# Dokumentation des IoT-Projekts Smartcart

**Percy Wünsch**
TH Köln
Gummersbach, Germany
percy.wuensch@gmx.de

**Daniel Weitz**
TH Köln
Gummersbach, Germany
daniel.weitz@smail.th-koeln.de

**Pierre Jack Wilken**
TH Köln
Engelskirchen, Germany
pierre_w@live.com

## EINLEITUNG

Der folgende Bericht behandelt die Entwicklung eines IoT-Device, mit dem Namen SmartCart, im Rahmen einer Wahlpflichtveranstaltung der TH-Köln.

Von der Konzeption der Idee bis zur Fertigstellung eines funktionsfähigen Prototyps und darüber hinaus zu Weiterentwicklungsmöglichkeiten in der Zukunft.

## IDEE

Der Grundgedanke, der sich hinter dem Namen SmartCart verbirgt ist, einen für manche Menschen alltäglichen Prozess, das Einkaufen in einem Supermarkt, zu unterstützen und somit zu erleichtern.

SmartCart, auf Deutsch "Intelligenter Einkaufswagen" ist also genau das was der Name aussagt, ein Einkaufswagen, der wesentlich mehr Funktionen bietet als ein herkömmlicher Einkaufswagen, welcher lediglich dazu dient die Einkäufe durch den Supermarkt bis zur Kasse, und eventuell noch bis zum Auto, zu transportieren.

Durch diese einfache Funktionsweise entstehen aber mehrere Probleme, die SmartCart versucht zu lösen. Das erste Problem muss nicht auf jede Person zutreffen die einen Einkauf tätigt, aber ist durchaus von Relevanz für Menschen die nur über ein begrenztes Budget verfügen oder aber gerne den Überblick über ihren Einkauf behalten möchten.

Das Problem besteht darin, dass man die Summe der Preise aller Artikel die man kaufen möchte im Kopf behalten muss wenn man sein gesetztes oder verfügbares Budget nicht überschreiten möchte. Sicherlich dürfte den meisten Menschen einfaches Addieren im Kopf nicht schwer fallen, sie würden diesen Vorgang aber dennoch als sehr mühselig betrachten.

SmartCart löst dieses Problem indem es alle Artikel die der Kunde einkaufen möchte registriert und ihm die ganze Zeit während des Einkaufs einen Überblick über seinen Einkaufswagen verschafft indem es ihm den Gesamtbetrag dieser Artikel anzeigt. Somit braucht der Kunde nicht ständig alle Preise im Kopf zusammenrechnen und sich die Summe merken und kann sich mit weniger Aufwand seinem Einkauf widmen.

Wenn der Kunde einen Artikel wieder aus dem Wagen entfernt, findet der gegenläufige Prozess statt und die Information wird wieder gelöscht. Dadurch dass SmartCart die Artikel vor dem Gang zur Kasse registriert, wird bereits das nächste Problem gelöst.

Normalerweise beendet der Kunde seinen Einkauf damit, dass er sich zur nächsten Kasse bewegt und die Waren auf das Band legt, damit die Kassiererin sämtliche Artikel einscannen kann und der Kunde diese schließlich bezahlt. Hier entstehen aber mehrere Situationen die das SmartCart vermeiden kann.

Zum einen müssen die Artikel die sich vorher im Einkaufswagen befunden haben wieder ausgepackt werden, und wenn der Kunde mit dem Auto zum Supermarkt gefahren ist, ein zweites mal eingepackt um sie danach zum Auto zu befördern. Das mehrmalige Ein- und Auspacken erzeugt also einen Zeitaufwand der wiederum Folgen hat.

Der Kunde muss weitere Arbeit verrichten und je nach dem wie belebt der Supermarkt ist, bildet sich hinter ihm eine Schlange, die zusätzlich für Stress sorgen kann, wenn eine besonders ungeduldige Person darauf wartet seinen Einkauf zu bezahlen.

Dieser Vorgang ist in seiner Gesamtheit, wenn das SmartCart zum Einsatz kommt, nicht mehr nötig bzw. vermeidbar.

Jedes SmartCart besitzt seine eigene Identität und hat alle Informationen über den Einkauf des Kunden. Wenn sich dieser jetzt zur Kasse bewegt, wird das SmartCart von der Kasse erkannt, liest diese Informationen aus und der Kunde kann direkt ohne Umwege den Bezahlvorgang einleiten. Es entsteht eine Zeitersparnis die allen Beteiligten zugute kommt.

Das Ein- und Auspacken fällt weg, das Scannen an der Kasse ist nicht mehr nötig und das Risiko einer Schlangenbildung wird vielleicht nicht vermieden aber zumindest um einen Großteil reduziert.

SmartCart übernimmt also die Registrierung der Artikel wodurch die Mitarbeiterin an der Kasse überflüssig wird. Der Supermarktbetreiber kann enorme Kosten sparen und zufriedenere Kunden gewinnen.

## TECHNOLOGIEN

### RFID

*Warum RFID?*

Bei der Auswahl der Technologie die für das Scannen der Waren zuständig ist, war die Auswahl vorerst auf RFID oder NFC gefallen. Die momentan handelsüblichen Barcodes vielen aufgrund der Tatsache, dass sie mit dem Barcode über einen Scanner gezogen werden müssen direkt raus.

Ebenso viel dann die Entscheidung zwischen RFID und NFC sehr leicht, da NFC eine maximale Distanz von weniger als 20

Zentimetern erreicht, wohingegen RFID mit passiven Ultra-High-Frequency Tags auf eine Reichweite von über 15 Metern kommt. Natürlich ist dies für unseren Anwendungsfall nicht nötig, aber es bietet uns die Reichweite die wir brauchen.

*Einsatz*

Der Einsatz von RFID findet in den Produkten selbst statt, sie werden mit Tags ausgestattet um sie unterscheiden zu können. Der Einkaufswagen selbst wird dann mit einem oder mehreren RFID Readern ausgestattet um die Produkte zu erkennen die sich momentan in ihm befinden.

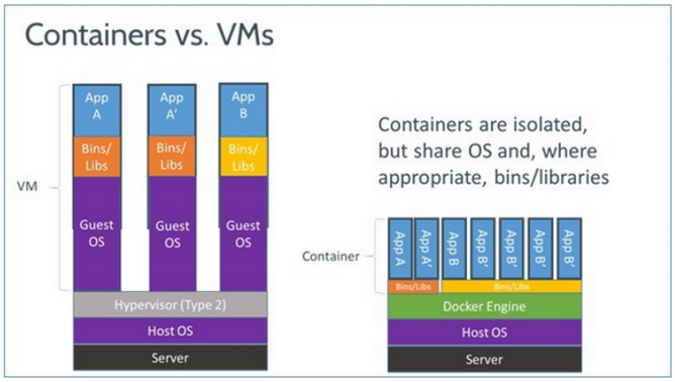
Ein wichtiger Faktor ist hierbei die Unterscheidung zwischen Produkten die sich im Einkaufswagen befinden und welchen die noch im Regal stehen. Lösungsansätze hierfür wäre unter anderem eine grobe Distanz vom Reader zum Tag mittels des Received Signal Strength Indicators(RSSI) zu ermitteln oder auch Produkte erst als im Wagen zu betrachten, sobald sie eine gewisse Zeit lang erkannt wurden.

### Raspberry Pi

Für einen Raspberry Pi als Kernstück unseres Systems haben wir uns entschieden, da dieser aufgrund der sehr umfassenden Community, die Tutorials für viele Komponenten und Schnittstellen bereitstellt, sehr einsteigerfreundlich ist. Ein weiterer Pluspunkt ist das integrierte WLAN-Modul und die sehr geringe Größe.

### Container Basiertes Datenbanksystem mit Docker

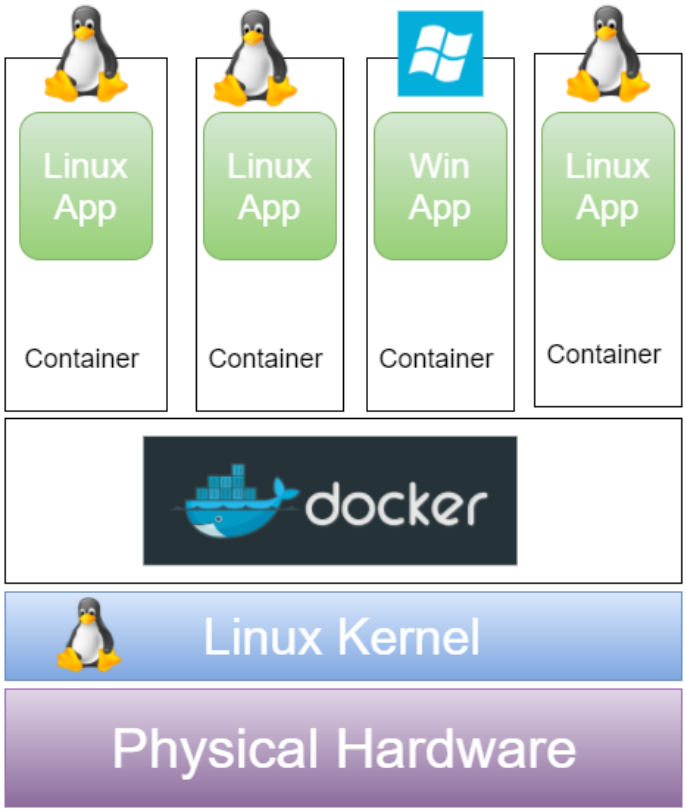
Als Datenbanksystem wurde ein Microsoft SQL Server als Container mit Einsatz von Docker gewählt. Hierfür soll nicht nur einmalig das System als Container eingerichtet sondern ein Dockerfile zur automatisierten Erstellung/Bereitstellung des Systems erstellt werden.




Quelle:http://zdnet2.cbsistatic.com/hub/i/r/2017/05/08/af178c5a-64dd-4900-8447-3abd739757e3/resize/770xauto/78abd09a8d41c182a28118ac0465c914/docker-vm-container.png

### Was ist Docker

Docker stellt gegenwärtig die marktführende Implementierung der Container-Technologie dar. Mit dieser können Anwendungen isoliert mit der Container-Virtualisierung bereitgestellt werden.




Quelle:http://csharpcorner.mindcrackerinc.netdna-cdn.com/article/getting-started-with-docker-and-Asp-Net-co-introduction/Images/1.png

Docker ist für die Virtualisierung mit Linux ausgerichtet, kann aber auch auf Windows und macOS verwendet werden. Unter Windows wird Hyper-V und unter macOS VirtualBox zur Virtualisierung des Docker Hostsystems eingesetzt.

### Was ist die Container-Technologie

Ein Container ist ein abgekapseltes System, das eine einzelne Anwendung mit allen Abhängigkeiten zusammenfasst. Der Unterschied zu einem Virtualisierten Betriebssystem ist, dass jeder Container ohne das Virtualisieren eines Betriebssystems von den anderen Containern unabhängig läuft. Das vermindert den Overhead stark.

Es wird in jedem Container nur der benötigte Kern virtualisiert, da diese im Kontext des Hostsystems laufen. Das Host System stellt bei Docker ein mit Hyper-V virtualisiertes Linux System dar.

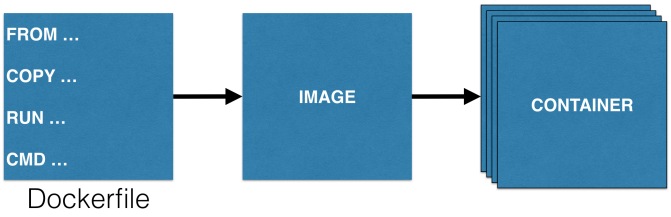
### Dockerfile->Image->Container

Ein Dockerfile ist eine Anleitung zur automatisierten Erstellung von Images für Docker. Als Textdokument beinhaltet dieses alle Befehle die zur Erstellung/Einrichtung des gewünschten Images aufgerufen werden.

Dies kann auch das Aufrufen von weiteren Shell Skripten beinhalten.

Ein Container ist eine aktive Instanz eines Images, der Container wird also aktuell ausgeführt. Sobald der Container keine

Aufgabe oder Beschäftigung mehr haben sollte, wird dieser automatisch beendet.




Quelle:http://blog.arungupta.me/wp-content/uploads/2016/06/docker-lifecycle.png

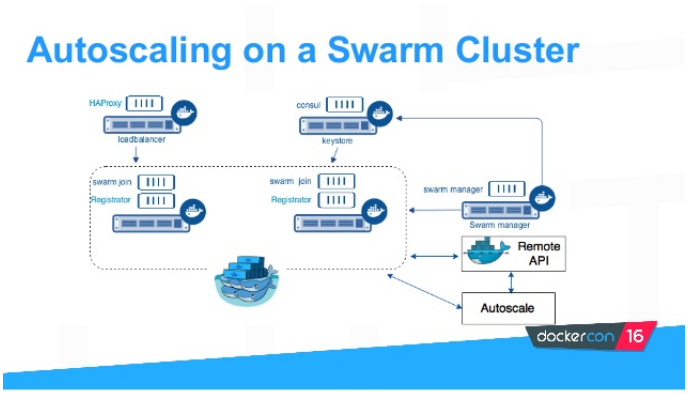
### Wieso ein Container basiertes Datenbanksystem

Der Einsatz von Container basierten Systemen bringt, wenn richtig eingesetzt, große Vorteile mit sich. Diese haben zwar bei der Entwicklung unseres Prototypen zum Teil noch keine Auswirkung, aber im Hinblick auf die Weiterentwicklungsmöglichkeiten sind diese notwendig.

Folgende die Weiterentwicklungsmöglichkeiten beeinflussende Vorteile gegenüber eines typischen Datenbankservers liegen vor.

*Skalierbarkeit*

Unser Datenbanksystem ist dank des Einsatzes von Docker skalierbar. Es lässt sich das Dockerfile bspw. in der Amazon-Cloud als Schwarm deployen.



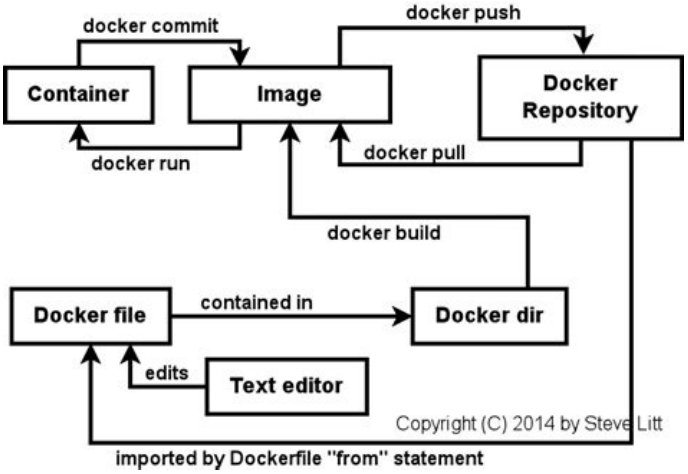

Quelle:https://image.slidesharecdn.com/dc16-present-kfaliagkas-160627163550/95/autoscaling-docker-containers-by-konstantinos\tolerance9999\emergencystretch3em\hfuzz.5\p@\vfuzz\hfuzz-faliagkas-docker-birthday-3-app-challenge\tolerance9999\emergencystretch3em\hfuzz.5\p@\vfuzz\hfuzz-winner-7-638.jpg?cb=1467045575

*Portierbarkeit*

Das Datenbanksystem kann als Container, per Befehl als Image gesichert und auf einem anderen Docker System wieder eingerichtet werden.

*Automatisierte Einrichtung*

Das Datenbanksystem kann mit Hilfe des Dockerfiles, auf einem beliebigen Dockersystem per Kommando als Image erstellt und daraufhin als Container gestartet werden. Die Einrichtung der Datenbank wird ebenfalls automatisiert anhand der Instruktionen im Dockerfile vorgenommen.




Quelle:http://www.troubleshooters.com/linux/docker/images/term_diag.svg

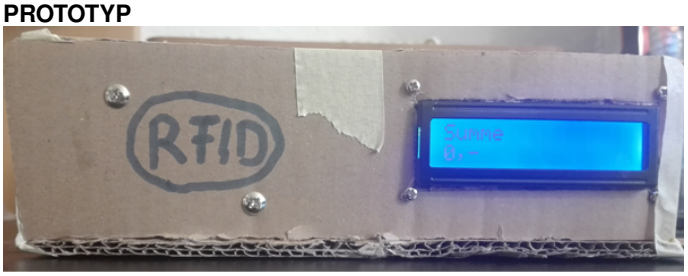
### Wieso MS SQL Server

Unser Projekt weist für das Datenbanksystem an sich, keine besonderen Ansprüche auf. Hierfür würde sich jede der populären relationalen Datenbanksysteme eignen.

Wir haben uns der Einfachheit Halber für einen MS SQL Server entschieden, da dieser momentan das 3. populärste Datenbanksystem ist, alle Anforderungen abdeckt, es 2 offiziell unterstützte Treiber für Python gibt und die Community Version kostenlos ist.

Aufgrund der anfallenden Lizenzkosten die bei einem produktiven Einsatz des Systems entstehen würden, würden wir uns nachträglich für einen MySQL Server entscheiden. Dieser bietet ebenfalls einen Python Treiber an, deckt alle Anforderungen ab und ist mit der Position 2 der populärsten Datenbanksysteme das populärste Open Source Datenbanksystem weltweit.

## PROTOTYP




### Überblick

Unser Prototyp besteht aus 5 Komponenten, von denen 4 Hardware sind. Einem Raspberry Pi 3 Model B als Grundlage und Kernstück des Systems, einem 16x2 LCD zur Anzeige der aktuellen Einkaufssumme, einem Akku Pack für die mobile

Stromversorgung, einem RFID Reader zum Lesen der Waren und unserem Datenbanksystem.

Die Hardwarekomponenten befinden sich in einem 20x20x6 cm Pappgehäuse, da unser erster 3D-Druck leider schiefging und nicht mehr genügend Zeit blieb einen neuen Versuch zu unternehmen.

Leider war bei diesem Gehäuse die Höhe des unteren Einschiebefachs um 1-2mm zu niedrig.

Bei niedrigster Druck-Qualtitätseinstellung betrug die Druckzeit circa 8,5 Stunden.

*Gehäuse aus Pappe*


Da wir nur einen RFID-Reader mit einer relativ kurzen Reichweite erwerben konnten, musste das Prinzip des Lesens der Waren abgeändert werden. Anstatt sie einfach nur in den Einkaufswagen zu legen, müssen sie hier noch über Reader gezogen werden, damit dieser sie erkennen kann.

Ebenso haben wir das Bezahlen der Einfachheit halber auf das Lesen eines bestimmten RFID-Tags reduziert.

### RFID

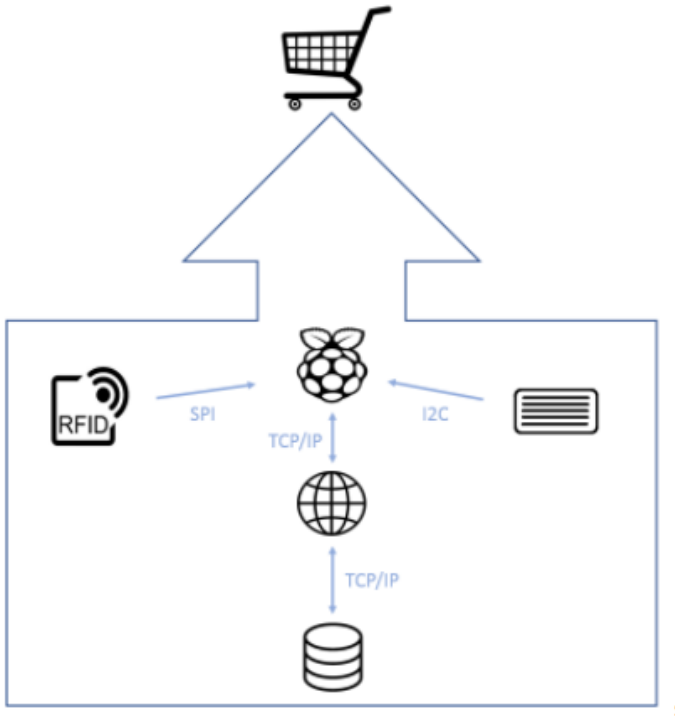
Aus Kostengründen mussten wir uns für den Prototypen leider mit dem SBC-RFID-RC522 RFID-Reader der Firma Joy-It begnügen.

Die Chips werden über eine Frequenz von 13,56 MHz beschrieben und ausgelesen, die Kommunikation zwischen dem Reader und dem Raspberry-Pi erfolgt über das Serial Peripheral Interface(SPI).

Zum Ansprechen des Readers benutzen wir die vorgefertigte SimpleMFRC522 Klasse(https://github.com/pimylifeup/MFRC522-python), die es uns ohne viel Aufwand erlaubt Daten auf die Chips zu schreiben oder von ihnen zu lesen.

Die Wahl dieses Readers hat uns, wie oben bereits genannt, aufgrund seiner sehr geringen Reichweite etwas in der Funktionalität eingeschränkt.

### Raspberry Pi und Programm




Der Raspberry Pi ist über I2C mit dem Display und SPI mit dem RFID-Reader verbunden und bezieht seinen Strom aus dem Akku Pack über ein ganz normales USB-Kabel. Sobald der Pi hochfährt, verbindet er sich automatisch mit einem verfügbaren WLAN, im Fall der Demonstration war es ein

Handy-Hotspot, startet TeamViewer für den Fall das eine Fernwartung nötig wird und das Hauptskript um den Prototypen zu steuern, ReadCart.py.

Im Hauptskript wird anfangs die Verbindung zur Datenbank hergestellt, es werden Objekte für LCD und Reader erstellt und für ein ShoppingCart-Object(ShoppingCart.py), welches für die Verwaltung der momentan im Korb liegenden Waren zuständig ist.

```
32          while(checkedOut==False):
33              try:
34                  lcd.lcd_print("Summe",1)
35                  lcd.lcd_print(shoppingListSumString,2)
36
37                  id,itemName=reader.read()
38                  if(id==48700008907): #checkout if id matches checkout tag, white card atm
39                      checkedOut=True
40                  else:
41                      temp = Item.Item(itemName, id)
42                      slist.addItem(temp)
43                      shoppingListSum = slist.getListSum()
44                      shoppingListSumString = str(shoppingListSum) + ",-"
45                      sleep(1)
```

Im Hauptteil des Programms werden solange Waren zum Einkaufskorb hinzugefügt bis die Kassenkarte gelesen wird oder wieder entfernt falls die einzigartige ID des RFID-Chips, also das eine spezielle Produkt sich schon im Wagen befindet. Anschließend wird die neue Einkaufssumme auf das Display geschrieben.

### Datenbanksystem

Das Datenbanksystem kann mit Hilfe des erstellten Dockerfiles automatisiert als Image erzeugt werden.

*Dockerfile*

12 lines (10 sloc) | 404 Bytes

```
1  FROM microsoft/mssql-server-linux:2017-GA
2  MAINTAINER Pierre Jack Wilken
3
4  VOLUME /docker-entrypoint-initdb.d
5  EXPOSE 1433
6
7  RUN ln -s /opt/mssql-tools/bin/sqlcmd /usr/local/bin/sqlcmd \
8      && ln -s /opt/mssql-tools/bin/bcp /usr/local/bin/bcp
9  COPY docker-entrypoint.sh /usr/local/bin/
10 COPY docker-entrypoint-initdb.sh /usr/local/bin/
11 COPY smartcart-db.sql /usr/local/bin/
12 ENTRYPOINT ["docker-entrypoint.sh"]
```

Unser Dockerfile setzt auf ein von Microsoft erstelltes linux mssql-server image auf.

Mit Expose geben wir den Port 1433 frei, dieser wird für die Datenbank-Kommunikation verwendet.

Kopieren des Einstiegspunkt-Skriptes und des Datenbank-Initialisierungs-Skriptes in das Container-Datensystem.

Kopieren des Datenbank-Scripts, dieses legt die Struktur der SmartCart-Datenbank fest und füllt diese mit Demodaten. ss Als Einstiegspunkt legen wir das Shell-Skript docker-entrypoint.sh fest.

*Skripte zur Einrichtung*

Das Shell-Skript “docker-entrypoint.sh” dient als Startpunkt, in diesem wird das Shell-Skript “docker-entrypoint.initdb.sh” zur Einrichtung der Datenbank aufgerufen.

In der Datenbank-Einrichtung wird das SA-Passwort gesetzt und der Datenbank-Status geprüft.

Daraufhin wird das Skript “smartcart-db.sql” aufgerufen, dieses erstellt die SmartCart-Datenbank, richtet diese strukturell ein und befüllt diese mit Demodaten.

*How to use it*

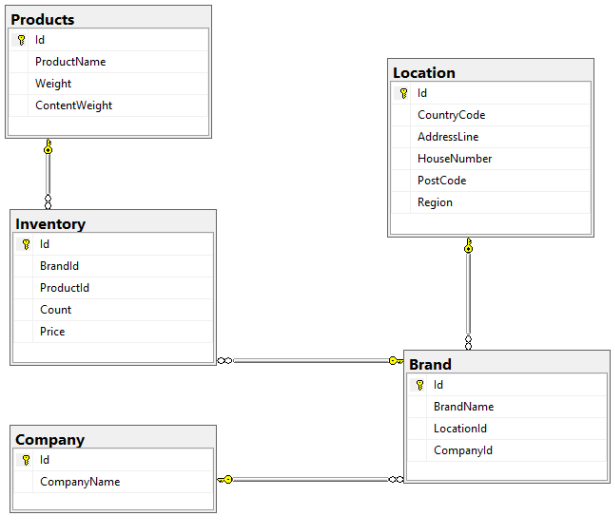
Das Dockerfile muss sich gemeinsam mit den Skripten an einem Platz befinden, zu diesem wird in der CMD “Eingabeaufforderung” gewechselt.

Mit dem Befehl “docker build -t smartcart .” erzeugen wir mit Hilfe des Dockerfiles unser Image mit dem Namen smartcart.

Als Container ausführen können wir das Image nun mit dem Befehl “docker run -d[1] –name smartcart -p 1433:1433[2] -e ACCEPT_EULA=Y[3] -e SA_PASSWORD=IoT20172018 smartcart”

*Die Datenbankstruktur*

Die Datenbankstruktur würde wir im richtigen Einsatz so nicht komplett übernehmen. Wir würden die Tabelle Company streichen, da es im Realeinsatz nicht passieren wird, das unterschiedliche Unternehmen sich eine Datenbank teilen. Auch auf den Hinblick von Konstrukten wie Mutter und Tochtergesellschaften ist das mehr als unwahrscheinlich.




Das ein Unternehmen mehrere Geschäftsniederlassungen zentral verwalten kann, halten wir sowohl für sinnvoll, wie auch technisch für machbar.

[1] Der Container wird im Hintergrund ausgeführt.
[2] Der Port 1433 des Containers wird auf den Port 1433 des Hostsystems umgeleitet “port forwarding”.
[3] Die Lizenzbestimmungen werden automatisch bestätigt “silent installation”.

*Die Kommunikation mit der Datenbank*

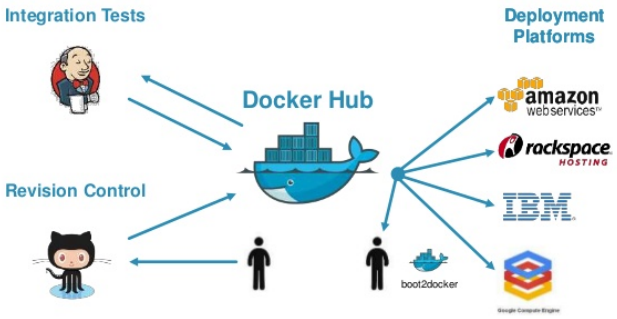
Die Kommunikation zwischen dem MS SQL Server und unserem Raspberry Pi wird in einem Microsoft-Kommunikationsformat formatiert “Tabular Data Stream-Paket (TDS)”.

In der Netzwerkprotokoll Ebene der SQL Server-Netzwerkschnittstelle (SQL Network Interface, SNI) wird das TDS-Paket in einem Standardkommunikationsprotokoll, z. B. TCP/IP oder Named Pipes verkapselt.

Wir verwenden den von Microsoft empfohlenen open-source Pyodbc-Treiber. Dieser vereinfacht den Zugriff auf ODBC Datenbanken und implementiert den DB API 2.0 Standard in der Python-Konvention.

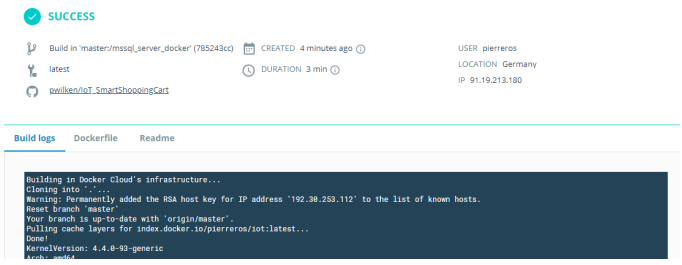
*Das automatisierte Deployen*

Unser Projekt ist auf Github-gehosted, wir besitzen ein Docker-Cloud-Repository und einen Amazon AWS Account. Diese 3 Dinge kombinieren wir um ein automatisches builden und Deployen des Docker-Images als Amazon Cloud Schwarm zu ermöglichen.



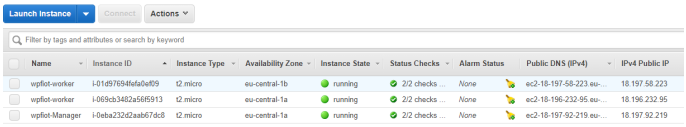

Quelle:http://karenyyng.github.io/MySlideDeck/img/docker_spring2016/Screen+Shot+2014-06-13+at+8.34.10+pm.png

In der Docker-Cloud wird unser Docker-Image, das in unserem Github-Projekt liegt, automatisch gebuilded sobald eine Änderung an diesem oder einer der zugehörigen Skripte gepusht wurde.




Quelle: Screenshot des Docker Cloud UIs

In der Docker Cloud haben wir einen Schwarm, speziell für Amazon AWS, verknüpft mit unserem AWS Account, erstellt.




Quelle: Screenshot des Amazon AWS UIs

## MÖGLICHE WEITERENTWICKLUNG

Die Weiterentwicklungsmöglichkeiten lassen sich in die folgenden speziellen Teilbereiche gliedern.

### Die Device-Entwicklungsmöglichkeiten

Als Hauptpunkt erkannten wir die Erweiterung der Interaktions- und Reaktionsmöglichkeiten des Devices mit dem Anwender und des Anwenders mit dem Device sowie die Verbesserung der Produkterkennung.

*Verbesserung der Aktoren*

Empfehlenswert ist ein auditives Feedback wenn ein Produkt erkannt wurde. Oft wurde ein Produkt über den Scanner gezogen und der Proband wusste nicht ob der Vorgang erfolgreich war und was er jetzt genau tun sollte. Ein einfacher Piepton wäre unserer Meinung nach hier ausreichend.

Ein weiterer Punkt wäre die Ersetzung des momentanen Displays, auf dem nur die Summe angezeigt wird durch ein größeres Display, auf dem man in einem GUI auch die momentanen Produkte angezeigt bekommt und Informationen zu ihnen.

Ebenso wäre die Erweiterung des Systems um eine Smartphone-App wünschenswert, auf der Kunden sich zum Beispiel ihre letzten Einkäufe anzeigen lassen können, aber auch Sachen wie Allergien oder Budgets einstellen können, bei denen der Einkaufswagen eine visuelle und auditive Benachrichtigung von sich gibt und den Kunden falls nötig warnt.

*Verbesserung der Sensoren*

Ein wichtiger Teil der Idee von SmartCart ist es, dass Kunden Artikel einfach in den Wagen legen können und diese automatisch erfasst werden. Aufgrund seiner Reichweite war dies mit dem von uns eingesetzten RFID-Reader nicht möglich. Für die Registrierung musste der Artikel an einem festen Punkt des Prototyps mit wenigen Zentimetern abstand gehalten werden. Es sollte also ein leistungsstärkerer RFID-Reader zum Einsatz kommen um der Idee gerecht zu werden.

Zusätzlich sollte der Boden die Funktion einer Waage besitzen, um Ungenauigkeiten des RFID-Sensors ausgleichen zu können und sicherzugehen, dass nur Produkte erkannt werden die auch wirklich im Einkaufswagen sind.

Wichtig: Es ist ausführlich zu testen, wie stark die Fülle des Einkaufswagen die Funktionalität der RFID-Reader einschränkt.

Für die ersten Tests mit dem Prototyp wurden verschiedene Artikel mit RFID-Chips versehen, indem sie von außen an der Verpackung befestigt wurden. Unter diesen Artikeln befand sich auch Obst, bei dem ein auf diese Weise angebrachter Chip keine geeignete Lösung sein kann.

Der Preis von Obst und Gemüse berechnet sich in der Regel aus dem Gewicht und nicht aus der Stückzahl. Dieses Problem macht also den Einsatz einer Waage erforderlich. Die herkömmliche RFID-Chips scheiden wie bereits erwähnt aus. Aber auch die manuelle Eingabe durch den Kunden ist kein gewünschtes Ziel, denn sie verletzt die Idee des Smart Cart von Einfachheit und Selbstständigkeit.

Ein Lösungsansatz wären Drucker für RFID-Etiketten. Der Kunde legt wie gewohnt das Obst auf die Wage, aber statt einem Barcode erhält das Etikett ein RFID-Tag mit Art des Obstes oder Gemüses und Gewicht, dass die im Wagen verbauten Reader einfach erkennen können.

Als Absicherung, dass Kunden nicht Etiketten drucken und dann nachträglich weitere Produkte in die Tüte legen, würde hier ebenfalls die oben bereits angesprochene Waage dienen, die Unterschiede zwischen Etikett und Realität feststellen könnte. Drucker und Waage befinden sich aber außerhalb des Einkaufwagen wodurch sie keinen Einfluss auf den Prototyp haben.

*Wechsel zu kostengünstigeren Device*

Der Bedarf an Rechenleistung und Funktionalität des Prototyps liegt weit unter dem was ein Raspberry Pi zu bieten hat. Es wäre also ratsam zu weniger leistungsstarken Komponenten zu wechseln um Ressourcen und somit Kosten zu sparen.

Eine Überlegung wäre zu einem Arduino zu wechseln, aber ob damit der Bedarf des Prototyps gedeckt werden kann müsste zuerst untersucht und getestet werden.

Eine andere und gleichzeitig optimale Lösung wäre es, ein eigenes eingebettetes System zu entwickeln. Dieser Weg wäre aber nur mit hohen Entwicklungskosten und dem notwendigen Know-How zu bestreiten, der unsere derzeitigen Möglichkeiten übersteigt.

### Cloud-Einsatz

*Cloud Device Management*

Eine sinnvolle Möglichkeit die SmartCart Devices zu verwalten, wäre der Einsatz eines Cloud IoT Device Managements.

Hier ist bspw. das AWS IoT Device Management empfehlenswert.

"Mithilfe von AWS IoT Device Management werden die Einbindung, Organisation, Überwachung und Fernverwaltung von verbundenen IoT-Geräten im großen Stil zum Kinderspiel. Mit IoT Device Management können Sie Ihre Geräte einzeln oder im Massenverfahren registrieren und durch die richtige Verwaltung von Berechtigungen dafür sorgen, dass die Geräte stets sicher sind. In der IoT Device Management-Konsole können Sie anschließend Ihre Geräte in Gruppen organisieren, die Gerätefunktionalität überwachen und eventuell aufgetretene Fehler beheben sowie über Fernzugriff Updates an Ihre Geräte senden. AWS IoT Device Management ermöglicht Ihnen die Skalierung Ihrer Geräteflotten und Verringerung der Kosten und des Aufwands einer Verwaltung von umfangreichen IoT-Gerätebereitstellungen."

Quelle:https://aws.amazon.com/de/iot-device-management/

*Cloud Schwarm Liveeinsatz*

Um auch dynamisch schwankenden Leistungsanforderungen entsprechen zu können, empfiehlt sich der Liveeinsatz des vorbereiteten Einsatzes eines Cloud Schwarms.

Hierbei gibt es immer eine ungerade Anzahl an Instanzen die das Management des Schwarms übernehmen (master), bspw. 3 Manager-Instanzen und z.B. 15 Schwarm Instanzen (worker).

Es liegen zwei Besonderheiten vor, die erste ist die Skalierbarkeit, es können Live neue Instanzen hinzu- und abgeschaltet werden und die zweite ist die Ausfallsicherheit/Hochverfügbarkeit des Systems. Selbst bei Ausfall der Leader Master-Instanz, übernimmt eine andere Master-Instanz die Leader-Position ohne Ausfall-Erscheinungen.

Bei Ausfall von Worker-Instanzen,übernimmt eine neue/andere Worker-Instanz dessen Aufgabe. Quelle:https://rock-it.pl/my-experience-with-docker-swarm-when-you-need-it/

Wie schon im Abschnitt Technologie erwähnt, würden wir in der weiteren Entwicklung auf MySQL umstellen, das wäre auch hierfür nach jetzigem Stand sinnvoll. Quelle:https://mysqlrelease.com/2016/08/trying-out-mysql-in-docker-swarm-mode/

### Automatic Payment / Automatisierte Kasse

Für die Marktreife wäre die Entwicklung eines Kassenterminal-Systems sinnvoll. Dieses wäre wie eine Schranke aufgebaut, die sich öffnet sobald beim durchqueren der Sensoren die Waren automatisiert bezahlt werden konnten.

Quelle:http://www.nfctimes.com/sites/default/files/imagecache/poster/ez-link-icon-u_0.jpg

Das Konzept des Systems ist zum Großteil angelehnt an das weltweit in vielen Ländern eingesetzte MRT- (U-Bahn) Bezahlsystem. Beispielländer sind z.B. Taiwan und Singapur.

Grundsätzlich gibt es bei diesem Konzept nur ein Bedenken, die deutsche Bevölkerung ist generell skeptisch gegenüber digitalen Bezahlvorgängen die in den Alltag eingegliedert werden. Dies stellt auch den Grund dar, wieso Deutschland in diesem Bereich gegenüber vielen anderen Staaten rückständig ist.

#### Künstliche Intelligenz

Richtig interessant und innovativ wird es wenn man das System mit dem Einsatz von künstlicher Intelligenz verbindet.

Hier ist beispielsweise vorstellbar, das die KI den persönlichen Geschmack anhand eines Einkaufs Profils und weiteren Merkmalen erkennt. Passend zu dem persönlichen Profil, der erkannten Stimmung, dem Wetter und etlichen weiteren Variablen, weist diese einem den passenden Weg zu den Zutaten die einen glücklich/zufrieden stimmen. Auf dieses Thema gehen wir näher in der Zukunftsvision aus.

## ZUKUNFTSVISION

#### Der Supermarkt der Zukunft

Ein gutes Beispiel für den Supermarkt der Zukunft, ist der Amazon Go Supermarkt in Seattle.

Fasst man den Supermarkt der Zukunft in ein paar Sätzen zusammen, ergibt sich das folgende. Der Supermarkt der Zukunft ist ein möglichst stark automatisierter und sicherer Supermarkt. Er gestaltet das Supermarkt-System sowohl für den Kunden, wie auch für den Unternehmer effektiver.

Bei dem System von Amazon wird eine Objekt und Gesichtserkennungs-Software gekoppelt mit Regalböden die eine Waage-Funktion haben eingesetzt. Bezahlt wird automatisiert beim Verlassen des Geschäfts.

In Zukunft werden sich solche automatisierten Supermärkte mit größter Wahrscheinlichkeit etablieren. Ebenso ist es wahrscheinlich das diese Systeme weitreichender fungieren werden. Es ist ebenso denkbar, das der Supermarkt der Zukunft autonom Waren nachbestellt und diese ebenso autonom zugeliefert bekommt.

Quelle: https://www.golem.de/news/amazon-go-kassenloser-supermarkt-oeffnet-1801-132285.html

Wie wir finden geht die Zukunftsvision weit über den Supermarkt der Zukunft hinaus. Der Supermarkt der Zukunft beschränkt sich lediglich auf den Supermarkt, dies ist sehr innovativ, aber die Thematik ist doch weit umfassender.

Besonders der Einsatz von Künstlicher Intelligenz und dem Aspekt des Internet of Things sowie die immer weiter voranschreitende Personalisierung von technischen Gegenständen, eröffnet neue Horizonte.

Unsere Zukunftsvision ist ein persönlicher allgegenwärtiger Assistent, der unter anderem auch zum Einkauf dient.

In einen Supermarkt gehen und sich Lebensmittel auszusuchen ist zeitaufwendig, die heutige moderne Variante online Lebensmittel zu kaufen ist es jedoch ebenfalls.

In einer Weboberfläche müssen Lebensmittel einzeln zusammengesucht werden, die beliebte Alternative sich von online Anbietern Lebensmittelpakete für komplette Gerichte zu bestellen ist zwar zeitsparend aber wenn ich bspw. ein spezielles Gericht haben möchte, stößt dieses schnell an seine Grenzen.

#### Der Assistent der Zukunft

Der Assistent der Zukunft ist wahrscheinlich der Urenkel von Alexa, Cortana und Siri.

Das Prinzip ist das gleiche, nur auf einem ganz neuen Level.

*Was macht ihn aus*

Er ist effektiv, sobald der Assistent in der Lage ist, in den meisten Fällen schneller als wir an einer Tastatur zu sein, sehen wir diesen Punkt als erfüllt.

Er ist zuverlässig, er versteht den Anwender in allen oder fast allen Fällen und fällt in 99,9% der Fälle nicht aus.

Er ist verbunden mit anderen Geräten (z.B. dem Kühlschrank) und dem Internet.

Er kennt den Anwender und seine Bedürfnisse und wahrscheinlich wird dieser in der Zukunft sogar einen vermenschlichten Charakter haben.

Gut dargestellt in dem Film “Blade Runner 2049”.

*Beispiel Szenario*

Durch den Satz “Bestell bitte für die Party am Samstag alles notwendige” wird gecheckt wer alles für Samstag zugesagt hat, wer vielleicht kommt, was diese mögen und anhand von vergangenen Erfahrungen passend eingekauft.

#### Ein intelligentes lokales sowie globales Warenwirtschaftssystem

In der Zukunft könnte die künstliche Intelligenz den Warenverbrauch der Bevölkerung vorhersagen und anhand dessen Produktionsanforderungen versenden.

Es würde in etwa nur das Produziert werden, das wirklich benötigt wird.

## FAZIT

Aufgrund des schmalen Zeitfensters und der Budgetbegrenzung, gestaltete sich die Findung einer Idee als ein Prozess bei dem viel Kreativität und Fantasie gefragt war.

Sowohl aus der kreativen, wie auch aus der technischen Sicht, war die Findung einer IoT-Idee sowie die Entwicklung eines Prototypen sehr lehrreich.

Der erste Prototyp hat die anfangs geplante Funktionalität erfüllt. Während der Testläufe mit an der Entwicklung unbeteiligter Personen, sind aber diverse Schwachstellen deutlich geworden wodurch der Prototyp als Basis für viele Verbesserungen und Möglichkeiten der Weiterentwicklung dient.

**Moralische Bedenken**

Angenommen SmartCart würde flächendeckend zum Einsatz kommen. Der Bedarf an Mitarbeitern an der Kasse fällt weg und viele Menschen würden wahrscheinlich ihre Arbeit verlieren.

Laut Bundesinstitut für Berufsbildung befindet sich der Ausbildungsberuf des Einzelhandelskaufmann/-frau nach Neuabschlüssen 2016 auf Platz 2 der Rangliste mit 25191 und der des Verkäufer/-in auf Platz 3 mit 23.850 abgeschlossen Ausbildungsverträgen.

Man kann nicht genau sagen, wie viele davon im Supermarkt arbeiten da hier der gesamte Einzelhandel in Deutschland mit einbezogen wird, aber es ist vermutlich ein nicht geringer Anteil.

SmartCart würde sich also in die Reihe der Technologien einfügen die in Zukunft Menschen ihre Arbeit abnimmt und somit "überflüssig" macht. Also genau an der Entwicklung beteiligt ist, die unsere Gesellschaft vor große Herausforderungen stellen wird, da der Anteil an Arbeitsplätzen der wegfallen würde ungleich höher ist, als der der Arbeitsplätze schafft. Man könnte also sagen, SmartCart erleichtert auf der einen Seite das Einkaufen und spart dem Unternehmen Geld, lässt auf der anderen Seite aber Arbeitsplätze wegfallen. Was ist nun vorzuziehen?

Wir denken das es zur Natur des Menschen gehört sich durch Entwicklung von Technologie von lästigen Aufgaben zu befreien und sich das Leben zu vereinfachen und es zu verbessern.

Wir vertrauen auf den Einfallsreichtum und die Fantasie des Menschen mit dieser Entwicklung umzugehen und die Technologie für sich zu nutzen um sich das Leben zu erleichtern, wie er es schon immer getan hat.

Quelle:https://www.bibb.de/dokumente/pdf/naa309_2016_tab67_0bund.pdf