Praktikumsbericht

DPS Engineering – Silas Hoffmann

Inhalt

Tätigkeiten der DPS:

Zu Beginn des Praktikums bekam ich eine kurze Einweisung in das Unternehmen. Mir wurden neben sämtlichen Räumlichkeiten Einblicke in diverse Projekte gegeben. Da das Unternehmen in erster Linie im Banking-Bereich tätig ist, wurde mir dies allerdings vor allem verbal erläutert um sensitive Daten nicht unnötig preiszugeben. Ich habe aber auch mehrere Prototypen sowie echte Bankautomaten zu sehen bekommen, die geöffnet wurden damit die Mitarbeiter zum Beispiel am Kartenlesegerät verschiedene Tests durchführen konnten, denn DPS ist nicht nur für das Backend sondern vor allem für die eingebettete Software in Bankautomaten und Kartenlesegeräten zuständig, wie sie zum Beispiel in Supermärkten verwendet werden. Da mein Vorgesetzter allerdings in anderen Bereichen tätig war, hatte ich mit diesem ebenfalls wenig zu tun.

Neben dieser eingebetteten Software, stellt die Firma auch eine Payment Lösung bereit, welche direkt als Schnittstelle zwischen den Banken genutzt wird. In Belgien sind Sie im Bereich der Instant Payment (Echtzeitüberweisungen) bereits Marktführer und streben dies im gesamten europäischen Raum ebenfalls an. Eine Echtzeitüberweisung zeichnet sich dadurch aus, dass sie innerhalb von maximal sieben Sekunden getätigt und vollendet werden kann. Dies ist ein neuer Standard, der in den kommenden Jahren im europäischen Raum etabliert werden soll. Für Überweisungen zwischen Konten derselben Bank stellt dies kein Problem dar, da hierbei lediglich die entsprechenden Datenbanken ausgewählt, und die Tabelleneinträge modifiziert werden müssen. Zwischen verschiedenen Banken stößt man allerdings auf Probleme, da es zu einer Kommunikation kommen muss. Um einen reibungslosen Austausch zu gewährleisten, stellt DPS eine Plattform zur Verfügung, die von den Banken genutzt werden kann. Diese läuft schon seit etwas über einem Jahrzehnt und ist das Hauptprodukt der DPS. Mein Vorgesetzter hatte eine Zeit lang dieses Projekt betreut ist jetzt allerdings auf ein neues Projekt angesetzt worden, an dem ich im Verlauf des Praktikums mitgearbeitet habe. Dieses Projekt beschäftigt sich mit der Digitalisierung von deutschen Behörden.

Praktikumsbeginn / Einarbeitungsphase

Einer der Gründe warum ich bei meinem Betreuer gelandet bin, ist der, dass die Firma ihre Implementierung der Payment Plattform mit neuer Technologie auswechseln möchte. Da dies ein sehr aufwendiges Unterfangen ist, sollte ich beispielhaft eine Anwendung mit diesen Technologien entwickeln um einmal einen groben Überblick zu bekommen, inwiefern dies überhaupt möglich ist und um die Performanz des Ganzen zu prüfen. Momentan läuft die Payment-Plattform Java Enterprise Application-Servern, im Unternehmen wird hierfür die open source Implementierung „Wildfly“ verwendet. Um eine Anwendung zu deployen wird eine .war oder .ear Datei gebaut, welche in den Container geladen wird und erst nach vollständiger Konfiguration bereit zum Start ist. Diese Art des Deployments zeichnet allerdings die monolithische Struktur aus. Es gibt im Normalfall nur diese eine Plattform in der sämtliche Abhängigkeiten einfach importiert werden. Gerade im Hinblick auf eine Continuous Delivery Pipeline, wie es bei der Unternehmens-Größe durchaus sinnvoll ist, stellt dieses System eine Hürde dar. Denn in einem solchen System ist es deutlich schwieriger einzelne Komponenten auszuwechseln. Um eine solche Applikation neu zu bauen muss der komplette Sourcecode gebaut werden oder es wird intern bereits auf kleiner Komponenten zurückgegriffen. Generell ist diese Herangehensweise aber relativ unflexibel. Um gerade diese benötigte Flexibilität abzubilden, werden heutzutage in vielen Unternehmen Architekturen verwendet, die sich auf sogenannte Microservices stützen. Diese Services können als logisch abgetrennte Einheiten eines Systems betrachtet werden, welche gerade im Zusammenspiel mit anderen Komponenten eine Applikation bilden. So ist es zum Beispiel möglich Teile des Systems auszuwecheln, ohne die Applikation überhaupt stoppen zu müssen. Um die abgetrennten Einheiten / Microservices zu realisieren, wird vermehrt auf eine Technologie namens „Docker“ zurückgegriffen. Um die Microservices zu erstellen, greift man hierbei auf sogenannte Docker-Container zurück. Ein Container stellt dabei die Runtime zur Verfügung die von dem Teil der Applikation verwendet wird und erinnert in vielerlei Hinsicht an eine virtuelle Maschine. Wenn man nun einen Service innerhalb eines solchen Containers laufen lassen möchte, generiert man sich zuallererst ein „image“. Dieses Image ist die Beschreibung aus der mehrere konkrete Container Instanzen gestartet werden können. Ein weitere wesentlicher Vorteil ist, dass der Server auf dem die Applikation laufen soll keine weiteren Abhängigkeiten braucht. Es reicht einen Docker-Daemon installiert zu haben um sämtliche Container zu starten oder sogar zu bauen (Stichwort multi-stage-build).

Um mich mit der Technologie vertraut zu machen, sollte ich in den ersten Wochen ein einfaches Producer – Consumer- Muster implementieren, in der eine Komponente eine Nachricht an einen Broker übergibt und eine zweite Komponente diese empfängt und verarbeitet. Für die Implementierung der Komponenten habe ich das Spring Boot Framework verwendet. Dieses stellt im Wesentlichen die gleichen Funktionalitäten bereit wie ein Java Enterprise Container, ist jedoch deutlich performanter und stellt zum Beispiel einen eigenen Webserver zur Verfügung. In der ersten Beispielapplikation wurde vorerst lediglich eine Rest-Schnittstelle erstellt, über die man

diverse Benchmark Anfragen an das System stellen konnte. Die Anfragen wurden von der Producer-Komponente entgegengenommen und an einen Broker übergeben. Ein Messagebroker ist eine weitere Komponente bei der sich die Produzenten sowie die Konsumenten registrieren können um Nachrichten über eine gemeinsame Schnittstelle auszutauschen. Broker zeichnen sich insbesondere dadurch aus, dass sie skalierbar sind. Es ist also möglich, sowohl mehrere Konsumenten bzw. Produzenten mittels Warteschlange kommunizieren zu lassen, als auch mehrere Broker für die gleiche Warteschlange zu erstellen um die Verwaltungslast aufzuteilen. Ähnlich wie bei verteilten Datenbanken gibt es hierbei entsprechende Protokolle die Datenkontrolle gewährleisten. Für mein Einsteigerprojekt habe ich allerdings von allen Komponenten erst einmal nur eine Instanz eingepflegt. Für den Broker habe ich eine Implementierung von Apache gewählt, die sich „activemq“ nennt, eine Alternative wäre hierbei das open source Projekt „rabbitmq“ gewesen. Dieses stellt eine native Unterstützung der gängigen Metrikwerkzeuge, wie zum Beispiel „Prometheus“, zur Verfügung. Da im Unternehmen aber bereits mit Activemq gearbeitet wird, habe ich mich ebenfalls darauf beschränkt. Sobald der Broker eine eingegangene Nachricht an einen Konsumenten geschickt hat, wird eine zufällige Verschlüsselungsoperation ausgeführt, um etwas CPU-Last zu erzeugen, bevor die Nachricht in eine Datenbank abgespeichert wird. Für die Datenbank-Komponente wurde ebenfalls ein eigener Docker-Container bereitgestellt. Als Datenbank wurde hierbei „Postgres“ verwendet, da das Unternehmen auf lange Sicht auf diese umsteigen möchte. An allen nennenswerten Kommunikationsendpunkten wurde ein Timestamp an die verschickte Nachricht angeheftet, um im Nachhinein die Performanz eines Durchlaufs untersuchen zu können. Es wurde auch auf verschiedenen Systemen getestet, die je nach Hardware sowie Kapazität unterschiedliche Metriken produzierten. Zu diesem Zeitpunkt habe ich sämtliche Metriken selbst analysiert, im späteren Verlauf wurde jedoch auf entsprechende Tools zurückgegriffen.

Prototyp Behördensoftware

Nachdem ich mich etwas mit den Technologien des Unternehmens beschäftigt habe, sollte ich einen Prototypen für einen neuen Projektzweig des Unternehmens erstellen. Da viele Behörden auch zum heutigen Zeitpunkt noch mit herkömmlichen Formularen auf Papier arbeiten, soll die spätere Applikation in der Lage sein solche Formulare einzuscannen, eine logische Prüfung der Eingabewerte durchführen und an die entsprechende Dienstelle weiterleiten. Sie soll also im Endeffekt als eine Art Verteiler für eingegangene Formulare dienen. Um dies zu gewährleisten wurde in den Grundzügen auf die gleiche Architektur wie bei meinem ersten Einarbeitungsprojekt zurückgegriffen. In den ersten Tagen ging es wieder darum ein Producer- Consumer- Muster zu implementieren. Dieses Mal wurde jedoch direkt eine Nachricht im XML Format mitgegeben. Dieses XML sollte in einer sinnvollen Struktur in einer Datenbank hinterlegt werden.

Im nächsten Schritt wurde eine Art Aufsatz für die Produzentenseite in Form einer grafischen Benutzeroberfläche gebaut. Hierfür habe ich auf das JavaScript Framework „ReactJs“ zurückgegriffen, welches mittels eines reaktiven States in der Lage ist in Echtzeit geschickte Informationen vom Backend im Frontend darzustellen. Der Benutzer konnte also im Browser

verfolgen wie seine eingegebene Nachricht durch das System wandert und an den verschiedenen Stellen verarbeitet wird. Es war ebenfalls möglich den Inhalt sowie diverse Metriken wie zum Beispiel Start- und Zielsystem zu beobachten. In einem weiteren Schritt wurden weitere Start- und Zielsysteme hinterlegt. So ist es zum Beispiel möglich eine Datei auf einem SFTP Server hochzuladen und diese mittels eines festgelegten Intervalls anzufragen und vom Backend aus zu erkennen. Die Datei wird dann in eine Zwischenablage kopiert bevor sie in das System gespeist wird. Zu diesem Zeitpunkt hatte ich alles eigenständig implementiert, nun kam jedoch ein Senior Entwickler dazu, der sich nun mit dem Auslesen von Metadaten aus einem eingegebenen PDF beschäftigte. Der Prototyp sollte nun in der Lage sein ein am Rechner ausgefülltes Formular einzulesen, um ein simples XML Format daraus zu generieren, das anschließend den üblichen Roundtrip durch das System bewerkstelligt. Nachdem er die Transformation kodiert hat, pflegte ich dies in die Applikation ein. Anschließend sollte es möglich sein einen Emailaccount abzufragen an den Formulare geschickt werden können. Diese Formulare werden vom System ausgelesen und anschließend vom Algorithmus des Senior Entwicklers konvertiert. Zu diesem Zeitpunkt fuhr mein Vorgesetzter in den Urlaub und beauftragte mich damit ihn in den Meetings mit dem Management sowie dem Firmeninhaber zu vertreten, da ich der einzige aus unserem kleinen Team war, der fließend Deutsch spricht. Außerdem bekamen wir noch Zuwachs von zwei Masterstudenten welche als Werksstudenten ebenfalls anfingen am Projekt mitzuarbeiten, sich zu dieser Zeit jedoch auf andere Teilaspekte (wie zum Beispiel Werkzeugen zur Erkennung von Handschrift) widmeten. Da ich allerdings auch die Teammeetings leitete musste ich mich auch etwas in diese Thematik einarbeiten. Zu Anfang der Vertretungszeit wurde ich vom Management damit beauftragt ein Mapping vom XML, welches vom Konvertierungsalgorithmus des Senior Entwicklers generiert wurde, hin zu einem Schema zu erstellen. Das Schema folgt einer XSD Spezifikation, die im öffentlichen Raum als Standard gilt. Eine XSD Spezifikation stellt in gewisser Hinsicht ein Regelwerk dar, wie eine XML Datei auszusehen hat. Da die Applikation in Zukunft skalierbar sein soll, war ein möglichst generischer Algorithmus gefragt. Hierfür generierte ich mir mit einem Tool zuallererst einen leeren Objektbaum den ich anschließend mit den einzelnen Feldern des flachen XMLs vom Kollegen befüllte. Die Regeln, nach denen ein Tag aus dem flachen XML in ein Element des Objektbaums abgebildet wird, habe ich in diversen Abstraktionsschichten versucht möglichst allgemeingültig darzustellen. Hierfür wurden vor allem Namen / Enums auf Interfacelevel verwendet, die ihre genaue Implementierung / Pfade aus Konfigurationsdateien auslesen. Nachdem ich auch diesen Teil in den gesamten Prototypen eingebaut habe, folgten mehrere Meetings in denen vermehrt Kleinigkeit an der Benutzeroberfläche angemerkt wurden, die ich dann in den kommenden Wochen entsprechend verbessert habe. Nachdem mein direkter Vorgesetzter aus dem Urlaub zurückgekehrt ist, habe ich noch an ein paar weiteren Meetings mit dem Management sowie der Unternehmensführung teilgenommen. Gegen Ende des Praktikums wurde jedoch angekündigt, dass der Ansprechpartner, sowie das gesamte Team bei der Behörde die den Prototypen als erstes einsetzen wollten nun erst einmal im Urlaub ist und das Projekt erst einmal ein paar Monate ausgesetzt wird, bis neue Spezifikationen ihrerseits angegeben werden.

In der Zeit in der ich auf neue Anforderungen seitens des Managements warten musste, habe ich außerdem eine Docker-Registry aufgesetzt. Eine Registry dient zur Versionskontrolle der

Container-Images und lässt sich mit Repository Plattformen wie git oder svn vergleichen. Auch bei den Image-Registries gibt es mehrere Alternativen aus denen es wählen gilt. Ich habe mich hierbei für ein Nexus-Deployment entschieden, da diese Lösung ebenfalls als Docker-Image erhältlich war und somit eine schlankere Lösung bot als eine Festinstallation. Um die Registry jedoch auf einem Server im Unternehmensnetz zu deployen musste ich mich mit dem Dienstleister des Unternehmens auseinandersetzen, der für die Unternehmensinfrastruktur verantwortlich ist, da das Unternehmen diesen Aspekt ausgelagert haben. Ich habe zwei Server organisiert, einen um die Docker-Registry laufen zu lassen und einen weiteren, um im späteren Verlauf des Projekt Lastentests durchführen zu können. Die zweite VM ist daher auch deutlich performanter als die erste. Den leistungsstärkeren Server werde ich auch zum Testen des Codes im Rahmen meiner Bachelorthesis verwenden. Sobald das Projekt ein größeres Team zur Verfügung gestellt bekommt, wird auf einem der Server ebenfalls ein Jenkins-Server deployt. Dieser wird zur Umsetzung einer Continuous Integration Pipeline verwendet um den Build-Prozess der Applikation automatisch durchführen zu können.

Ausblick

In Zukunft wird die Behördenapplikation definitiv mehr Mitarbeiter zugeteilt bekommen und es wird wahrscheinlich eine eigene Abteilung gebildet. Da die Applikation früher oder später nach heutigen Standards deployt werden soll, müssen sämtliche Komponenten skalierbar aufgebaut werden. Es soll also möglich sein, zu jedem Zeitpunkt ohne großen Aufwand oder daraus resultierenden Problemen diverse Replikate für alle Services zu erzeugen. Hierfür müssen Orchestrierungsplattformen wie „Kubernetes“ oder „Docker Swarm“ verwendet werden. Diese Projekte bieten die Möglichkeit Container in großen Mengen automatisch zu verwalten, sodass es zum Beispiel möglich ist, über automatische health checks / heartbeats festzustellen, ob ein Container noch läuft oder ob er neu gestartet werden muss. Hierfür werden dann auch Metriken erstellt, die über allgemein genutzte Schnittstellen im Ökosystem erreichbar sind. Den Aspekt der Skalierbarkeit werde ich in meiner Bachelorthesis ausführlich behandeln. Hierzu werde ich allerdings nicht Kubernetes verwenden, da dies bereits im Unternehmen vorhanden ist und mein Vorgesetzter gerne wissen möchte wie es sich mit Docker Swarm verhält. Docker Swarm ist ein Projekt, das keine komplette Neuinstallation wie Kubernetes benötigt, sondern bereits im Docker-Daemon integriert ist. Da Docker eine hervorragende Unterstützung für eine Beschreibungssoftware namens „docker-compose“ bildet, werde ich diese beiden Projekt zusammen mit mehreren open source Lösungen zum Skalieren einiger beispielhafter Services nutzen. Hierbei wird außerdem auch auf die Nutzung von Werkzeugen zur Darstellung sowie Analyse der erhaltenen Daten eingegangen.

Fazit

Ich war positiv überrascht vom gesamten Praktikum. Die Arbeitsatmosphäre war größtenteils sehr angenehm und ich hatte mit freundlichen Kollegen und Ansprechpartnern zu tun. Besonders gut fand ich, wie sehr ich direkt von Anfang an eingebunden und gefordert wurde. Besonders die Meetings mit der Geschäftsführung und dem Management waren sehr aufschlussreich. Dennoch hatte ich manchmal das Gefühl zu viel Verantwortung übertragen bekommen zu haben. Es gab leider keine Codereviews oder dergleichen und generell kennt sich auch sehr wenige der Kollegen mit den Technologien aus die ich verwenden sollte. Wie qualitativ hochwertig mein Prototyp ist, kann ich leider überhaupt nicht einschätzen. Das hängt natürlich auch damit zusammen, dass ich gerade zum Ausprobieren dieser Technologien eingestellt wurde, dennoch hätte ich mir an manchen Stellen etwas mehr Unterstützung gewünscht. Generell war die Zeit aber sehr spannend. Ebenfalls hervorheben möchte ich, dass die Arbeit im Homeoffice nach einigen Wochen Einarbeitung auch sehr gut funktioniert hat und es bis auf wenige Ausnahmen (Verbindungsprobleme meinerseits) zu keinerlei Problemen kam.