtext erweitert wurden. Die Übertragung von Alarmtexten erfolgt im Gegensatz zum Monitoring von Messwerten nur nach einer Wertänderung des OPC-Nodes. Je nach Zustand wird ein zusätzlicher Text angehängt, beispielsweise bei einer neu auftretenden Meldung (Meldung1 = true) wird der Zusatz K für Kommend angehängt. Beiden Listen liegen ebenfalls Textdateien als Datengrundlage zugrunde. Die Einträge (z.B. Alarmtext oder Node-ID) sind wiederum direkt in der jeweiligen Listenanzeige oder im Textfile editierbar.

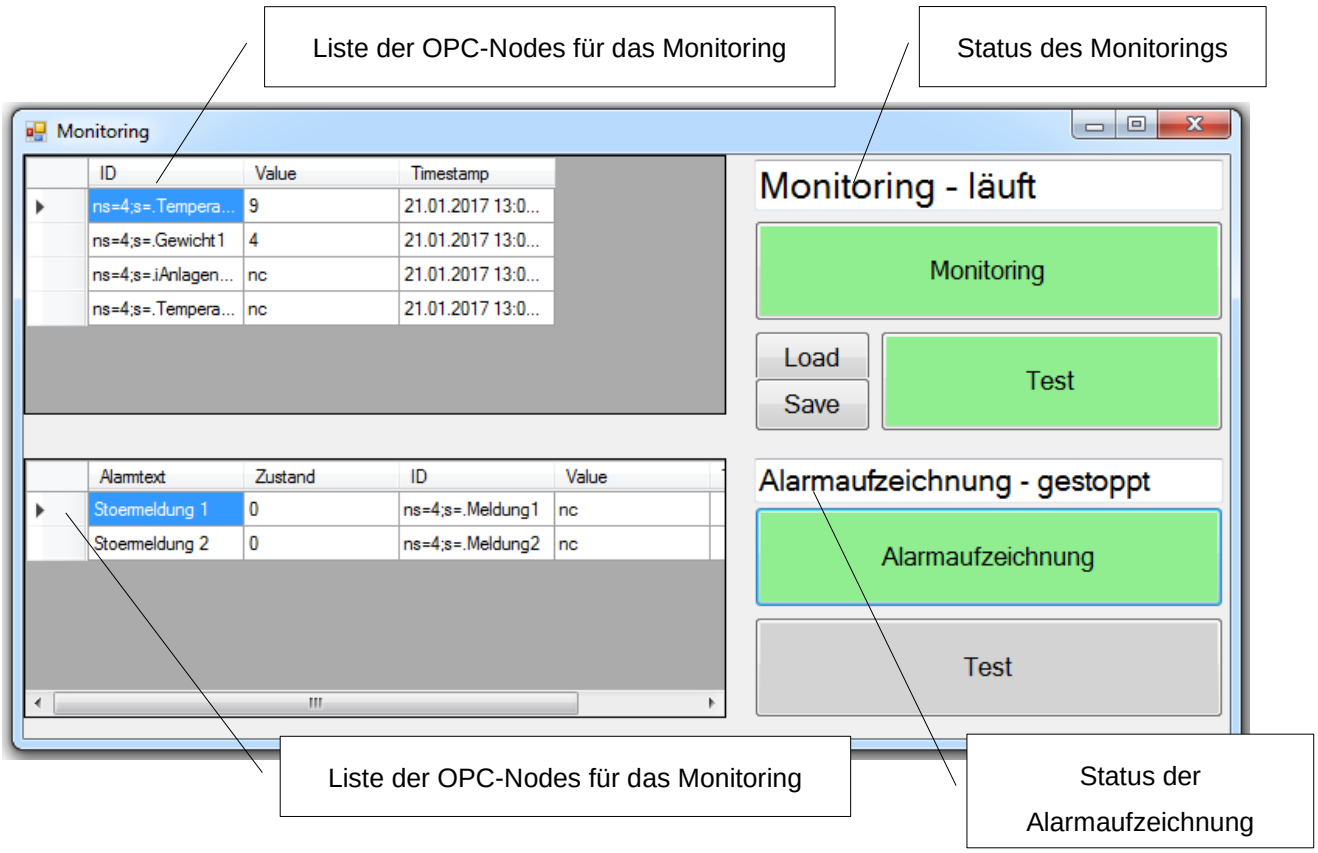


Abbildung 53: Monitoring-Fenster der Wrapper-Applikation, Quelle: Eigene Darstellung.

6.3.2 Datenaustausch mit der Webdatenbank

Nach der Vorstellung der Benutzeroberfläche wird nun gezeigt, wie ein Datenaustausch mit der Webdatenbank geschehen kann und wie die Aufträge verarbeitet werden. Die bereits zuvor erwähnte Abbildung 50 zeigt bereits die prinzipielle Funktionsweise der Applikation. Die Anwendung ruft über die HTTP-POST-Methode ein PHP-Skript am Webserver auf und verarbeitet anschließend die Textausgabe. Hier sind jedoch zwei verschiedene Anwendungsfälle zu unterscheiden. Einerseits können Daten mit POST von der Wrapper-Applikation gesendet und im PHP-Skript in die Datenbank gesichert werden, beispielsweise Prozessdaten für die Aufzeichnung. Andererseits sind die vom Webshop in die Datenbank gesicherten Auftragsdaten auszulesen und an die Wrapper Anwendung weiterzugeben.

Für den in Kapitel 5 geforderten sicheren Datenaustausch wird auf HTTPS zurückgegriffen. Dazu wurde in C# eine eigene Klasse (Web) implementiert. Sie enthält derzeit nur eine Methode, welche einen HTTP-Request generiert. Als Übertragungsprotokoll kommt jedoch, wie bereits in Kapitel 5 gefordert HTTPS zum Einsatz. Der Codeauszug in Abbildung 54 zeigt die Generierung eines Objekts (w) der Webklasse und den anschließenden Aufruf der Methode zur Übertragung der Daten an das PHP-Skript. Die für den Aufruf notwendige URL (hier URL4) des PHP-Skripts ist ein Listeneintrag der zuvor gezeigten globalen Einstellungen (siehe Abbildung 52). Die selbst erstelle Methode basiert auf der in der .NET-System Bibliothek beinhalteten WebRequest- und der Webresponse-Klasse. Die Verwendung von HTTPS erfordert jedoch eine zusätzliche Erweiterung des Objekts um Eigenschaften der darauf aufbauenden HttpWebRequest-Klasse. Beispielsweise muss ein sogenannter User Agent überlagert werden und die Authentifizierungsreferenzen (Credentials) sind ebenfalls anzugeben. In diesem Fall wurden sie als

Default definiert. Für die Versendung von quasi ungesicherten Requests mittels HTTP wäre dies nicht nötig. Mittels Webrequest könnten auch GET-Anfragen gesendet werden, jedoch wird in diesem Fall eine POST-Anfrage bevorzugt, da die Daten hier nicht in der URL kodiert werden sondern sich im Body der Anfrage befinden. Die als String angegeben Parameternamen (z.B. Parameter1) spezifizieren den darauffolgenden Variablenwert. Somit können die Werte beispielsweise mit $_POST[] im aufgerufenen PHP-Skript dem Namen zugeordnet und in die Datenbank geschrieben werden. Diese Vorgangsweise entspricht der Übergabe von Bestelldaten aus dem Formular (ebenfalls POST-Methode), was bereits im Zuge der Auftragsgenerierung im Kapitel 6.2.2.2 behandelt wurde.

```
Web w = new Web();
// HTTP Request senden
w.MyHTTPSREquest(GD.URL4,
    "Parameter1", item.Value,
    "Parameter2", AlarmText,
    "Parameter3", "",
    "Parameter4", "",
    "Parameter5", l.Item2);
```

Abbildung 54: Aufruf der HTTP-Request-Methode, Quelle: Eigene Darstellung.

Die Methode wird im Falle der Alarmgenerierung zyklisch im Sekundentakt in einem eigenen Thread aufgerufen um die Benutzeroberfläche nicht zu blockieren, dazu kommt die BackgroundWorker-Klasse zum Einsatz. Wie bereits im vorrangehenden Kapitel angesprochen, wird ein Alarm nur gesendet wenn eine Freigabe (Button) besteht und sich der zuvor abgefragte Wert des OPC-Nodes geändert hat. Dieser Wert (z.B. item.Value), der zugeordnete Alarmtext sowie der gewünschte Table (z.B. l.Item2) der Webdatenbank sind ebenfalls Parameter. Aufgrund der damit hervorgehenden möglichen Dynamisierung des Tables können die in der Datenbank gesicherten Daten besser organisiert werden. Dies ist besonders für die aufgezeichneten Prozessdaten und die Störmeldungen von Relevanz. Für jede Anlage wäre demnach eine eigene Table sinnvoll. Der Wert für die zu benutzende Table für die Monitoring-Daten ist wiederum in den globalen Einstellungen (siehe Abbildung 52) zu editieren.

Die Methode MyHTTPSRequest wird ebenfalls für die zyklische Übertragung von Prozessdaten und dem Auslesen von Aufträgen benutzt und wird ebenfalls zyklisch in einem eigenen Thread ausgeführt. Die Zykluszeit beträgt hier jedoch ca. fünf Sekunden.

Im Zuge des Monitorings sind die gewünschten (in der Liste konfigurierten) zuvor gelesenen Werte der OPC-Nodes als Parameter zu übergeben. Der eigentliche Aufruf erfolgt jedoch erst nach der Freigabe durch den Button Monitoring (siehe Abbildung 53). War die Übertragung erfolgreich, gibt das aufgerufene PHP-Skript für Alarmaufzeichnung und Monitoring eine Bestätigung (z.B. „neu hinzugefügt“) aus, ansonsten wird ein Fehler ausgegeben. Diese Ausgabe wird als Response gelesen und in Form eines Strings von der MyHTTPSRequest-Methode weitergegeben. Dieser Rückgabewert wird in weiterer Folge für die jeweilige Statusanzeige (siehe Abbildung 53) abgefragt.

Der Ablauf für die Vermittlung von Aufträgen von der Webdatenbank zur Steuerung der Produktionsanlage ist in Abbildung 55 ersichtlich. Diese Logik wird ebenfalls zyklisch in einem eigenen Thread ausgeführt. Die in der entsprechenden Liste Control (siehe Abbildung 51) spezifizierten OPC-

Nodes werden zyklisch von der Wrapper-Anwendung gelesen. Sie bilden den derzeitigen Betriebsstatus der Anlage sowie Variablen zum Starten, Stoppen und Freigeben eines Ablaufes ab.

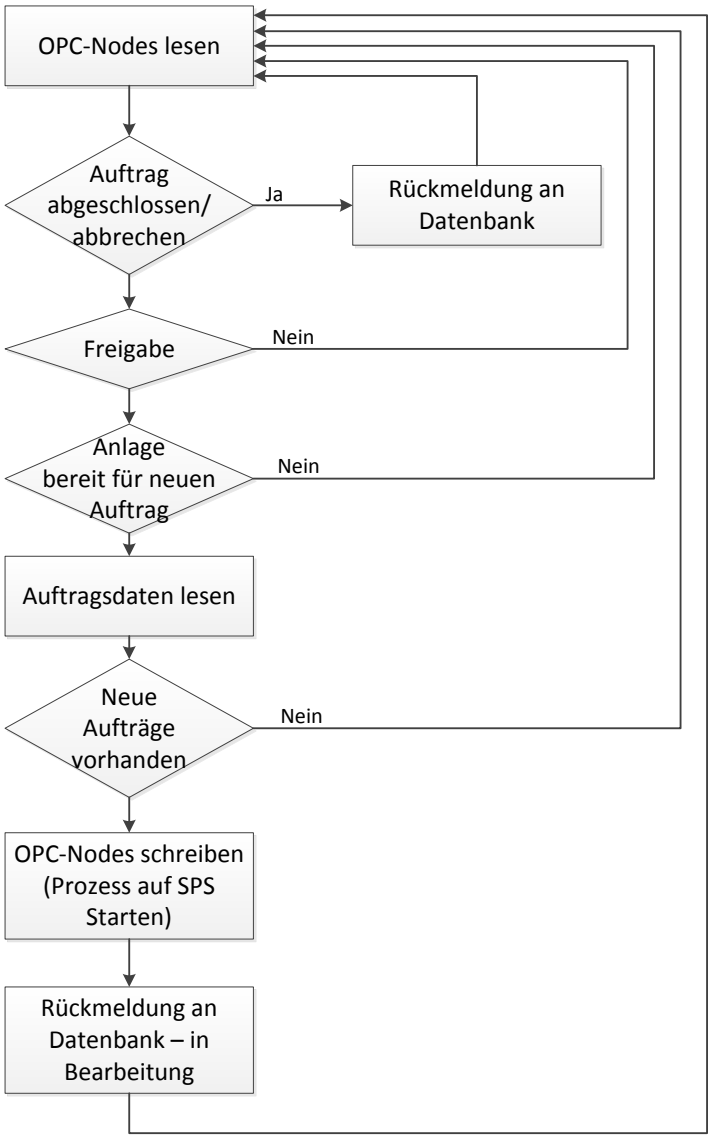


Abbildung 55: Ablauf der Auftragsvermittlung, Quelle: Eigene Darstellung.

Zunächst wird der Anlagenstatus abgefragt, ob ein Auftrag abgeschlossen oder eine Anforderung zum Abbrechen besteht. Ist dies der Fall, so wird sofort der derzeitige Status (z.B. in Bearbeitung) des bearbeiteten Auftrags geändert. Dafür wird wiederum die zuvor geschilderte MyHTTPSRequest-Methode aufgerufen, um ein eigenes PHP-Skript aufzurufen welches den mittels POST übergebenen neuen Auftragsstatus (z.B. Abgeschlossen) und der ID des Auftrags-Datensatzes den Auftragsstatus ändert. Wie bereits erwähnt, kann ein Auftrag jederzeit anhand der Betätigung des Button Stopp, siehe Abbildung 51, oder an einem Button auf der Anlagenvisualisierung abgebrochen werden. Besteht keine Anforderung zum Abbrechen des Auftrags, ist eine Freigebe für den weiteren Ablauf (z.B. Button „From Web“ siehe Abbildung 51) gegeben und die Anlage ist für einen neuen Auftrag bereit, wird der aktuellste Auftrag welcher noch nicht bearbeitet wurde mittels der MyHTTPSRequest-Methode von der Webdatenbank gelesen. Beinhaltet die Ausgabe von dem aufgerufenen PHP-Skript keine neuen Aufträge (Kein Datensatz mit Auftragsstaus ToDo) wird die weitere Bearbeitung abgebrochen und der bisherige Ablauf

zyklisch wiederholt, bis ein neuer Auftrag abzuarbeiten ist. Dadurch wird die in Kapitel 5.1 festgelegte Funktionalität erfüllt, dass sich die Produktionsanlage die Aufträge selbst holt. Nach dem Auslesen der Daten erfolgt nun das eigentliche Mapping der gelesenen Auftragsdaten mit den OPC-Nodes nach der Liste Aufträge (siehe Abbildung 51), indem die ausgelesen Werte in die OPC.UA-Nodes der konfigurierten Reihenfolge nach gespeichert werden. Anschließend sind zuerst die aktualisierten Rezepturdaten und weitere im Auftrag enthaltene Informationen an den OPC-UA-Server zu schreiben (siehe nächstes Kapitel) und in weiterer Folge der Produktionsprozess zu starten (Variable bStarteMischung setzen). Abschließend erfolgt die Rückmeldung an die globale Datenbank anhand der zuvor geschilderten Anpassung des Auftragsstatus.

6.3.3 Kommunikation mit dem OPC-UA-Server

Nach der Schilderung des Datenaustausches zwischen der Wrapper-Applikation und der Webdatenbank sowie dem Ablauf der Auftragsabarbeitung wird nun die Kommunikation mit dem OPC-Server bzw. die Verwendung der OPC-UA-API näher behandelt.

Das Grundgerüst der Anwendung basiert auf der Verwendung von OPC-UA-Client-API-Klassen, welche in Form einer Demoanwendung im OPC-UA Simple Client der Siemens AG beinhaltet sind. Das Klassendiagramm in Abbildung 56 zeigt die verfügbaren Klassen und deren Methoden der verwendeten OPC-UA-Client-API. All diese Klassen sind in der .NET-Assembly Siemens.OpcUA.dll zusammengefasst und haben Abhängigkeiten zu der ebenfalls in der API beinhalteten .NET-Client-SDK-Baugruppe Opc.Ua.Client.dll und der Opc.Ua.Core.dll. Grundsätzlich bietet ein OPC-UA-Server eine Reihe verschiedener Services an (siehe Kapitel 4.3.6), diese sind mittels der Methoden der drei zur Verfügung gestellten Klassen von der .NET-Anwendung ansprechbar. Es sind jedoch bei weitem nicht alle Services der OPC-UA-Spezifikation verfügbar, das für die Anwendung ausreichende Zugriffmodell Data Access und die Base Services werden auch vom verwendeten OPC-UA-Server unterstützt.

Für den gewünschten Datenaustauch zwischen der Wrapper-Anwendung (OPC-UA-Client) und dem auf der SPS-implementierten OPC-UA Server kommt in erster Linie ein Objekt der Server-Klasse zur Verwendung. Dieses Objekt bietet Methoden zum Verbinden sowie zum Unterbrechen der Verbindung mit einem bekannten Server (Connect, Disconnect). Das Lesen und Schreiben von Nodes wird anhand der Methoden ReadValues und WriteValues ausgeführt. Im Gegensatz zu dem in OPC-DA spezifizierten synchronen oder asynchronen Zugriff auf Variablen (mit jeweils einer eigenen Methode) bietet die verwendete ReadValues-Methode beide Kommunikationsvarianten an.

Ist keine Timeoutzeit für den lesenden Zugriff definiert, wird der Zugriff asynchron ausgeführt. Da keine besonders zeitkritischen Anwendungen zu erwarten sind wird auf den performanteren synchronen Zugriff verzichtet. Im Zuge der Auftragsverarbeitung bzw. des Monitorings (Alarme) sind die Nodes maximal im Sekundentakt abzufragen.

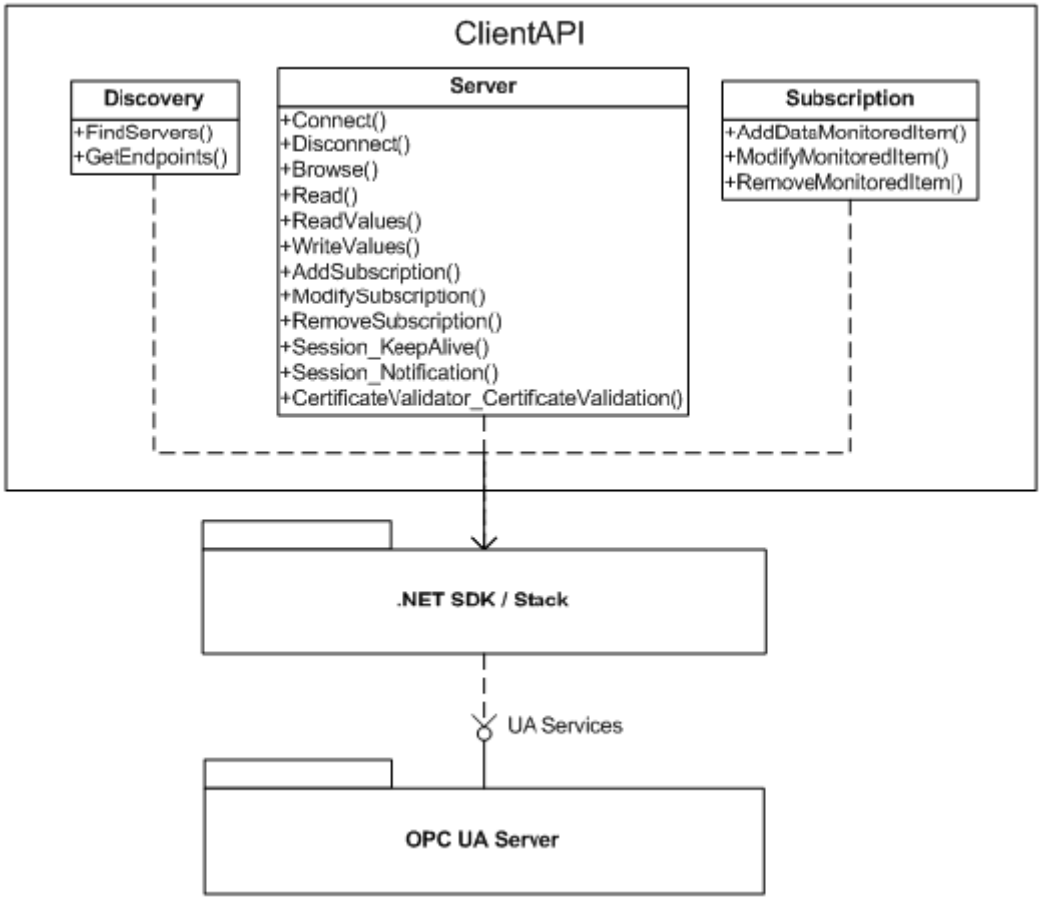


Abbildung 56: Klassendiagramm der verwendeten OPC-UA-Client-API: Quelle: Siemens AG (2010).

Abbildung 57 zeigt einen beispielhaften Aufruf der ReadValue-Methode der verwendeten OPC-UA-API, um zu zeigen wie OPC-Nodes (von der SPS) gelesen werden können. Das statische Serverobjekt (m_Server) wird beim Laden des Grundfensters (Siehe Abbildung 51) aus der bereits behandelten Serverklasse gebildet. Die URL des zu verbindenden Servers ist im Kopfbereich oder in einer Konfigurationsdatei editierbar. Das Server-Objekt kann anhand der statischen Definition verschiedenen weiteren Objekten und Threads verwendet werden und es muss nicht ständig eine neue Instanz gebildet werden. Außerdem entfällt der sonst notwendige Verbindungsaufbau.

In der Wrapper-Applikation wurde zum Lesen von Werten am Server eine eigens erstellte ScanValues Klasse gestaltet, welche die ReadValues-Methode des Serverobjekts beinhaltet und die in den Listen konfigurierten OPC-Nodes (z.B. Node-IDs für das Monitoring, siehe Abbildung 53) für diesen Methodenaufruf aufbereitet.

```
OPC.m_Server.ReadValues(
    nodesToRead,
    out m_currentValues);
```

Abbildung 57: Beispielhafter Aufruf der ReadValue-Methode der verwendeten API: Quelle: Eigene Darstellung.

Der in Abbildung 57 gezeigten ReadValue-Methode wird beim Aufruf ein Objekt (nodesToRead) übergeben das alle zu lesenden Node-IDs enthält. Der Rückgabewert (m_currentValues) stellt die vom

Server (asynchron) gelesenen Werte der Nodes und deren Zeitstempel dar. Diese Werte werden wiederum in die für Anwender ersichtlichen Listen geschrieben und stehen für die weitere Verarbeitung (Beispielsweise zum Abfragen des Anlagenstatus für die Auftragsabarbeitung) als List-Objekt zur Verfügung.

Für das Schreiben von Werten wird ebenfalls eine eigene Klasse um die konfigurierten Listen (z.B. Mapping der Auftragsdaten) gestaltet. Deren Methode OPCWriteValues wird beispielsweise ein List-Objekt (z.B. Liste Auftragsdaten dargestellt in Abbildung 51) übergeben. Zusätzlich wurde noch eine Überladung der Methode implementiert die nur die Node-ID (als String) sowie den zu schreibenden Wert (ebenfalls als String) als Übergabewert akzeptiert. Hier wird die bereits erwähnte WriteValues-Methode des Server-Objekts genutzt. Sie erfordert neben einem Objekt mit dem zu schreibenden Node-IDs ein weiteres Objekt mit den gewünschten Werten. Im Gegensatz zum Lesen der Variablen ist hier der Datentyp der zu beschreibenden Variablen von besonderer Bedeutung. Da dieser nicht vorher spezifiziert wurde (z.B. Eintrag in Konfigurationsliste) und somit nicht bekannt ist, wird vor dem schreibenden Zugriff Lesend auf die Node zugegriffen und dadurch der Datentyp (sowie der aktuelle Wert) festgestellt. Vor dem schreibenden Zugriff wird der zu schreibende Wert in das festgestellte Datenformat umgewandelt. Der Rückgabewert der aufgerufenen Methode des Server-Objekts gibt ein Objekt mit den jeweiligen Status des Nodes zurück (z.B. Bad Value).

Eine Abfrage auf Wertänderung eines OPC-UA-Nodes könnte mittels der in Kapitel 4.3.6 behandelten MonitoredItem Service Sets und den Subscription Service Sets erfolgen, welche einerseits im Subcription-Objekt (siehe Abbildung 56) und anderseits im Server-Objekt als Methoden zur Verfügung gestellt werden.

Eine weitere Methode welche vom Server-Objekt der OPC-UA-Client-API zur Verfügung gestellt wird, ist die Validierung des Serverzertifikats um dessen Identität zu bestätigen. Dieses Zertifikat könnte clientseitig dauerhaft im Windows Certificate Store abgelegt werden. Ist dies noch nicht erfolgt, erscheint beim Starten der Applikation ein Fester (siehe Abbildung 58) welches abfragt ob der Benutzer das vom OPC-UA-Server verwendete, selbstsignierte Zertifikat zur App- bzw. Transport-Security (siehe Kapitel 4.3.8) dennoch akzeptiert.

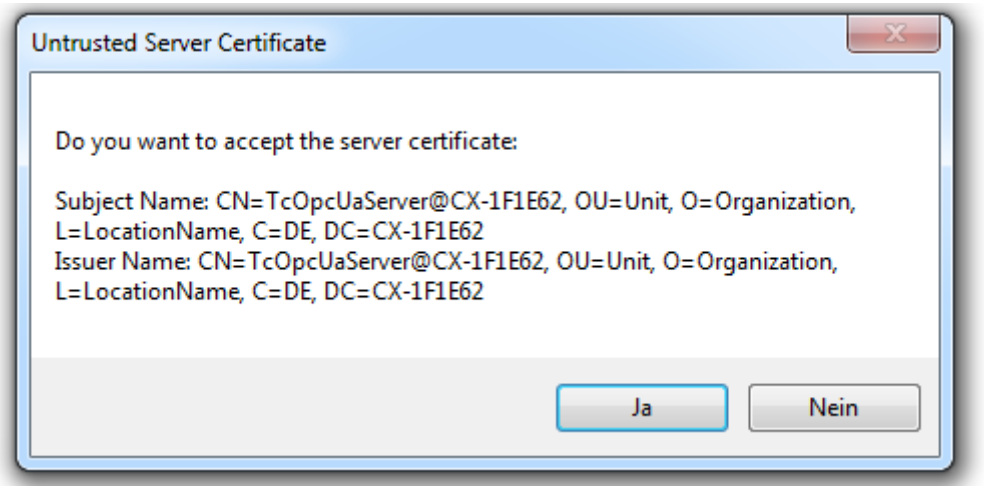


Abbildung 58: Popup für Serverzertifikatsabfrage, Quelle: Eigene Darstellung.

Die Client-API legt beim erstmaligen Start der Anwendung (unbedingt mit Administratorrechten) das (öffentliche) Zertifikat des Clients im Windows Certificate Store an und überträgt es an den OPC-Server. Der Server muss dieses Zertifikat akzeptieren, um wiederum die Identität des Clients zu bestätigen.

6.3.4 Implementierung des OPC-UA-Servers auf der SPS

OPC-UA erlaubt es aufgrund der Plattformunabhängigkeit einen Server direkt auf einer Steuerung zu implementieren. Der Hersteller Beckhoff bietet für diese Anwendung geeignete Softwarelösung für Windows CE an. Die Installation auf der Steuerung erfolgt mit einer auf der Steuerung ausführbaren CAB-Datei, welche im Zuge der Installation (und Registrierung) der Software am Engineering PC (z.B. mit Windows 7) generiert wird. Nach Neustart der SPS ist der OPC-UA Server online. SPS-Variablen welche über den OPC-UA-Server erreichbar sein sollten, sind nur mit einem speziellen Kommentar in der Variablendeklaration zu kennzeichnen.

Der verwendete OPC-UA Server TwinCAT OPC-UA Server CE (TS6100-0030) erlaubt grundsätzlich die unautorisierte (anonymous) sowie die autorisierte Benutzeranmeldung mit Benutzername und Passwort. Die Benutzerauthentifizierung von OPC-UA (Siehe Kapitel 4.3.8) könnte zusätzliche Sicherheit bieten, denn sie gewährleistet die Echtheit des Clients. Die User-Security wurde für den Testbetrieb der Anlage jedoch nicht weiter verwendet, wird aber in weiterer Folge noch nachgerüstet. Zur Sicherstellung der App- bzw. Transport-Security ist Sign&Encrypt (Signieren und Verschlüsseln von Nachrichten) als Security-Level zu wählen. Das im vorrangehenden Kapitel genannte Zertifikat des Clients muss dafür im Dateisystem (z.B. im Pfad: InstallDir\UA\PKI\CA\certs) des Servers hinterlegt werden.

6.4 Funktionstest an der Produktionsmaschine

Nach der Vorstellung des grundsätzlichen Aufbaus und der Funktionsweise des Systems in den vorrangehenden Kapiteln wird nun ein Funktionstest des Gesamtsystems durchgeführt. Dies soll einerseits zeigen, wie die Auftragsdaten (Bestellungen) automatisiert vom Webshop auf die Steuerung übertragen und gelesene Werte an die Datenbank gesendet werden. Dies soll zeigen, dass die bereits in der Aufgabenstellung spezifizierten Funktionen durch das System erfüllt werden können.

6.4.1 Testaufbau

Abbildung 59 zeigt den Testaufbau in der Versuchsanlage, angelehnt an das im Kapitel 5 ausgewählte Konzept. Die Wrapper-Anwendung wird auf einem lokalen PC (siehe Abbildung) ausgeführt, welcher über Ethernet mit der SPS im Steuerschrank Verbunden ist. Die Kommunikation erfolgt über das performante OPC-UA-Binary-Protokoll. Die benötigte Internetverbindung wird über ein handelsübliches USB-Modem mit SIM-Karte aufgebaut. Eine zusätzliche Sicherung mittels externer Firewall ist während der Testphase aus wirtschaftlichen Überlegungen nicht erfolgt.

Die beiden Webanwendungen werden mit dem zweiten PC aufgerufen (Web-Client), dieser ist über seine WLAN-Schnittstelle mit dem Internet verbunden.

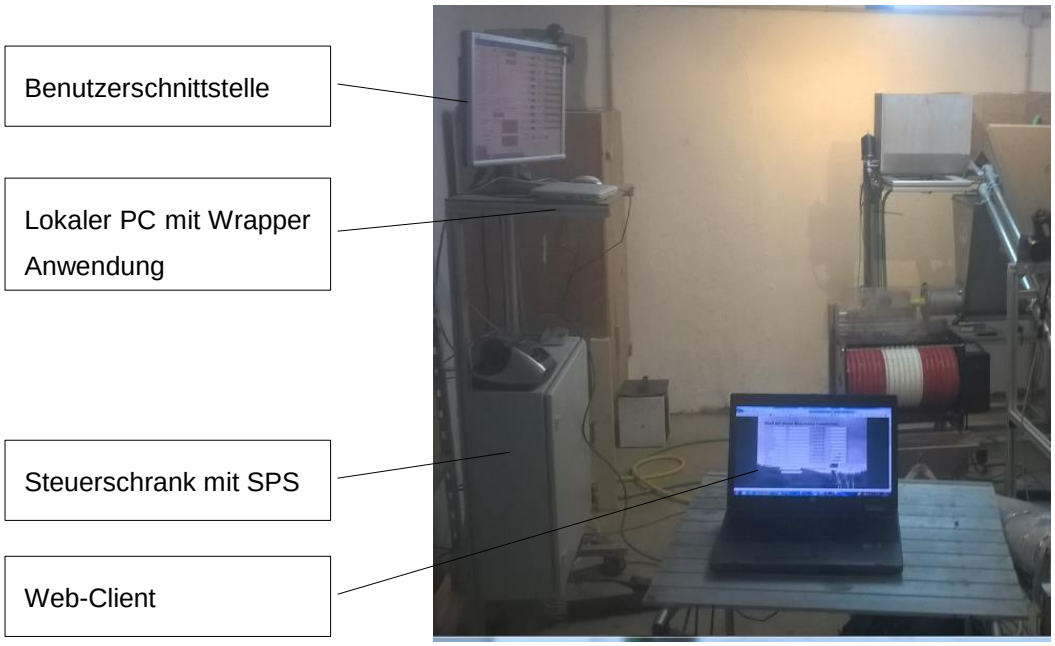


Abbildung 59: Steuerung der Versuchsanlage, Quelle: Eigene Darstellung.

6.4.2 Automatische Verarbeitung der Bestellungen

Hier wird die in der Aufgabenstellung geforderte automatische Verarbeitung von im Webshop getätigten Bestellungen auf der Versuchsanlage getestet. Dafür wird am Web-Client (Beliebiger PC oder Smartphone) die Webshop-Webanwendung (https://tpboilie.at/TP/) im Internetexplorer 11 aufgerufen und eine in Abbildung 60 gezeigte, beliebige Produktzusammensetzung konfiguriert.

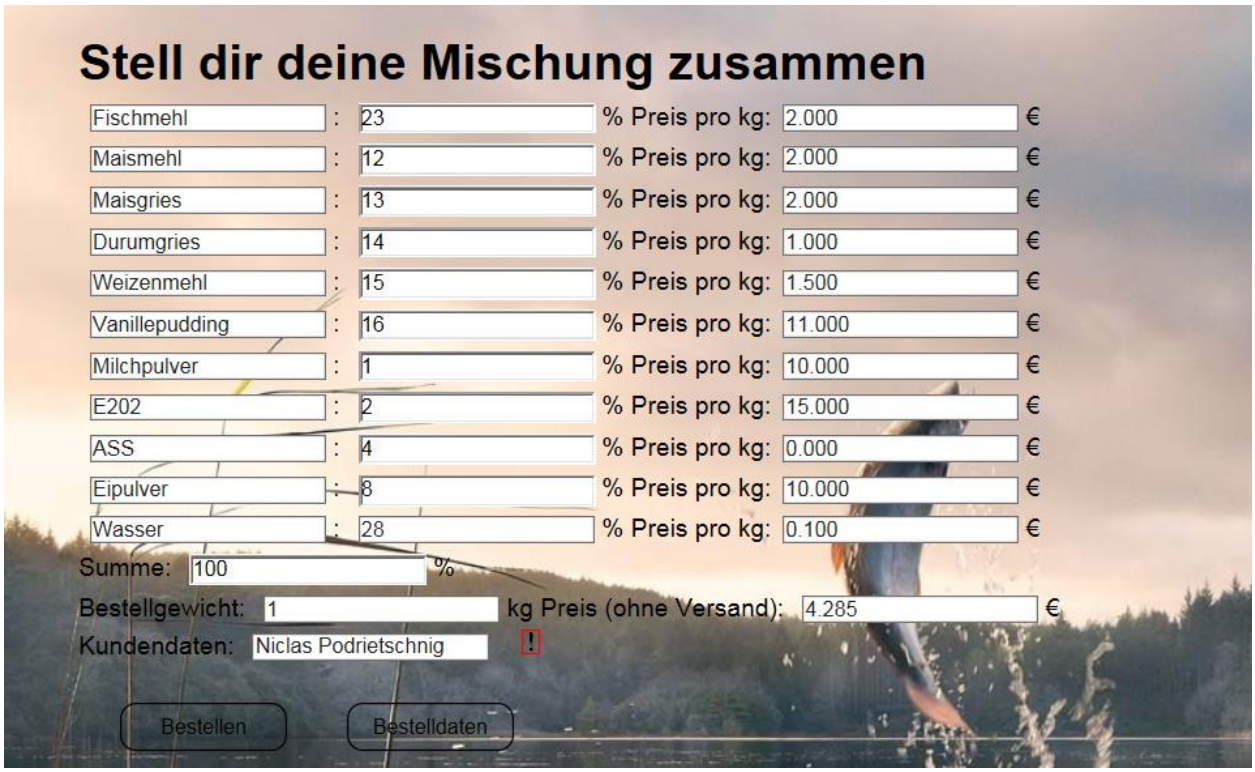


Abbildung 60: Konfigurierte Bestellung in der Webshop-Webanwendung, Quelle: Eigene Darstellung.

Wie bereits in Kapitel 6.2.2. gezeigt, erfolgt die eigentliche Bestellung der Waren mit der Betätigung des in Abbildung 60 ersichtlichen Bestellen-Buttons. Damit die im HTML-Formular konfigurierten Werte mittels HTTP-POST an das in Kapitel 6.2.2 ausführlich behandelte PHP-Skript übergeben werden, muss ein Benutzer angemeldet sein. Zusätzlich zur Überprüfung der Eingaben während der serverseitigen Auftragserstellung (im PHP-Skript) übernimmt das in die Webseite eingearbeitete JavaScript eine zusätzliche Überprüfung der Benutzereingaben (Entspricht die Gesamtsumme der Zutaten nicht 100 % wird sie orange eingefärbt). Die konfigurierte Rezeptur muss 100 Prozent ergeben, außer die Produktnamen sind Eipulver oder Wasser, denn diese Zutaten sind als unabhängig von der restlichen Rezeptur zu verstehen.

ID	Zutat1	MV1	Zutat2	MV2	Zutat3	MV3	Zutat4	MV4	Zutat5	MV5
11	Maismehl	25	Fischmehl	25	Durumgries	25	Weizenmehl	25	Vanillepudding	0
12	Maismehl	25	Fischmehl	25	Durumgries	25	Weizenmehl	25	Vanillepudding	0
13	Eipulver	5	Maismehl	10	Maisgries	15	Fischmehl	20	Durumgries	10
14	Fischmehl	25	Maismehl	25	Maisgries	25	Durumgries	25	Weizenmehl	0
15	Fischmehl	23	Maismehl	12	Maisgries	13	Durumgries	14	Weizenmehl	15

Bezahlt	Gewicht	Kunde	Preis	Status
1	5	Kunde 1	8.12500	Done
1	5	Kunde 1	8.12500	Done
1	5	Kunde 2	24.25000	Done
1	5	Kunde 2	13.88500	Done
1	1	Niclas Podrietschnig	4.28500	ToDo

Abbildung 61: Konfigurierte Bestellung in der Web-Datenbank, Quelle: Eigene Darstellung.

Nach der erfolgreichen Bestellung (derzeit ohne Bezahlfunktion) werden die Aufträge in der MySQL-Datenbank abgelegt. Abbildung 61 zeigt einen Auszug (nicht alle Rezepturdaten) der zuvor konfigurierten Auftrags mit der internen ID 15. Da dieser Datensatz noch nicht von der Produktion bearbeitet wurde, ist sein Status auf ToDo. Bereits erledigte Aufträge (siehe Ablauf in Kapitel 6.3.2) sind mit dem Status Done markiert. Die in Kapitel 6.3 behandelte Wrapper-Anwendung, welche am lokalen PC ausgeführt wird ruft nach Freigabe durch einen Anwender zyklisch (Ablauf siehe Kapitel 6.3.2) ein PHP-Skript am Webserver auf, das den ersten Auftrag mit dem Status „Do“ ausgibt. In diesem Fall wird der Auftrag mit der ID 15 ausgegeben und mittels HTTPS gesichert an die .NET-Anwendung zurückgegeben.

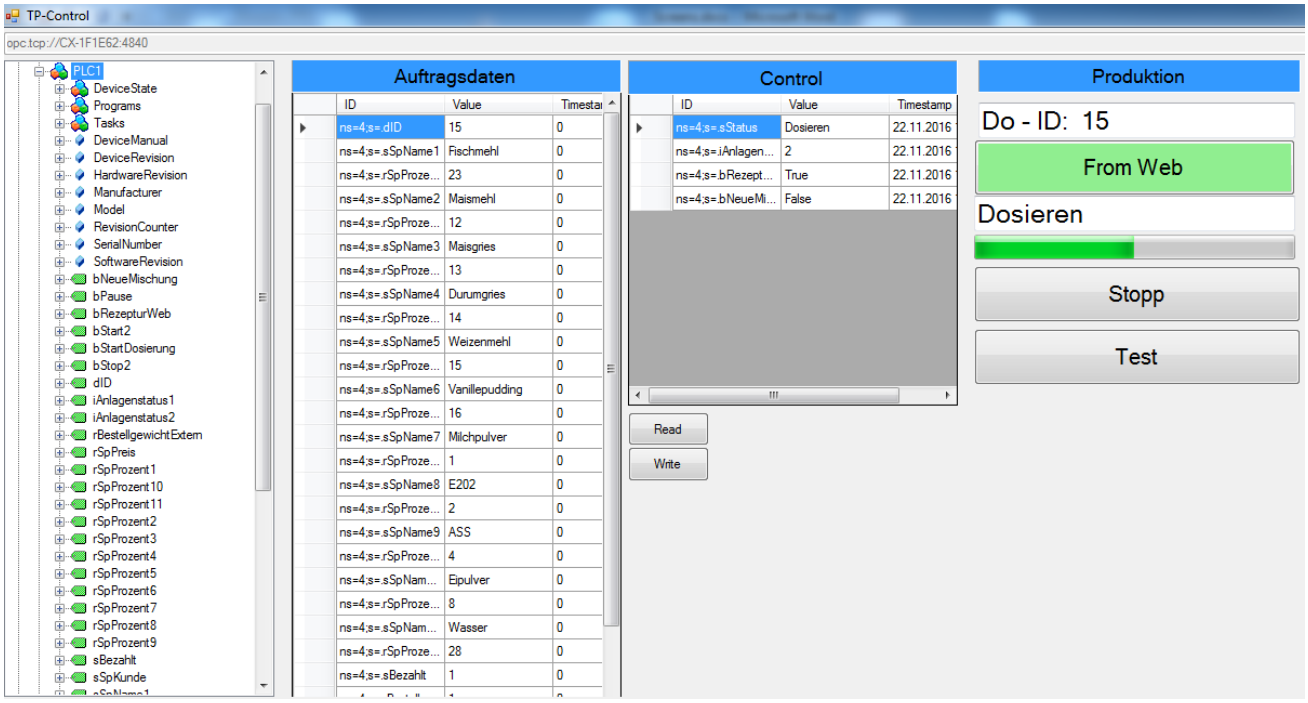


Abbildung 62: Grundbild der Wrapper-Anwendung mit aktuellem Auftrag, Quelle: Eigene Darstellung.

Anhand der Verwendung von klassischen HTTP-Requests ist hier jedoch der übertragene Overhead jedoch höher als mit neueren Technologien wie beispielsweise Websocket. Deshalb werden hier die Auftragsdaten nur bei Freigebe und wenn die Anlage auch wirklich bereit ist für einen neuen Auftrag gelesen. Anhand der typischen Abarbeitungszeit von ca.15 Minuten fällt dieser Overhead während des dauerhaften Betriebs jedoch nicht ins Gewicht.

Die empfangenen Auftragsdaten werden den konfigurierten OPC-Nodes zugeteilt und an den OPC-UA Server welcher auf der SPS installiert wurde übertragen. Die aktuellen Werte sind an der in Abbildung 62 ersichtlichen Benutzeroberfläche in der Liste Auftragsdaten ersichtlich. Die Statusanzeige zeigt die derzeit bearbeitete ID des Auftrags (hier 15) an. Nach Übernahme eines neuen Auftrags wird auch der Produktionsprozess gestartet. Der derzeitige Produktionsschritt der Anlage ist ebenso an der Benutzeroberfläche ersichtlich (hier Dosieren). Abbildung 63 zeigt die von der Wrapper-Anwendung gelesenen Auftragsdaten (ID 15) auf der Benutzerschnittstelle der SPS, sprich die Werte der vom OPC-Client geschriebenen SPS-Variablen.

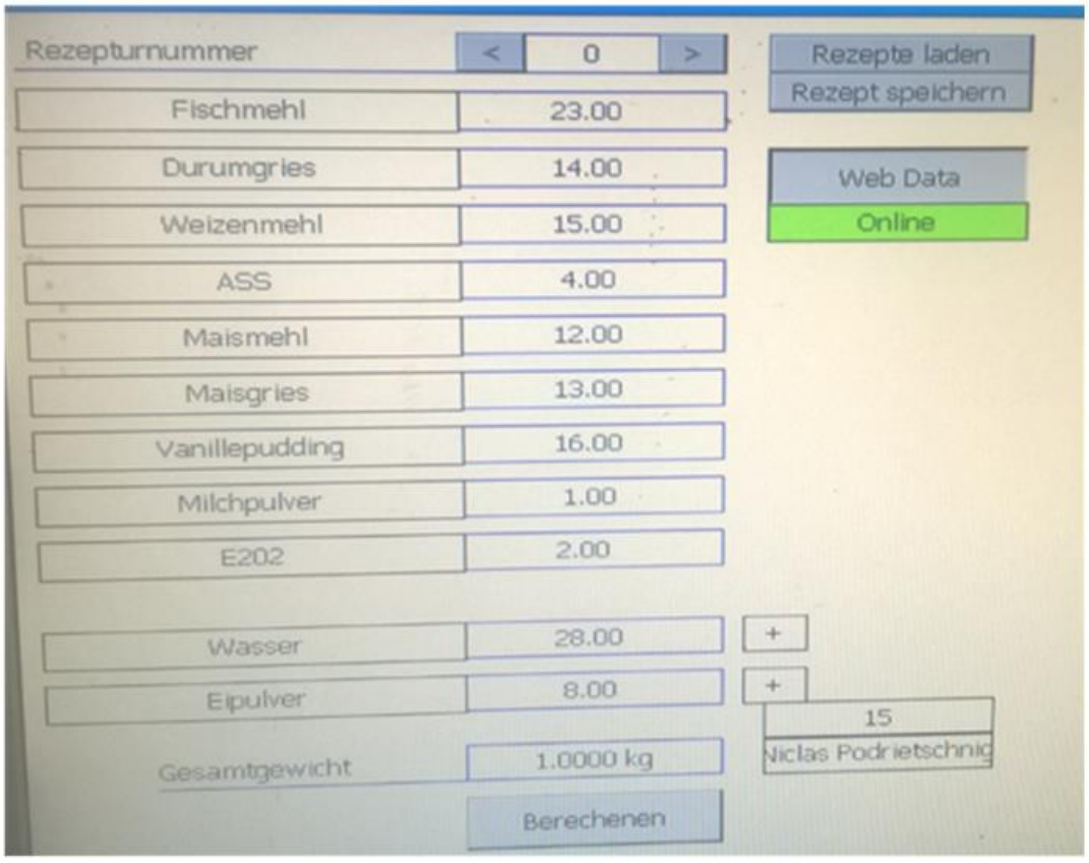


Abbildung 63: Aktuelle Auftragsdaten auf der SPS, Quelle: Eigene Darstellung.

Der somit bis auf die Produktionsmaschine übertragene Auftrag wird nun von der Versuchsanlage automatisch abgearbeitet (Ablauf siehe Kapitel 6.1) und der Staus des Auftrags in der Datenbank auf Do gesetzt. Die im Web konfigurierte Rezeptur wird demnach automatisch zu Boilies verarbeitet. Ist der automatische Prozess beendet (Waage leer oder Überwachungszeit verstrichen) ändert die Wrapper-Anwendung den Status des aktuellen Auftrags auf „Done“.

Wie bereits in Kapitel 6.1 veranschaulicht, ist der Prozess auf der gewählten Versuchsanlage nicht vollständig automatisiert. Die Trocknung und der Versand sind noch manuell auszuführen. Eine automatische Rechnungslegung an Kunden ist ebenso noch nicht implementiert, ist aber (meist) Teil der Bezahldienste. Außerdem wird derzeit nur mit einer Versuchsanlage getestet, das System könnte jedoch auf mehreren dezentralen Anlagen zur Verwendung kommen.

6.4.3 Monitoring von Prozessdaten

Der zweite Teil des Funktionstests umfasst die in der Aufgabenstellung geforderte und in Kapitel 5 präzisierte Monitoring von Prozessdaten und als spezielle Herausforderung das Anzeigen von aktuellen Störmeldungen.

Abbildung 64 zeigt die Monitoring-Benutzeroberfläche der entwickelten Wrapper-Anwendung. Die Alarmaufzeichnung sowie das Monitoring (zyklische Übertragung von Prozessdaten) sind über die Buttons freigegeben. Die in den beiden Listen konfigurierten OPC-UA-Nodes (z.B. Gewicht1) werden im Falle des Monitorings in einem Zyklus von ca. 5 Sekunden aktualisiert und im Anschluss an die Webdatenbank übertragen (Dauer des Aufrufs der in Kapitel 6.3.2. behandelten MyHTTPSRequest-Methode kommt hinzu). Die aktuellen Werte und der Zeitstempel der gelesenen Nodes sind in der Liste ersichtlich. Ist eine Node-ID fehlerhaft deklariert, wird in der Liste der Wert (null) angezeigt.

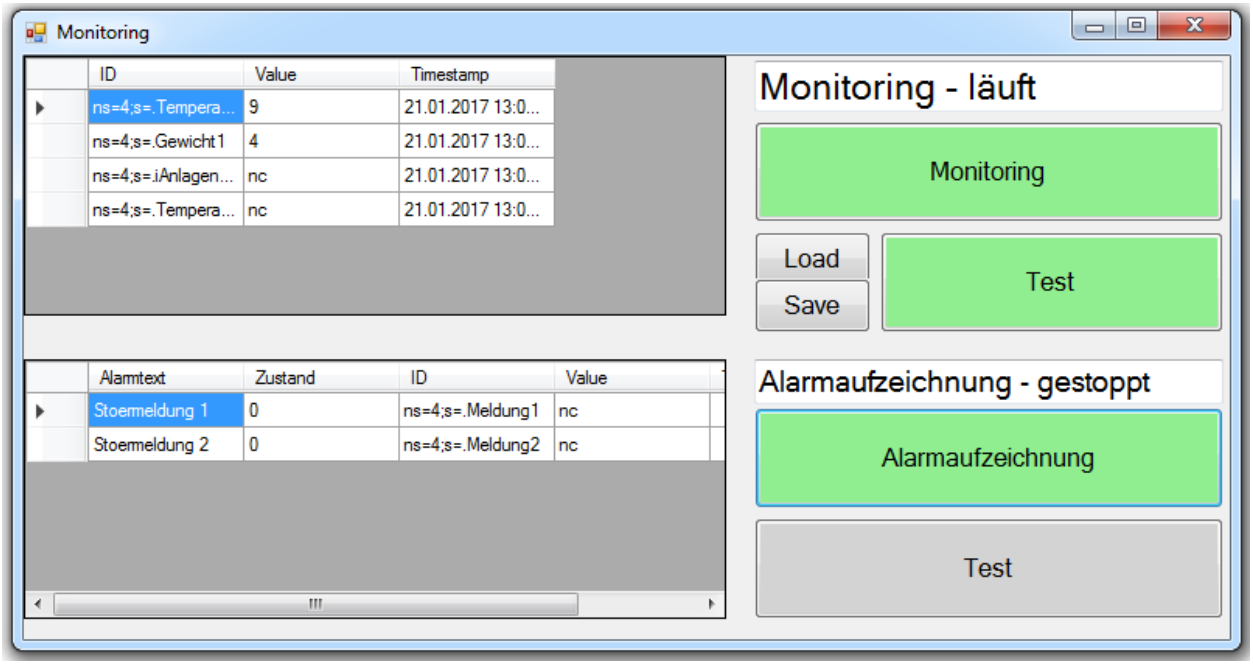


Abbildung 64: Monitoring-Fenster, Quelle: Eigene Darstellung.

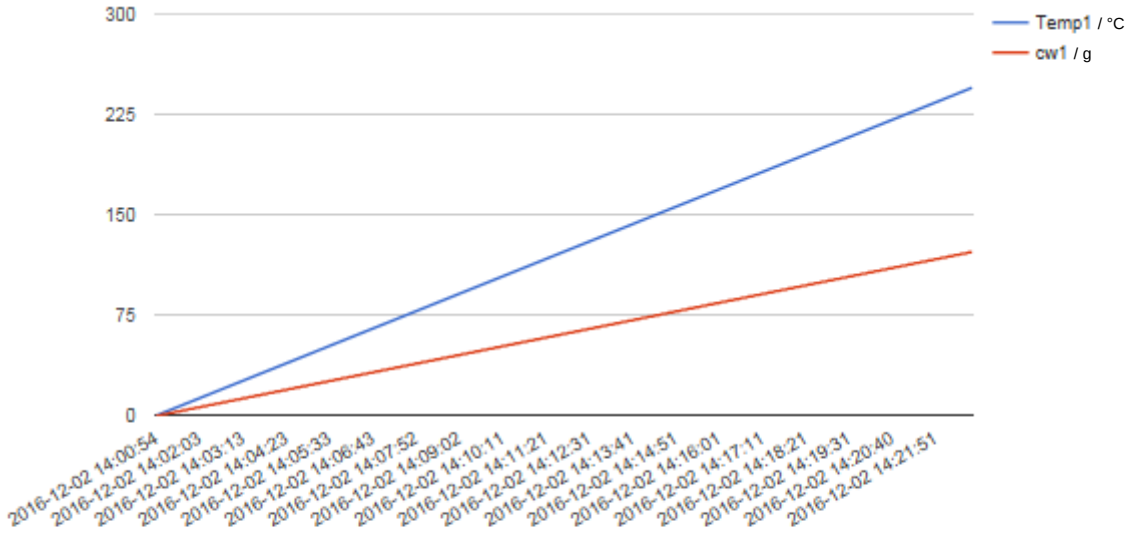
Tritt nun eine Störung an der Produktionsanlage auf (z.B. Motorschutzschalter fällt) wird die SPS-Variable Meldung1 gesetzt. Die Wrapper-Anwendung aktualisiert die als Meldung deklarierten Nodes zyklisch und setzt bei einem Zustandswechsel des Nodes von False auf True (Meldetext und Kürzel K für Kommend) und bei einem Zustandswechsel von True auf False (Fehler behoben - KG) einmalig eine Fehlermeldung ab. Die selbst entwickelte MyHTTPSRequest-Mothode ruft ein PHP-Skript am Webserver auf und speichert den konfigurierten Alarmtext in die Webdatenbank.

Der Inhalt des Table Stoermeldungen wird in der Monitoring-Webanwendung (siehe Kapitel 6.2.3) ebenfalls zyklisch aktualisiert. Eine auftretende Meldung, siehe Abbildung 65 ist somit (bei geöffneter Monitoring-Ansicht im Webbrowser) innerhalb kürzester Zeit für Anlagenbetreiber sichtbar.

	Time	Text
1	2016-11-27 20:12:43	Stoermeldung 1 (KG)
2	2016-11-27 20:06:14	Stoermeldung 1 (K)

Abbildung 65: Alarmmeldungen in der Monitoring-Ansicht (Webbrowser), Quelle: Eigene Darstellung.

Die Aufzeichnung von Prozessdaten ist ebenfalls in der Monitoring-Ansicht sichtbar. Bei Aufbau der Webseite werden die in der Webdatenbank gespeicherten Prozessdaten als Kurven temp1 (Temperaturmessung) und cw1 (Gewichtsmessung) in dem Diagramm (siehe Abbildung 66) angezeigt. Zum Testen der Datenaufzeichnung wurde ein Testmodus implementiert welcher auf der SPS eine Zählervariable (z.B. Temperatur1) alle 5 Sekunden inkrementiert, die Gewichtsanzeige wird ebenfalls simuliert (Zähler, durch Zwei ohne Kommastellen).



Time	Temp1	cw1
2016-12-02 14:00:54	0	0
2016-12-02 14:00:59	1	0
2016-12-02 14:01:04	2	1
2016-12-02 14:01:10	3	1
2016-12-02 14:01:15	4	2
2016-12-02 14:01:21	5	2
2016-12-02 14:01:26	6	3

Abbildung 66: Aufzeichnung von Prozessdaten, Quelle. Eigene Darstellung.

In dem aufgezeichneten Zeitraum konnten keine Verbindungsausfälle beim Monitoring festgestellt werden. Ist die Internetverbindung jedoch nicht besonders performant, ist sicherlich mit einer hohen Anzahl an nicht übertragenen Werten zu rechnen, da ein erneutes Senden nach nicht erfolgreicher Übertragung der Daten derzeit nicht implementiert ist. Für den Benutzer wird jedoch eine Fehlermeldung ausgegeben.

7 RESÜMEE

Ziel dieser Masterarbeit war es, ein flexibles und sicheres System im Sinne der 4. industriellen Revolution zu entwickeln, das einen Datentransfer zwischen verteilten Produktionsanlagen und einem Webshop ermöglicht. Die vom Endkunden in einem demonstrativen Shop getätigten Bestellungen eines individuell konfigurierten Produkts, sind möglichst automatisch an einer örtlich unabhängigen Produktionsanlage abzuarbeiten. Außerdem war dem Anlagenbetreiber in Form einer Webanwendung eine Möglichkeit zum Monitoring der Produktion zur Verfügung zu stellen. Die Konzeption von möglichen Systemarchitekturen, welche einen Informationsfluss zwischen SPS-gesteuerten Produktionsanlagen sowie den Web basierten Schnittstellen zu Endkunden und Betreibern gewährleisten, war besonders zu fokussieren. Eine grundlegende Idee eine zentrale Datenbank als Informationsspeicher für die Produktionsanlagen, sowie für die Webanwendungen wurde bereits in der Aufgabenstellung festgehalten. Außerdem sollte eine ausgewählte Konzeption für das Produktionssystem auf einer eigens dafür vorgesehen Produktionsanlage getestet werden. Schlussendlich sollte das System eine Alternative zu kostenintensiven ERP- und MES-Systemen darstellen, wobei ein Onlineshop die Schnittstelle zum Endkunden bildet.

Bevor im theoretischen Teil der Arbeit näher auf den technischen Aufbau eines Onlineshops bzw. von Webanwendungen und auf mögliche Kommunikationsvarianten des zu realisierenden Systems eingegangen wurde, war zunächst der Innovationsgrad der vorgeschlagenen Lösung, nämlich einen Webshop mit einer Maschine zu verbinden, sowie eine Webanwendung zum Monitoring von Prozessdaten zu verwenden, zu unterstreichen. In diesem Kapitel wurde gezeigt, dass diese Idee einen Beitrag zur vierten industriellen Revolution leistet und welche Prozesse in einem Unternehmen übernommen werden können. Zusätzlich wurde hier die grundsätzliche Funktion des zu konzipierenden Systems hier weiter präzisiert.

Im darauf folgenden Kapitel wurden der Aufbau einer Webanwendung, deren Kommunikationsmöglichkeiten mit einem Webserver, sowie möglich zu verwendende Programmiersprachen und Softwarekomponenten näher dargestellt. Anschließend, dies ist vor allem für den Webshop relevant, waren die Funktionsweisen von gängigen Bezahlsystemen von Interesse .Dieses Kapitel sollte einen Teil der Basis bilden, um die Entwicklung von Konzepten und die Umsetzung einer Systemarchitektur für eine Verbindung zwischen den Webanwendungen der zentralen Datenbank und den dezentralen Produktionsmaschinen zu ermöglichen. Anschließend wurden Technologien analysiert, welche eine bidirektionale Verbindung zwischen einer zentralen Datenbank und einer SPS gesteuerten Produktionsanlage ermöglichen, um anschließend die Entwicklung von Konzepten für eine Verbindung zwischen den beiden Webanwendungen und der Steuerung zu unterstützen und folgend eine geeignete Systemarchitektur zu finden.

Anschließend wurde ein Lösungsansatz für eine praktische Umsetzung für ein möglichst sicheres und flexibles System auszuwählen, um die automatische Auftragsabarbeitung, von der Bestellung im Onlineshop bis zur Fertigung des gewünschten individuellen Produkts, sowie die Datenerfassung von Prozessdaten an einer Testanlage zu erproben.

Zunächst wurden die Anforderungen an die Applikation anhand der erlangten Kenntnisse des theoretischen Teils der Arbeit zusätzlich präzisiert. Außerdem wurden mögliche Bedrohungen aufgezeigt, um grundsätzliche Sicherheitsanforderungen und zu verwendende Protokolle der Konzepte zu formulieren. Im Folgenden wurden Konzepte vorgestellt, welche einen Informationsfluss zwischen Produktion und den Web basierten Schnittstellen zu Endkunden und Betreibern gewährleisten, um mehrere Möglichkeiten aufzuzeigen wie das System aufgebaut sein könnte. Alle vier erarbeiteten Konzepte ermöglichen die geforderte verteilte Systemarchitektur und den laut Kapitel 5.2 geforderten sicheren Datenaustausch zwischen den Produktionsstandorten und der zentralen Datenbank über das Internet. Zusätzlich bietet jedes Konzept einen Webserver an, somit können beispielsweise, nach Aufruf der Webseite mit der Shop-Webanwendung Daten in die Datenbank gespeichert werden. Die Verbindung zwischen Client (Browser) und Webserver erfolgt laut den Ergebnissen aus Kapitel 5.2 bei jedem Konzept verschlüsselt. Auf Seiten der Webanwendungen (Webshop und Monitoring) ist einerseits auf die Integrität des Datenstroms und anderseits auf die Authentizität möglicher Kunden bzw. des Betreibers zu achten, wobei die Kommunikation mittels bewährten Technologien wie HTTPS oder verschlüsselter VPN Verbindung als sicherer Datenaustausch angesehen wird. Zur Feststellung der Authentizität wurde eine klassische Benutzeranmeldung angedacht. Bestellungen sollten auch im Webshop nur nach Bezahlung automatsch weiterverarbeitet werden, um Missbrauch durch Dritte zu verhindern. Die Webanwendungen wurden demnach mittels HTTPS und einer Benutzeranmeldung gesichert, außerdem sollte der Informationsfluss vom Client zur Datenbank (z.B. Aufträge sichern) die Eingaben prüfen um mögliche Angriffe zu erschweren. Anhand der Art des Informationsflusses zwischen der Produktionsanlage und der Web-Datenbank sowie die Art des Webservers, beispielsweise ein Rootserver oder ein virtueller Server sowie mögliche serverseitige Programmiersprache unterscheiden sich die konzipierten Systeme erheblich in der Ausführung und der Flexibilität. Folgend wurden die ausgearbeiteten Konzepte anhand der in Kapitel 5.1 festgelegten Anforderungen analysiert, sowie Vor- und Nachteile der Lösungsvorschläge aufgezeigt, welche die Grundlage zur Entscheidung bilden und mit Punkten bewertet wurden. Tabelle 1 zeigt demnach, dass das Konzept mit dem gehosteten Webspace und der lokalen Wrapper-Anwendung (siehe 5.3.4 und 5.4.4) am besten für das System geeignet ist. Besonders aufgrund der leichten Integrierbarkeit und der hohen Realisierbarkeit wird diese Systemarchitektur bevorzugt. Zusätzlich werden alle in Kapitel 5.2 aufgezeigten Sicherheitsanforderungen ausreichend erfüllt und es ermöglicht als einziges die Verwendung eins gemieteten Webspaces. Der dafür zusätzlich benötigte PC könnte sicherlich für weitere Aufgaben in der Produktion eingesetzt werden.

Nach der Auswahl einer geeigneten Konzeption für das System und der genaueren Spezifikation der Anforderungen wird im Zuge dieses Kapitels die grafische Benutzeroberfläche der erstellten Anwendungen vorgestellt. Außerdem soll die Implementierung der Wrapper-Anwendung OPC-UA-Clients in C# und der Aufbau der beiden Webanwendungen anhand von kurzen Auszügen erläutert werden. Der demonstrative Webshop soll zeigen wie ein von Kunden konfiguriertes Produkt in die zentrale Web-Datenbank gesichert wird. Die implementierte Monitoring-Ansicht dient zur Anzeige von historischen Prozessdaten und zur Anzeige von aktuellen Meldungen der Produktionsanlage, wobei die asynchrone Verarbeitung von Daten im Vordergrund stand. Die Vorstellung der Versuchsanlage und ein abschließender Test der in der Aufgabenstellung geforderten und in Kapitel 5 präzisierten Funktionen sind ebenfalls Teil des letzten Abschnitts der Arbeit. Die durchgeführten Tests zeigten, dass die in der

Aufgabenstellung gewünschte Funktionalität der Applikation gegeben ist. Außerdem konnten keinerlei Verbindungsausfälle oder nicht durchgeführte Requests festgestellt.

Das entwickelte System sorgt für einen sicheren Datenaustausch mit Web-Applikationen und ist besonders einfach in bestehende Systeme zu integrieren. Darüber hinaus ermöglicht es eine automatische Abwicklung von im Online-Shop getätigten Aufträgen und eine Datenüberwachung für den Betreiber. Anhand dieser Funktionalitäten und der hohen Flexibilität bietet das System eine wirtschaftlich, attraktive Alternative zu bestehenden MES- oder ERP-Systemen.

Basierend auf diesen Ergebnissen sind weitere Langzeitversuche erforderlich, um die höhere Leistung der Produktion aufgrund der automatischen Auftragsabarbeitung zu bestätigen und weitere Verbesserungen an den entwickelten Prototypen vorzunehmen. Obwohl das System für die Kommunikation mit mehreren Anlagen konzipiert wurde, sind die Tests nur auf eine einzige Versuchsanlage beschränkt. Außerdem können die bestellten Produkte derzeit noch nicht, im demonstrativen Onlineshop bezahlt werden. Im weiteren Verlauf könnte die Webshop-Schnittstelle mit einem schlüsselfertigen Shopsystem abgedeckt werden. Für die geforderte Funktionalität des Systems ist die Ausprägung der Shop-Lösung jedoch nicht von Relevanz. Das entwickelte System ermöglicht es Prozessdaten und Meldezustände ausschließlich für den Betreiber ersichtlich in einer Webanwendung darzustellen. Darüber hinaus kann die Auftragsabarbeitung anhand der implementierten Logik selbständig durch die Maschine erfolgen. Die Produktionsmaschine holt sich demnach die Aufträge vom Webshop.

Zusätzlich ist hervorzuheben, dass das im Zusammenhang mit Industrie 4.0 erwähnte Konzept der Selbstkonfiguration der Produktionsmaschine nicht implementiert ist. Die Anlagenkonfiguration, sprich die Anzahl der Zutaten ist begrenzt und bei Änderungen dieser müsste das gesamte System angepasst werden. Die Anpassungen sind jedoch überschaubar. Im Gegensatz dazu ist es besonders einfach eine weitere Produktionsmaschine in das System zu integrieren. Dazu müsste nur eine weitere Instanz der Wrapper-Anwendung konfiguriert werden (neuer OPC-UA-Servername) und eine identische Produktionsmaschine (identisches SPS-Programm) könnte ihre Arbeit augenblicklich und standortunabhängig beginnen.

LITERATURVERZEICHNIS

Gedruckte Werke (10)

Abts, Dietmar (2015): *Masterkurs: Client/Server-Programmierung mit Java*, 4. Auflage, Springer-Verlag, Heidelberg.

Bauernhansel, Thomas; ten Hompel, Michael; Vodel-Heuser, Birgit (Hrsg.) (2014): *Industrie 4.0 in Produktion, Automatisierung und Logistik*, 1. Auflage, Springer Vieweg, Wiesbaden.

Cantelon, Mike; Harter, Marc; Holowaychuk, T.J.; Rajlich, Nathan (2014): *Node.js in Action*, Manning Publications Co, Shelter Island, NY 11964.

Hammer, Norbert; Bensmann, Karen (2011): *Webdesign für Studium und Beruf: Webseiten planen gestalten und umsetzten*, 2. Auflage, Springer-Verlag, Heidelberg.

Hollosi, Arno (2012): *PHP: Von Geodaten bis NoSql*, Carl Hanser Verlag, München.

Friedmann,Vitaly (2009): *Praxisbuch Web 2.0:Moderne Webseiten programmieren und gestalten*, 2. Auflage, Galileo Press, Bonn.

Lange, Jürgen (2014): *OPC*: *Von Data Access bis Unified Architecture,* 5. Auflage, VDE Verlag GmbH, Berlin.

Stender, Peter (2014): *Webprojekte realisieren nach neuesten OOP-Kriterien*, 1. Auflage, Vieweg+Teubner Verlag, Wiesbaden.

Steyer, Ralf (2014): *JavaSkript: Die universelle Sprache zur Web-Programmierung*, Carl Hanser Verlag, München.

Straub, Markus; Schiepp, Thomas (2010): *Der Praxisorientierte Weg zum Schlanken Produktionssystem* in: Manz, Carsten (Hrsg.): *Konstanzer Managementschriften*,1.Auflage, Hochschule Konstanz, Technik, Wirtschaft und Gestaltung, Konstanz.

Online-Quellen (19)

Adobe(2016): *Webanwendungen*
https://helpx.adobe.com/de/dreamweaver/using/web-applications.html [Stand: 12.07.2016].

Adaptivepath (2005): Ajax: A New Approach to Web Applications
http://adaptivepath.org/ideas/ajax-new-approach-web-applications/ [Stand: 12.07.2016].

Ascolap (2010): *OPC Unified Architecture - Technical overview and short* description
http://www.ascolab.com/images/stories/ascolab/doc/ua_whitepaper_technicaloverview_e.pdf [Stand: 07.09.2016].

Beckhoff (2016): *Beckkoff Informations System*
https://infosys.beckhoff.com [Stand: 03.07.2016].

Bundesamt für Sicherheit in der Informationstechnik (2016): *Sicherheitsanalyse OPC UA* https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/OPCUA/OPCUA.pdf?__blob=publicationFile&v=2 [Stand: 07.09.2016].

Frauenhofer Institut (2013): *OPC Unified Architecture - Wegbereiter der 4. industriellen (R)Evolution* http://www.iosb.fraunhofer.de/servlet/is/21752/OPC-UA-Wegbereiter-der-I40.pdf?command=downloadContent&filename=OPC-UA-Wegbereiter-der-I40.pdf [Stand: 07.09.2016].

Hochschule Koblenz (2016): *Lean Manufacturing* https://www.hs-koblenz.de/einrichtungen/zentrum/angebote/lean-manufacturing/ [Stand: 03.07.2016].

Leanion (2016): *Das Pull Prinzip - Das Kernelement des Lean Managements* http://www.leanion.de/lean-produktion/das-pull-prinzip.html [Stand: 03.07.2016].

MatrikonOPC (2009): *OPC – was ist das eigentlich? : Das OPC-Handbuch für Jedermann* http://www.matrikonopc.com/portal/downloads/whitepapers/Guide_to_OPC_DE.pdf [Stand: 03.09.2016].

Munstermann, Marco (2006): *Datenaustausch in der Prozessautomatisierung Teil 2*. http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master05-06/munstermann/abstract.pdf [Stand: 03.09.2016].

Microsoft Technet (2016): *Distributed Component Object Model* https://technet.microsoft.com/en-us/library/cc958799.aspx [Stand: 06.09.2016].

OPC Foundation (2015): *OPC Unified Architecture - Interoperabilität für Industrie 4.0 und das Internet der Dinge* https://opcfoundation.org/wp-content/uploads/2015/04/OPC-UA-Interoperability-For-Industrie4-and-IoT-DE1.pdf [Stand: 07.09.2015].

Portmann, René (2007): OPC UA https://prof.hti.bfh.ch/uploads/media/Feldbus_OPC_UA_070122.pdf [Stand: 07.09.2016].

Heinemann, Jürgen (2014): *(HTTPS) Hypertext Transmission Protokol Secure* http://www.hjcms.de/pub/HypertextTransmissionProtokolSecure.pdf [Stand: 15.09.2016].

SOFORT (2016): *SOFORT ÜBERWEISUNG – API* DOKUMENTATION https://www.sofort.com/integrationCenter-ger-DE/content/view/full/2513#h1[Stand: 15.09.2016].

Swift Kanban (2016): *Kanban Board* http://www.swiftkanban.com/kanban/what-is-a-kanban-board/ [Stand: 03.07.2016].

PayPal (2016): *PayPal Developer Documentation* https://developer.paypal.com/docs/classic/products [Stand: 15.09.2016].

Siemens AG (2010): Programmierung eines OPC UA .NET Client mit C# für den Simatic Net OPC UA Server https://cache.industry.siemens.com/dl/files/088/42014088/att_4419/v1/42014088_opc_uaclient_doku_v10_e.pdf [Stand: 10.05.2015].

VEZ (2015): *Bargeldloses Zahlen* http://www.vez-epay.ch/bargeldloses-zahlen/ [03.11.2016].

ABBILDUNGSVERZEICHNIS

Abbildungsverzeichnis

Abbildungsverzeichnis

ABKÜRZUNGSVERZEICHNIS

ADS	Automation Device Specification
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ANSI	American National Standards Institute
CA	Certificate Authority
CIM	Computer Integrated Manufacturing
CSS	Cascading Style Sheets
DCOM	Distributed Component Object Model
DOM	Document Object Model
DNS	Domain Name Service
ECMA	European Computer Manufactures Association
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HTML	Hypertext Markup Language
IIS	Microsoft Internet Information Services
ISO	International Organization for Standardization
KVP	Kontinuierlicher Verbesserungsprozess
MIME	Multipurpose Internet Mail Extension
OLE	Object Linking and Embedding
OPC	OLE for Process Control
OPC-DA	OPC Data Access
OPC-UA	OPC Unified Architecture
PHP	PHP: Hypertext Preprocessor
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
SQL	Structured Query Language
TLS	Transport Layer Security
UTC	Universal Time Coordinated

URL	Uniform Resource Locator
VPN	Virtual Private Network
RPC	Remote Procedure Call
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XMR	XMLHttpRequest