Fachhochschule Vorarlberg
University of Applied Sciences

Projektdokumentation

Transportüberwachungssystem Roadrunner

Projektteam: Franziskus Domig, B.Sc.; Stefan Gassner, B.Sc.;
Wolfgang Halbeisen, B.Sc.; Matthias Schmid, B.Sc.

Bearbeitung: Dornbirn, im Sommersemester 2011

Betreuer: Prof. (FH) DI Wolfgang Auer

Zusammenfassung

In dieser Arbeit wird die Erstellung eines Transportüberwachungssystems für Arzneimittel erläutert. Es werden die Komponenten einer mobilen Applikation für die Android Plattform beschrieben, ein Backendserver mit CouchDB erläutert sowie die Überwachung mit einer Webapplikation dargestellt. Um das implementierte System für ein reales Szenario verwenden zu können, werden abschließend die wirtschaftlichen Aspekte analysiert.

Abstract

This paper describes the development of a monitoring and tracing system for pharmaceutical products. Furthermore the development of a mobile application for the Android platform as well as the backendserver with a CouchDB database and the corresponding webapplication are described in detail. An economical reflection on the developed system is given in a concluding section.

Inhaltsverzeichnis

1 Motivation

Ein Transportüberwachungssystem kann aus mehreren Blickwinkeln betrachtet werden. In diesem Semesterprojekt ist für die Erstellung eines vollständigen Systems zu wenig Zeit vorhanden und somit wird in diesem Projekt der Fokus auf die Erstellung einer mobilen Applikation, die Replizierung von Daten auf einen Backendserver sowie die Verwaltung des Systems mit einer Webapplikation gelegt.

Aus oben genannten zeitlichen Gründen wird keine eigene Hardware entwickelt und somit die Android Plattform als Host-System für eine mobile Applikation verwendet. Zudem werden bis auf eine Ausnahme keine realen Sensoren verwendet sondern benötigte Sensoren simuliert.

Dieses Projekt konzentriert sich auf die Entwicklung von Software und wir mit Hilfe neuer Technologien, teilweise sogar in Beta-Versionen, implementiert.

Auf dem Backendserver wird mit CouchDB ein unkonventioneller Ansatz der Datenpersistierung gewählt. Zur Verwaltung von Aufträgen (Lieferungen) dient eine Webapplikation.

Als Softwareentwicklungsprozess wird *Test-Driven-Development* gewählt. Hierzu wurden nahezu alle entwickelten Komponenten mit *Unit-Tests* getestet sowie wenn möglich auf einem *Continuous-Integration*-Server bei jeder Änderung automatisiert getestet.

2 Projektanforderungen

Das Ziel dieses Projekts ist es, ein System zur lückenlosen Transportüberwachung zu entwickeln. Hierzu sollen Produkte, welche erst im Rahmen des Projekts zu spezifizieren sind, überwacht werden. Es soll nach einem Transport möglich sein, eindeutig nachvollziehen zu können, welche Sensor-Daten zu jedem Zeitpunkt aufgezeichnet wurden.

Ausgehend von den Anforderungen an ein System für die Transportüberwachung von Arzneimitteln in Österreich [ARGE Pharmazeutika 07] wird ein System erstellt, welches die Temperatur- sowie die Positionsdaten von Lieferungen bzw. den darin enthaltenen Paketen aufzeichnet und überwacht.

Es soll am Ende dieses Projekts möglich sein, einen einfachen Anwendungsfall vollständig durchzuführen.

3 Systembeschreibung und -architektur

In diesem Abschnitt werden die in diesem Projekt verwendeten Technologien erläutert. Insbesondere werden die Gründe beschrieben, weswegen diese Technologien eingesetzt und anderen vorgezogen werden. Die Vor- sowie Nachteile der entsprechenden Technologien werden gegenübergestellt und besprochen. Zugleich werden die entsprechenden Technologien auf ihre Tauglichkeit in einem real logistischen Szenario geprüft.

3.1 Mobiles Gerät auf Basis von Android

Bei diesem Transportüberwachungssystem wird laut den in Kapitel 2 spezifizierten Anforderungen, eine lückenlose und ständige Überwachung sichergestellt werden. Somit müssen Sensorendaten laufend auf ein Backendsystem übertragen werden. Hierzu bietet sich die *Android*[1] Plattform als Grundlage für eine mobile Applikation an.

Mit Android als Grundlage können gleich mehrere Aspekte abgedeckt werden. Jedes Fahrzeug wird mit einem Android Smartphone ausgestattet und ist somit nicht nur telefonisch erreichbar sondern gleichzeitig kann die komplette Transportüberwachung damit realisiert werden.

Mit der entwickelten Applikation können Gegenstände als in einem Transportmittel "einladen" werden. Hierzu werden diese via Barcode-Scanning in das System übernommen werden. Ab diesem Zeitpunkt wird dieser Gegenstand ständig überwacht und via CouchDB (siehe Abschnitt 3.2) mit dem Backensystem synchronisiert. Auf der Webapplikation (siehe Abschnitt 3.3) können die Produkte bzw. die Lieferung, welche aus mehreren Produkten bestehen kann, auf einer Karte nachverfolgt, sowie die jeweiligen Daten der Temperatursensoren (siehe Abschnitt 3.4) bis zum Entladevorgang überprüft werden.

Auch in einem realen Szenario lässt sich Android hervorragend einsetzen. Es ist mittlerweile in Version 3.1 (15. Juni 2011) verfügbar und weit verbreitet. Die von Google bereitgestellten *Google Apps for Business*[2] ermöglichen eine Administration der Smartphones per Fernwartung. Dabei können auch Applikations-Updates an alle registrierten Smartphones verteilt werden. Somit ist auch eine einfache Lösung für das Deployment von neuen Versionen gegeben.

[1] vgl. `http://www.android.com/`

[2] vgl. `http://www.google.com/apps/intl/de/business/index.html`

Im Kapitel 4 wird die in diesem Projekt erstellte Android Applikation beschrieben.

3.2 Verteiltes Datenbanksystem CouchDB

Bei einem Transportüberwachungssystem ist die Datensicherung ein wichtiger Aspekt. In diesem Abschnitt wird die Datenverwaltung betrachtet und erläutert welches System für dieses verwendet wird.

In diesem Projekt wird durch die in Kapitel 2 spezifizierten Anforderungen ein Fokus auf die Verteiltheit des Systems gelegt. Es fallen durch die mobile Transportüberwachung Daten auf mobilen Geräten an, welche mit einem Backendsystem synchronisiert werden müssen. Aus diesem Grund wird für das Datenbanksystem kein klassisches System in Betracht gezogen. Um einen neuen Ansatz in der Datenpersistierung zu erlernen, wurde das verteilte und dokumentbasierte Datenbankmanagementsystem *CouchDB*[3] verwendet.

In einem real-logistischen Szenario muss auf die Skalierbarkeit sowie die Robustheit von CouchDB betrachtet werden. Klassische relationale Datenbankmanagementsysteme bringen durch die bereits sehr gut entwickelten Versionen vor allem einen Vorteil in der Robustheit und Stabilität. Dennoch, CouchDB wird bereits seit 2005 entwickelt und liegt aktuell in der stabilen Version 1.1.0 (6. Juni 2011) vor und wird bereits in mehreren kommerziellen Projekten wie beispielsweise in Ubuntu [Murphy 09][S. 1] eingesetzt.

Einen detaillierten Überblick der Verwendung von CouchDB in diesem Projekt wird in Kapitel 5 gegeben.

3.3 Webapplikation mit dem Framework Silex

Um ein benutzerfreundliches und einfaches Backendsystem für dieses Projekt zu erstellen, wird auf mehrere bereits bestehende Frameworks zurückgegriffen. *Silex*[4] ist ein Mikroframework für PHP 5.3. Es basiert wiederum auf dem Kern des *Symfony2*[5] Frameworks.

Mit diesem Framework lassen sich einfache Webapplikationen sehr effizient in einer Model-View-Controller Architektur implementieren. Durch die schöne Trennung

[3] vgl. `http://couchdb.apache.org/`
[4] vgl. `http://silex-project.org`
[5] vgl. `http://symfony.com`

der jeweiligen Schichten sowie der leichten Testbarkeit ist Silex für dieses Projekt hervorragend geeignet.

Für ein Szenario in der Realität kann Silex sehr gut eingesetzt werden, solange das System einfach und klein ist. Bei steigender Anzahl an implementierter Anwendungsfälle im Backendsystem, sollte ein Wechsel zu Symfony2 in Betracht gezogen werden, da sich damit deutlich komplexere Anwendungsfälle implementieren lassen. Ein solcher Wechsel ist durch die bereits in Silex verwendeten Komponenten von Symfony2 leicht zu vollziehen, da die bereits bestehenden Komponenten weiterverwendet werden können.

In Kapitel 6 wird eine detaillierte des Backendsystems gegeben.

3.4 Sensoren-Simulation mit Node.js

Für dieses Projekt werden Temperatursensoren sowie Zeitsynchronisation mit Hilfe von *Node.js*[6] simuliert. In diesem Projekt liegt das Augenmerk vor allem auf der mobilen Applikation mit Android, unter Verwendung einer verteilten Datenbank, sowie der Webapplikation. Somit werden bis auf Positionssensoren (GPS) des Android Smartphones keine realen Sensoren verwendet.

Node.js ist ein ereignisgesteuertes I/O Framework für die V8 JavaScript Engine [Wikipedia 10a]. Diese Engine ist in C++ sowie JavaScript entwickelt und liegt in einer MIT-Lizenz vor, welches es für dieses Projekt einsetzbar macht und zugleich auch in einem realen Szenario eingesetzt werden könnte.

Mit Node.js können mit wenigen Zeilen Code Server-Applikationen programmiert werden. Es wird hierzu auf einem Interface (IP) sowie einem beliebigen Port eine JavaScript Callback-Funktion registriert, welche bei einem Zugriff aufgerufen wird. Dies macht es sehr einfach, Sensoren in diesem System zu simulieren, welche via HTTP-Requests “ausgelesen” werden können.

In einem real-logistischen System werden Sensoren nicht simuliert und somit spielt Node.js nur in diesem simulierten Szenario eine Rolle.

In Kapitel 7 werden die in diesem System mit Node.js simulierten Sensoren erläutert. Zugleich werden die entsprechenden realen Sensoren beschrieben, welche in einer nicht simulierten Umgebung verwendet werden könnten.

[6] vgl. `http://nodejs.org/`

4 Android Applikation

Als Hauptsystem in diesem Projekt wird eine Applikation für die Android Plattform erstellt. Diese Applikation dient der mobilen Überwachung von Lieferungen bzw. den Gegenständen einer Lieferung.

Android[7] ist ein Betriebssystem sowie auch eine Software Plattform für mobile Geräte. Es werden Smartphones, Netbooks, Mobiltelefone und Tablets unterstützt. Entwickelt wurde das Betriebssystem von der *Open Handset Alliance*[8], einem Konsortium von 80 Firmen zur Schaffung offener Standards für mobile Geräte, die von Google im Jahr 2007 gegründet wurde (vgl. [Open Handset Alliance 07]).

In diesem Projekt wird Android gewählt, da es die gewünschten Anforderungen an unsere mobile Applikation am besten erfüllt. Ein wichtiger Aspekt für diese Entscheidung ist, dass Android Open-Source ist und ständig verbessert wird.

Android unterstützt ebenfalls CouchDB, welche auch in unserem Backend System einsetzen, was ein weiters wichtiges Kriterium ist. Das Android SDK ist als Plugin für Eclipse[9] verfügbar und ermöglichte es uns somit ein plattformunabhängige Entwicklung in einer gewohnten Entwicklungsumgebung. Ein weiterer Vorteil von Android ist, dass die Smartphones von mehreren verschiedenen Herstellern angeboten werden, was dem Kunden einen Gewissen Spielraum bei der Anschaffung der Smartphones gibt. Ebenfalls ist ein einfaches Deployment, sowie Updates für die Anwendung wichtig. Dies wird mit *adb* oder App-Installer ermöglicht. Ein Nachteil könnte die fehlerhafte Bedienung des Benutzers sein, die die Funktionsweise der mobilen Applikation beeinträchtigten könnte.

4.1 Benutzung

Nach dem sich ein Benutzer mit seinem Benutzernamen, seinem Passwort und der ausgewählten Transporteinheit angemeldet hat, gelangt er auf den Home Screen. Die Benutzeroberfläche des Home Screens ist einfach und intuitiv gestaltet. Der Benutzer hat dann die Möglichkeit Barcodes zu scannen sowie die Lieferdetails zu den aktuell geladenen Paketen anzusehen. Nachdem der Barcode eines Pakets gescannt wurde, kann abhängig vom Status des Paketes (aufgeladen - nicht aufgeladen), das Paket

[7] vgl. `http://www.android.com/`
[8] vgl. `http://www.openhandsetalliance.com/`
[9] vgl. `http://www.eclipse.org/`

geladen, entladen oder abgeliefert werden. Wenn ein Paket abgeliefert wird, kann der Kunde die Lieferung mit seiner Unterschrift bestätigen.

Mit einem Klick auf den Menüpunkt *My Deliveries* sieht der Benutzer die Adressinformation des Senders bzw. des Empfängers für jede Lieferung. Weiters können alle Pakete und deren Status einer ausgewählten Lieferung angezeigt werden. Außerdem kann die Karte mit der eingezeichneten Route und der aktuellen Positionen des Benutzers angezeigt werden.

4.2 Implementierung

Bei der Implementierung der Benutzeroberfläche werden bekannte *UI-Patterns* verwendet. Auf dem Home Screen der Applikation wird das *Dashboard Pattern* verwendet um die Navigation in der Applikation klar und einfach zu gestalten. Des Weiteren enthält jeder Screen eine *Action Bar* im oberen Teil, die dem Benutzer anzeigt in welchem Kontext der Applikation er sich gerade befindet (*Breadcrumb Navigation*). Mit einem Klick auf das Home Symbol in der Action Bar hat der Benutzer jederzeit die Möglichkeit wieder auf den Home Screen zu gelangen.

Die Dienste zur Überwachung der Temperatur und der GPS-Position sowie zum replizieren der Daten mit der serverseitigen Datenbank sind als *Android Services*, die im Hintergrund laufen, implementiert. Somit wird gewährleistet, dass diese wichtigen Dienste unabhängig von der Applikation, die auch vom Benutzer geschlossen werden kann, durchgeführt werden.

4.3 Mögliche Erweiterungen

Mögliche Erweiterungen, die in diesem Projekt nicht umgesetzt werden, wären beispielsweise eine automatische Berechnung der besten Route, abhängig von den zu erledigenden Lieferungen. Weiters könnte die berechnete Route in das Navigationssystem Navigation von Google übernommen werden. Ein weiterer Nutzen für den Benutzer wäre auch die Darstellung der aktuellen Temperaturwerte seiner geladenen Güter, sowie eine Benachrichtigung wenn die Maximale- bzw. Minimale-Temperaturgrenze über- bzw. unterschritten wird. Eventuell könnte in Zukunft auch eine Benachrichtigung an das Fahrzeug, beispielsweise zum Abholen eines Pakets direkt aus der Web-Anwendung gesendet werden.

5 CouchDB Applikation

Apache CouchDB ist ein Dokument-Orientiertes-Datenbanksystem für die Verwendung mit JavaScript. CouchDB bietet inkrementelle Replikation mit bi-directionaler Konflikt-Erkennung und -Lösung.

CouchDB bietet eine RESTfull-API [Fowler 10][S. 1] via *JavaScript Object Notation* (JSON) an, welche von jeder beliebigen Umgebung mit Hilfe von HTTP-Requests abgerufen werden kann. CouchDB ist zusätzlich ein System, welches eine beliebige Skalierbarkeit sowie Erweiterbarkeit anbietet [CouchDB 11][S. 1].

In diesem Projekt wird CouchDB eingesetzt, um ein neues Gebiet der Datenpersistierung zu erlernen. Durch die einfache Replizierung von Daten, kann CouchDB sowohl auf der Backend-Webapplikation (vgl. Kapitel 6), als auch auf den mobilen Geräten (vgl. Kapitel 4) eingesetzt werden.

5.1 Implementierung

5.1.1 JSON und Schema-Validierung

CouchDB verwendet JSON als Dokumentstruktur. Vor der Speicherung eines JSON-Dokuments in die Datenbank werden die Validierungsmethoden von allen Designdokumenten in der Datenbank aufgerufen. Nur wenn alle Validierungen erfolgreich sind wird das Dokument gespeichert. Obwohl JSON schemalos ist, kann trotzdem eine Schema-Validierung durchgeführt werden. Als Schema wird das JSON-Schema verwendet [Internet Engineering Task Force 11][S. 1].

In einem Designdokument können verschiedene Validierungen eingeführt werden. Zusätzlich zu Validierungen der Benutzerrechte werden in diesem Projekt alle Dokumente auf das definierte JSON-Schema validiert.

5.2 Datenverteilung

Daten können auf unterschiedliche Arten in einem System verteilt werden. Eine Möglichkeit ist, die Daten aus der mobilen Datenbankinstanz in die Applikation zu lesen und auf der Applikationsschicht die anfallenden Daten an die Master-Datenbank zu senden.

Mit CouchDB wird die Synchronisierung der Daten in diesem Projekt von den Datenbankinstanzen selbst durchgeführt. Diese Synchronisierung nennt sich Repli-

zierung. Bei dieser kann explizit angegeben werden, welche Daten von welcher Instanz auf welche Instanz synchronisiert werden sollen.

5.2.1 Dokumentänderung - MapReduce

MapReduce[10] ist ein Framework von Google, das entwickelt wurde damit sehr große Datenmengen parallel bearbeitet werden können. CouchDB verwendet ebenfalls einen ähnlichen Ansatz um Daten aus der Datenbank zu lesen. Anhand eines Beispiels wird die Funktionsweise von MapReduce nachfolgend erläutert.

Map - Phase Auf jedes Dokument in der Datenbank wird die Map-Methode angewendet. In einer Map-Methode werden Key-Value-Paare gebildet. Jedes Dokument in der Datenbank kann eine beliebige Anzahl an Key-Value-Paare generieren. Diese Key-Value-Paare werden in einem B-Baum (vgl. [Ottmann 96][S. 317-327]) gespeichert. Ändert sich nun ein Dokument müssen nur die entsprechenden Paare im B-Baum angepasst werden.

Reduce - Phase In dieser Phase wird auf jedem Knoten im Baum die Reduce-Methode angewendet. Ziel der Reduce-Methode ist es die Datenmenge zu minimieren.

5.3 Mögliche Erweiterungen

Mögliche Erweiterungen wären zusätzliche *Views* (ähnlich eines SQL-Views) mit denen bestimmte Informationen aus der Datenbank gelesen werden können. In diesem Projekt wird eine View umgesetzt, welche den Status eines Produkts (geladen, geliefert, etc.) ermitteln kann.

Wichtig ist eine View, die den Status einer ganzen Lieferung ermittelt oder eine View zur Ermittlung an welchen Positionen sich die einzelnen Fahrzeuge gerade befinden. Damit kann die aktuelle Fahrzeugposition beispielsweise auf einer Karte dargestellt werden.

[10] vgl. `http://labs.google.com/papers/mapreduce-osdi04.pdf`

5.4 Verwendung in einem realen Projekt

Bei der Umsetzung in einem realen System liegen möglicherweise bereits Daten des Unternehmens vor. Da relationale Datenbanken weit verbreitet sind, müssen die bereits vorhandenen Daten in CouchDB importiert werden.

Weiters muss die Datenbank per Proxy abgesichert werden, sodass über das Internet nur über klar definierte Schnittstellen auf die Datenbank zugegriffen werden kann. Die externe Erreichbarkeit der Datenbank ist notwendig, da die mobile Android-Applikation die Datenbank erreichen können muss um Dokumente zu replizieren.

6 Webapplikation als Backendsystem

Als unterstützendes Backendsystem wird in diesem Projekt eine Webapplikation erstellt. Hiermit ist es möglich, eine Lieferung mit entsprechenden Gegenständen zu erstellen. Nach erfolgreichem Erstellen einer Lieferung kann diese auf einer Karte nachverfolgt werden. Gleichzeitig kann in einem Diagramm die Temperatur überwacht werden.

6.1 Implementierung

Das Backendsystem ist in der Programmiersprache *PHP*[11] implementiert. PHP ist eine dynamische Skriptsprache, die speziell für den Einsatz auf Webservern entwickelt wurde. Zum schnelleren Entwickeln, werden mehrere Frameworks zur Unterstützung verwendet.

6.1.1 Silex Framework

Silex[12] ist ein auf *Symfony2*[13] basierendes Mikro-Webapplikations-Framework für PHP 5.3. Es bietet eine überschaubare und intuitive API an, ist einfach zu erweitern und durchgängig mit Unit-Tests (PHPUnit[14]) getestet.

In diesem Projekt wird eine Model-View-Controller Architektur (vgl. [Schmidt 09][S. 354]) umgesetzt. Es wird die Erstellung, Bearbeitung sowie Betrachtung von Lieferungen implementiert. Zusätzlich wird eine Verwaltung für Transport-Einheiten (z.B. LKWs) und die Benutzerverwaltung für die mobile Applikation in die Webapplikation integriert.

6.1.2 Doctrine2 ODM für CouchDB

Doctrine2[15] ist ein Framework zur Datenbankabstraktion. Im ursprünglichen Framework war nur ein *Object-Relational-Mapper* (ORM) basierend auf dem *Active-Record-Pattern* [Schmidt 09][S. 380] vorhanden. Dadurch konnten nur relationale Datenbanksysteme (z.B. Oracle oder MySQL) abstrahiert werden. Durch die Ver-

[11] vgl. `http://www.php.net`
[12] vgl. `http://silex-project.org/`
[13] vgl. `http://symfony.com/`
[14] vgl. `http://www.phpunit.de/`
[15] vgl. `http://www.doctrine-project.org/`

wendung einer Dokumenten-Orientierten-Datenbank (sieht Kapitel 5) wird in diesem Projekt allerdings ein *Object-Document-Mapper* (ODM) benötigt.

Dafür bietet sich eine Erweiterung von Doctrine2 durch einen ODM für CouchDB an, welcher sich allerdings erst in einem frühen Alpha-Stadion befindet. Dennoch wird diese Erweiterung verwendet und in einigen Teilen sogar verbessert.

6.1.3 JavaScript Framework jQuery

Das JavaScript Framework *jQuery*[16] ist eine schnelle und einfach zu bedienende Bibliothek um in HTML-Dokument Manipulationen, Ereignis-Behandlung, Animationen sowie Effekte und Ajax- Interaktionen durchzuführen.

jQuery wurde entwickelt um schneller Webapplikationen zu erstellen. Es wurde der Fokus vor allem auf die Art, wie JavaScript in Webapplikationen verwendet wird, gelegt.

In diesem Projekt wird die Darstellung von Graphen, Karten sowie die Validierung von Benutzereingaben mit jQuery realisiert.

6.1.4 Blueprint CSS Framework

Für die Erstellung eines einfachen *Grid-Layouts* [W3C 11][S. 1] mit Hilfe von *Cascading-Style-Sheets* (CSS) wird in diesem Projekt das CSS-Framework *Blueprint*[17] verwendet.

Hiermit lässt sich schnell ein grobes Grundgerüst für moderne Webapplikationen erstellen. Es verwendet eine ansprechende Typographie und bietet dem Designer bzw. Entwickler einen guten Ansatz für die Layoutgestaltung. Bei Bedarf kann dieses Framework mit einigen Plugins erweitert werden.

6.2 Mögliche Erweiterungen

Aus unternehmerischer Sicht wäre eine Anbindung an eine Kundendatenbank von großer Bedeutung. Durch diese können Lieferungen an wiederkehrende Auftraggeber einfacher eingetragen werden.

Eine andere wichtige Erweiterungsmöglichkeit wäre die Implementierung einer entsprechenden Zugriffskontrolle. Ein entsprechendes Rechtesystem für diese Webapplikation wäre vor allem in einem realen System von großer Bedeutung.

[16] vgl. `http://jquery.com/`

[17] vgl. `http://www.blueprintcss.org/`

6.3 Verwendung in einem realen System

In einem realen System ist möglicherweise bereits eine Backend-Applikation vorhanden, welche um die Komponenten dieses Projekts erweitert werden müsste. Die Backend-Datenbank (siehe Kapitel 5) kann auch an eine andere Applikation angebunden werden. Beispielsweise könnte eine Integration in eine SAP[18] Umgebung erfolgen.

Sollten keine eigene Backend-Applikation im Unternehmen bestehen, sollte diese Webapplikation entsprechend erweitert werden. Beispielsweise muss eine Zugriffskontrolle, eine Kundendatenbank etc. implementiert werden.

[18] vgl. `http://www.sap.com/germany/index.epx`

7 Transportüberwachung mittels Sensoren

In diesem Projekt wird die Transportüberwachung mittels Sensoren, welche von der Android Applikation (siehe Kapitel 4) überwacht werden, realisiert.

Für die in diesem Projekt spezifizierten Anforderungen (siehe Kapitel 2) wird die Temperatur sowie die aktuelle Position eines Gegenstands überwacht. Hierzu werden zwei unterschiedliche Sensortypen, welche in den beiden nachfolgenden Abschnitten erläutert werden, verwendet. Zusätzlich wird die Zeitsynchronisation der mobilen Geräte mittels eigens entwickelter Synchronisierung (wie in Abschnitt 8.1 erläutert) realisiert.

7.1 Temperaturüberwachung

Temperatursensoren werden in diesem Projekt simuliert. Alle benötigten Temperatursensoren werden mit *node.js*, wie in Abschnitt 3.4 erläutert, simuliert.

7.2 Positionsüberwachung

In diesem System werden in einem definierten Zeitintervall die aktuelle Position eines sich auf dem Transportweg befindenden Gegenstands aufgezeichnet. Die Positionsdaten werden von der Android Applikation bzw. der Service Applikation ausgelesen. Hierzu werden die Positionsdaten via GPS bzw. UMTS oder WLAN ermittelt und ebenfalls aufgezeichnet.

Die ermittelten Positionsdaten werden in der Webapplikation bei den Lieferungen jeweils auf einer Karte dargestellt.

8 Applikationssicherheit

Die Anforderungen an die Sicherheit einer Applikation (Programm) sollen bereits zu Beginn der Entwicklung ermittelt und abgestimmt werden. Eine nachträgliche Implementierung von Sicherheitsmassnahmen ist bedeutend teurer und bietet im Allgemeinen weniger Schutz als Sicherheit, die von Beginn an in den Systementwicklungsprozess oder in den Auswahlprozess für ein Produkt integriert wurde.

In diesem Projekt werden die Zeitsynchronisierung sowie die Zugriffskontrolle für CouchDB unter dem Aspekt der Applikationssicherheit betrachtet.

Mit Hilfe der Zeitsynchronisierung wird die Systemzeit zwischen Android Smartphone und Backendsystem abgeglichen um eine einheitliche Zeit gewährleisteten zu können.

Die Zugriffskontrolle für CouchDB dient als Sicherheitsmechanismus um die Daten vor Zugriffen Dritter zu schützen.

8.1 Zeitsynchronisierung

In diesem Abschnitt werden Probleme besprochen, die durch fehlerhafte respektive mangelhafte Zeitsynchronisierung entstehen können.

Problem durch falsche Zeitstempel bei Logeinträgen Betrachtet wird das Szenario "Umladevorgang eines Produktes". Das mobile Gerät der Transporteinheit wird benutzt um den Ausladevorgang aus einem Container im System zu verarbeiten. Mit dem Scannen des Produkts wird auf dem mobilen Gerät der Transporteinheit ein Logeintrag in dessen lokale Datenbank erstellt. Genauso wird beim darauffolgenden Ladevorgang der Umladestation ein Logeintrag auf dessen Gerät erstellt. Wenn das System mit absoluter Zeit arbeitet und die Uhrzeit des Geräts der Transporteinheit vor jener der Umladestation ist, dann würde im System der Übernahmevorgang der Umladestation vor dem Ausladevorgang der Transporteinheit stattfinden.

Lösungsansatz Um dieses Problem zu lösen muss relative Zeit eingeführt und synchronisiert werden. Für die Zeitsynchronisierung können bekannte Algorithmen für verteilte Systeme verwendet werden. Mögliche Algorithmen sind Christian's Algorithmus [Christian 89][S. 146-158] oder Berkley Algorithmus[19].

[19] vgl. `http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?&arnumber=29484`

In diesem Projekt wurde Christian's Algorithmus implementiert. Jeder Client misst seine Differenz zur Serverzeit und verwendet diese zur Erzeugung der Zeitstempel. Somit können die unterschiedlichen Uhren bis zu einer gewünschten Genauigkeit synchronisiert werden.

Die Systemzeit wird bis zu einer gewählten Genauigkeit synchronisiert.

8.2 Zugriffskontrolle

Die Benutzerrechte in diesem Projekt werden mit verschiedene Benutzergruppen abgebildet. Die Benutzerauthentifizierung wird von CouchDB durchgeführt.

8.2.1 Administratoren und System-Benutzer

Auf einer Datenbank können in CouchDB Administratoren und normale Benutzer definiert werden.

Administrator Ein Administrator hat sämtliche Rechte auf der Datenbank. Er kann die Datenbank löschen, Designdokumente verändern oder Benutzerrechte ändern.

Benutzer Ein Benutzer hat lesenden und schreibenden Zugriff auf alle Dokumente bis auf die Designdokumente.

8.2.2 Rollen im System

Einem Benutzer können keine bis mehrere Rollen zugewiesen werden. Bei jeder Veränderung von Dokumenten wird von CouchDB eine Benutzerauthentifizierung durchgeführt. Bei dieser Benutzerauthentifizierung wird das Zugriffsrecht auf die Datenbank überprüft und zudem eine Validierung durchgeführt. Bei der Validierung werden alle definierten Validierungsmethoden aufgerufen. Nur wenn alle Validierungen gültig sind wird die gewünschte Änderung an den Dokumenten durchgeführt.

9 Wirtschaftliche Betrachtung

In diesem Kapitel werden die finanziellen Aufwendungen für ein Unternehmen, welches dieses System einsetzt, betrachtet. Es werden die Kosten der Server-Infrastruktur, der Android Smartphones, der Sensoren und der Tarife betrachtet.

9.1 Infrastruktur für Server/Client

Für das Backendsystem wird ein *Apache* Webserver sowie ein *CouchDB* Datenbankserver benötigt. Bei einem kleinem System, bezogen auf die aktiv benutzten Smartphones und Transportmittel, können der Webserver und der Datenbankserver auf einem einzigen physikalischen Server installiert werden. Bei einem größeren Anwendungsszenario muss dieses System entsprechend skaliert werden. Der Preis für ein *Rackmount* liegt derzeit zwischen 700 bis 1.000 Euro.

9.2 Android Smartphones

Der Kaufpreis für ein Android Smartphone liegt aktuell im einem Bereich von 250 bis 400 Euro. Jeder Benutzer (FahrerIn) benötigt ein eigenes Gerät.

9.3 Temperatur Sensoren

Temperatur Sensoren (Bluetooth) kosten derzeit zwischen 150 und 250 Euro. Jedes Transportmittel muss mindestens über einen Sensor verfügen.

9.4 Tarife

Für Unternehmen gibt es inzwischen in den meisten Ländern Komplettpakete die einen Gesprächstarif sowie unlimitiertes Datenvolumen beinhalten.

Diese Komplettpakete kosten je nach Land und Umfang zwischen 15 und 80 Euro. Dabei gilt es zu beachten, dass sich diese Tarife nur auf das Inland beziehen. Im Ausland fallen Roaming Kosten an, die erheblich teurer sind. Bei den meisten Netzbetreibern gibt es mittlerweile spezielle Roaming Tarife.

Für das Deployment der mobilen Applikation sowie die Administration der Android Smartphones im Unternehmen wird *Google Apps for Business*[20] empfohlen. Die jährlichen Kosten pro Smartphone belaufen sich derzeit auf ca. 40 Euro.

[20] vgl. `http://www.google.com/apps/intl/de/business/features.html`

10 Projektentwicklung

In diesem Projekt wird ein agiler Softwareentwicklungsansatz gewählt. Es wird die Entwicklung von Software in den Vordergrund gestellt und der klassischen, formalisierten Vorgehensweise geringe Bedeutung zugeteilt. Diese Vorgehensweise kann mit der von [Beck 98][S. 25] et al. entwickelten Methode *Extreme-Programming* verglichen werden.

Um diesen, durch fortlaufende Iterationen und den Einsatz mehrerer Einzelmethoden, sich stets ändernden Prozess erfolgreich anwenden zu können, ist eine entsprechende Disziplin sowie Kommunikationsbereitschaft im Team notwendig.

Durch den sogenannten *Best-Practice* Ansatz kann in dieser Art der Entwicklung auf bereits vorhandene Lösungsansätze zurückgegriffen werden. Somit kann der Entwicklungsprozess erheblich beschleunigt werden.

Da die jeweiligen Iterationen nur kleine Änderungen und in der Regel nur ein neues Feature in das System einführen, können auch sehr einfach neue Technologien getestet werden. Sollte sich herausstellen, dass eine neue Technologie nicht die erwarteten Verbesserungen bringt, kann durch diese Vorgehensweise auch wieder schnell auf eine vorherige Iteration zurück gewechselt werden.

Konkret wird in diesem Projekt in abwechselnden Zweierteams entwickelt und jedes neue Feature bzw. neuer Quellcode durch beide Entwickler besprochen und bei Bedarf verbessert.

Mit der Unterstützung eines verteilten Versionskontrollsystem (siehe Abschnitt 11.1) kann auch sehr einfach gleichzeitig an mehreren Features gearbeitet werden.

11 Projektunterstützende Werkzeuge und Hilfsmittel

In diesem Projekt werden mehrere projektunterstützende Werkzeuge verwendet. Durch den sehr agilen Entwicklungsprozess (siehe Kapitel 10) wird vor allem auf eine entsprechende Versionskontrolle sowie ein Continuous-Integration-Server zur Überwachung des jeweiligen Entwicklungsstands geachtet. Diese beiden Systeme sind in den nachfolgenden Abschnitten erläutert.

11.1 Versionskontrollsystem GIT mit github.com

Git ist ein verteiltes Revisions-Kontroll-System, dessen Schwerpunkt auf Geschwindigkeit liegt. Git wurde ursprünglich von Linus Torvalds für Linux-Kernel-Entwicklung [Torvalds 07] entworfen und entwickelt. Jedes Git-Arbeitsverzeichnis ist ein vollwertiges *Repository* mit kompletter Historie und vollständigem Commit-Tracking. Git ist nicht abhängig von einem Netzwerkzugang oder einen zentralen Server, das heißt es kann auch nur lokal entwickelt werden. Git ist freie Software unter der GPL2 (GNU General Public License Version 2) verteilt.

In diesem Projekt wird Git für alle Teile des Projekts eingesetzt. Es wird sowohl die CouchDB Applikation, die Android Applikation, die Webapplikation sowie die Dokumentation via Git verwaltetet. Alle Projektmitglieder können alle Komponenten einsehen und bearbeiten.

Um Git auch Online zu synchronisieren, wird in diesem Projekt gihub.com[21] verwendet. Via github.com ist es möglich, zusätzliche projektunterstützende Werkzeuge (wie z.B. Issue-Tracking, Wiki, etc.) zu verwenden.

11.2 Continuous Integration mit Jenkins

Jenkins ist eine Open-Source-Continuous-Integration (CI)-Tool, welches in Java geschrieben ist. Jenkins bietet kontinuierliche Integrations-Services für Software-Entwicklung an. Es ist ein server-basiertes System mit einem Servlet-Container wie Apache Tomcat. Es unterstützt SCM-Tools wie CVS, Subversion, Git und Clearcase. Zudem werden die Build-Tools Apache Ant und Apache-Maven, sowie beliebige Shell-Skripte und Windows-Batch-Befehle unterstützt. Jenkins ist freie Software und wird unter der MIT-Lizenz (Massachusetts Institute of Technology License) veröffentlicht.

[21] vgl. `http://github.com`

In diesem Projekt wird sowohl die Android Applikation (Java) sowie die Webapplikation (PHP) automatisiert via Jenkins getestet. Bei jedem neuen Commit, welcher zu github.com synchronisiert wird, wird automatisiert ein Checkout von Jenkins in ein neues Verzeichnis erstellt. Danach wird das jeweilige Ant-Build-Script *build.xml* ausgeführt und die Software kompiliert. Anschließend werden die definierten Analysewerkzeuge mit der jeweiligen Konfiguration ausgeführt.

Bei der Android Applikation werden JUnit[22] Tests ausgeführt.

Bei der Webapplikation werden PHPUnit[23], PHP-Check-Style, PHP-Mess-Detector sowie PHP-Copy-Paste-Detector verwendet um die Qualität zu überprüfen.

[22] vgl. `http://www.junit.org/`
[23] vgl. `http://www.phpunit.de/`

12 Fazit

Die in diesem Projekt getroffene Entscheidung, ein “Software Projekt” zu realisieren und keine eigene Hardware zu entwickeln war schlussendlich ausschlaggebend für einen erfolgreichen Projektabschluss.

Durch das Verwenden eines agilen Software-Enwicklungsprozesses konnte zu jedem Zeitpunkt ein lauffähiges Produkt gezeigt werden. Mit den Verwendeten Methoden des Extreme-Programming inklusive eines Test-Driven-Development Ansatzes konnten weitgehend alle Teile dieses Projekt “fehlerfrei” umgesetzt werden.

Mit einem neuen Ansatz der Datenpersistierung in einer Dokumenten-Orientierten-Datenbank wurde viel an Erfahrung gesammelt, sowie eine sehr interessante neue Technologie kennengelernt.

Mit dem Einsatz neuer Tools, Software, Frameworks und Applikationen wurde gezielt das Anti-Pattern “Golden-Hammer” [Brown 98][S. 111] vermieden. Dieses besagt, ein Projekt mit den selben Tools und Systemen zu erstellen, welche sich bereits bewährt hat, führt dazu, dass oftmals die falsche Software eingesetzt wird, weil die Entwickler sich daran gewöhnt hat.

Leider mussten wir in diesem Projekt auch oftmals mit unfertigen Tools und Frameworks arbeiten, was manchmal auch Nachbesserung an diesen von Seiten des Projektteams benötigte. Jedoch hat das “Experimentieren” und “Ausprobieren” neuer Software viel Spaß gemacht und interessante neue Ansätze aufgezeigt.

Die entwickelte Software wäre mit entsprechenden Erweiterungen durchwegs für ein reales Szenario zu verwenden und könnte in einem unternehmerischen Umfeld eingesetzt werden.

Literatur

[Beck 98] K. Beck, W. Cunningham, R. Jeffries: Chrysler Goes To “Extremes”, Distributed Systems: Case Study, S. 25-28 October 1998, online abrufbar: http://www.xprogramming.com/publications/dc9810cs.pdf.

[Brown 98] W. H. Brown, R. C. Malveau, H. W. McCormick III, et al.: Anti Patterns Refactoring Software, Architectures, and Projects in Crisis. New York: John Wiley & Sons, Inc., 1998.

[Christian 89] F. Cristian: Probabilistic clock synchronization, Distributed Computing (Springer) 3 (3), 1998.

[Gamma 94] E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns: Elements of Reusable Object-Oriented Software, MA: Addison-Wesley, 1994.

[Ottmann 96] T. Ottmann, P. Widmayer: Algorithmen und Datenstrukturen-3, Heidelberg; Berlin; Oxford: Spektrum, Akad. Verlag 1996.

[Schmidt 09] S. Schmidt: PHP Design Patterns: Entwurfsmuster für die Praxis., Köln: O’Reilly Verlag, 2. Auflage 2009.

Web-Referenzen

[ARGE Pharmazeutika 07] Arbeitsgemeinschaft des pharmazeutischen Großhandels Österreichs: Codex für den Transport von Arzneimitteln in Österreich http://www.argepgh.at/kwpc_Downloads/Pharmarecht%20%D6sterreich/Codex%20Transport%201007.pdf, besucht am 15.03.2011.

[CouchDB 11] Apache CouchDB: Technical Overview http://couchdb.apache.org/docs/overview.html, besucht am 22.06.2011.

[Fowler 10] M. Fowler: Richardson Maturity Model: Steps towards the glory of REST http://martinfowler.com/articles/richardsonMaturityModel.html, besucht am 20.06.2011.

[Internet Engineering Task Force 11] Internet Engineering Task Force: A JSON Media Type for Describing the Structure and Meaning of JSON Documents http://tools.ietf.org/html/draft-zyp-json-schema-03, besucht am 11.05.2011.

[Murphy 09] E. Murphy: CouchDB in Ubuntu http://mail-archives.apache.org/mod_mbox/couchdb-dev/200910.mbox/%3C4AD53996.3090104@canonical.com%3E, besucht am 16.06.2011.

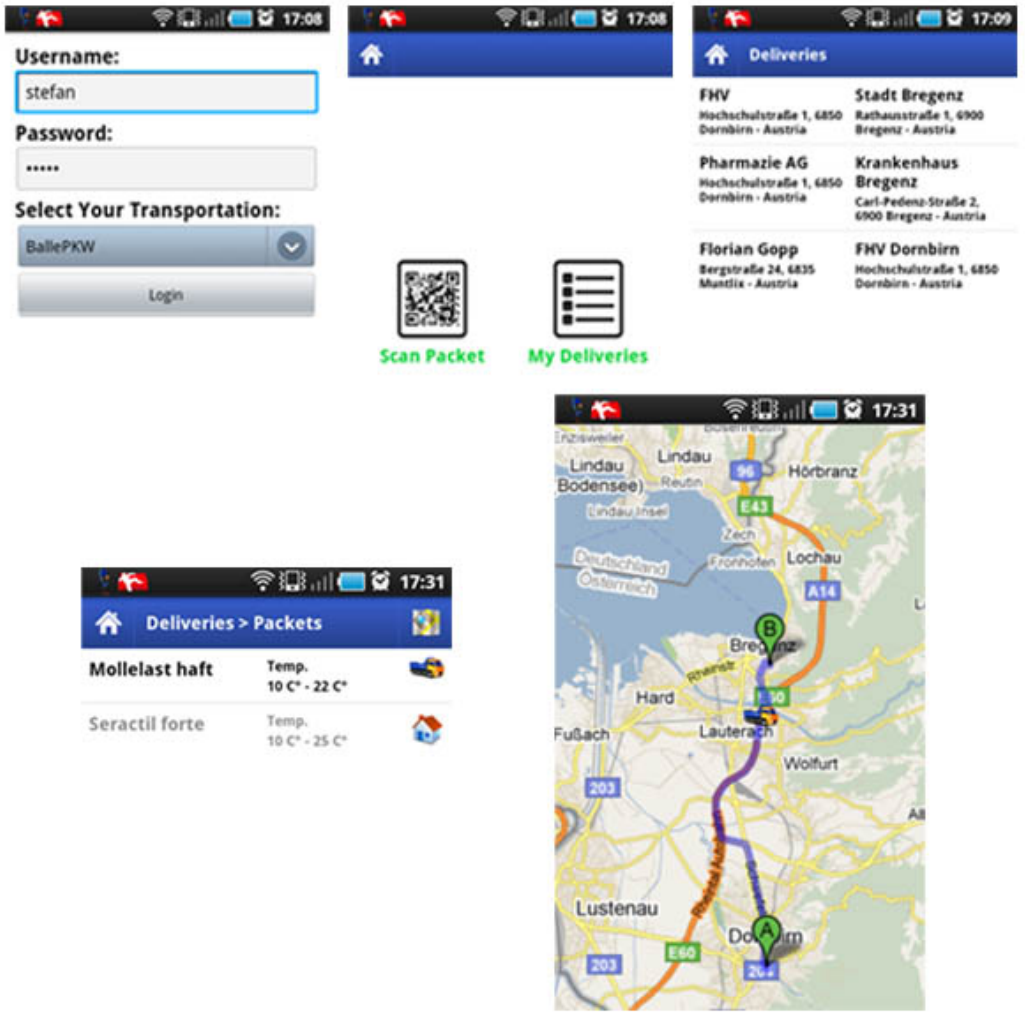
[Open Handset Alliance 07] Open Handset Alliance http://www.openhandsetalliance.com/press_110507.html, besucht am 23.06.2011.

[Torvalds 07] Linus Torvalds (2005-04-07). “Re: Kernel SCM saga..”. The linux-kernel mailing list. “So I’m writing some scripts to try to track things a whole lot faster.” http://marc.info/?l=linux-kernel&m=111288700902396, besucht am 18.06.2011.

[W3C 11] W3C Working Draft 7 April 2011: Grid Layout http://www.w3.org/TR/css3-grid-layout/, besucht am 22.06.2011.

[Wikipedia 10a] Wikipedia: Node.js http://de.wikipedia.org/wiki/Node.js, besucht am 20.04.2011.

A Android Applikation - Screenshots



B Kommerzielle Temperatursensoren

Hygrosens TLOG20-BLUE Das ist *NICHT* unsere Lösung. `http://shop.hygrosens.com/Messsysteme-acma/Messsysteme-fuer-Temperatur/Temperaturmesssysteme/Temperaturmesssysteme-BLUETOOTH/BLUETOOTH-Temperaturmesssystem-20-Kanaele.html` hygrosens.com/TLOG20-BLUE, Zugriff am 16.04.2011

Ampedrf BT11 Das ist *NICHT* unsere Lösung. `http://www.ampedrf.com/modules.htm` BT11 Class1, Zugriff am 16.04.2011 `http://www.ampedrf.com/datasheets/BT11_Datasheet.pdf` BT11 Datasheet

$149 Programmable Universal Key Fob Sensor Wir haben uns für das BlueRadios BR-FOB-SEN-LE4.0 Device entschieden, weil es eine komplette und etablierte Lösung für Temperatur, Beschleunigungs- und Licht-Messung ist. http://www.blueradios.com/BR-FOB-SEN-LE4.0-S2A.pdf Blueradios BR-FOB-SEN-LE4, Zugriff am 16.04.2011

http://www.blueradios.com/hardware_sensors.htm Blueradios BR-FOB-SEN-LE4