

Business Technology

Architektur & Management Magazin

Expertenwissen für IT-Architekten, Projektleiter und Berater

CLOUD COMPUTING

SONDERDRUCK
der Firma arlanis Software AG

arlanis 

Wolkige Geschäfte –
Wozu überhaupt Cloud?

Die Enterprise Cloud 

Umdenken für Architekten





Den besten Servicemitarbeiter schnell vor Ort in den Businessworkflow einbeziehen

Kritische Kunden- probleme in Echtzeit lösen

Fast jede neue (oder zu erweiternde) Businessanwendung oder jedes neue Portal wird heute auf die Realisierbarkeit mittels Platform as a Service, respektive Cloud-Technologien, überprüft. Im Rahmen dieses Artikels werden wir eine Software-as-a-Service-Anwendung vorstellen, mit der Kundenprobleme erfasst und zeitnah weiter verarbeitet werden können. Dabei wird bei kritischen Fällen der Servicemitarbeiter in Echtzeit lokalisiert (via Android oder iPhone Mobile), alle relevanten Daten online überspielt und der Tageszeitplan mit der Zentrale synchronisiert.

Seit einiger Zeit lässt sich ein Wandel in der IT-Dienstleistungswelt beobachten. Branchensoftware wird immer mehr in Richtung der Cloud verschoben, speziell in die von den Herstellern bereit gestellten Plattformen [1]. Einige Beispiele dazu sind Azure von Microsoft [2], AWS von Amazon [3] oder auch Force.com von salesforce.com [4]. Der hinter diesem Konzept stehende Gedanke scheint auf den ersten Blick nur Vorteile zu haben. Anwender müssen keine Software mehr installieren und auch die Wartung entfällt. Entwickler müssen sich nicht mehr um Infrastruktur, benötigte Software und Verteilung der Ergebnisse kümmern. Eigentlich ist das die perfekte Welt für alle Beteiligten. Stück für Stück wurden so in den letzten Jahren die Dienste in das Internet gelegt (**Abb. 1**), nicht unbemerkt, aber eher ohne den sonst allgegenwärtigen Hype. Heute können wir fast alles als Dienst beziehen, seien es Festplattenspeicher, die Rechenleistung oder eben der Platz für ganze Anwendungen.

DAS PROJEKT IM ÜBERBLICK

Das Projekt hatte das Ziel, Firmen stärker mit dem Außendienst, den Servicemitarbeitern, zu vernetzen. Zeitgleich sollte im Büro, in der Rechnungslegung und Zeiterfassung sowie im Auftrags-Dispatching eine Infrastruktur mithilfe von Cloud-Lösungen aufgebaut werden. Für das gesamte Projekt wird keine lokale Software bei der Handwerksfirma mehr benötigt, außer natürlich Browser und Betriebssystem. Am besten können wir uns das anhand eines kleinen Beispiels und den Aktionen über die Zeit vorstellen:

1. **Kunde:** Nehmen wir einmal an, der Geschirrspüler ist eines Morgens defekt. Der potenzielle Kunde wird einen Handwerksbetrieb anrufen.
2. **Firma/Störungsannahme:** Alle Fälle werden am Telefon erfasst und in eine zentrale Falldatenbank eingegeben. Durch die Nutzung von Computer Telefonie Integration (CTI) ist der Kunde eventuell schon bekannt und hat somit weitere Vorteile: Er muss seine Adresse etc. nicht neu angeben.
3. **Firma/Dispatcher:** Der Dispatcher sieht alle neuen Fälle und kann aufgrund eines Location-based Systems den nächstgelegenen Handwerker (Servicemitarbeiter) finden, kennt über den integrierten Kalender alle Termine der Handwerker und stellt einen neuen Termin ein.
4. **Handwerker:** Der Handwerker sieht auf seinem mobilen Endgerät, dass ein neuer Termin eingetragen ist, und kann diesen akzeptieren oder ablehnen.
5. **Handwerker:** Der Handwerker kommt beim Kunden an, startet auf dem mobilen Endgerät die Zeiterfassung und stoppt diese nach getaner Arbeit. Gleichzeitig sieht der Dispatcher, dass der Fall gerade in Arbeit ist.
6. **Firma/Dispatcher:** Der Dispatcher sieht, dass die Arbeit erledigt ist und veranlasst eine Rechnung an den Kunden.
7. **Kunde:** Der Kunde freut sich und bezahlt die Rechnung.

Durch das Akzeptieren des Auftrags kann der Kunde informiert werden.

ARCHITEKTONISCHE HINTERGRÜNDE

Architekturbetrachtung: Eine der wichtigsten Anforderungen an das Projekt ist die Nutzbarkeit als reine Cloud-Lösung, das heißt, dass auf der Seite des Anwenders (Firma, Dispatcher und Servicemitarbeiter) keine Software installiert werden sollte. Zusätzlich sollte das System auf mobilen Endgeräten nutzbar sein und mit der Anzahl der Servicemitarbeiter skalieren. Ein kleiner Betrieb mit wenigen Mitarbeitern kann somit genauso die entwickelte Lösung nutzen wie auch einer mit vielen Mitarbeitern.

Im letzten Abschnitt haben wir exemplarisch den Ablauf eines Kundenfalls beschrieben. Dazu gehörten unter anderem der Anruf des Kunden, die Weiterleitung an einen Servicemitarbeiter und natürlich auch die Lösung des Falls. Alles zusammen wird jetzt in die Cloud gelegt, umgangssprachlich. Dazu schauen wir uns erst einmal an, was wir benötigen und wie es verknüpft werden kann (**Abb. 2**).

Die Teile des Ganzen: Betrachtet man das System im Überblick genauer, ist es eigentlich gar nicht mehr so schwer, den einzelnen Teilen die entsprechenden Aufgaben zuzuweisen. Sehr schön ist ersichtlich, dass sich im Internet der Dienste für alle benötigten Komponenten das Passende finden lässt:

1. **CRM-Lösung:** Eine Anwendung, die mit Kundendaten umgeht, sollte als Backend eine Customer-Relationship-Management-(CRM-)Lösung spendiert bekommen. Das kann wie in unserem Fall das CRM der Firma salesforce.com sein [5]. Hier werden die Anrufe dem entsprechenden Kunden zugeordnet und ein neuer Vorgang generiert. Auch die Dispatcher Console findet hier ihren Platz, denn die einzelnen Fälle müssen dem jeweiligen Servicemitarbeiter auf Basis der Standortermittlung übermittelt werden. Unsere CRM-Lösung bildet also eine Klammer um alle anderen Bestandteile.

2. *Location-based Data Store und Standortermittlung (Amazon EC2 und UDC)*: Die mobilen Endgeräte der Servicemitarbeiter senden den aktuellen Standort periodisch an einen zentralen Service [3]. Dort werden die empfangenen Geo-Daten aufbereitet [6] und bei Bedarf dem Dispatcher zur Verfügung gestellt.
3. *Mobile Endgeräte (Servicemitarbeiter, Android oder iOS)*: Die Servicemitarbeiter benutzen ein mobiles Endgerät. Dort befindet sich eine Anwendung, die den aktuellen Standort an den zentralen Geo-Location Service sendet und die zugewiesenen Arbeitsaufträge darstellt.
4. *Internetcomputer (Dispatcher, beliebiger Browser)*: Der Dispatcher benötigt einen beliebigen internetfähigen Computer. Er nutzt das CRM-System und die Dispatch Console, um einerseits neue Fälle für den Kunden anzulegen und andererseits den Kundenauftrag zu verfolgen und zu steuern.

Web Services als ideale Schnittstellen: Die Kommunikation zwischen den technischen Komponenten geschieht mithilfe von Web Services. Erst durch die Standardisierung der Beschreibung von Schnittstellen lassen sich Elemente aus verschiedenen Plattformen miteinander verbinden. Interessant ist, dass sich neben den fachlichen Services diese Technik auch für jegliche Form von Schnittstellen einsetzen lässt. Somit verlassen wir zwar den reinen SOA-Gedanken, sind aber offen für die restliche Welt. Weitere technische Informationen lassen sich unter [7] finden. Nun können wir von der Betrachtung eines einzelnen Web Service ausgehen und das Ganze auf eine Cloud-Infrastruktur ausrollen. Man stelle sich hier einfach eine Menge in Echtzeit vernetzter Teilsysteme vor. Alle sind kompatibel miteinander, da die Schnittstellen von jedem Teilsystem verstanden werden. Allerdings ist manchmal die Anpassung der Daten notwendig. Alle Informationen können in kürzester Zeit transportiert werden, da über die Web Services sehr direkt mit dem eigentlichen Teilsystem kommuniziert wird.

GESTALTUNG DER DISPATCH CONSOLE

Wie bereits erwähnt, ist die Dispatch Console ein zentraler Baustein des Projekts. Sie fragt die zur Verfügung stehenden standortbezogenen Daten ab und erlaubt so letztlich die Vermittlung eines Handwerkers an den Kunden. Dabei sammelt die Konsole auch Informationen, die eine aktive Betreuung des Kunden durch den Dispatcher erlauben. Für die Umsetzung dieses Projektteils bietet sich in der As-a-Service-Welt eine Webapplikation an, die in diesem speziellen Fall aufgrund der zentralen Bedeutung direkt in die CRM-Umgebung von salesforce.com integriert wird. Damit stehen für die Umsetzung serverseitig die Programmiersprache Apex zur Abbildung der Geschäftslogik und VisualForce Pages zur Gestaltung der Oberfläche zur Verfügung. Bei beiden Technologien handelt es sich um Eigenentwicklungen von salesforce.com, die später bei der Analyse des typischen Entwicklungsprozesses noch einmal genauer betrachtet werden.

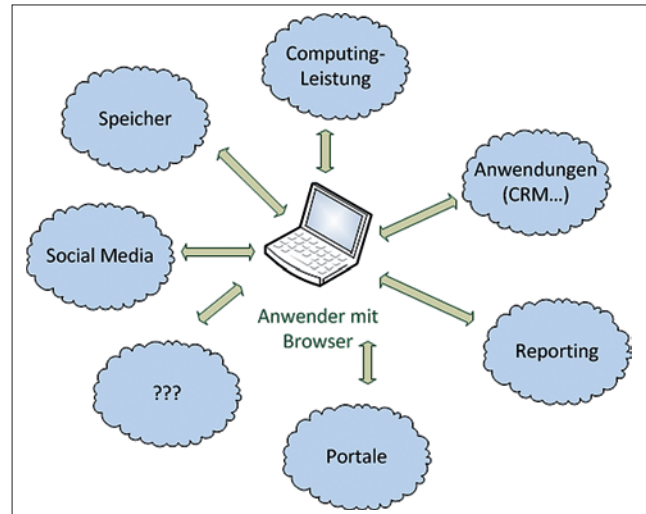


Abb. 1: Verschiedenste Software ist heute als Dienst im Internet vorhanden

Für die Benutzerinteraktion mit der Anwendung werden AJAX-Technologien (asynchronous JavaScript and XML) eingesetzt, insbesondere werden dabei Anfragen an den Server nach dem initialen Laden der Seite als XMLHttpRequest (XHR) gesendet, und die Antwort des Servers enthält nur den tatsächlich zu aktualisierenden Teil der Seite. Trotz dieser Optimierung entsteht eine merkbare Wartezeit, sobald eine Anfrage an die Cloud gestellt werden muss. Um nicht bei jedem Klick in der

com integriert wird. Damit stehen für die Umsetzung serverseitig die Programmiersprache Apex zur Abbildung der Geschäftslogik und VisualForce Pages zur Gestaltung der Oberfläche zur Verfügung. Bei beiden Technologien handelt es sich um Eigenentwicklungen von salesforce.com, die später bei der Analyse des typischen Entwicklungsprozesses noch einmal genauer betrachtet werden.

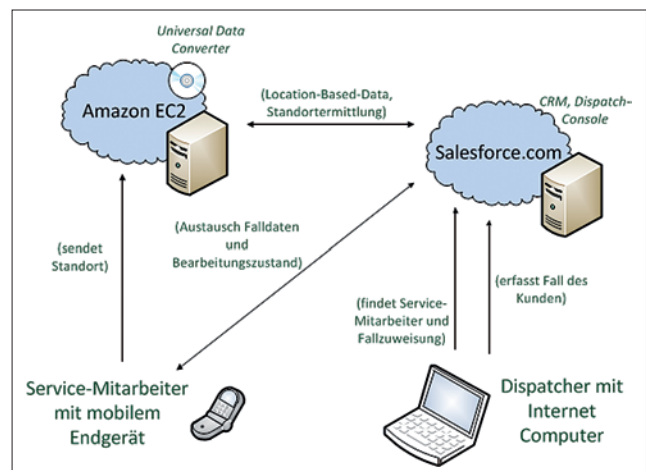


Abb. 2: Unser System im Überblick

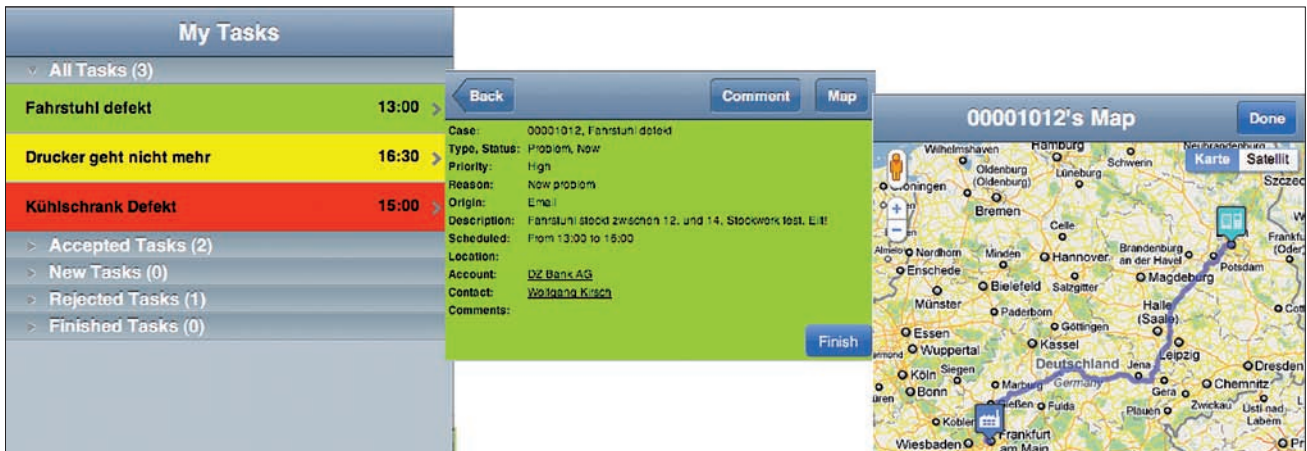


Abb. 3: Beispielhafte Ansicht von Seiten des mobilen Clients mit Google Maps Integration

Dispatch Console auf eine Antwort des Servers warten zu müssen, wird der Einsatz von JavaScript clientseitig erweitert, etwa zur Änderung der Sortierfolge von bereits geladenen Wertelisten oder zum Wechsel zwischen verschiedenen vorgeladenen Ansichten der Benutzeroberfläche. Neben der Optimierung der Geschwindigkeit durch Reduktion der zwischen Client und Server übertragenen Datenmenge können die Fähigkeiten von JavaScript in Kombination mit geeigneten Stylesheets auch zur Modifikation des Document Object Model (DOM) genutzt werden, um Aussehen und Bedienung der Dispatch Console so weit wie möglich einer Desktopanwendung anzunähern: So schafft zum Beispiel die Einblendung modaler Dialoge die Möglichkeit, Informationen zur richtigen Zeit in vollem Umfang einzublenden, was die Übersichtlichkeit der Bedienoberfläche immens erhöht.

Da salesforce.com als mehrmandantenfähiges System die Basis für die Dispatch Console ist, muss schon während der Designphase durch die Wahl der Datenstrukturen und der Art der Integration die Unabhängigkeit von der Ausprägung einer konkreten Instanz sichergestellt werden. Ferner ist für den erfolgreichen Vertrieb und die Installation in beliebigen salesforce.com-Instanzen eine Zertifizierung als AppExchange-Paket unerlässlich.

ANBINDUNG MOBILER ENDGERÄTE

Das mobile Endgerät, das von den Außendienstmitarbeitern beziehungsweise Handwerkern verwendet wird, muss zur Erfüllung seiner Aufgabe in diesem Projekt natürlich mit einem GPS-Chip ausgestattet sein, da nur so die Ermittlung des aktuellen Standorts des einzelnen Mitarbeiters vorgenommen werden kann. Ferner muss das Gerät in der Lage sein, mobile Daten zu senden und

zu empfangen. Viele der heute im Handel erhältlichen Smartphones erfüllen diese beiden Bedingungen schon. Es muss jedoch beachtet werden, dass für den erhöhten Energiebedarf der permanenten GPS-Ortung geeignete technische Maßnahmen ergriffen werden müssen, um die Stromversorgung des mobilen Geräts für einen ganzen Arbeitstag sicher zu stellen. Softwareseitig stehen im Fokus der Betrachtung zunächst Endgeräte, die unter iOS oder Android betrieben werden. Da unter den beiden genannten Betriebssystemen standardmäßig WebKit-Browser zum Einsatz kommen, die bereits einige Bestandteile von HTML5 unterstützen sowie über eine leistungsfähige JavaScript-Engine verfügen. Dadurch kann der mobile Client ganz im Sinne des as-a-Service-Gedankens als HTML5-Webapplikation bereitgestellt werden, wodurch eine Softwareinstallation auf dem Endgerät gänzlich vermieden werden kann. Ferner steht durch dieses Design den Unternehmen mit heterogenen Umgebungen ein einheitliches Featureset sowie eine einheitliche Oberfläche auf den unterschiedlichen mobilen Endgeräten zur Verfügung. Die vom Browser darzustellenden Seiten werden auf derselben Plattform betrieben, die auch die Dispatch Console bereitstellt, was zudem eine Bündelung der Entwicklungsarbeit ermöglicht.

Es darf jedoch nicht verschwiegen werden, dass die mit HTML5, JavaScript und CSS realisierten Anwendungen nur einen Teil des Leistungsspektrums der Mobilgeräte ausschöpfen können, da dem Browser aus Sicherheitsgründen der Zugriff auf einige Funktionen der Systemsoftware und Hardware verboten ist. Daher kann es je nach Anforderung erforderlich werden, native Apps für die einzelnen Mobilplattformen zu entwickeln. Bei diesem Vorgehen eröffnet die Verwendung eines Cross Compilers wie Phone Gap [8]

einen sanften Migrationspfad von der HTML5 App hin zur nativen App, denn die Entwicklung für die unterschiedlichen Plattformen kann immer noch auf einer gemeinsamen Codebasis stattfinden. Sobald jedoch eine native App vorliegt, muss diese auch alle für die jeweilige Plattform geltenden Vorgaben erfüllen, da erst nach einer Zertifizierung für AppStore beziehungsweise Marketplace ein erfolgreicher Vertrieb des Produkts möglich wird.

Eine besondere Herausforderung bei der Entwicklung für mobile Geräte stellt auch die Anpassung des Inhalts an die jeweiligen Endgeräte dar, denn sowohl auf einem kleinen Smartphonebildschirm als auch auf einem Tablet-PC muss im Sinne der Benutzbarkeit der Anwendung eine gut lesbare und übersichtliche Darstellung gewährleistet sein.

ORTSBEZOGENE DIENSTE

Bevor genauer auf den Entwicklungsprozess der soeben vorgestellten Komponenten eingegangen wird, sind noch ein paar Details zur Ortung von Interesse. Der GPS-Chip des mobilen Geräts ermittelt während des Betriebs fortlaufend den aktuellen Standort und sendet diesen in fest vorgegebenen Intervallen, etwa ein Mal pro Minute, an die für die Verarbeitung der Geo-Informationsdaten vorgesehene Plattform. Wie aus der Architektur des Systems hervorgeht (Abb. 2), wird zum Sammeln der Ortungsdaten der Außendienstmitarbeiter eine andere Plattform genutzt als für die Bereitstellung von Dispatch Console und mobiler App. Das geschieht, um die Skalierbarkeit des Projekts für eine beliebige Anzahl mobiler Endgeräte zu gewährleisten, denn salesforce.com stellt nicht beliebig hohe CPU-Ressourcen für die Datenverarbeitung bereit. Somit ist die Wahl einer reinen Infrastrukturplattform wie [3], die bei Bedarf einfach mit zusätzlichen CPU-Ressourcen erweitert werden kann, in Kombination mit dem UDC zur Bereitstellung der Web-Services-Schnittstelle und zur Durchführung der Datenaufbereitung die richtige Wahl. Nachdem die Daten aller Außendienstmitarbeiter vom UDC aufbereitet und gesammelt wurden, werden sie von der Dispatch Console periodisch abgefragt, damit der Dispatcher stets auf dem aktuellen Informationsstand ist.

Neben der Information über Längen- und Breitengrad sowie die Messgenauigkeit steht auf dem mobilen Endgerät bei gutem Satellitenkontakt auch die Geschwindigkeit des Geräts und die Höhe über dem Meeresspiegel des aktuellen Standorts zur Verfügung. Zur Interpretation dieser Rohdaten bietet sich wiederum ein anderer Dienst aus der Cloud an, z. B. Google Maps [9]: Neben der reinen Darstellung der Positionsangabe auf einer Karte besteht beispielsweise auch die Möglichkeit, die

Rohdaten in eine reguläre Adresse zu konvertieren und eine Route von der aktuellen Position zu einem anderen Ort berechnen zu lassen (Abb. 3).

Mit diesen Features lässt sich im Rahmen des Projekts sowohl für den mobilen Client als auch für die Dispatch Console ein guter Mehrwert erzielen, wenn gleich bei der Umsetzung natürlich auf die Nutzung des Dienstes gemäß den geltenden Bedingungen geachtet werden muss.

EINBLICKE IN DEN ENTWICKLUNGSPROZESS

Nachdem das Zusammenspiel und die Funktionsweise der einzelnen Komponenten beschrieben wurde, bleibt noch der Blick auf den Entwicklungsprozess selbst. Für die Programmierung der UDC-Komponente kommt Java zum Einsatz. Die Entwicklung findet lokal statt und unterscheidet sich praktisch nicht von der bekannten Vorgehensweise. Als Arbeitsumgebung kommt vorzugsweise Eclipse [10] zum Einsatz, da der UDC auf dieser Plattform aufbaut.

Für die salesforce.com-Komponenten lohnt sich ein genauerer Blick auf die verfügbaren Werkzeuge. Bei der Programmiersprache Apex, in der die Logik der Geschäftsprozesse abgebildet wird, handelt es sich um eine objektorientierte Interpreter-Sprache, deren Syntax relativ nahe an jener von Java oder C# anzusiedeln ist. Eine VisualForce Page, die zum festlegen der Benutzeroberfläche eingesetzt wird, ist ein XML-basiertes Markup, in dem die speziellen Tags für die serverseitige Verarbeitung durch standardkonformes XHTML angereichert werden. Bei einer Anfrage der Seite wird aus diesem XML ein dynamisch generiertes XHTML-Dokument, wobei der Preprozessor mit der zur Seite gehörenden Apex-Klasse interagieren kann. Vom Konzept her handelt es sich also analog zu Java Server Faces um ein Framework, das nach der Model-View-Controller-Architektur verfährt. Die Aufgabe des Datenmodells übernimmt dabei die dem CRM-System salesforce.com zugrunde liegende Datenbank (Abb. 4).

Während Anpassungen beziehungsweise Erweiterungen des Datenmodells und die Verwaltung der Zugriffsrechte sich auch über die Setup-Funktion der salesforce.com-Weboberfläche komfortabel durchführen lassen, eignet sich sowohl für Apex-Klassen als auch für VisualForce Pages die force.com IDE besonders. Mit diesem Eclipse-Plug-in hat man direkten Zugriff auf die Metadaten und Ressourcen der Plattform und kann Quelltexte in einem vollwertigen Editor bearbeiten. Unabhängig davon, ob Änderungen über die Weboberfläche oder die force.com IDE eingepflegt werden, werden die Datenstrukturen in den Metadaten der Datenbank

```

1 <apex:page controller="MobileController" standardStylesheets="false" sidebar="false" showHeader="false">
2 <html>
3   <head>
4     <title>Mobile Client</title>
5
6     <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
7     <meta name="layout" content="webkit" />
8     <meta name="viewport" content="width=device-width, initial-scale=1.0, m
9
10    <!-- mobile style sheets -->
11    <apex:outputText value="{!styleLinkTags}" escape="false" />
12    <style type="text/css">
13      .sfdcLink { color:black; }
14    </style>
15
16    <!-- JavaScript libraries -->
17    <script type="text/javascript">
18      var djConfig = {
19        isDebug: false,
20        parseOnLoad: true,
21        locale: '{!userLocale}',
22        extroLocale: ['en-us']
23      };
24      if (djConfig.isDebug) {
25        djConfig.baseUrl = "https://ajax.googleapis.com/ajax/libs/dojo/
  
```

```

166 public ID currentTaskId {get;set;}
167 public String currentTaskAction {get;set;}
168 public String geolocTimestamp {get;set;}
169 public Double geolocLatitude {get;set;}
170 public Double geolocLongitude {get;set;}
171 public Double geolocAccuracy {get;set;}
172 public String geolocAddress {get;set;}
173
174
175 public MobileController() { }
176
177
178 public String getUserLocale() {
179   if (userLocale == null) {
180     userLocale = UserInfo.getLanguage().toLowerCase().replace('_', '-');
181   }
182   return userLocale;
183 }
184
185
186 private Configuration__c getConfig() {
187   if (config == null) {
  
```

Abb. 4: Syntaxbeispiel einer VisualForce Page mit der zugehörigen Apex-Klasse

abgelegt, und für die Quelltexte finden sich spezielle Tabellen in derselben Datenbank. Wegen dieses auf die Datenbank zentrierten Ansatzes, unterscheidet sich der Entwicklungsprozess im Cloud-Umfeld von salesforce.com in einigen wesentlichen Punkten vom herkömmlichen Ansatz. Denn obwohl die force.com IDE lokal auf dem Entwicklungsrechner installiert ist, besteht keinerlei Zugriff auf einen lokalen Compiler oder Debugger, um den entwickelten Code direkt testen zu können. Vielmehr erfordert jede noch so kleine Änderung am Code ein Deployment auf die Plattform, wo dann der Apex Interpreter zwecks Validierung des hochgeladenen Codes seine Arbeit verrichtet. Ferner verlangt eine salesforce.com Policy, dass der überwiegende Teil des Codes durch Unit Tests abzudecken ist, die in Form von *testMethods* direkt in den Code eingebunden werden. Bei umfangreichen Projekten entsteht daher beim Deployment von Code eine merkliche Wartezeit, die unter anderem auf die erzwungene Ausführung der Unit Tests zurückzuführen ist. Auf einen Debugger muss man als Entwickler derzeit sogar noch gänzlich verzichten. Zur Fehlersuche stehen ausschließlich textuelle Ausgaben im Debug-Log zur Verfügung, dessen maximale Größe durch salesforce.com beschränkt ist. In Anbetracht des enormen Nutzens für den Geschäftsprozess ist man jedoch bereit, diese Einschränkungen hinzunehmen, und wartet darauf, dass mit einem der nächsten Releases das ein oder andere vermisste Feature nachgerüstet wird.

FAZIT

Seit über einer Dekade haben sich die Konzepte von Software as a Service, Platform as a Service und Cloud Computing von den ersten Anfängen bis hin zu den heutigen Lösungen entwickelt. Im Artikel haben wir gezeigt, wie Cloud-Konzepte im realen Alltag einer

Firma genutzt werden können. Dabei skaliert die Anwendung vom Klein- und Mittelständler bis hin zu großen Firmen. Das das möglich ist, hat nicht zuletzt den Grund, dass viele vorgefertigte Konzepte und auch Funktionalitäten aus der Plattform genutzt werden können.



Georg Spengler arbeitet als Consultant für die arlanis Software AG und ist seit mehreren Jahren spezialisiert auf die Beratung und Umsetzung von Datenintegrationen im As-a-Service-Umfeld mit besonderem Fokus auf Aspekte der Datensicherheit.



Andreas Holubek arbeitet als VP Engineering für die arlanis Software AG und blickt auf eine langjährige Arbeit in Konzeption, Design und Programmierung von Software zurück. Daneben beschäftigt er sich mit der Einführung und Umsetzung von modernen Technologien in der Praxis.

Links & Literatur

- [1] Metzger, C.; Reitz, T.; Villar, J.: Cloud Computing, Hanser Verlag, ISBN 978-3-446-42454-8
- [2] Microsoft Azure Plattform: <http://www.azure.com>
- [3] Amazon Web Services (AWS): <http://aws.amazon.com>
- [4] Salesforce.com Force.com Plattform: <http://www.force.com>
- [5] Salesforce.com CRM: <http://www.salesforce.com>
- [6] Universal Data Converter/Integration Plattform: <http://www.arlanis.de>
- [7] Heuser, O.; Holubek, A.: Java Web Services in der Praxis, dpunkt.verlag 2010, ISBN 978-3-89864-596-6
- [8] PhoneGap: <http://www.phonegap.com>
- [9] Google Maps JavaScript API V3: <http://code.google.com/intl/de-DE/apis/maps/documentation/javascript/>
- [10] Eclipse: <http://www.eclipse.org>

arlanis 

arlanis Software AG

Friedrich-Ebert-Straße 51

D-14469 Potsdam

T: +49-331-23189950

F: +49-331-23189979

www.arlanis.com