
Konzeption eines Softwaresystems zur Verwaltung von Hochschulporteinrichtungen

Diplomarbeit von Dirk Beckmann



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Humanwissenschaft
Institut für Sportwissenschaft

Konzeption eines Softwaresystems zur Verwaltung von Hochschulsporteinrichtungen

Vorgelegte Diplomarbeit von Dirk Beckmann

1. Gutachten: Prof. Dr. Joseph Wiemeyer
2. Gutachten: Dietbert Schöberl

Tag der Einreichung:

Bitte zitieren Sie dieses Dokument als:

URN: urn:nbn:de:tuda-tuprints-...

URL: [http://tuprints.ulb.tu-darmstadt.de/..](http://tuprints.ulb.tu-darmstadt.de/)

Dieses Dokument wird bereitgestellt von tuprints,

E-Publishing-Service der TU Darmstadt

<http://tuprints.ulb.tu-darmstadt.de>

tuprints@ulb.tu-darmstadt.de



Die Veröffentlichung steht unter folgender Creative Commons Lizenz:

Namensnennung – Keine kommerzielle Nutzung – Keine Bearbeitung 2.0 Deutschland

<http://creativecommons.org/licenses/by-nc-nd/2.0/de/>

Erklärung zur Diplomarbeit

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den ...

(Dirk Beckmann)

Inhaltsverzeichnis

1	Einleitung	6
2	Analyse der Fachdomäne Hochschulsport	7
2.1	Historische Entwicklung des Hochschulsports	7
2.1.1	Hochschulsport heute	8
2.2	Typische Aufgaben	10
2.3	Organisationsformen	11
2.4	Befragung ausgewählter Standorte	12
2.4.1	Aktuelle Lösungen	13
2.4.2	Bewertung der Wichtigkeit von Funktionalitäten	16
2.4.3	Zusätzliche Anmerkungen	20
2.5	Probleme und Herausforderungen	20
3	Cloud Computing	21
3.1	Merkmale	22
3.2	Service Modell	22
3.3	Bereitstellung Modell	23
3.3.1	Private Cloud	24
3.3.2	Public Cloud	24
3.3.3	Hybrid Cloud	25
3.4	Pro & Contra	26
3.4.1	Vorteile	26
3.4.2	Nachteile	31
3.5	Software-as-a-Service Anwendungen (SaaS)	32
3.5.1	Vorteile	33
3.5.2	Herausforderungen	34

4	Software Architektur und Design	36
4.1	Softwarearchitektur	36
4.1.1	Monolithisch	36
4.1.2	Client/Server-Architekturmodell	39
4.1.3	Dienstbasiert	42
4.2	SaaS spezifische Besonderheiten	44
4.2.1	Multi-Tenant Application (MTA)	44
4.2.2	Security	47
5	System Konzeption	48
5.1	Planungsphase	48
5.2	Definitionsphase	49
5.3	Entwurfsphase	49
6	Diskussion, Zusammenfassung und Ausblick	50

Abbildungsverzeichnis

2.1	Zufriedenheit mit der aktuellen Lösung	15
2.2	Häufigkeit der Beeinträchtigung zu Anmeldebeginn	16
2.3	Zufriedenheit mit der aktuellen Lösung	17
2.4	Wichtigkeit einzelner Funktionalitäten	18
3.1	3-Ebenen Modell von Cloud Computing	22
3.2	Public und Private Cloud Architektur	24
3.3	Hybrid Cloud Architektur	25
3.4	Provisionierung anhand von Lastspitzen	30
3.5	Unterprovisionierung	30
4.1	Traditionelle monolithische Webanwendungs-Architektur	38
4.2	Client Server Model	39
4.3	Logische drei Schichten	41
4.4	Dreischichtige Architektur	41
4.5	Richardson Maturity Model	44
4.6	Separierte Datenbanken	45
4.7	Geteilte Datenbank und separierte Schemata	46
4.8	Geteilte Datenbank und Schema	47

Tabellenverzeichnis

2.1	Übersicht befragter Hochschulen	12
4.1	Verwendungsbeispiele von mehrschichtigen Architekturen Sommerville (2007) . .	42
4.2	Rest Operationen	43

1 Einleitung

2 Analyse der Fachdomäne Hochschulsport

Die Aufgaben und Anforderungen, die an deutsche Hochschulen gestellt werden sind vielfältig und erstrecken sich in viele verschiedene Bereiche. Neben den Kernaufgaben der universitären Lehre und wissenschaftlichen Forschung gehört es auch zu den Aufgaben, das studentische Leben in sozialen und gesellschaftlichen Belangen zu unterstützen und zu fördern. Neben Einrichtungen wie dem Studierendenwerk oder dem Allgemeinen Studierendenausschuss (AStA) ist auch der Hochschulsport eine Einrichtung, die unterschiedlichste Aufgaben im Umfeld der Fachhochschulen und Universitäten übernimmt. Im Vergleich zu amerikanischen Universitäten erscheint der Hochschulsport in Deutschland oft als rein schmückendes Element. Allein der Mitteleinsatz für die Erreichung eines hohen Niveaus des Hochschulsportprogrammes in Amerika übersteigt die Ausgaben um ein Vielfaches. In deutschen Hochschulsporteinrichtungen lagen die Aufwendungen für Personal- und Sachmittel im Jahre 2004 für Universitäten bei etwa 300.000€ und bei Fachhochschulen bei rund 30.000€ (vgl. Hachmeister, Tabor & Bürgel, 2004, S.9). Der erste Teil dieser Diplomarbeit soll ein Verständnis für die Domäne Hochschulsport und deren Entwicklung, Organisationsformen, Aufgaben sowie Probleme und Herausforderungen vermitteln.

2.1 Historische Entwicklung des Hochschulsports

Das Sporttreiben an deutschen Hochschulen begann zwar nicht erst nach dem zweiten Weltkrieg, die historische Betrachtung soll sich hier jedoch auf die Zeit zwischen 1948 und 2009 beschränken, um den Rahmen der Arbeit nicht zu sprengen. Da die Organisation von Bildung in die Zuständigkeit der Bundesländer fällt, hat die Entwicklung durchaus unterschiedliche Wege genommen. Ich konzentriere mich in diesem Fall auf den geschichtlichen Rückblick mit dem Schwerpunkt NRW, da hier die umfangreichsten Dokumentationen verfügbar waren. Der 1999 von 29 Bildungsministern initiierte Bologna-Prozess zur Schaffung eines einheitlichen Hochschulraums (vgl. Zandonella, 2005)

2.1.1 Hochschulsport heute

Durch die gesetzliche Verankerung im Hochschulrahmengesetz und die Festschreibung in den einzelnen Landeshochschulgesetzen hat sich eine vorerst gesicherte Grundlage zur Förderung des Sports an Hochschulen manifestiert. Die einzelnen Hochschulsporteinrichtungen (HSP) besitzen eine gefestigte Position im Hochschulkontext und müssen nicht mehr um ihre grundlegende Daseinsberechtigung kämpfen. Trotz gesicherter Grundlagen haben etwa 40 % der Hochschulsporteinrichtungen mit Mittelkürzungen zu kämpfen (Hachmeister et al., 2004, S.9). Das Grundanliegen des HSP ist und bleibt jedoch die Förderung des Sporttreibens der Studierenden, die sich in speziellen Lebensphasen befinden und durch besondere Umstände geprägt sind. Dass diese nicht generalisiert werden können, lässt sich leicht nachvollziehen, da unterschiedliche Studiengänge, Wohnumstände und Studiumsformen die Lebensphase in sehr heterogener Weise beeinflussen können. Der Einfluss auf den Hochschulsport und im speziellen auf die Akzeptanz und Teilnahme lässt sich an folgenden Kriterien festmachen, die speziell bei der Hochschulsportorganisation explizit berücksichtigt werden müssen:

1. Sportinteresse des Studierenden
2. Ort der Sporteinrichtungen
3. Soziale Kontakte
4. Belastungen und Entbehrungen

Somit ergibt sich für den Hochschulsport heute die besondere Herausforderung den Bedürfnissen der Studierenden gerecht zu werden und ihnen die Gelegenheit zu geben, gemäß ihrer Interessen und Vorstellungen Sport treiben zu können.

Der bereits erwähnte Bologna-Prozess führte allerdings zu weiteren Herausforderungen im Verhältnis Hochschulsport-Studierende (vgl. Berthold & Leichsenring, 2007, S.7). Die Verkürzung der Studienzeit führte zu einem strenger vorgegebenen Studienplan und einer deutlichen Verknappung der persönlichen Gestaltungs- und Freizeitmöglichkeiten. Diese Faktoren beeinflussen die Möglichkeiten der Studierenden an der Teilnahme im Hochschulsport erheblich, sodass sich das Hochschulsportprogramm diesen speziellen Anforderungen noch stärker durch mehr Flexibilität und neue Angebotsformen stellen muss.

Neben dem Einfluss des Bologna-Prozesses auf die Studierenden zeigt die einher gehende Restrukturierung der Hochschulen auch direkten Einfluss auf die Hochschulsporteinrichtungen. Durch die Messung der Hochschulen anhand ihrer Erfolge in den Kernprodukten, schrumpfen die Möglichkeiten für zusätzliche Angebote, was sich auch in der Bereitstellung der Finanz- und Sachmittel widerspiegelt. Berthold und Leichsenring sehen dabei folgende vier Bedrohungen für den Hochschulsport:

1. **Einsparungen:** Linear an der allgemeinen Sparbemühungen beteiligt zu sein, kann den Hochschulsport bei bisher generell eher magerer Ausstattung vollends marginalisieren.
2. **Leistungsorientierung:** je mehr die Hochschulen von den Länder über eine leistungsorientierte Mittelverteilung finanziert werden, desto eher könnten die Hochschulen einen Anreiz empfinden, sich auf die in Kennzahlen ausgedrückten Kernleistungen zu konzentrieren.
3. **Studienzeitverkürzung:** Wenn im Rahmen der Umstellung auf gestufte Studiengänge tatsächlich eine striktere Einhaltung der Regelstudienzeit erreicht werden sollte, dann werden die Studierenden intensiver studieren. Das dürfte dazu führen, dass sie weniger Zeit für Mitwirkung am Sport zur Verfügung haben und das sie sich weniger ehrenamtlich in der Betreuung engagieren.
4. **Studiengebühren:** Ist bereits das Studium kostenpflichtig, so werden sich die Studierenden eher fragen, ob sie für Sportangebote zusätzlich zahlen wollen. Sollten sie in den Konflikt zwischen intensiver Belastung durch das Studium und seiner Finanzierung durch Nebentätigkeiten geraten, so dürfte die Betätigung im Sport das sein, was geopfert würde.

(siehe Berthold & Leichsenring, 2007, S.16)

Nach der bundesweiten Abschaffung der Studiengebühren ist diese Bedrohung zur Zeit nicht mehr so prekär, wie einst angenommen.

Neben den Studierenden gehören aber zunehmend auch andere Personengruppen zur Zielgruppe des Hochschulsports. Besonders die Bediensteten der Hochschule werden zunehmend in den Programmen mit speziellen Angeboten angesprochen. Speziell im Bereich Gesundheitssport haben sich verschiedene Programme an den Hochschulen oder auch durch übergeordnete Gremien, wie den Allgemeinen Deutschen Hochschulsportverband (adh), etabliert. Das Angebot ist dabei sehr unterschiedlich und stark abhängig von den verfügbaren Ressourcen, erstreckt

sich von der Teilnahme am allgemeinen Programm über spezielle Bediensteten Kurse bis hin zur individuellen Betreuung am Arbeitsplatz im Büro. **[Entwicklung von Gesundheitskursen]** Die Nachfrage nach gesundheitsfördernden Maßnahmen dieser Art erfreut sich derzeit einer hohen Nachfrage und wird sowohl vom adh als auch von den Hochschulsporteinrichtungen in der Mehrheit als aktuelle und zukünftige Kernaufgabe des Hochschulsports gesehen (vgl. Baumgarten, Lür & Dreiboldt, 2009). Die Potenziale des Hochschulsports, der wichtige Beitrag zur sozialen und integrativen Unterstützung sowie die positiven, gesundheitsfördernden Effekte auf die Studierenden werden von Göring und Möllenbeck (2010) zusammen gefasst.

Neben den Bediensteten lassen sich aber auch externe Teilnehmer als weitere Zielgruppe bestimmen. Die Einbindung dieser Personenkreise unterscheidet sich am stärksten in den unterschiedlichen Einrichtungen und birgt großes Potenzial und Gefahr gleichzeitig. Je nach Organisationsform stehen dem Hochschulsport ganz unterschiedliche Möglichkeiten offen aber auch das regionale Umfeld muss hier analysiert werden. Mit der Öffnung des Hochschulsports für externe Teilnehmer begibt man sich in direkte Konkurrenz mit kommerziellen Anbietern. Die Integrationsformen variieren von Kooperationen mit Vereinen, Firmen im Rahmen von Betriebssportangeboten, Rehakliniken und Gesundheitszentren bis hin zur freien Öffnung für Jedermann. Allgemein lässt sich jedoch beobachten, dass diese Gruppe vor allem dann angesprochen wird, wenn eine Auslastung durch Bedienstete und Studierende nicht möglich ist. Im Bereich Organisationsformen werde ich auf die Unterschiede noch detaillierter eingehen.

2.2 Typische Aufgaben

Dauerhafte Motivation und Möglichkeit zum Sporttreiben

Ausgleich zu den alltäglichen und universitären Belastungen

Gesundheitsförderung und entwickeln einer umfassenden Mitverantwortung für eine gesunde Lebensführung

Erwerb von Soft-Skills

Bereicherung des studentischen Lebens an der Hochschule und Integration über den Sport

Ausbildung von Übungsleitern

Profilbildung Hochschule

Erprobung neuer Trends

Unterstützung von studierenden Spitzensportlern bei der Vereinbarkeit von Studium und Sport

2.3 Organisationsformen

Wie bereits bemerkt lassen sich unterschiedliche Organisationsformen der Hochschuleinrichtungen finden. Radde (1996, S. 15) gliedert diese in drei Formen:

Die zentralen Einrichtungen

Sie ist die am häufigsten anzutreffende Organisationsform seit der Festschreibung des Hochschulsportes im Hochschulrahmengesetz. Unabhängig von einem Fachbereich und direkt dem Senat unterstellt hat sie die Aufgabe den Hochschulsport eigenständig zu organisieren und zu verwalten.

Die in das Aufgabenfeld der Institute für Sportwissenschaft integrierte Organisation des Hochschulsports

Diese enge Verbindung mit der Sportlehrer/innenausbildung kann sich auf das Hochschulsportangebot äußerst positiv auswirken, bringt aber auch durch notwendige „Prioritätensetzungen“ die Gefahr der Benachteiligung mit sich. Nicht selten kommt es zu konkurrierenden Ansprüchen bei Hallenzeiten, Personal und Finanzierungen.

Der durch studentische Selbstverwaltung organisierten Hochschulsport

Eine solche Form hat sich vor allem an kleineren Hochschulen bzw. Fachhochschulen etabliert, wo keine hauptamtlichen Sportlehrer/innen zur Verfügung stehen. Die Übernahme der Verant-

wortung durch Studierende begrenzt die Leistungsfähigkeit der Einrichtung, ermöglicht aber gleichzeitig den Erwerb von Kompetenzen für Studierende, die in anderen Organisationsformen nicht möglich sind.

2.4 Befragung ausgewählter Standorte

Im Rahmen der Anforderungsanalyse an ein Hochschulsportverwaltungssystem wurde eine Befragung an acht Hochschulsporteinrichtungen durchgeführt zur Ermittlung existierender Systeme, wichtiger Kernfunktionen und Prioritäten (Tabelle 2.1).

Name	Bundesland	Typ	TN/Jahr	Mitarbeiter	Übungsleiter	Kurse/Jahr
Hochschulsport Göttingen	Niedersachsen	übergreifend	15.000	20	150	550
Hochschulsport Münster	Nordrhein-Westfalen	übergreifend	20.000	30	500	n.a.
Hochschulsport Osnabrück	Niedersachsen	übergreifend	3.000	10	100	380
Hochschule Darmstadt	Hessen	exklusiv	1.000	2	40	120
Hochschulsport Uni des Saarlandes	Saarland	übergreifend	4.000/Woche	8	150	500
Hochschulsport Hamburg	Hamburg	übergreifend	10.000/Pers.	15	350	1400
Hochschulsport Berlin	Berlin	übergreifend	18.000	20	300	1150
Hochschulsport Kiel	Kiel	übergreifend	3.000	20	100	300

Tabelle 2.1: Übersicht befragter Hochschulen

Die Stichprobe stellt einen Ausschnitt unterschiedlicher Hochschulsporteinrichtungen bezüglich Teilnehmerzahl, Kursangebot, Bundesland und Mitarbeiter dar. Alle der befragten Einrichtungen gaben dabei an, als zentrale Einrichtung an der Universität bzw. Hochschule organisiert zu sein. Die Angaben zu den Teilnehmerzahlen im Jahr konnte nicht immer genau beantwortet werden, was auf folgende Gründe zurück zu führen ist:

- Die Teilnehmerzahlen können unterschieden werden nach Personenanzahl, Anzahl der Besucher je Kurs und Anzahl der Besucher je Kurstermin
- Die verwendeten Systeme konnten keine genaue Aufschlüsselung darübermachen, wie viele Personen und Kursbesuche in einem Definierten Zeitraum stattfinden
- Anmeldefreie Angebote werden in der Regel nicht mit Teilnehmerzahlen erfasst

Die daraufhin erfolgen Angaben sollen in diesem Fall als Schätz- und Näherungswerte dienen, um den Nutzerkreis etwa beurteilen zu können.

Die Befragung gliederte sich in vier Teile:

1. Allgemeine Angaben zur Hochschulsporteinrichtung (siehe Tabelle 2.1)
2. Was für ein System wird aktuell verwendet und wie zufrieden ist der HSP mit der aktuellen Lösung
3. Bewertung der Wichtigkeit einzelner Funktionalitäten für den HSP
4. Zusätzliche Anmerkungen

2.4.1 Aktuelle Lösungen

Um einschätzen zu können, mit Hilfe welcher Systeme die Organisation erfolgt, wurde ermittelt wie viele Einrichtungen eine spezielle Software für die Verwaltung des Hochschulsports nutzen. Der überwiegende Teil der Befragten mit 87,50 % gab an eine spezielle Software zu verwenden und nur 12,50 % nutzten eine andere Lösung in Form von Microsoft Excel und der Hochschulwebsite. Bei der Verwendung einer Hochschulsportverwaltungssoftware ergab sich ein Verhältnis von 71,43 % zum System BuchSys und 28,57 % mit der Software HSPinONE. Dies zeigt bereits die Verteilung der Systeme in der Hochschulsportlandschaft wobei das BuchSys-System noch weiter verbreitet erscheint, als schon angenommen. Die Verbreitung von HSPinONE ist dagegen als geringer einzuschätzen, da die Entwicklung abgebrochen wurde und diese Vertreter die einzigen Nutzer sind. Auf Nachfrage bei dem ehemaligen Entwickler, der Hochschulsport Marketing GmbH, wurde angegeben, dass der Bedarf an einem neuen System durchaus besteht, die Entwicklung aber nicht zur Ausrichtung des Geschäftsmodells passt. Drei der sechs Befragten, die nicht HSPinONE nutzen gaben ebenfalls an, die Software als Alternative evaluiert und in für die Zukunft in Betracht gezogen hätten. Die befragte Hochschule ohne spezielles Verwaltungssystem befindet sich derzeit in der Evaluierungsphase, welche Software sich für eine Anschaffung eignet.

Bei der Art des Betriebsmodells ob On-Premise oder Software-as-a-Service (vgl. Abschnitt 3.5) gaben 75,00 % der Befragten an das On-Premise Modell zu nutzen und 25,00 % das SaaS Modell. Die Nutzer des On-Premise Modells entschieden sich in erster Linie aus Datenschutzrechtlichen Gründen für dieses Modell und gaben teilweise an, dass keine Alternative zur Verfügung

stand. Die Nutzung einer SaaS Lösung hat keine dieser Hochschuleinrichtungen bisher in Erwägung gezogen. Die Nutzer des SaaS Modelles entschieden sich aufgrund der einfachen Wartung, unkomplizierte Umsetzung und die Anforderungen der Software für die Nutzung dieses Modells.

Die Nutzer der SaaS Lösung gaben an die Infrastruktur zu 100 % durch den externen Dienstleister betreuen zu lassen und die Kosten entsprechend dafür zu tragen. Die Nutzer der On-Premis Lösung hingegen gaben alle an, dass die Wartung und Instandhaltung für die Einrichtung ohne Kosten verbunden ist. Die Aufgabe der Betreuung wurde hingegen unterschiedlich verteilt. Hochschulsportmitarbeiter waren bei 83,3 %, Hochschulrechenzentren bei 50 % und externe Dienstleister bei immerhin 33,3 % der Hochschulsporteinrichtungen an der Betreuung beteiligt. Die Aussagen zu den Betreuungskosten ist in diesem Zusammenhang zu hinterfragen, da sie sich in versteckten Kosten äußern kann. Sind Hochschulsportmitarbeiter in die Betreuung involviert, so fällt es in deren Arbeitszeit, die entsprechend vergütet wird und die Arbeitskraft in dieser Zeit nicht für andere Tätigkeiten zur Verfügung steht. Die Unterstützung des Hochschulrechenzentrums kann in diesem Fall als Unterstützung der Hochschule gesehen werden, so lange hier für den HSP keine Kosten anfallen.

Wichtige Erkenntnisse sollte die Befragung im Fokus auf die Zufriedenheit liefern. Dazu musste auf einer sechsstufigen Skala eine Bewertung zwischen *gar nicht zufrieden (1)* und *sehr zufrieden (6)* gemacht werden. Die Hochschule ohne spezielles System soll hier nicht detailliert aufgeführt. Es zeigt sich jedoch dass nur in den Bereichen Kosten (zufrieden) und Ausfallsicherheit, Betreuung/Support (eher zufrieden) leicht positive Bewertungen gemacht werden konnten.

Untersucht wurde die Zufriedenheit im Hinblick auf die Aspekte Funktionsumfang, Anpassung der Darstellung, Anpassung der Logik, Dokumentation, Betreuung/Support, Weiterentwicklung, Kosten, Ausfallsicherheit, Performance und Bedienbarkeit (Abbildung 2.1).

Sehr zufrieden bis zufrieden wurden die Systeme in den Bereichen *Kosten* (\bar{x} 5,86; $\pm 0,38$), *Ausfallsicherheit* (\bar{x} 5,57; $\pm 0,79$), *Performance* (\bar{x} 5,57; $\pm 0,79$), *Bedienbarkeit* (\bar{x} 5,43; $\pm 0,79$) und *Funktionsumfang* (\bar{x} 5,14; $\pm 1,07$) bewertet. Die geringe Varianz bestätigt hier ein recht homogenes Bild in der Bewertung.

Unterschiede in der Beurteilung werden bei den weiteren Aspekten deutlich. Die Anpassungs-

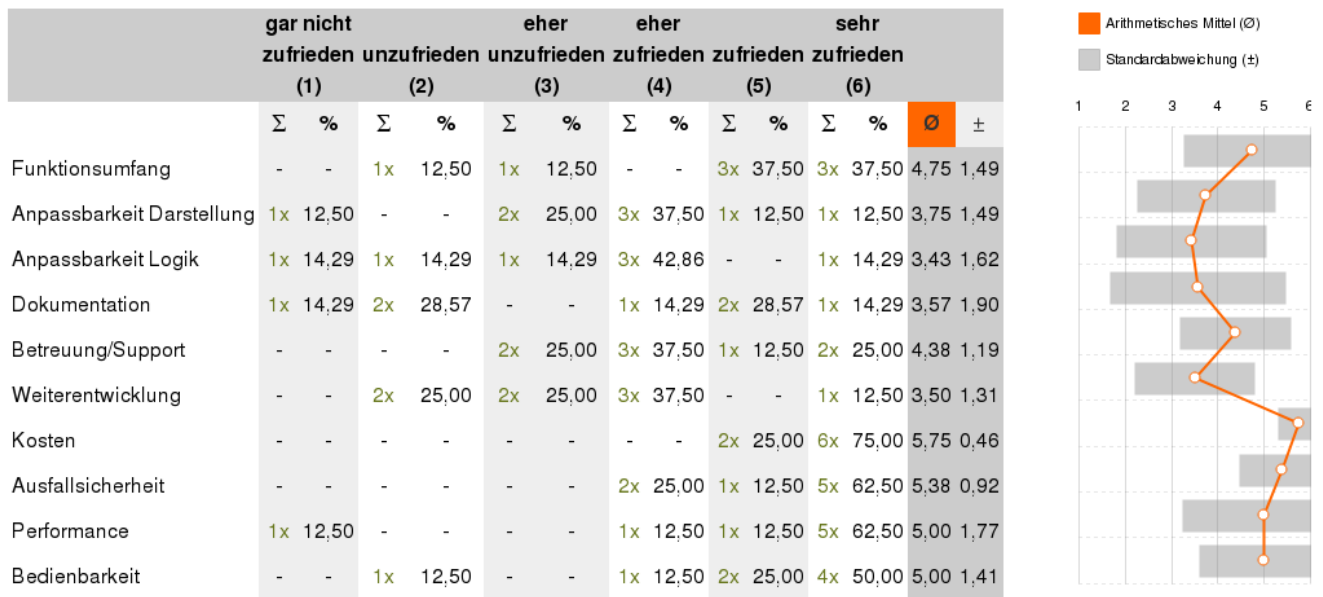


Abbildung 2.1: Zufriedenheit mit der aktuellen Lösung

möglichkeiten bezüglich *Darstellung* (\bar{x} 4,14; \pm 1,07) und *Logik* (\bar{x} 3,83; \pm 1,33) sowie die *Betreuung/Support* (\bar{x} 4,43; \pm 1,27) zeigen eine höhere Varianz, auch wenn die durchschnittliche Wahrnehmung noch gut ist. Die größten Unterschiede sehen die Hochschulsporteinrichtungen jedoch bei den Themen *Dokumentation* (\bar{x} 3,83; \pm 1,94) und *Weiterentwicklung* (\bar{x} 3,71; \pm 1,25). Es ist an dieser Stelle zu bemerken, dass es sowohl zum System BuchSys als auch zum System HSPinONE keine offizielle Dokumentation gibt. Dies mag für HSPinONE im Hinblick auf die Beendigung des Projektes noch nachvollziehbar sein, für ein System wie BuchSys mit hoher Verbreitung ist es jedoch unverständlich. Dass die Bewertung hier nicht negativer ausfällt, erscheint überraschend, jedoch wurde von den Hochschulsporteinrichtungen angegeben, dass eine grundlegende Dokumentation für das BuchSys System besteht, die aber unter strengen Sicherheitskriterien nur für Kunden zugänglich gemacht wird.

Als eine der kritischen Performance Stellen wurde das Verhalten des Systems zu Anmeldebeginn ausgemacht. Eine Anmeldung im Modus einer *Race Condition* oder *First come, first serve* unterliegt immer der Gefahr, bei sehr nachgefragten Inhalten zur Startzeit sehr hohe Nutzerzahlen bedienen können zu müssen. Die Frage nach der Häufigkeit von Beeinträchtigungen zu Anmeldebeginn wurde von 42,86 % mit gelegentlich, 28,57 % mit selten und 28,57 % mit nie beantwortet. Regelmäßige oder immer Beeinträchtigungen wurden nicht genannt. Eine Quan-

tifizierung schwierig und der Grad der Beeinträchtigungsbeurteilung subjektiv geprägt. 71,4 % der Befragten war jedoch schon von Problemen betroffen (vgl. Abbildung: 2.2).

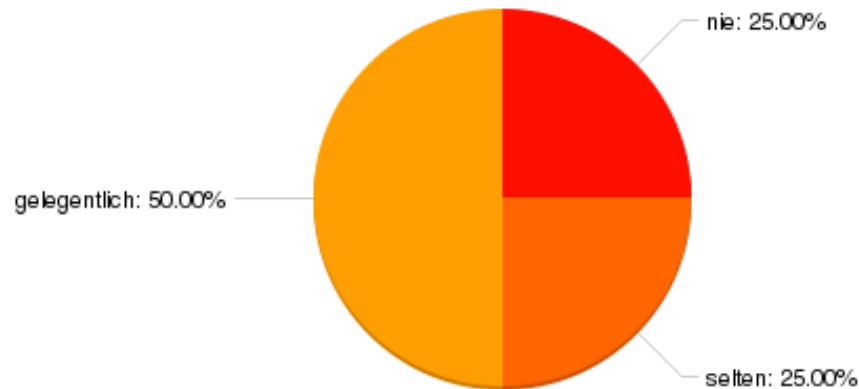


Abbildung 2.2: Häufigkeit der Beeinträchtigung zu Anmeldebeginn

Die anschließend Frage ob dafür Anpassungen in der Anmeldung gemacht werden mussten, wurde von 71,43 % der Einrichtungen mit nein und nur von 28,57 % mit ja beantwortet. Aus methodischer Sicht ist die Frage in diesem Fall unglücklich gewählt, da in anschließenden Gesprächen eine sehr unterschiedliche Interpretation zur Beantwortung führte. So liegt es im Ermessensspielraum, ob die Auslagerung der Buchungen aus dem verwendeten Content Management System (CMS) zu Anmeldestart eine Anpassung am Anmeldeprozedere ist oder nicht. Ebenso verhält es sich mit der bewussten Staffelung von Anmeldungen über einen gewissen Zeitraum, um die Last zu entzerren.

In der abschließenden Frage der allgemeinen Zufriedenheit antworteten sechs der acht Befragten mit ihrem System zufrieden zu sein.

2.4.2 Bewertung der Wichtigkeit von Funktionalitäten

Nachdem ein Überblick über die Bewertung der aktuell verwendeten Systeme gewonnen werden konnte, sollte im dritten Teil die Wichtigkeit einzelner Anforderungen und Funktionalitäten evaluiert werden. Dabei konnten wie Befragten wieder auf einer sechsstufigen Skala von + + +

(1) bis — — — (6) wählen. Zum Einstieg wurden erneut die nicht funktionalen Anforderungen abgefragt (Abbildung: 2.3).

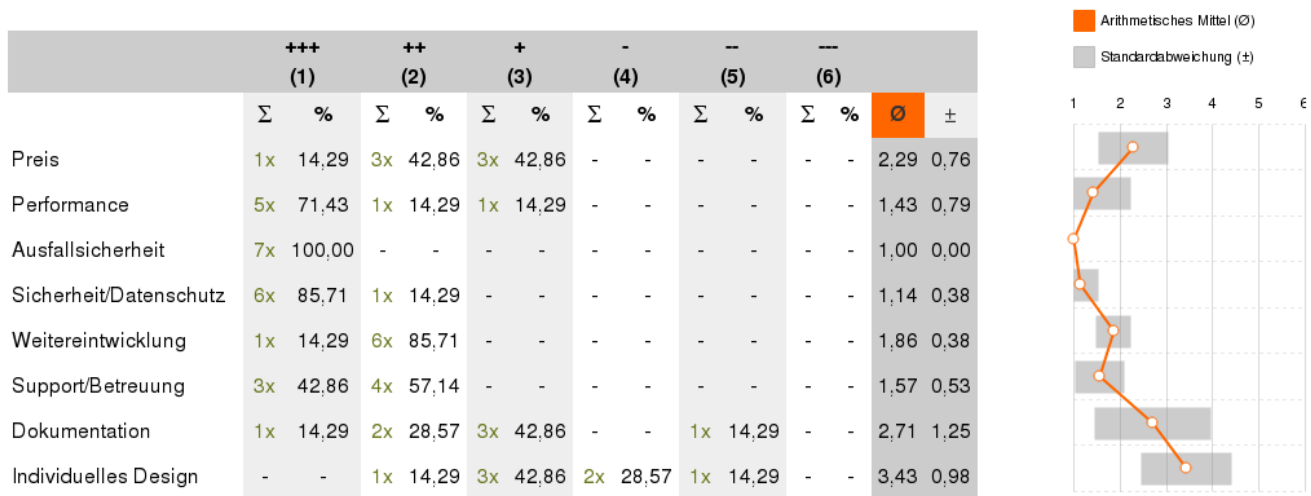


Abbildung 2.3: Zufriedenheit mit der aktuellen Lösung

Als wichtigste Faktoren wurden von allen Hochschulsporteinrichtungen die Aspekte *Ausfallsicherheit* (Ø 1,00; ±0,00) und *Sicherheit/Datenschutz* (Ø 1,14 und ±0,38) genannt. Dieses Resultat ist wenig überraschend, wurde doch der Datenschutz als entscheidendes Kriterium für die Auswahl eines Modelles genannt und durch die hohe Unterstützungsleistung im täglichen Geschäft ist eine höchst mögliche Ausfallsicherheit zwingend notwendig. Ebenfalls als sehr wichtig eingeschätzt wurden *Performance* (Ø 1,43; ±0,79), *Support/Betreuung* (Ø 1,57; ±0,53) und *Weiterentwicklung* (Ø 1,86; ±0,38). Bemerkenswert ist, dass der *Preis* mit Ø 2,29 und ±0,76 im Vergleich eine eher untergeordnete Rolle spielt. Deutlich unterschiedlich wird dagegen die Wichtigkeit der Bereiche *Dokumentation* (Ø 2,71; ±1,25) und *Individuelles Design* (Ø 3,43; ±0,98) wahrgenommen.

Eine anschließende Befragung sollte Aufschluss darüber geben, welche Endgeräte primär unterstützt werden sollten. Die Wichtigkeit von *Web* Ø 1,13; ±0,35 und *Mobil (Smartphone/Tablet)* Ø 1,38; ±0,74 wird als annähernd gleich wichtig gesehen während die *Desktop Anwendung* mit Ø 2,50; ±2,00 in der Wichtigkeit sehr unterschiedlich beurteilt wurde. Dies lässt sich vermutlich darauf zurück führen, was als Desktop Anwendung gesehen wird. Ist der Webbrowser in dem die Internetseite dargestellt wird jetzt zum Web zuzuordnen oder zur Desktop Anwendung? Was ist wenn der Webbrowser auf einem mobilen Endgerät verwendet wird? Es lassen sich in diesem

Fall Schwächen in der Fragestellung erkennen, jedoch wird ebenfalls deutlich, dass eine Internetbasierte Lösung von den meisten Befragten präferiert wird.

Die im Hinblick auf die Konzeption eines neuen Systemes wichtigste Frage beschäftigte sich mit der Einschätzung bestimmter Funktionalitäten. Die Befragten sollten 14 Funktionalitäten entsprechend ihrer Wichtigkeit beurteilen (siehe Abbildung 2.4).



Abbildung 2.4: Wichtigkeit einzelner Funktionalitäten

Insgesamt zeigt sich bei vielen der abgefragten Funktionalitäten ein sehr heterogenes Bild und eine hohe Streuung. Dies ist wenig überraschend, so sind die Anforderungen und Bedürfnisse an jeder Hochschulsporteinrichtung doch sehr verschieden. Insgesamt einig waren sich die Befragten bei der Wichtigkeit der Funktionen *Kursverwaltung* und *Teilnehmerverwaltung*. Sie wurden als elementar und mit der höchsten Wichtigkeit eingestuft (Ø 1,00; ±0,00). Diese Funktionen gehören zu den Eckpfeilern einer jeder Hochschulsporteinrichtung und ohne sie ist eine Verwaltung nicht möglich. Ähnlich verhält es sich mit den Übungsleitern, deren Verwaltung mit Ø 1,50 und ±0,76 ebenso hoch eingestuft wurden. Alle anderen Funktionalitäten sind gekennzeichnet durch eine hohe Standardabweichung von ±1,25 bis 2,00. Als wichtig eingestuft

werden zusätzlich die *Fitnessstudio* (\bar{x} 2,13; \pm 1,25), *Statistische Auswertung* (\bar{x} 2,50; \pm 1,41), *Verträge/Mitgliedschaften* (\bar{x} 2,50; \pm 1,60), *Automatisierte Zutrittskontrolle* (\bar{x} 2,50; \pm 2,00) und *granulares Rechtekonzept* (\bar{x} 2,63; \pm 1,51). Die Varianz kann am Beispiel Zutrittskontrolle gut fest gemacht werden. Während für Einrichtungen mit entsprechenden Vorrichtungen (z.B. Drehkreuz) die Funktionalität sehr wichtig ist, so ist sie für diejenigen die nicht darüber verfügen eher zu Vernachlässigen.

Als weniger wichtig beurteilten die befragten Einrichtungen die Funktionalitäten wie *Rechnungswesen* (\bar{x} 3,00; \pm 1,93), *Point of Sales* (\bar{x} 3,13; \pm 1,46), *Verschiedene Platzvergabe Modi* (\bar{x} 3,88; \pm 1,96), *Verleih* (\bar{x} 4,25; \pm 1,49) und *(Geschenk)Gutscheine* (\bar{x} 4,63; \pm 1,30). Bei der Betrachtung gilt es jedoch zu beachten, dass auch hier die Varianz meist sehr groß ist und diese Funktionalitäten für einige Einrichtungen doch eine deutlich höhere Wichtigkeit besitzen. Einzig bei der Funktionalität *Zeiterfassung für Mitarbeiter* (\bar{x} 5,50; \pm 1,41) ist man sich in der Unwichtigkeit weitgehend einig.

Betrachtet man die Resultate dieser Frage, so bestätigt sich das bereits gewonnene Bild der unterschiedlichen Anforderungen und Herausforderungen der einzelnen Hochschulsporteinrichtungen. Funktionen die für einzelne Einrichtung sehr wichtig sind, können für Andere sehr unwichtig sein. Die sehr allgemeine Fragestellung lässt hier jedoch viel Interpretationsmöglichkeiten, wie die einzelnen Begrifflichkeiten belegt werden. So ist das Konzept der Gutscheine inzwischen weit verbreitet in Form von Geschenkkarten für zum Beispiel iTunes, Amazon, Google Play, Ikea, etc. Eine Etablierung im Hochschulsport Umfeld bedürfte jedoch einer zusätzlichen Anpassung im Geschäftsprozess, da die Rechnungsstellung darauf Rücksicht nehmen muss.

Betrachten wir die Ergebnisse unabhängig von der Bewertung der einzelnen Funktionen, so wird sehr deutlich, welche Flexibilität von einem Hochschulsportverwaltungssystem erwartet wird, wenn annähernd alle Bedürfnisse berücksichtigt werden sollen. Um diese Anforderungen zu einigen Funktionalitäten noch weiter auszudifferenzieren wurden den Probanden drei weitere Fragen zu den Themenfeldern Point of Sales, Rechnungswesen (Bezahloptionen) und Fitnessstudio gestellt. Dabei ergaben sich folgende Ergebnisse:

Fitnessstudio

Die Bewertung der Anforderungen an spezielle Fitnessstudio Funktionen erwies sich als sehr stark abhängig von der Tatsache, ob in der Hochschulsporteinrichtung bereits ein Fitnessstudio

betrieben wird bzw. eines in Planung ist. Ein separates Programm (\bar{x} 2,50; \pm 1,97) war für die Probanden die wichtigste Funktionalität. 50 % der Befragten sahen es als sehr wichtig an. Die Möglichkeiten einer Trainerbuchung bzw. Personal Trainings (\bar{x} 3,43; \pm 1,62) und einer Warenwirtschaft (\bar{x} 4,00; \pm 1,41) wurden deutlich weniger wichtig angesehen.

Rechnungswesen (Bezahloptionen)

Im Rahmen einer Hochschulsporteinrichtung im Umfeld einer Hochschule haben sich viele Bezahloptionen etabliert. Die Veränderungen durch die Einführung von SEPA im Lastschrift Verfahren, hat diese Heterogenität weiter unterstützt und für Unsicherheit gesorgt. Bei der Frage wie wichtig derzeit die Unterstützung einzelner Optionen ist, lässt sich eine deutliche Abstufung erkennen. Das Lastschrift wird von allen Einrichtungen als zwingend notwendig erachtet (\bar{x} 1,00; \pm 0,00). Dies liegt vor allem in der gelebten Praxis begründet. So ist diese Verfahren bekannt und lässt sich leicht an die Finanzabteilung der Hochschule auslagern. Weiterhin als wichtig wird zudem die Möglichkeit der Rechnungsstellung erachtet (\bar{x} 3,13; \pm 2,23). Die hohe Standardabweichung zeigt jedoch schon hier, wie unterschiedlich die einzelnen Interessen gelagert sind. Dies setzt sich bei den Optionen Bargeld (\bar{x} 3,75; \pm 1,98), Unikarte (\bar{x} 3,75; \pm 2,12), EC-/Kreditkarte (\bar{x} 3,75; \pm 1,75) und Online-Bezahlssysteme (\bar{x} 3,88; \pm 1,96) weiter fort.

Point of Sales

Eine weitere Frage bezog sich auf die Unterstützung von Point of Sales (Verkaufs-)Plätzen. Ein PC basiertes Kassensystem (\bar{x} 2,88; \pm 2,10) und die Unterstützung von Kartenlesegeräten (\bar{x} 2,88; \pm 1,64) wurde im Mittel als wichtigste Unterstützungsleistung angesehen. Ein Barcode Scanner dagegen erscheint weniger wichtig zu sein (\bar{x} 3,63; \pm 1,69). Die Unterstützung zusätzlicher Kassensysteme auf mobiler Basis (\bar{x} 4,00; \pm 1,91) oder als Kassensystem von Drittanbietern (\bar{x} 4,86; \pm 1,21) steht auf der Prioritätenliste der Befragten hinten an.

2.4.3 Zusätzliche Anmerkungen

adfafdadfadfa

2.5 Probleme und Herausforderungen

3 Cloud Computing

Das erste Kapitel hat deutlich gemacht, welche Anforderungen im Hochschulsport an ein Software System gestellt werden, sowie welchen Dynamiken und Besonderheiten das fachliche Umfeld unterliegt. In diesem Kapitel soll ein Verständnis für unterschiedliche Softwarearchitekturen und Vertriebsmodelle geschaffen werden. Dabei soll vor allem auf das Thema Cloud Computing eingegangen werden, welches an Bedeutung gewinnt und Gegenstand vieler akademischen und wirtschaftlichen Beiträgen in der Fachpresse ist.

Für den Begriff Cloud Computing finden sich in der Literatur vielfältige Definitionen. Die häufigste Verwendung findet jedoch die Definition des National Institute of Standards and Technology (NIST), dessen Definition sich auch die European Network and Information Security Agency (ENISA) angeschlossen hat, in der Form:

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

(vgl. Mell & Grance, 2011, S.2)

Diese Definition des ist auch Grundlage für die Definition des Bundesamt für Sicherheit in der Informationstechnik (BSI) welches den Begriff Cloud Computing folgendermaßen festlegt:

Cloud Computing bezeichnet das dynamisch an den Bedarf angepasste Anbieten, Nutzen und Abrechnen von IT-Dienstleistungen über ein Netz. Angebot und Nutzung dieser Dienstleistungen erfolgen dabei ausschließlich über definierte technische Schnittstellen und Protokolle. Die Spannbreite der im Rahmen von Cloud Computing angebotenen Dienstleistungen umfasst das komplette Spektrum der Informationstechnik und beinhaltet unter anderem Infrastruktur (z. B. Rechenleistung, Speicherplatz), Plattformen und Software.

(vgl. Bundesamt für Sicherheit in der Informationstechnik, o. J.)

3.1 Merkmale

3.1.0.1 On-demand self-service

3.1.0.2 Broad network access

3.1.0.3 Resoruce pooling

3.1.0.4 Rapid elasticity

3.1.0.5 Measured service

(vgl. Mell & Grance, 2011, S.2)

3.2 Service Modell

In der Definition von (Mell & Grance, 2011, S.2) sind drei unterschiedliche Service Modelle beschreiben, die sich in der Cloud Community etabliert haben. Diese decken sich mit den Kategorisierungen von (Tharam, Chen & Elizabeth, 2010, S. 28) und (Jadeja & Modi, 2012, S. 878).

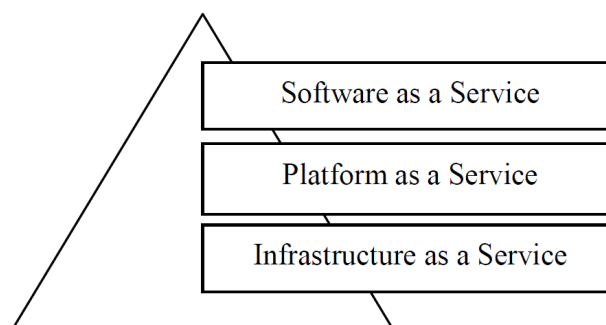


Abbildung 3.1: 3-Ebenen Modell von Cloud Computing

Software-as-a-Service (SaaS) stellt eine Anwendung in der Cloud bereit, die von Nutzern verwendet werden kann, ohne eine eigene Installation vornehmen zu müssen. Der Nutzer hat dabei keinen Zugriff auf die Cloud Infrastruktur. Wichtige Merkmale einer SaaS Anwendungen sind Netzwerk basierter Zugriff, Nutzer können limitierte spezifische Konfigurationen vornehmen und sind meistens als Multi-Tenancy-System aufgebaut. Beispiele sind Anwendungen wie Salesforce.com, Google Mail, Google Docs, Office 365, Quicken Online, etc.

Platform as a Service (PaaS) bezeichnet Software und Entwicklungstools, die der Nutzer verwenden kann, um eigene Anwendungen zu entwickeln und zu veröffentlichen. Der Nutzer hat dabei keinen Zugriff auf die darunter liegende Cloud Infrastruktur, kann jedoch im Vergleich zu SaaS die eigene Anwendung frei konfigurieren. Beispiele für PaaS Services ist Google AppEngine.

Infrastructure as a Service (IaaS) ist die unterste Ebene der Cloud Computing Pyramide. Dem Nutzer wird virtualisiert Hardware wie Datenspeicher, Virtuelle Maschinen, Netzwerke, etc. zur Verfügung gestellt, die frei verwendet werden können. IaaS Services werden gewöhnlich auf pay-per-use Basis bezahlt. Amazon EC2, VMWare und Windows Azure sind nur einige Beispiele.

In einigen Fällen werden diese drei Ebenen noch um zusätzliche Service Formen ergänzt. So verwendet (Tharam et al., 2010, S. 28) noch die Einteilung in storage as a Service (DaaS) während (Mahmood, 2011, S. 123) von other Provision as Service spricht und darunter Formen wie Storage as a Service, Database as a Service, Security as a Service, Communication as a Service, etc gruppiert. All diese Services können jedoch auch als spezialisierte IaaS Services angesehen werden und sollen daher in dieser Arbeit keine weitere Betrachtung erfahren.

3.3 Bereitstellung Modell

Ein weiterer zentraler Punkt in der Definition von Cloud Computing ist das Bereitstellungsmodell, das primär die Art beschreibt, wer Zugriff auf die Daten und Anwendungen hat.

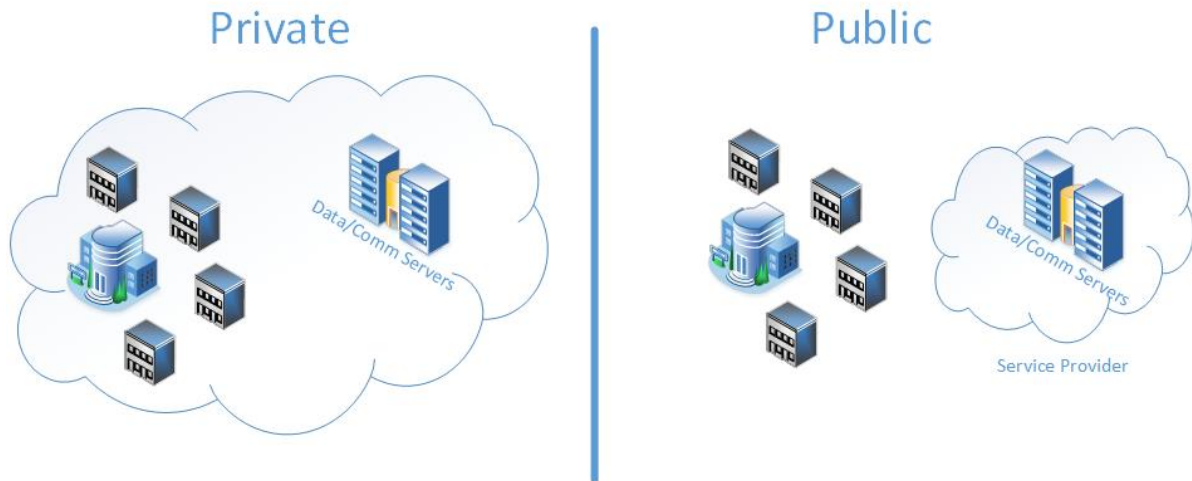


Abbildung 3.2: Public und Private Cloud Architektur

3.3.1 Private Cloud

Als Private Cloud wird das Szenario bezeichnet, wenn die Infrastruktur nur von einem Kunden genutzt wird. Im Normalfall ist sie im Organisationseigenen Rechenzentrum beheimatet. Sicherheit, Datenschutz, Wartung sind so einfacher durch eigenes Personal oder externe Dienstleister zu organisieren und zu überwachen. Die Private Cloud lässt sich daher mit einer erweiterten Art des Intranets vergleichen. (Jadeja & Modi, 2012, S. 879)

Tharam et al. (2010) nennen fünf Gründe, die für eine Private Cloud sprechen können:

1. Maximieren und optimieren bestehender, eigenen Ressourcen
2. Sicherheitsbedenken und Datenschutz
3. Datentransferraten
4. Vollständige Kontrolle über kritische Aktivitäten
5. Forschungs- und Ausbildungszwecke

3.3.2 Public Cloud

Dieses Modell ist derzeit in der Öffentlichkeit am besten bekannt. Nutzer können die Cloud über ein Web-Interface ansteuern und bezahlen für die Dauer der genutzten Services. Im Vergleich

zu anderen Modellen ist die Public Cloud weniger sicher, da sie leichter zugänglich für bösartige Angriffe ist. Bekannte Public Cloud Anbieter sind Amazon EC2, S3, Google AppEngine, Force.com und Microsoft Azure.

(vgl. Jadeja und Modi (2012) und Tharam et al. (2010))

3.3.3 Hybrid Cloud

Bei der Hybrid Cloud handelt es sich um eine Kombination aus zwei oder mehr Cloud Arten, die für sich selbst existieren, aber miteinander verbunden sind. Dadurch lassen sich eine Private und Public Cloud einfach kombinieren und Sicherheitsanforderungen kann gerecht geworden werden. Dies ist in erster Linie möglich wenn Datenschutz und Sicherheitskritische Anwendungsteile in die Private Cloud verlagert werden, weniger wichtige aber hoch nachgefragte Teile in die Public Cloud gestellt werden.

Hybrid Clouds sind die Treiber zur Notwendigkeit einer Standardisierung und Cloud Interoperabilität damit die Clouds miteinander kommunizieren können.

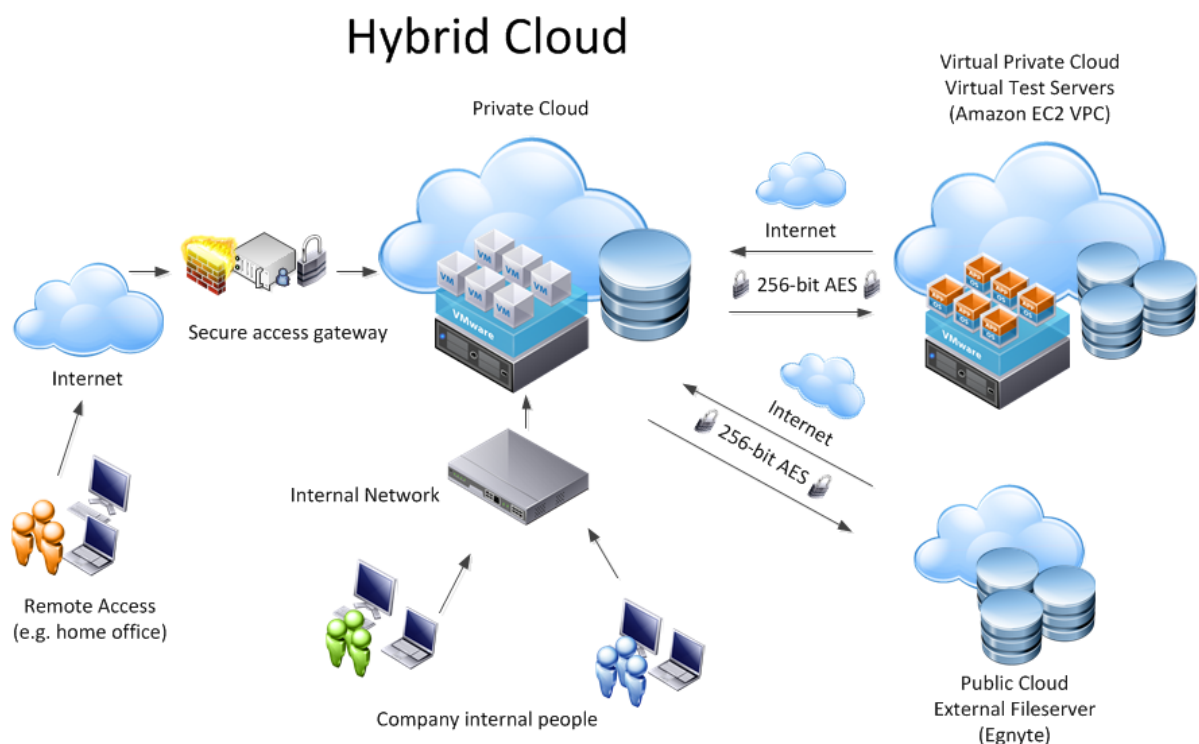


Abbildung 3.3: Hybrid Cloud Architektur

3.4 Pro & Contra

Nachdem der Begriff Cloud Computing, sowie die beinhalteten Service und Bereitstellungsmodelle erläutert wurden stellt sich die Frage nach den Vor- und Nachteilen dieses neuen Modells. Wie bei den meisten neuen Entwicklungen und Technologien ist die Entstehung getrieben aus den Unzulänglichkeiten bestehender Ansätze und so können die Merkmale, die diese Neuentwicklung definieren in vielen Fällen auch als Vorteile angesehen werden. Nachdem einige Gründe der Notwendigkeit bereits in Abschnitt ?? erläutert wurden sollen hier die Vor- und Nachteile jedoch noch einmal explizit zusammen gefasst werden. Der Fokus liegt hierbei vor allem auf den Services IaaS und PaaS, die etwas anders betrachtet werden können als SaaS. Auch wenn viele der Vor- und Nachteile auch für SaaS Modelle gelten, werden diese im Abschnitt 3.5 noch einmal explizit betrachtet. Verglichen werden dabei Cloud Anbieter mit dem traditionellen Rechenzentrum.

3.4.1 Vorteile

3.4.1.1 Einfachheit

Cloud Angebote leben von ihrer einfachen Einrichtung. Speziell als Anwender bzw. Kunde genügt in den meisten Fällen ein Webbrowser, um sich auf der entsprechenden Oberfläche die benötigten Ressourcen anzufordern und automatisiert bereit gestellt zu bekommen. Es handelt sich hier um eine Art Outsourcing, sodass die Fachkenntnis auf Hardware- und Softwareebene nicht mehr in der Tiefe notwendig ist, wie sie bei einem eigenen Betrieb vorhanden sein muss, um ein gleiches Qualitätslevel zu erreichen.

3.4.1.2 Kostenersparnis

Der Faktor Kostenersparnis gilt als einer der größten Vorteile, wobei die Berechnung der Kostenersparnisse für IT nicht all zu trivial ist, da sich je nach Modell diese ggf. erst nach Jahren

rentieren. Die Kosten mit eigener Verantwortung für die Bereitstellung und Unterhaltung ergeben sich primär aus den Beschaffungskosten für Hardware, Software und zusätzlich anteilig an den Personal- oder Dienstleisterkosten für Wartung und Instandhaltung. Mit einbezogen werden muss dabei zusätzlich, dass ggf. Neuanschaffungen und Erweiterungen über die Jahre notwendig sein werden.

Die Preismodelle von Cloud Anbietern wie Amazon Web Services (AWS), Microsoft Azure (Azure) oder Google AppEngine (AppEngine) unterscheiden sich dabei sehr von den traditionellen Abrechnungsmodellen. Auch wenn nicht jeder Anbieter alle Modelle unterstützt, so haben sich doch Abrechnungen nach Datenvolumen (wie viel Speicherplatz wird bereit gestellt), Computer Ressourcen (wie viel Arbeitsspeicher und CPU wird beansprucht), Datentransfervolumen (wie viel Datenvolumen wird über das Netzwerk transferiert) und einige andere in der Praxis etabliert. Der große Unterschied liegt jedoch in der Kombination mit dem Skalierbarkeitsfaktor. So wird im Gegensatz zum traditionellen Rechenzentrum nur das bezahlt, was wirklich benutzt wird und überschüssige Ressourcen können bei Bedarf einfach zurück gegeben werden. In der Literatur gibt es für diese Abrechnungsform jedoch bisher keinen einheitlichen Begriff und wird unterschiedlich als utility-style pricing, pay-as-you-go oder pay-per-use bezeichnet. (vgl. Jadeja und Modi, S. 879; Achimugu, Oluwagbemi und Popoola, S. 313)

Wie groß die Kostenersparnisse im Detail sein können ist sehr abhängig von den aktuellen Preisen, der Anwendungsarchitektur und der kalkulierten Laufzeit. In den letzten Jahren hat sich bei den Anbietern Amazon, Microsoft und Google ein richtiger Preiskrieg entwickelt, der zu stetig fallenden Preisen geführt hat und so die Kostenersparnisse weiter vergrößert hat.

3.4.1.3 Skalierbarkeit und Flexibilität

Elastizität und Skalierbarkeit ist ein weiter Hauptvorteil von Cloud Umgebungen. Datenspeicher und Computing Power steht in fast unbegrenzter Menge zur Verfügung und können mit wenigen Handgriffen vom User über einen Self-Service dazu provisioniert werden. So wie die Erweiterung der Ressourcen einfach möglich ist, so lassen sich diese auch wieder zurück geben wenn sie nicht mehr benötigt werden. Zusammen mit einer pay-per-use Abrechnung führt dies dazu, dass zusätzlich Ressourcen ggf. nur für bestimmte Zeiträume in Anspruch genommen werden müssen.

In traditionellen Rechenzentren ist das kurzzeitige hinzu schalten neuer Ressourcen und das anschließende Zurückgeben schwieriger. Werden neue Ressourcen benötigt, so stehen diese nur begrenzt zur Verfügung. Durch unterschiedliche Virtualisierung Umgebungen hat dies zwar auch zu mehr Flexibilität geführt, jedoch sind die Möglichkeiten begrenzt. Auch die Zeit, die bis zur Bereitstellung neuer Ressourcen benötigt wird ist im Normalfall deutlich länger, da der Prozess weniger automatisiert abläuft. Je mehr diese Automatisierung integriert ist, desto eher bewegt sich die Lösung im Bereich der Private Cloud.

3.4.1.4 Zukunfts- und Ausfallsicherheit

Ausfälle von Hardwarekomponenten ist in der Informationstechnologie ein Fakt der sich nicht vermeiden lässt und unterscheidet sich nicht in Cloud und eigenen Rechenzentren. Gerade in den großen Rechenzentren mit tausenden von Servern ist die Wahrscheinlichkeit für ein Auftreten eines solchen Ausfalles deutlich höher. Mit Redundanz und das Vermeiden von single point of failures wird versucht den Hardware Ausfall für den Endkunden unsichtbar zu machen und den Betrieb ungestört aufrecht erhalten zu können.

Durch die reine Anzahl von Hardware und spezialisiertem Personal sind die Cloud Anbieter in ihren Rechenzentren hier deutlich im Vorteil einen störungsfreien Betrieb zu garantieren. Zudem ist durch den Austausch kaputter oder veralteter Hardware eine sukzessive Erneuerung der Hardware ohne weiter Kosten für den Endkunden gegeben.

3.4.1.5 Nachhaltigkeit

Green IT, der umweltschonende Betrieb und Herstellung von Informationstechnologie, ist in den letzten Jahren von immer größerer Bedeutung geworden. Cloud Computing kann in diesem Bereich einiges beitragen, da Energiekonzepte einfacher optimiert und Ressourcen besser genutzt werden können. Siegele (2008) geht von einer durchschnittlichen Auslastung von Hardware Ressourcen mit 5%-20% in traditionellen Rechenzentren aus. Was auf den ersten Blick sehr wenig erscheint begründet sich darin, dass in Lastspitzen die benötigten Ressourcen um ein 2 bis 10-faches steigen. Um diesen Anforderungen gerecht zu werden, wird in vielen Fällen eine Provisionierung auf Basis des maximalen Workloads vorgenommen und führt zur Ressourcen- und Energieverschwendung in weniger aktiven Zeiten.

Durch das einfache hinzufügen von Ressourcen und die interne Ressourcen Verwaltung modernen Cloud Rechenzentren kann dies die Auslastung der einzelnen Systeme auch bei geringer oder moderater Belastung verbessern und so seinerseits zur Umweltschonung beitragen.

3.4.1.6 Beispiel für Vorteile

Diese Vorteile lassen sich an einem kleinen Beispiel im Hochschulsport mit fiktiven Zahlen verdeutlichen. Wie bereits in Abschnitt ?? beschrieben kommt es zu einem enormen zusätzlichen Performance Bedarf für das Buchungssystem zum Buchungsstart. Als Lastspitze wird ein fiktiver Bedarf von 100 Servern zu Grunde gelegt. Diese Lastspitzen treten etwa 10 mal im Jahr auf und halten für 24 Stunden an. Den Rest der Zeit besteht ein Workload für etwa 20 Server. Daraus ergibt sich über das Jahr ein durchschnittlicher Bedarf von 532,6 Serverstunden je Tag. Für eine Veranschaulichung des Kostenfaktors wird eine Serverstunde mit 50 Cent veranschlagt.

$$\text{Serverstunden je Tag: } \frac{((355 * 20) + (10 * 100)) * 24}{365} = \mathbf{532,6}$$

$$\text{Kosten je Tag: } \frac{532,6 * 50}{100} = \mathbf{266,3\text{€}}$$

Um den Kunden auch in den Lastspitzen die benötigte Performance zusichern zu können, so müsste in einem traditionellen Rechenzentrum die Bemessung des Bedarfs am Peak Load erfolgen und somit für 100 Server. Gemessen am durchschnittlichen Bedarf über das Jahr stellt dies jedoch eine deutliche Überprovisionierung da (3.4).

$$\text{Serverstunden je Tag: } 100 * 24 = \mathbf{2400}$$

$$\text{Kosten je Tag: } \frac{2400 * 50}{100} = \mathbf{1200\text{€}}$$

Dies entspricht einem erhöhten Preis um Faktor 4,5.

Ein alternativer Ansatz wäre eine moderatere Bemessung des Bedarfes mit z.B. 50 Servern und somit einer Unterprovisionierung möglich. Dies würde dazu führen, dass die Lastspitzen jedoch nicht mehr entsprechend versorgt werden könnten und es zu Einschränkungen und so-

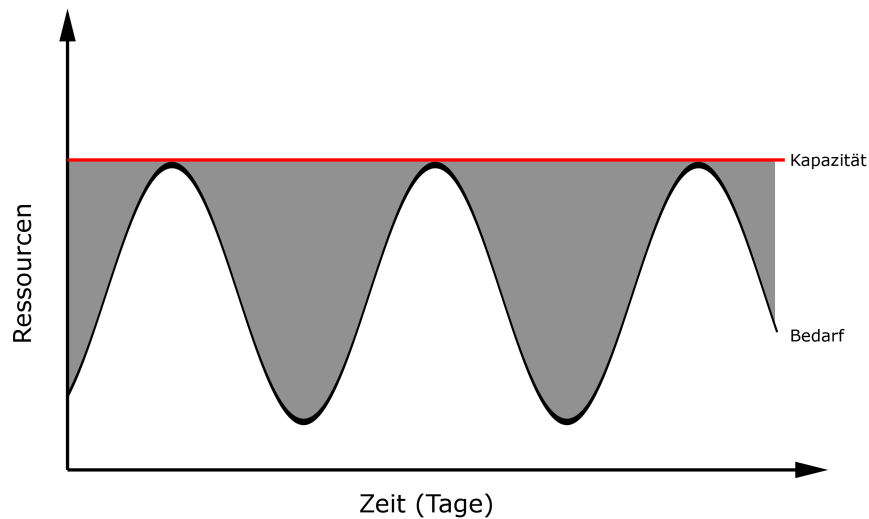


Abbildung 3.4: Provisionierung anhand von Lastspitzen

gar Ausfällen des System in diesen Zeiten kommen könnte (3.5). Mit diesem Ansatz ließen sich

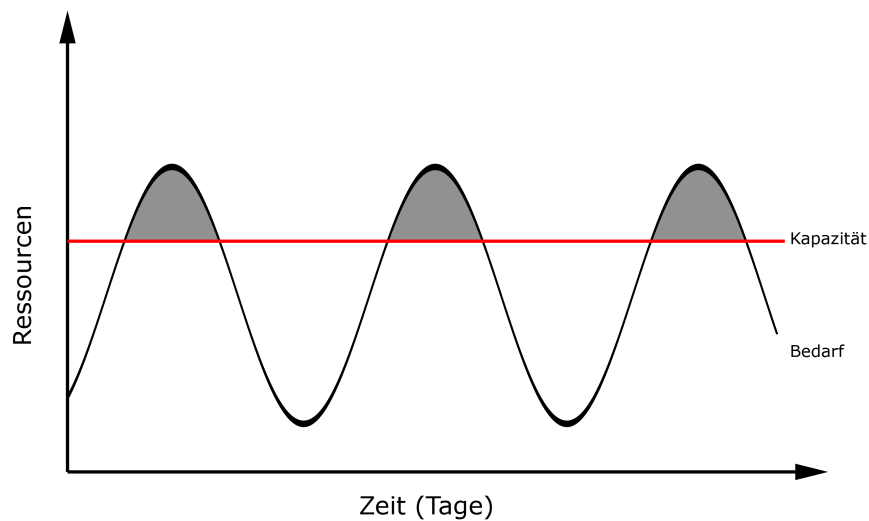


Abbildung 3.5: Unterprovisionierung

deutliche Einsparungen erzielen, jedoch muss einkalkuliert werden, dass es zu deutlichen Einnahmeeinbußen kommen könnte, da nicht allen Kunden eine Buchung möglich wäre oder sich Kunden sogar vollständig vom diesem Dienst entfernen.

Serverstunden je Tag: $50 * 24 = 1200$

$$\text{Kosten je Tag: } \frac{1200 * 50}{100} = \mathbf{600\text{€}}$$

Dieses auf fiktiven Zahlen basierende Beispiel ist in der realen Welt häufig anzutreffen und es müssen mit dem traditionellen Modell Kompromisse eingegangen werden, die unterschiedlich stark ausfallen können. Kosten-Nutzen-Optimierungen sind immer notwendig, bekommen mit dem Cloud Computing aber neue Möglichkeiten. Eine Umsetzung im Cloud Umfeld könnte wie folgt aussehen. Um einen reibungslosen Betrieb in den Phasen mit weniger Aktivität zu gewährleisten und einen gewissen Puffer zu integrieren, werden für das Jahr 25 Server provisioniert. Für die Anmelde Zeiträume werden jeweils für 48 Stunden weiter 75 Server dazu provisioniert. Daraus ergäbe sich folgende Rechnung:

$$\text{Serverstunden je Tag: } \frac{((355 * 25) + (20 * 100)) * 24}{365} = \mathbf{715,07}$$

$$\text{Kosten je Tag: } \frac{715,07 * 50}{100} = \mathbf{357,53\text{€}}$$

Dies entspräche dem 1,34-fachem des durchschnittlichen Bedarfes und ist in erster Linie dem Puffer geschuldet. Im Vergleich zur Überprovisionierung betragen die Kosten aber nur das 0,3-fache und im Vergleich zur Unterprovisionierung das 0,6-fache ohne die einhergehenden geschäftlichen Verluste. Dieses vereinfachte Beispiel macht so klar, welche Vorteile vor allem die Punkte Kosteneinsparung, Skalierbarkeit und Flexibilität in Cloud Umgebungen bringen zu können.

3.4.2 Nachteile

Neben den bereits genannten Vorteilen haben sich neue Technologien und Modelle mit einigen Problemen und Herausforderungen auseinander zu setzen, die hier ebenfalls vorgestellt werden sollen.

3.4.2.1 Sicherheit und Datenschutz

3.4.2.2 Abhängigkeit

3.4.2.3 Internetzugang

Da alle Public Cloud Dienste über das Internet angebunden werden, ergibt sich eine zusätzliche Internet Abhängigkeit, die ein Arbeiten bei gestörten Verbindungen unmöglich macht. Ob die Störung dabei beim Kunden oder beim Cloud Anbieter liegt ist dabei unerheblich. Im Vergleich zu den traditionellen eigenen Rechenzentren erhöht sich damit die Komplexität und die Anforderungen an die Internet-Bandbreite, die in lokalen Netzwerken leichter bereit gestellt werden konnte.

3.4.2.4 Verfügbarkeit

Die Zuverlässigkeit eines Anbieters gilt weiter als Nachteil, da mehr Schnittpunkte existieren. Ausfälle bei Amazon, Microsoft oder Google können zum Ausfall aller Systeme führen. Es muss dabei aber gesagt werden, dass diese Ausfälle auch in lokalen Rechenzentren jederzeit passieren und passieren können. Die Ausfallzeiten der etablierten Anbieter beschränkte sich in den letzten Jahren auf ein Minimum (QUELLE?)

3.4.2.5 Interoperabilität

Viele Cloud Anbieter bieten je nach Service viele eigenen Techniken, Produkte und Services an mit denen gearbeitet wird. Prinzipiell ist dabei ein Umzug zu einem anderen Cloud Anbieter nicht vorgesehen, da es zu Inkompatibilitäten führen kann. Dies kann zu großen Problemen führen, wenn diese Anbieter ggf. insolvent gehen oder der Kunde aus anderen Gründen den Anbieter wechseln möchte.

3.5 Software-as-a-Service Anwendungen (SaaS)

Wie bereits in den vorherigen Abschnitten beschrieben, bietet das Cloud Computing viele neue Möglichkeiten. In Abschnitt 3.2 wurde der Begriff Software-as-a-Service erläutert und gilt heute als einer der bekanntesten Begriffe aus dem Cloud Computing Service Katalog. Da sich diese Arbeit mit der Konzeption eines Softwaresystems für Hochschulsporteinrichtungen beschäftigt soll dieses Modell in diesem Abschnitt noch einmal genauer betrachtet werden.

Per Definition gibt es für SaaS-Anwendungen keine speziellen Einschränkungen, wie diese aufgebaut sein muss. Aus Anbietersicht lohnt sich eine SaaS-Anwendung jedoch nicht in allen Fällen, sodass das Ziel sein muss, mit einer Anwendung möglichst viele Kunden gleichzeitig versorgen zu können, ohne spezielle Anpassungen, Installationen etc. machen zu müssen. Damit wird das Multi-Tenancy Konzept zu einem zentralen neuen Bestandteil der meisten Anwendungen. Betrachtet man die Anwendungsmerkmal genauer, so lassen sich weitere Charakteristika erkennen, die im Normalfall erfüllt sein müssen:

- Generalisiert und anpassbar, sodass eine breite Nutzerbasis angesprochen werden kann
- Intuitive, einfache Bedienung und Navigation
- Modular und Service-orientiert aufgebaut
- Integrierte Messung und Monitoring für nutzungsbasierte Abrechnung
- Integrierter Rechnungsstellung
- Konstante Weiterentwicklung mit neuen Features
- Gewährleistung von Datenschutzrichtlinien
- Unterstützung von mehreren gleichzeitigen Kunden (Multi-Tenancy)

3.5.1 Vorteile

Betrachtet man die Vorteile, die auf Kundenseite durch SaaS-Anwendungen entstehen, so lässt sich leicht feststellen, dass diese zum Teil auf den Vorteilen von Cloud Computing basieren.

3.5.1.1 Kostenersparnis

Die Kostenersparnisse setzen sich aus mehreren Faktoren zusammen, um die SaaS Variante als Vorteil bezeichnen zu können. Betrachtet man die Kosten für die Anwendungsnutzung isoliert, so sind SaaS-Anwendungen in vielen Fällen sogar teurer als traditionelle on-premise Anwendungen. Bezieht man die Kosten für IT-Infrastruktur, Skalierungsmöglichkeiten und Wartung mit ein, so ändert sich das Verhältnis zu Gunsten der SaaS-Lösung, da diese für on-premise Lösungen ebenfalls aufgebracht werden müssen.

3.5.1.2 Zeitersparnis

Da die Software bereits beim Cloud Anbieter betriebsbereit ist, entfällt die Zeit für Installation und Einrichtung der Software und aller beteiligter IT-Infrastruktur Komponenten.

3.5.1.3 Fokus auf Geschäftsaufgaben anstatt Infrastruktur

Der Kunde wird von kostenintensiven, zeitaufwändigen Unterstützungsaufgaben für den Betrieb der IT-Infrastruktur befreit und kann sich auf die Kernaufgaben konzentrieren.

- Beschaffung und Wartung der hauseigenen IT-Infrastruktur, um die Software vor Ort zu betreiben
- Sicherstellung von Sicherheit, Reliabilität und Skalierbarkeit mit redundanter Hardware
- Aufrechterhalten eines Update und Upgrade-Prozesses

3.5.1.4 Direkter Zugriff auf Neuerungen

In herkömmlichen Anwendungen muss bis zum neuen Release gewartet werden, um neue Features nutzen oder Fehlerbehebungen einspielen zu können. Der Update-Aufwand und die Release Frequenz können diesen Prozess oft in die Länge ziehen. In SaaS-Anwendungen wird das Update vom Cloud Anbieter eingespielt und steht sofort allen Kunden zur Verfügung. Dem Anbieter ist im Rahmen des Wettbewerbsvorteils meist sehr daran gelegen, neue Features zeitnah zu Verfügung zu stellen und wird dem Kunden der Update Prozess abgenommen.

3.5.2 Herausforderungen

Viele Herausforderungen, die für herkömmliche Anwendungen gelten, sind auch bei SaaS-Anwendungen anzutreffen. Hinzu kommen jedoch weiter, neue Herausforderungen, die bisher nicht so häufig zu Tage traten jetzt aber eine höhere Relevanz bekommen. Eine der zentralen Herausforderungen umfasst das Thema Security. Daten und Anwendung werden an den Dienstleister weiter gegeben und unterliegt deren Verantwortlichkeit. Für viele Firmen, Organisationen und Anwender führt dies zu erheblichen Bedenken, die von Seiten der Anbieter besei-

tigt werden müssen und ein Vertrauen aufgebaut werden muss. Eine weitere Herausforderung ergibt sich in diesem Zusammenhang mit der Multi-Tenancy Architektur. Da SaaS-Anwendungen meist für mehrere Kunden ausgelegt sind, muss sichergestellt werden, dass jeder Kunde auch nur auf seine eigenen Daten zugreifen kann. Durch das Multi-Tenancy Prinzip muss die Anwendung mit einer hohen Zugriffszahlen und sehr unterschiedlichen Kundenwünschen zurecht kommen können. Dies stellt spezielle Anforderungen an das Architekturdesign und die Implementierung. Da SaaS-Anwendungen als ein spezieller Service des Cloud Computing zu sehen sind, beanspruchen sie für sich die Cloud Computing Vorteile für sich bestens nutzbar machen zu können. Damit werden besondere Ansprüche an die Kernpunkte Skalierbarkeit und Flexibilität gestellt. Diese drei Kernherausforderungen sollen im Anschluss betrachtet werden.

4 Software Architektur und Design

Software Architektur und Design sind komplexe Konstrukte, die von vielen Seiteneffekten **BE-NENNEN** beeinflusst werden. Ein passendes Konzept für die jeweilige Anwendung zu finden ist daher äußerst schwierig und kann sich im Lebenszyklus einer Anwendung durchaus ändern. Auch gibt es für SaaS-Anwendungen nicht die richtige oder falsche Architektur jedoch lassen sich anhand der Anforderungen geeignete und ungeeignete Modelle identifizieren.

4.1 Softwarearchitektur

Die Definitionen, was Softwarearchitektur ist und wie dieser Term zu definieren ist, unterscheidet sich bei einzelnen Autoren. Bass, Clements und Kazman (2013, S. 4) verwenden für mich eine sehr passende Definition des Begriffes:

The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.

Vor allem die Zusammenhänge sind hier von Bedeutung, denn auch wenn das Design der Architektur nicht endgültig sein muss und sich im Laufe der Entwicklung noch Veränderungen und Anpassungen ergeben könne, so ist es doch von Anfang an wichtig eine Vorstellung über die geplanten Zusammenhänge im gesamten Team zu entwickeln.

Für das Verständnis, welche Prinzipien sich für eine SaaS-Anwendung eignen, will drei grundlegende Architekturen darstellen, die sich in den Zusammenhängen unterscheiden. Jede diese Architekturen lässt sich optimieren, anpassen und so besser nutzbar machen und oftmals gibt es auch unterschiedliche Kombinationen die zum gewünschten Erfolg führen.

4.1.1 Monolitisch

Eine der am weitest verbreiteten Software Architekturen ist die monolithische Architektur. Gerade zentrale Server Anwendungen sind häufig nach diesem Prinzip aufgebaut. Die Stärken dieser Architektur liegen in:

-
- Einfach zu entwickeln - Das Ziel bestehender Entwicklungstools ist es die Erstellung monolithischer Programme zu unterstützen
 - Einfach auszuliefern - Nur die Anwendung als Ganzes muss ausgeliefert werden
 - Einfach zu skalieren - Hinter einem Lastverteiler können einfach mehrere Kopien der Anwendung laufen

Durch diese Vorteile hat sich dieses Architektur Prinzip weit verbreitet und ist jedem Entwickler bekannt. Gerade für kleine bis mittelgroße Projekte sind die genannten Faktoren entscheidend. Der interne Aufbau der Anwendung kann dabei jedoch sehr unterschiedlich gestaltet sein. In den meisten Fällen ist allerdings ein Schicht-Aufbau zu erkennen, der unterschiedliche Komponenten intern trennt. Gängige Schichten in einer Webanwendung sind folgende:

- Präsentation - Darstellung und Verwalten von HTTP Anfragen und Ergebnissen in HTML oder JSON/XML Format
- Geschäftslogik - Interne Logik für Geschäftsabläufe
- Datenbank Zugriff - Zuständig für alle Zugriffe auf die Datenbank
- Anwendungsintegration - Nachrichten Transfer, Email etc.

Richardson (2014) sieht jedoch auch eine Reihe von Nachteilen dieser Architektur, die mit zunehmender Größe der Anwendung und wachsender Teamgröße immer signifikanter werden.

- Die große monolithische Codebasis kann neue Entwickler schnell einschüchtern. Änderungen und Verständnis können schwierig sein und die Entwicklung verlangsamen. Durch fehlende Modulgrenzen geht im Laufe der Zeit die Modularität verloren.
- Mit wachsender Größe der Codebasis werden die Integrierte Entwicklungsumgebung (IDE) langsamer und bremsen die Produktivität
- Anwendungsstart wird zunehmend länger und hat einen großen Effekt auf die Produktivität der Entwickler und die Auslieferung. Wartezeit ist verschwendete Zeit.
- **Continuous deployment is difficult**

Traditional web application architecture

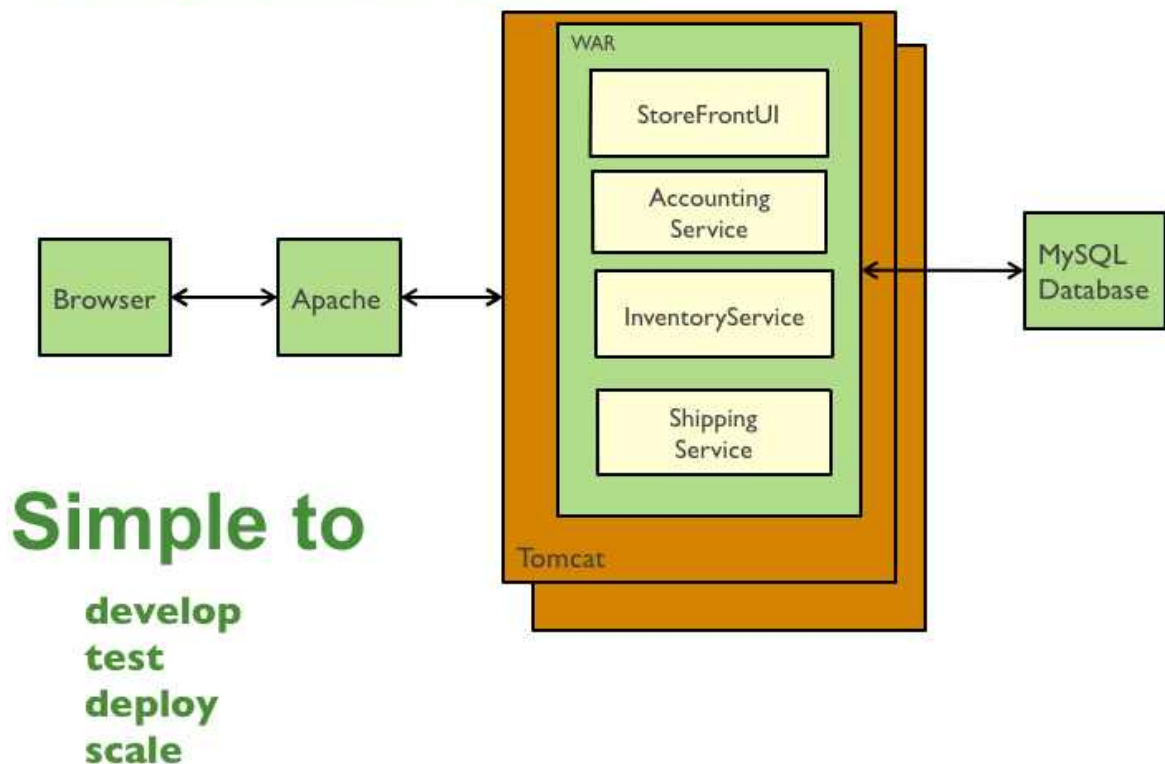


Abbildung 4.1: Traditionelle monolithische Webanwendungs-Architektur

- Das Skalieren der Anwendung kann schwierig werden, da es nur in eine Dimension skalieren kann. Mit wachsenden Transaktionen kann das System durch zusätzlich Instanzen skaliert werden. Hingegen kann bei einem wachsenden Datenvolumen diese Architektur nicht gut skalieren. Jede Instanz greift auf alle Daten zurück und führt zu wenig effektivem Caching, höherem Speicherverbrauch sowie zusätzlichem I/O traffic. Außerdem können unterschiedliche Anwendungsbereiche verschiedene Anforderungen z.B. CPU haben. Das skalieren von einzelnen Komponenten ist jedoch unmöglich.
- Die monolithische Architektur hat Grenzen wie die Entwicklung skalieren kann. Bei einer entsprechenden Anwendungsgröße werden Teams gebildet, die für spezielle Bereiche verantwortlich sind z.B. UI Team, Buchhaltungs Team etc. Die enge Verzahnung der einzelnen Bereiche erschwert jedoch die unabhängige Arbeit.
- Mit dem Beginn der Entwicklung muss sich auf eine Technologie und ggf. eine Version festgelegt werden, die es sehr schwer macht neue Technologien zu adaptieren. Eine in-

krementelle Umstellung gestaltet sich als schwierig, wobei eine vollständige Ersetzung der aktuellen Technologie sehr kostenintensiv ist und das Problem im Kern nicht behebt, da sich wieder auf eine neue Technologie für das Ganze festgelegt werden muss.

4.1.2 Client/Server-Architekturmodell

Im Gegensatz zur monolithischen Architektur handelt es sich beim Client/Server Modell um ein verteiltes System. Das Client/Server-Architekturmodell ist ein Systemmodell, das aus einer Anzahl von zugehörigen Diensten und Servern besteht, sowie aus Clients, die diese Dienste nutzen und auf sie zugreifen. Das einfachste Modell ist dabei das zweischichtige Client/Server Modell, die zusammen ein komplettes System bilden mit unterschiedlichen Zuständigkeiten. Dabei lassen sich drei logische Hauptbestandteile unterscheiden.

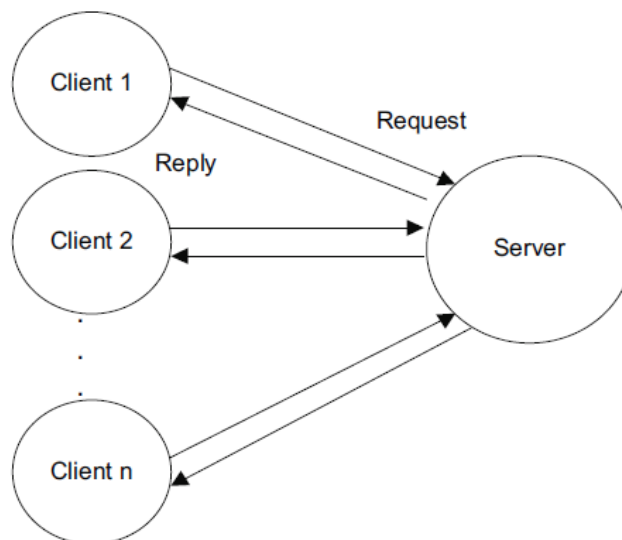


Abbildung 4.2: Client Server Model

4.1.2.0.1 Server

Der Server ist die zentrale Einheit der Architektur. Sie stellt Services und Daten anderen Einheiten zur Verfügung. Der Server muss nicht zwingend wissen, welche und wie viele Einheiten die von ihm zur Verfügung gestellten Dienste und Daten nutzen.

4.1.2.0.2 Client

Clients können in beliebig großer Anzahl existieren. Sie nehmen die Daten oder Services, die vom Server bereitgestellt werden in Anspruch. Um mit den Diensten interagieren zu können, muss ein Client von der Existenz und dessen Adresse wissen. Der Client sendet eine Anfrage an den Server und bekommt ein Resultat zurück geliefert. Sommerville (2007, S. 302) unterscheidet folgende zwei Client-Modelle:

Thin-Client-Modell

Bei diesem Modell werden die gesamten Anwendungsprozesse und Datenhaltung auf dem Server erledigt. Der Client ist nur für die Darstellung verantwortlich. Daraus ergibt sich der Vorteil, dass sehr wenig Ressourcen für den Client benötigt werden und dieser sich auch auf kleinen Systemen (z.B. Terminals) implementieren lässt. Der Nachteil dieses Modelles liegt jedoch darin, dass so sämtliche Arbeit Serverseitig geleistet werden muss. Gerade bei Anwendungen mit großer Client Anzahl oder rechenintensiven Operationen kann so der Server schnell zur kritischen Komponente werden und das System vor Skalierungsprobleme stellen. Zusätzlich führt der Thin-Client zu einer erhöhten Netzwerkbelastung, da alle Daten vom Server zum Client transportiert werden müssen.

Fat-Client-Modell

Dieses Modell verlagert die Anwendungslogik vollständig oder in großen Teilen vom Server direkt in den Client. Dadurch kann der Server deutlich entlastet werden und die Rechenleistung des Client Rechners zusätzlich genutzt werden. Dies kann sich in speziellen Situationen allerdings auch als Nachteile erweisen, wenn die Client Computer die notwendige Rechenleistung nicht selbstständig bereitstellen könne. Der Server fungiert dabei nur noch als Daten Service und die Netzwerklast wird reduziert. Die Verlagerung der Anwendungslogik auf den Client bringt aber auch Nachteile mit sich. Müssen hier Änderungen vorgenommen werden, so müssen alle Clients auf die neue Version aktualisiert werden, was zu einem komplexen Systemmanagement führen kann. Der Aufwand dafür steigt mit zunehmender Zahl der Clients.

4.1.2.0.3 Netzwerk (optional)

In der Literatur wird das Netzwerk häufig noch als dritter Bestandteil aufgeführt, der jedoch nicht zwingend notwendig ist da prinzipiell Server und Client auf einem Computer ausgeführt

werden können und so kein Netzwerk zur Kommunikation benötigen. Da dieses Modell aber in der Praxis nahezu ausschließlich als verteiltes System angewendet wird, ist in diesen Fällen das Netzwerk grundlegende Kommunikationsgrundlage.

Generell hat das zweischichtige Client/Server Modell das Problem, dass die drei logischen Schichten auf zwei unterschiedlichen Computersystemen abgebildet werden müssen.



Abbildung 4.3: Logische drei Schichten

Als Weiterentwicklung hat sich dafür die dreischichtige System-Architektur entwickelt. Sie stellt für jede logische Schicht ein separates Computersystem zur Verfügung. Die Vorteile liegen dabei in der besseren Skalierbarkeit, da jedes System individuell skaliert werden kann, geringere Netzwerklast im Vergleich zum Thin-Client-Modell und Anwendungsprozesse sind zentralisiert und lassen sich dort leichter anpassen und aktualisieren.



Abbildung 4.4: Dreischichtige Architektur

Sommerville (2007) gibt in seinem Buch Software Engineering Beispiele, wann die Anwendung der jeweiligen Architektur sinnvoll ist.

Architektur	Anwendung
Zweischichtige C/S-Architektur mit Thin-Clients	Anwendungen mit Legacy-Systemen, bei denen die Trennung von Anwendungsprozessen und Datenmanagement nicht praktikabel ist. Rechenintensive Anwendungen wie zum Beispiel Compiler mit geringem oder keinem Datenmanagement. Datenintensive Anwendungen (Browser oder Abfragesysteme) mit wenig oder keinen Anwendungsprozessen
Zweischichtige C/S-Architektur mit Fat-Clients	Anwendungen, bei denen die Prozesse durch Standardanwendungen (z.B. Microsoft Excel) auf dem Client verarbeitet werden. Anwendungen mit rechenintensiver Datenverarbeitung (z.B. Datenvisualisierung). Anwendungen mit relativ stabiler Endbenutzerfunktionalität, die in einer Umgebung mit gut strukturiertem Systemmanagement angeordnet sind
Drei- oder mehrschichtige C/S-Architektur	Anwendungen großen Umfangs mit Hunderten oder Tausenden von Clients. Anwendungen, bei denen sowohl die Daten als auch die Anwendung selbst ständigen Veränderungen unterliegen. Anwendungen, bei denen Daten aus vielfältigen Quellen verarbeitet werden

Tabelle 4.1: Verwendungsbeispiele von mehrschichtigen Architekturen Sommerville (2007)

4.1.3 Dienstbasiert

Neben den beiden bereits genannten Architekturen hat sich eine weitere Form verbreitet. Der Schwerpunkt wird dabei auf eigenständige Dienste als primäre Komponente gelegt, auf die über eine Art von Remote-Protokoll zugegriffen werden kann. Die Art des Protokolls ist dabei nicht festgelegt oder auf eines beschränkt wie z.B. Representational State Transfer (REST), Simple Object Access Protocol (SOAP), Advanced Message Queuing Protokoll (AMQP) oder andere. Dienstbasierten Architekturen werden signifikante Vorteile im Bezug auf Skalierbarkeit, Entkopplung, Kontrolle über die Entwicklung, Testing und Deployment zugeschrieben. Verteilte Systeme tendieren zudem dazu weniger abhängig und besser modular aufgebaut zu sein. Im Zusammenhang mit dienstbasierten Architekturen bedeutet das, dass jeder Dienst eine in sich

geschlossenen Einheit bildet, die von einem Team selbst designed, entwickelt, getestet und ausgeliefert wird. Abhängigkeiten zu anderen Komponenten sollten dabei nicht, oder nur minimal bestehen.

Diese Vorteile bringen jedoch auch einige Nachteile mit sich, so zählen erhöhte Komplexität und höhere initiale Kosten zu den meist genannten Argumenten.

4.1.3.0.4 Representational State Transfer (REST)

REST wurde erstmals 2000 von Roy Thomas Fielding in seiner Dissertation beschrieben. Im Gegensatz zu bekannten Protokollen wie Simple Object Access Protocol (SOAP) oder XML-RPC handelt es sich bei REST um einen Architekturstil, wie Anwendungen basierend auf dem HTTP Protokoll miteinander kommunizieren können. Unterstützt eine Anwendung die REST-Stil, so bezeichnet man sie als RESTful. (vgl. Melzer, 2010)

REST basiert auf einem Konzept von Ressourcen, über die der Service Kenntnis hat. Der Server erstellt dabei verschiedene Darstellungen, wobei diese entkoppelt ist von der internen Speicherung. HTTP und HTTPS sind die primär benutzten Protokolle bei der Verwendung von REST. HTTP Caching Proxies, Load-Balancers und Monitoring Tools besitzen eine tiefe Unterstützung für HTTP von Haus aus, die in der Entwicklung hilfreiche Dienste leisten. Den HTTP Verben GET, POST, PUT und DELETE lösen in RESTful Anwendungen spezielle Aktionen aus (siehe Tabelle 4.2).

HTTP Verb	REST Aktion
GET	Abfragen einer Ressource ohne Änderungen vorzunehmen
POST	Erstellt eine neue Ressource. Der Link zur neuen Ressource wird zurück gegeben
PUT	Eine Ressource wird angelegt. Wenn die Ressource bereits existiert, wird sie geändert.
PATCH	Ein Teil der Ressource wird geändert
DELETE	Löscht eine Ressource

Tabelle 4.2: Rest Operationen

Der große Vorteil und der Grund für die schnelle Verbreitung von REST Architekturen liegt in der Einfachheit. Methoden für die Nutzung des HTTP Protokoll sind in allen modernen Programmiersprachen enthalten und können einfach genutzt werden. Das Ressourcen Konzept ist leicht anwendbar und bietet umfangreiche Möglichkeiten der eigenen Anpassung.

Die Bandbreite der Möglichkeiten, die mit der REST Architektur umgesetzt werden können zeigt Leonard Richardson in seinem "Richardson Maturity Model" indem er vier REST Level beschreibt.

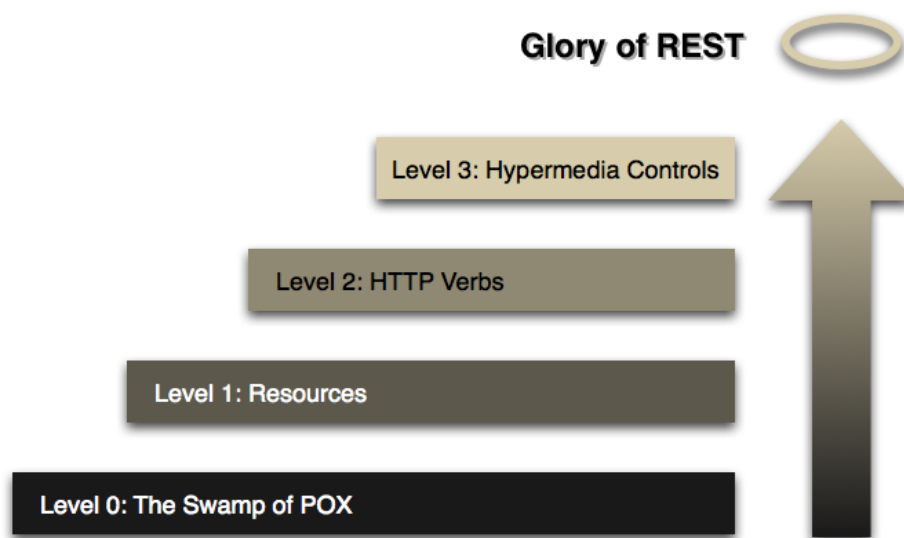


Abbildung 4.5: Richardson Maturity Model

(vgl. Fowler, 2010)

4.2 SaaS spezifische Besonderheiten

4.2.1 Multi-Tenant Application (MTA)

Bevor die MTA im Detail erklärt werden kann, ist es wichtig ein gemeinsames Verständnis von Tenant (dt. Mieter) und Multi-Tenancy (dt. Mehrfach Mieter) im Kontext von Software-Anwendungen zu schaffen. (vgl. Krebs, Momm & Kounev, 2012, S.2)

Tenant: Bezeichnet eine Gruppe von Benutzern, die die gleiche Sicht auf eine Software teilen. Dies bezieht sich auf die Punkte wie zugreifbare Daten, Konfiguration, User Management und

Funktionalität. Ein Tenant ist häufig eine (Teil-)Organisation wie zum Beispiel eine Hochschuleinrichtung.

Multi-Tenancy: Beschreibt ein Modell bei dem eine Anwendungsinstanz von mehreren Tenants genutzt wird aber jeder einen eingeschränkten, isolierten Zugriff erhält. Dabei werden Performance und Datenschutz Kriterien je Tenant behandelt.

Multi-Tenant Application: Teilt eine Anwendungsinstanz unter mehreren Tenants, um zusätzliche Kosten zu reduzieren.

Die primäre Herausforderung in MTA liegt daher in der sauberen Trennung von Abläufen und Daten zwischen den einzelnen Tenants. Chong, Carraro und Wolter (2006) zeigen in ihrem Artikel drei unterschiedliche Möglichkeiten, wie Multi-Tenant Daten verwaltet werden können.

4.2.1.0.5 Separierte Datenbanken

In diesem Ansatz werden die Daten für jeden Tenant in einer eigenen Datenbank gespeichert (vgl. Abbildung 4.6). Die Hardware Ressourcen und der Anwendungscode werden dabei geteilt. Innerhalb des Datenbank Management Systems (DBMS) bekommt jeder Tenant jedoch eine eigene Datenbank und auf Datenbankzugriffsebene wird eine Trennung zu anderen Nutzern hergestellt. Dadurch können mit dieser Methode Sicherheits- und Datenschutzaspekte am besten bedient werden. Dieser Ansatz macht es einfach das Datenmodell individuell zu erweitern und Daten aus einem Backup wiederherzustellen. Andererseits führt es zu höheren Wartungskosten, da mehrere Datenbanken verwaltet werden müssen und das Backup Prozedere komplizierter wird. Zusätzlich ist in einigen DBMS die Anzahl der Datenbanken beschränkt, sodass bei einer hohen Tenant Anzahl zusätzliche DBMS benötigt werden.

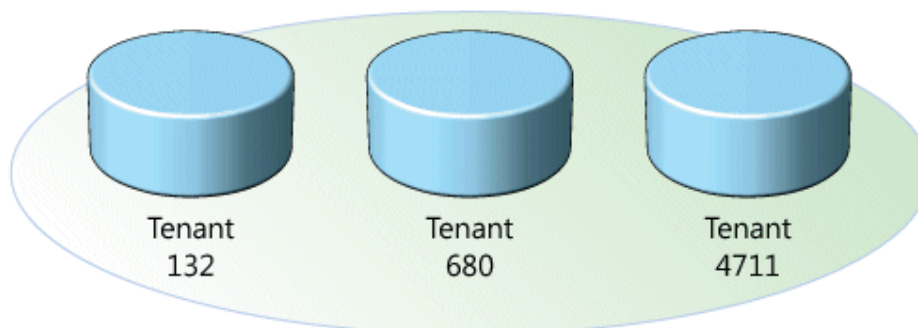


Abbildung 4.6: Separierte Datenbanken

vgl.

4.2.1.0.6 Geteilte Datenbank, separiertes Schema

Ein alternativer Ansatz ist es, alle Tenant Daten innerhalb einer Datenbank zu speichern, wobei jeder ein eigenes Schema und Set von Tabellen mit entsprechenden Zugriffsrechten bekommt. Die Implementierung ist relativ einfach und erreicht ein gemäßigtes Maß an logischer Isolation (vgl. Abbildung 4.7). Im Gegensatz zum Ansatz mit separierten Datenbanken lassen sich so jedoch deutlich mehr Tenants unterstützen. Das Wiederherstellen von Tenant spezifischen Daten aus einem Backup gestaltet sich in diesem Ansatz deutlich schwieriger. Da ein Datenbank Backup nicht einfach zurück gespielt werden kann, müssen zuerst alle Daten aller Tenants aus einem Backup in einen Temporären Server zurück gespielt werden, um sie anschließend von dort zurück in das Produktivsystem spielen zu können. Diese Vorgehensweise führt zu einem erheblichen Mehraufwand im Falle von Datenverlust.

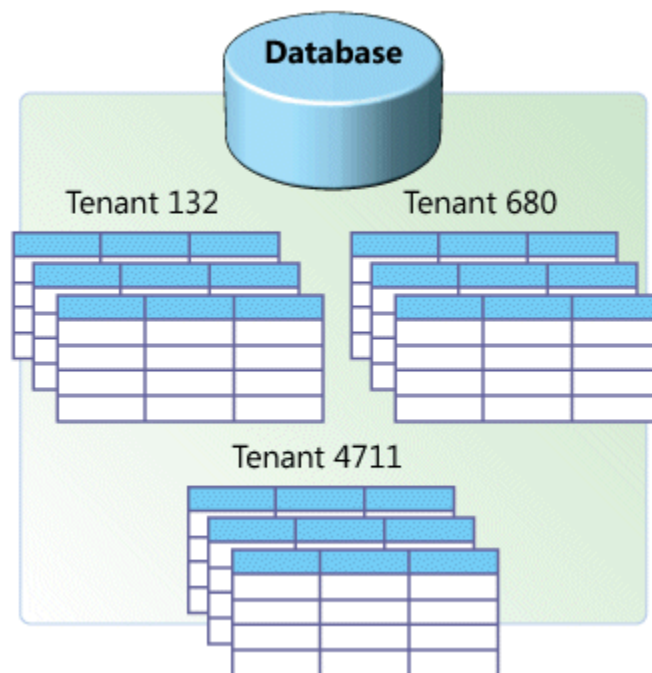


Abbildung 4.7: Geteilte Datenbank und separierte Schemata

vgl.

4.2.1.0.7 Geteilte Datenbank, geteiltes Schema

Dieser Ansatz basiert auf einer Datenbank und einem Set von Tabellen. Die Trennung der Daten erfolgt über das Mitführen einer Tenant-ID in jedem Datensatz wie in Abbildung 4.8 zu sehen. Diese Lösungsstrategie erlaubt die Versorgung einer großen Tenant-Anzahl per Datenbank und

erfordert daher die geringsten Hardware und Backup Kosten. Gleichzeitig erfordert es auch den höchsten Aufwand die Trennung von Tenant Daten anwendungsseitig sicherzustellen, um alle Sicherheits- und Datenschutzbedingungen zu erfüllen. Die Probleme beim Wiederherstellen von Backups verhalten sich wie beim Ansatz mit separiertem Schema. vgl.

TenantID		CustName	Address	
4	TenantID	ProductID	ProductName	
1	4	TenantID	Shipment	Date
6	1	4711	324965	2006-02-21
4	6	132	115468	2006-04-08
	4	680	654109	2006-03-27
		4711	324956	2006-02-23

Abbildung 4.8: Geteilte Datenbank und Schema

adfkadf

4.2.2 Security

5 System Konzeption

Nachdem in den ersten Kapiteln dieser Arbeit die grundlegenden fachspezifischen Anforderungen der Hochschulsporteinrichtungen an ein Softwaresystem sowie einige wichtige Konzepte der Systementwicklung vorgestellt wurden, soll im vierten Teil ein Konzept erstellt werden, wie eine Umsetzung unter Betrachtung der Anforderungen aussehen kann. Des Weiteren sollen in dieses Konzept Anforderungen aus Sicht des Software Erstellers mit berücksichtigt werden, so dass ein entstehendes Konzept theoretisch auch in der Praxis umgesetzt werden kann.

Der Aufbau der Konzeption erfolgt dabei in Anlehnung an Balzert (1996) und beinhaltet die Phasen:

1. Planungsphase
2. Definitionsphase
3. Entwurfsphase

5.1 Planungsphase

- Lastenheft
- Nutzeranforderungen
- Systemanforderungen
- Multi-Tenantcy
- PaaS, SaaS
- Glossar

- WAS und nicht WIE
- Priorisierung?

5.2 Definitionsphase

- Pflichtenheft

-

5.3 Entwurfsphase

- WIE

- Entwerfen einer Software-Architektur

- Zerlegung des definierten Systems in Systemkomponenten

- Strukturierung des Systems durch geeignete Anordnung der Systemkomponenten

- Beschreibung der Beziehungen zwischen den Systemkomponenten

- Festlegung der Schnittstellen, über die die Systemkomponenten miteinander kommunizieren.

Microservice Diagram

<https://dzone.com/articles/building-microservices-inter-process-communication-1>

<https://insidethecpu.com/2015/07/17/microservices-in-c-part-1-building-and-testing>

6 Diskussion, Zusammenfassung und Ausblick

Literaturverzeichnis

- Achimugu, P., Oluwagbemi, O. & Popoola, V. (2012). Architecting the Cloud: Prospects and Problems. *International Journal of Information*, 2 (3).
- Balzert, H. (1996). *Lehrbuch der Software-Technik - Software-Entwicklung*. Heidelberg [u.a.]: Spektrum Akadem. Verl.
- Bass, L., Clements, P. & Kazman, R. (2013). *Software architecture in practice* (3rd ed Aufl.). Upper Saddle River and NJ: Addison-Wesley.
- Baumgarten, K., Lüer, D. & Dreiboldt, S. (2009). Zukunftsperspektiven von Gesundheitssportangeboten im Hochschulsport. In A. Göring (Hrsg.), *Quo vadis Hochschulsport?* (Bd. 2, S. 69–85). Göttingen: Univ.-Verl. Göttingen.
- Berthold, C. & Leichsenring, H. (2007). Der Hochschulsport und die Reformen der Hochschulen. In I. Behrens & A. Göring (Hrsg.), *Abschied von Humboldt?* (Bd. Bd. 1, S. 5–19). Göttingen: Univ.-Verl. Göttingen.
- Bundesamt für Sicherheit in der Informationstechnik. (o.J.). *Cloud Computing Grundlagen*. Zugriff am 27.08.2016 auf <https://www.bsi.bund.de/DE/Themen/DigitaleGesellschaft/CloudComputing/Grundlagen/CloudComputing-Grundlagen.html>
- Chong, F., Carraro, G. & Wolter, R. (2006). *Multi-Tenant Data Architecture*. Zugriff am 17.08.2016 auf <https://msdn.microsoft.com/en-us/library/aa479086.aspx>
- Fowler, M. (2010). *Richardson Maturity Model*. Zugriff am 06.07.2016 auf <http://martinfowler.com/articles/richardsonMaturityModel.html>
- Göring, A. & Möllenbeck, D. (2010). Gesundheitspotenziale des Hochschulsports. *Prävention und Gesundheitsförderung*, 5 (3), 238–242. doi: 10.1007/s11553-010-0245-2
- Hachmeister, C.-D., Tabor, O. & Bürgel, E. D. (2004). *Hochschulsport 2004*. Dieburg/Gütersloh. Zugriff auf http://www.che-ranking.de/downloads/Hochschulsport_2004.pdf
- Jadeja, Y. & Modi, K. (2012). Cloud computing - concepts, architecture and challenges. In *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on* (S. 877–880). doi: 10.1109/ICCEET.2012.6203873

-
- Krebs, R., Momm, C. & Kounev, S. (2012). *Architectural Concerns in Multi-Tenant SaaS Applications*. At Setubal and Portugal.
- Mahmood, Z. (2011). Cloud Computing: Characteristics and Deployment Approaches. In *11th International Conference on Computer and Information Technology (CIT)* (S. 121–126). doi: 10.1109/CIT.2011.75
- Mell, P. M. & Grance, T. (2011). *SP 800-145. The NIST Definition of Cloud Computing*. Gaithersburg and MD and United States: National Institute of Standards & Technology.
- Melzer, I. (2010). *Service-orientierte Architekturen mit Web Services: Konzepte - Standards - Praxis* (4. Aufl.). Heidelberg: Spektrum Akademischer Verlag.
- Radde, G. (1996). Hochschulsporteinrichtungen - Eine vergleichende Betrachtung. *dvs-Informationen*, 11 (3), 15–17. Zugriff am 05.07.2016 auf http://www.sportwissenschaft.de/fileadmin/pdf/dvs-Info/1996/1996_3_radde.pdf
- Richardson, C. (2014). *Pattern: Monolithic Architecture*. Zugriff am 05.07.2016 auf <http://microservices.io/patterns/monolithic.html>
- Siegele, L. (2008). Let it rise: A special report on corporate IT. *Economist Newspaper* (8603).
- Sommerville, I. (2007). *Software-Engineering* (8. Aufl.). München and Boston [u.a.]: Pearson Studium.
- Tharam, D., Chen, W. & Elizabeth, C. (2010). Cloud Computing: Issues and Challenges. In *24th IEEE International Conference on Advanced Information Networking and Applications* (S. 27–33). Zugriff auf <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5474674> doi: 10.1109/AINA.2010.187
- Zandonella, B. (2005). *Pocket Europa - EU-Begriffe und Länderdaten* (1. Aufl Aufl., Bd. [04]). Bonn: Bpb.