openHPI - Eine flexible und erweiterbare Service-orientierte MOOC Plattform

Jan Renz, Thomas Staubitz, Christoph Meinel
Hasso-Plattner Institut, Potsdam, Germany

Abstract: Mit der Internet-basierten MOOC Plattform open.HPI.de war das Hasso-Plattner-Institut der erste europäische Anbieter von Massive Open Online Courses (MOOCs), der sich sowohl mit der Erstellung von Kursen als auch mit der Entwicklung der hinter dem Angebot stehenden technologischen Plattform beschäftigte. Nach dem initialen Launch der Plattform im September 2012, basierend auf einem bestehenden Learning Management System, wurde die Plattform seit Sommer 2013 den Prinzipien einer Service-orientierten Architektur folgend von Grund auf neu konzeptioniert. Aus den Erfahrungen während des Betriebs der ersten Version wurde die Lehre gezogen, dass die Anforderungen an ein LMS grundlegend andere sind als an eine MOOC-Plattform. Gleichzeitig sollte eine Plattform erschaffen werden, die als flexibles und erweiterbares Fundament für die laufenden Forschungsvorhaben im Bereich Web-University dienen kann. Im April 2014 startete schließlich der erste Kurs auf der neuen Version der Plattform. In dem vorliegenden Paper wird die Architektur dieser aktuellen Generation von openHPI erläutert. Weiterhin werden auf Grundlage der Erfahrungen aus dem Betrieb mit inzwischen über 50 Kursen und über 750.000 Kurseinschreibungen die Frage beantwortet, inwieweit sich die zugrundeliegende Architektur bewährt hat und somit als Best Practise für skalierende E-Learning Anwendungen dienen kann.

1 Vom generischen LMS zur spezialisierten MOOC Plattform

Im September 2012 startete openHPI als erste europäische MOOC Plattform mit dem damals neuen Konzept von massiven E-Learningkursen [GMM+13]. Der erste angebotene Kurs auf der Plattform, In-Memory Data Management von Prof. Hasso Plattner, wurde zur Laufzeit von über 13.000 Teilnehmern besucht[RTR15]. Die für die Ausrichtung des Kurses verwendete Plattform basierte dabei auf dem Open-Source LMS Canvas [1]. Canvas ist eine auf Basis von RubyOnRails umgesetzte monolithische Anwendung, die die damaligen Anforderungen weitestgehend zu erfüllen schien. Schon während der Durchführung des ersten Kurses stellten sich im Betrieb mehrere Nachteile dieser Architektur heraus. Trotz weiterer Umbauten und Anpassungen, konnten weder die gewünschte Skalierbarkeit noch die notwendige Flexibilität erreicht werden. Auch die Betriebsstabilität war nur schwer zu gewährleisten, da die verwendete Version von Canvas noch nicht über umfassende Tests verfügte. Oft genug tauchten durch Änderungen verursachte Fehler an ganz anderen Stellen erst Tage später auf und verursachten aufwendige Debugging-Sessions. Die zentralen Anforderungen an das neue System, wie sie etwa in [MW13] postuliert wurden, lauten:

[1] https://www.canvas.net/

- Einsatz einer service basierten Architektur
- Möglichkeit der Skalierung für mehrere Tausend gleichzeitige Nutzer
- Einfache Erweiterbarkeit durch neue Funktionen und interaktive praktische Übungen
- Fokussierung und Beschränkung des Funktionsumfanges auf das MOOC Lernformat
- Mehr Self Service Möglichkeiten für die Teaching Teams und Kursadministratoren
- Bessere Unterstützung von mobilem Lernen

2 Architektur

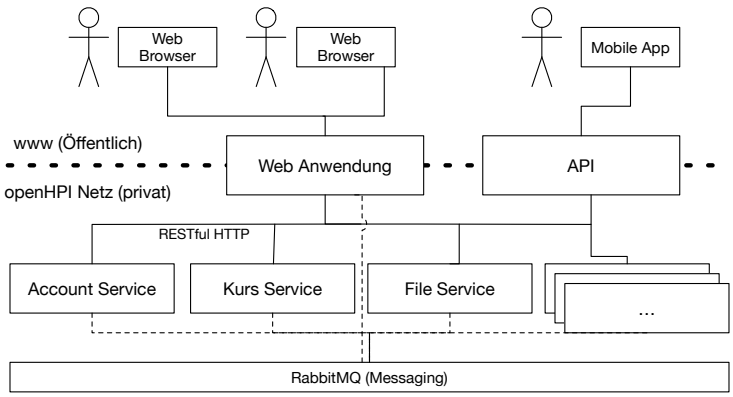

Abbildung 1: Service-basierte Architektur der neuen openHPI Plattform

openHPI besteht aus einer zweistelligen Anzahl von Microservices[2]. Ein Microservice ist für die Bereitstellung einer zusammenhängenden Funktionsgruppe zuständig. Jeder Microservice besitzt eine eigene Datenhaltung (Datenbank). Außer seiner Datenbank ist jeder Service stateless und gehorcht den Prinzipien der sog. *12 Factor Apps*[3]. Durch dieses Prinzip wird eine gute Skalierbarkeit gewährleistet, da problemlos mehrere Instanzen eines Services deployed werden können. Nach außen hin gibt es drei Verbindungen über die ein Service erreichbar sein kann:

- Eine REST-API über welche der Service seine Funktionalität anderen Services bzw. der Web Applikation zur Verfügung stellt.

[2]Zur Begrifflichkeit der Microservices siehe http://martinfowler.com/articles/microservices.html
[3]http://12factor.net/

- Eine Verbindung zu einem Messaging System (im konkreten Fall RabbitMQ) über welches Nachrichten von anderen Services empfangen werden können bzw. gesendet werden können. Diese Kommunikation folgt dem Publisher/Consumer-Pattern.
- Eine Verbindung zu einem applikationsweiten Cache. Dieser Cache kann zwar auf mehrere Instanzen skaliert werden, es ist aber zwingend erforderlich, dass dieser Cache zentral angesiedelt ist, da Erzeugung bzw. Abruf sowie das Invalideren eines Cacheeintrages von verschiedenen Services getriggered werden kann.

Alle Services bei openHPI verfügen über eine REST-API, einige über ein Cachinglayer (auf Basis von REDIS) und/oder eine Verbindung zum Messagingsystem. Als Datenbanktechnologie kommt hauptsächlich PostgreSQL zum Einsatz.

In Richtung Nutzer steht eine Webapplikation zur Anzeige der Daten im Browser und eine API als Schnittstelle zu den mobilen Applikationen zur Verfügung.

2.1 Optimierung der synchronen Kommunikation

Insbesondere die Web Applikation muss zum Erzeugen einer Webseite zahlreiche Ressourcen von verschiedenen Services anfragen. Da hierbei jedes mal eine Netzwerkverbindung aufgebaut werden muss, entsteht hier ein gewisser Overhead. Selbst wenn alle Services es schaffen diese Anfragen in wenigen Millisekunden zu beantworten, so gab es in den ersten Prototypen Performance Probleme. Ein Beispiel hierfür ist die Dashboardseite. Diese wird dem Lernenden nach dem Login präsentiert und liefert eine Übersicht über seine laufenden Kurse, Kursempfehlungen, anstehende Termine und Nachrichten. Zur Anzeige dieser Seite werden ca. 20 einzelne Anfragen an verschiedene lokale Dienste benötigt. Um an dieser Stelle eine zufrieden stellende Antwortzeit zu erreichen wurden zwei Konzepte zur Performancesteigerung evaluiert und implementiert.

Zum Einen wurde ein transparentes Cachingverfahren implementiert, so dass Ressourcen—sofern sie im Cache vorliegen und vom Service als cache-fähig gekennzeichnet sind—direkt aus dem Cache geliefert werden können. Zum Anderen wurde dafür gesorgt das Ressourcen parallel abgerufen werden können. Dies betrifft alle lesenden Zugriffe (also alle per HTTP-GET abgerufenen Ressourcen). Hierzu werden zunächst alle Abhängigkeiten beim Abruf geprüft, dann werden die Requests parallel unter Zuhilfenahme der der Typhoes Bibliothek abgerufen. Die entsprechende Funktionalität wurde in einer Bibliothek (Gem) realisiert, welche auf GitHub veröffentlicht wurde[4].

2.2 Liste der Services

Bereits zum Start der neuen Plattform bestand openHPI aus 14 einzelnen Anwendungen / Services. Die nachfolgende Liste gibt dabei nicht nur ein Überblick über die Architektur,

[4] https://github.com/jgraichen/acfs

```
# Start                 Fin
#   |===================|       'Acfs.run'
#   |====|                      /users/5
#   |    |==============|       /comments?user=5
#   |    |======|              /users/1
#   |    |=======|             /users/4
#   |    |======|              /users/10
```

Abbildung 2: Schema eines optimierten parallelisierten Ressourcenaufrufes (Quelle: ACFS Dokumentation)

sondern bietet auch ein guten Eindruck in die funktionalen Anforderungen einer MOOC-Plattform:

- Die **Web Applikation** und die (später hinzugekommene) **API Applikation** stellen die Daten der Services dem Endverbraucher zur Verfügung. Im Falle der Web Applikation sind der Endverbraucher direkt die Stakeholder des Kurses (Teilnehmer, Teacher, etc.). Im Falle der API Applikation sind es andere Programme, wie beispielweise unsere mobilen Applikationen. Beide besitzen keine eigene Datenhaltung.
- Der **Account Service** übernimmt die Verwaltung der Nutzerdaten (Registrierung, Login, Profildaten, Benutzereinstellungen) sowie Rechte und Rollenmanagement.
- Der **Course Service** hält Informationen über die Kurse, die Struktur der Inhalte und die Kurseinschreibungen. Er speichert auch den Lernfortschritt der Nutzer. Der Lernfortschritt wurde ursprünglich in einem eigenen Service behandelt. Aus Performancegründen wurde er aber in den Kurs Service integriert.
- Der **Certificate Service** übernimmt das Rendern der PDF-Zeugnisse sowie der Teilnahmebestätigungen und das Verwalten der entsprechenden Vorlagen.
- Der **File Service** speichert und verwaltet Dateien. Hierzu zählen Lernmaterialen wie Folien, Bilddateien, aber auch Uploads der Nutzer (bspw. das Profilfoto).
- Der **RichText Service** speichert Texte und deren Lokalisierungen und übernimmt deren Versionierung.
- Der **Helpdesk Service** nimmt Supportanfragen entgegen und leitet diese an ein separat gepflegtes, externes Ticketingsystem [5](OTRS) weiter.
- Der **Collab Space Service** stellt die Lernraum / Kollaborationsfunktionalitäten zur Verfügung. In den Lernräumen können die Lerner in einem eigenen Forum diskutieren oder Dateien austauschen. Eingebundene externe Tools ermöglichen synchronisiertes Lernen mehrerer Nutzer (Together.js) und Gruppenvideochats (Google Hangouts).

[5]https://www.otrs.com/

- Der **Quiz Service** verwaltet die Daten der Selbsttests, Hausaufgaben und Surveys. Der **Submission Service** hingegen verwaltet die Abgaben der Teilnehmer zu Selbsttests und Hausaufgaben.
- Der **Video Service** verwaltet sämtliche Videoinhalte der Plattform. Die Videoinhalte selbst werden dabei von einem oder mehreren externen Streaminganbieter ausgeliefert.
- Im **News Service** werden Nachrichten an die registrierten Nutzer verwaltet. Weiterhin besteht die Möglichkeit Nachrichten entweder nur auf der Plattform anzuzeigen oder sie per E-Mail an die Teilnehmer zu versenden.
- Der **Notification Service** übernimmt das Versenden von Mails an die Kursteilnehmer sowie die Erzeugung der auf der Plattform angezeigten Benachrichtigungen.

Nach dem Launch der neuen Version kamen durch funktionale Erweiterungen und Forschungsprojekte folgende neue Services hinzu:

- Der **LTI Service** verwaltet in den Kursen verwendete externe Aufgaben und deren Provider, die über den Learning Tools Interoperability (LTI 1.1) [6] Standard angebunden sind.
- Der **Gamification Service** stellt die in [SWRM14] vorgestellten Gamification Elemente wie Badges, Gamification-Points und Userstates zur Verfügung.
- Der **Grouping Service** dient zur Durchführung von AB-Tests. Hierdurch können die Auswirkungen neuer Features auf das Nutzungsverhalten und das Lernergebnis untersucht werden, indem sie vorübergehend nur für eine definierbare Testgruppe zur Verfügung gestellt werden.
- Der **Peer-Assessment Service** erweitert die Möglichkeiten zur Evaluierung des Lernfortschritts der Teilnehmer bedeutend. Insbesondere kreative und komplexe Aufgabenstellungen sind automatisiert nur schwierig zu bewerten. Auch die Teaching Teams können diese Aufgabe aufgrund der Masse der Abgaben unmöglich bewältigen. Als Ausweg bietet sich das Peer-Grading an, um diese Aufgabe auf möglichst viele Schultern zu verteilen (crowd-sourcing).
- Der **Learning Analytics Service** dient als zentrale Instanz (Single Point of Truth) für das Erfassen und Abrufen von Nutzungsdaten. Dies dient der wissenschaftlichen Auswertung bei der Erforschung des MOOC-Lernformates. Gleichzeitig können andere Services auf diese Daten zugreifen. Dies kann dazu verwendet werden den Teaching-Teams, den Lernenden und anderen Stakeholdern Einblicke in das Nutzungsverhalten zu geben.
- Der **Social Networking Service** ermöglicht die Vernetzung der Nutzer über "Freundschaftsbeziehungen". Dies ermöglicht z.B. dem Gamification Service einen sozialen Kontext zu liefern und gibt Nutzern die Möglichkeit über Aktivitäten von befreundeten Nutzern informiert zu werden.

[6] http://www.imsglobal.org/lti/

2.3 Einbindung externer Tools via LTI

Unterschiedlichste praktische Übungen können mittels der integrierten LTI Schnittstelle (LTI 1.1) eingebunden werden. Das grundlegende Prinzip hierbei ist folgendes: Aus der Lernplattform heraus wird ein beliebiges web-basiertes Tool, das ebenfalls über eine LTI Schnittstelle verfügt, geöffnet. Eine eindeutige Kennung des Nutzers und der zu absolvierenden Übung wird übergeben. Nun arbeitet der User in dem externen Tool. Dieses entscheidet eigenständig inwieweit der Nutzer die Anforderungen der Aufgabenstellung erfüllt und gibt zusammen mit der Kennung des Nutzers einen Wert zwischen Null (0 nicht NULL) und Eins zurück. Dieser kann auf der Lernplattform mit einem beliebigen Faktor multipliziert werden, um die Punkte festzulegen die der Nutzer für diese Leistung erhält. Somit können bspw. Domain-spezifische interaktive Übungsaufgaben in einzelne Kurse eingebunden werden, ohne dass diese die Komplexität der Plattform erhöhen. Die Entwickler dieser Aufgaben müssen über kein spezielles Wissen betreffend der Architektur der Plattform besitzen, es genügt Fachwissen über den verwendeten Interoperabilitätsstandard. Die Erstellung der einzelnen Anwendungen ist durch Bibliotheken, die für sehr viele Programmiersprachen erhältlich sind, vereinfacht. Da der Standard von vielen anderen Lernplattformen und LMS unterstützt wird besteht auch die Möglichkeit diese Tools anderen Anbietern zur Verfügung zu stellen.

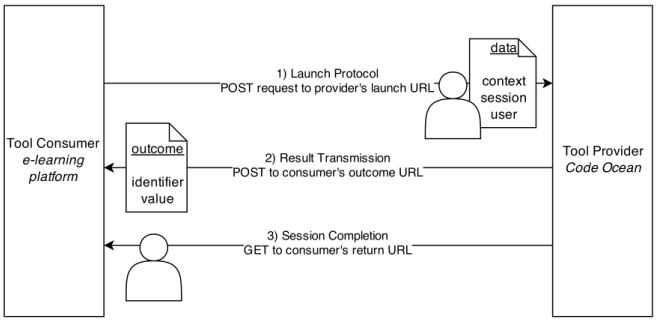

Abbildung 3: Funktionsweise der LTI Schnittstelle

Die Vielseitigkeit lässt sich an drei Beispielen für den Einsatz der LTI Schnittstelle auf openHPI darlegen:

- RDF-Validator im Kurs *Knowledge Engineering with Semantic Web Technologies*: Das Teaching Team entwickelte hier ein Tool in dem man die URL einer RDF-Datei eingeben kann. Je nach Konfiguration werden dann bestimmte Kriterien geprüft, z.B. ob die Datei existiert und lesbar ist, ob sie valide RDF-Daten in einem bestimmten Format (RDF/XML, Turtle, etc.) enthält, oder ob in der Beschreibung bestimmte Triple enthalten sind z.B. <#JohnDoe>rdf:type foaf:Person.
- E-Mail Aufgaben im Kurs *Sicherheit im Internet*: Das Teaching Team stellte hier ein Tool zur Verfügung mit dem die Teilnehmer Verschlüsseln und Signieren mit

GPG/PGP praktisch üben konnten. Hierzu mussten die Teilnehmer ihre E-Mail Clients mit GPG/PGP Plugins versehen, ihre Public Keys in das Tool laden, bzw. den Public Key des Tools herunterladen, um dann signierte Emails an das Tool zu senden bzw. verschlüsselte Mails des Tools zu empfangen. Das Tool kann die Signatur der signierten E-Mails anhand des hinterlegten Public Key der Teilnehmer überprüfen und im Erfolgsfall wurden dem Teilnehmer direkt Punkte gut geschrieben. Im Falle der verschlüsselten E-Mails die das Tool an die Teilnehmer schickte, wurde das bereits in anderem Zusammenhang bewährte STEAP-Pattern [SRW+14] eingesetzt. Der Grundgedanke dieses Patterns ist: dem Nutzer bei erfolgreicher Lösung einer Aufgabe ein Codewort zukommen zu lassen—anders gesagt: das Codewort **ist** die Lösung der Aufgabe—das dieser dann wiederum in eine Freitext Frage eines Quizzes auf openHPI eintragen kann um so an seine Punkte zu kommen.

- Coding Aufgaben über CodeOcean im Kurs *Java für Einsteiger*: CodeOcean, ein Open Source Tool zum automatisierten Bewerten von Programmieraufgaben, entstand aus einer Masterarbeit im Rahmen von openHPI. Dieses Tool ermöglicht die automatisierte Bewertung von Programmieraufgaben für eine breitgefächerte Auswahl von Programmiersprachen. Zur Bewertung der Aufgaben werden native Test-Frameworks in der verwendeten Programmiersprache eingesetzt.

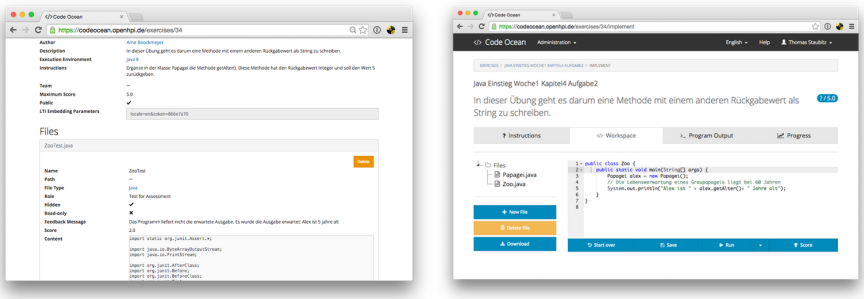

Abbildung 4: CodeOcean: Links Aufgabenstellung durch den Kursbetreuer (unten sieht man wie die JUnit Tests eingebunden werden). Rechts: Die gestellte Aufgabe aus Nutzersicht.

2.4 Gewährleistung der Softwarequalität durch automatisiertes Testen

Auf Basis der Erfahrungen des Betriebs der ersten Version war von Beginn der Arbeiten an der neuen Version klar, dass nur durch ein umfassendes vollautomatisches Testing eine gleichbleibende Softwarequalität gewährleistet werden kann. Den Entwicklern wurde nahegelegt nach den Prinzipien des Test Driven Development zu arbeiten und entsprechende interne Schulungen wurden durchgeführt. Zum automatisierten Testen der Platt-

form wurde ein zweistufiges Testkonzept konzeptioniert und implementiert. Für die einzelnen Dienste sind hierbei Teste auf Basis der verwendeten Basistechnologie vorgesehen. Da alle momentan existierenden Services auf Ruby on Rails basieren kommt hierbei immer das RSpec Testframework zur Anwendung. Die Anzahl der durchgeführten Tests liegt zwischen 15 (Helpdesk) und 1270 (Course). Auf den Servern ausgeführt liegt die Ausführungsdauer dieser Tests unter 2 Minuten (wobei diese Tests parallel durchgeführt werden können).

Zusätzlich gibt es Integrationstests. Diese dienen als Benutzerakzeptanztests und simulieren die Nutzung durch den Nutzer und werden auf Basis von Cucumber[7] in einem Browser durchgeführt[8]. Diese Tests benötigen momentan 45 Minuten zur Ausführung, wobei momentan knapp 150 einzelne Tests durchlaufen werden. Ein Grossteil der Zeit wird hierbei für das Aufsetzen des Testsystems verwendet (Installieren, Initialisieren und Starten der Services).

Problematisch wurde hierbei der Test von Komponenten die auf eine Vielzahl von Ressourcen aus anderen Diensten zugreifen. Im Gegensatz zu dem Stubbing von Datenbankinhalten, für welche es geeignete Bibliotheken gibt gab es hier keine verwendbaren fertigen Lösungen. Auch bei den Integrationstests wurde das Stubbing problematisch. Wurden hier alle für die einzelnen Testfälle benötigten Daten über die Benutzeroberfläche angelegt so erhöhten die Ausführungszeiten der Integrationszeiten erheblich. Um dieses Problem zu lösen wurden die Services so erweitert, dass diese innerhalb der Integrationsumgebung via REST performant Ressourcen zu Testzwecken erstellen können.

3 Evaluation und Ausblick

Nach nunmehr einem Jahr Produktiv-Betrieb auf der neuen Version der openHPI-Plattform lassen sich erste fundierte Aussagen treffen. Zusammenfassend kann festgestellt werden, dass die Erwartungen die an den Einsatz einer Service basierten Architektur gestellt wurden, vollumfänglich erfüllt werden konnten. Mit der beschriebenen Architektur konnten sowohl funktionale Erweiterungen als auch forschungsgetriebenen Prototypen realisiert werden. Traten Performance oder Stabilitätsprobleme auf, so konnten diese durch das Monitoring sofort auf den entsprechenden Dienst eingegrenzt werden und somit schneller behoben werden. Ungewollte Seiteneffekte traten kaum auf, und wenn wurden sie durch die automatisierten Tests noch im Entwicklungsprozess entdeckt und konnten behoben werden. Bewährt hat sich auch die Ausgliederung komplexerer Aufgabentypen in eigenständige LTI-Anwendungen. So konnten Aufgaben von einzelnen Teaching Teams entwickelt werden, ohne das sich diese mit der Komplexität der Platform beschäftigen mussten. Auch ein weiterer Einsatz der erstellten Aufgaben in anderen LMS, beispielsweise zum Einsatz in der universitären Lehre wird dadurch ermöglicht.

Natürlich bringt diese Architektur Probleme eigener Art mit sich, es ist allerdings davon auszugehen, dass gerade durch die wachsende Komplexität der Software, eine Umsetzung

[7] https://cukes.info/

[8] Hierfür wurde ein Cucumber Wrapper entwickelt der unter https://github.com/jgraichen/gurke zu finden ist

als monolithische Applikation perspektivisch wesentlich größere Probleme mit sich gebracht hätte.

Hierzu zählen:

- Probleme durch das Management einer zunehmenden Anzahl von externen Bibliotheken und deren Abhängigkeiten.
- Schwierigeres Skalieren: Grundsätzlich hätte auch eine dem Prinzip der ”12 Factor Apps”[9] folgende, monolithische Anwendung gut skaliert werden können. Durch den Einsatz der dienstbasierten Architektur wird diese Skalierbarkeit aber erzwungen.
- Erweiterungen oder Anpassungen des Portals hätten zwangsläufig zu mehr unbeabsichtigten Seiteneffekte geführt.

Die hier beschriebene und bei openHPI praxiserprobte Architektur eignet sich nicht nur als Basis für skalierbare E-Learning Anwendungen, sondern bietet auch Raum und Möglichkeiten für Erweiterungen zur Untersuchung von wissenschaftlichen Fragestellungen, für Abschlussarbeiten und Seminare. Der Schritt für den Relaunch von openHPI auf eine komplett neue Codebasis zu setzen und kein Code der ersten Version wiederzuverwenden ist in seinen Auswirkungen sicherlich drastisch und in dieser Form nicht für alle E-Learning Anwendungen realisierbar. Für bestehende monolithische Applikationen ist aber denkbar, diese teilweise auf eine dienstbasierte Architektur zu überführen, indem einzelne Komponenten in eigenständige per REST angebundene Dienste migriert werden.

Ein altes Sprichwort besagt: ”There is no free lunch”. Die durch die Architektur erzielten Vorteile gehen aber nicht ohne damit verbundene Nachteile einher. Hier ist unter anderem Folgendes zu nennen:

- Die Erstellung neuer Funktionen ist aufwendiger als in einer monolithischen Anwendung. Dies gilt zumindest solange keine der oben beschriebenen Probleme auftreten und dieses Verhältnis umkehren. Gründe für den erhöhten Aufwand:
  - Änderungen müssen oft an mehreren Komponenten vorgenommen werden (bspw. Dienst, API und Frontend)
  - Das Testing ist umständlicher und insbesondere Integrationstest dauern länger.
  - Einsetzbare Bibliotheken und Best Practises sind oft nicht direkt benutzbar sondern müssen angepasst oder neu erstellt werden.
- Trotz der Modularität ist durch die Architekturspezifischen Eigenheiten (selbst für erfahrende RubyOnRails Entwickler) ein hoher Einarbeitungsaufwand erforderlich.
- Die Installation einer lokalen Entwicklungsumgebung ist sehr aufwendig.
- Ein laufendes System erzeugt eine hohe Last auf lokalen Entwicklungsmaschinen. Die Arbeit auf z.B. älteren Laptops ist somit wenig effizient.

[9] http://12factor.net/

3.1 Ausblick

Die durch die Architektur ermöglichten Vorteile werden noch nicht in der Tiefe genutzt, wie dies wünschenswert wäre. Zum einen ist die technologische Vielfalt der Dienste noch nicht ausgeprägt, so wurden bspw. erst mit dem Learning-Analytics Dienst andere Datenbanktechnologien eingeführt. Die produktiv laufenden Services selbst sind alle in Ruby-OnRails implementiert. Dies hat vorrangig mit der vorhandenen Technologiekompetenz der vorhandenen Entwickler in diesem Technologiestack gut ausgebildet sind, aber auch mit der mangelnden Flexibilität des Deployments. Hier wird momentan untersucht inwiefern durch den Einsatz von Technologien wie Linux-Container (Docker) eine einfacherer Einsatz von unterschiedlichen Technologie Stacks ermöglicht werden kann und wie mit den hiermit verbunden Herausforderungen umgegangen werden kann. Auch werden die erstellten Dienste momentan noch exklusiv durch die openHPI Platform verwendet. Es wäre mittelfristig wünschenswert hier Synergien mit anderen Anwendungen und Angeboten zu schaffen und zu nutzen. So könnte bspw. der Learning Analytics Dienst auch von anderen Angeboten genutzt werden um Nutzungsdaten zu erfassen und auszuwerten oder der Account Service verwendet werden. Eine Zusammenlegung von API und Frontend Applikation ist geplant. Reine Darstellungslogik soll dabei perspektivisch eher vom Client übernommen werden. Unter Einsatz von Frameworks wie EmberJS oder Angular ermöglicht dies nicht nur eine höhere Responsivität sondern ermöglicht auch das Lernen bei nicht bestehender Internetverbindung.

Literatur

[GMM+13] Franka Grünewald, Elnaz Mazandarani, Christoph Meinel, Ralf Teusner, Michael Totschnig und Christian Willems. openHPI: Soziales und Praktisches Lernen im Kontext eines MOOC. In *DeLFI*, Seiten 143–154, 2013.

[MW13] Christoph Meinel und Christian Willems. *openHPI: das MOOC-Angebot des Hasso-Plattner-Instituts*, Jgg. 79. Universitätsverlag Potsdam, 2013.

[RTR15] Thomas Staubitz Ralf Teusner, Keven Richly und Jan Renz. Enhancing Content between Iterations of a MOOC – Effects on Key Metrics. In *Proceedings of eMOOCs 2015*, 2015.

[SRW+14] Thomas Staubitz, Jan Renz, Christian Willems, Johannes Jasper und Christoph Meinel. Lightweight Ad Hoc Assessment of Practical Programming Skills at Scale. In *Proc. CHI 2014*, number 10.1109/EDUCON.2014.6826135, Seiten 475–483. IEEE, 2014.

[SWRM14] T. Staubitz, S. Woinar, J. Renz und C. Meinel. Towards Social Gamification-Implementing a Social Graph in an XMOOC Platform. In *ICERI2014 Proceedings*, 7th International Conference of Education, Research and Innovation, Seiten 2045–2054. IATED, 17-19 November, 2014 2014.