Dank Alexa auf Bildschirm, Maus und Tastatur verzichten

Franziska Höcker
DB Systel
Berlin

Schlüsselwörter
Amazon Echo, Oracle Apex, Sprachsteuerung, Node.js, AWS Lambda

Einleitung
Aktuell groß im Trend sind sprachgesteuerte Geräte für Zuhause. Damit können Sie nicht nur Ihre Smart Home-Geräte oder Ihre Musik steuern, sondern auch Ihre Oracle Apex Anwendungen. So stellen Sie dem Endanwender ein völlig neues User Interface zur Verfügung bei dem man auf Bildschirm, Maus und Tastatur gänzlich verzichten kann. Aber wie soll das in der Praxis aussehen?

In einer Live-Demo wird eine exemplarische Oracle Apex Anwendung gezeigt, die über alle typischen Features verfügt, wie Einfügen eines Datensatzes, Auslesen eines und mehrerer Datensätze. All das wird über die Sprachsteuerung durch Alexa mit einem Echo Dot Gerät von Amazon durchgeführt.

Neben der Demonstration wird aufgezeigt wie die Entwicklung einer solchen Anwendung und die der nötigen Schnittstellen durchgeführt werden kann. Hierzu gehört nicht nur die Entwicklung eines RESTful Webservices in Apex, sondern auch die Anwendung einer sog. Lambda-Funktion (JavaScript) in AWS und die Erstellung der zugehörigen App für Alexa. Am Ende des Vortrages haben Sie sämtliche Grundlagen und ein paar wissenswerte Details um die Vorzüge der Sprachsteuerung auch in Ihrer Apex Anwendung in einfachen Schritten zu integrieren.

Idee und Anwendungsfälle
Die Geschichte des Computers reicht bis weit in die Antike zurück. Die ersten Ansätze waren sogenannte Rechenmaschinen (z.B. der Abakus). Der erste Digitalrechner wurde 1937 gebaut und wenig später wurde die Zuse Z1 entwickelt, der erste frei programmierbare mechanische Rechner. Seit dem entwickelte sich die Computertechnik sehr schnell weiter und schon 1983 gab es den ersten Computer mit grafischer Benutzeroberfläche, Maus und Tastatur (Apple Computer Lisa). Seit dem sind Computer nicht mehr wegzudenken aus unserer heutigen Gesellschaft.

Aktuell befinden wir uns wieder an einem Wendepunkt der Technik. Der Trend geht zur Sprachsteuerung. Bisher haben wir unsere Daten für Anwendungen am Computer eingegeben, aber in Zukunft kann es auch ohne Bildschirm, Maus und Tastatur funktionieren. Die Entwicklung von Intelligenten Persönlichen Assistenten auf Smartphones war der erste Schritt in eine sprachgesteuerte Zukunft. Es gibt mittlerweile viele Sprachgesteuerte Geräte zum Steuern von Smart-Home Geräten, zum Musik hören, zum Festhalten für die neue Einkaufsliste und vieles mehr. Aber auch im Büro bzw. Arbeitsalltag können sich viele Nutzergruppen über Sprachsteuerung freuen. Hierzu zählen beispielsweise:

- Sehbehinderte Menschen
- Anwender mit schmutzigen oder vollen Händen z.B. ein Koch in der Küche oder ein Arzt im OP-Saal
- Anwender die Hilfe brauchen bzw. geführt werden müssen

Es stellt sich die Frage warum sollte man die Sprachsteuerung nicht nutzen, um simple Formulare in Oracle Apex Anwendungen und anderen Software Anwendungen zu bedienen? Damit würde man eine bisher noch nicht dagewesene Form der Bedienung erreichen. Diese Art der Bedienung würde für viele verschiedene Anwendungsfälle einen Mehrwert bringen, zum Beispiel für:

- Anwendungen zum Protokollieren (z.B. Tagebuch, Journal, Fallakte)
- Telefonbuch zum Eintragen und Abfragen von Einträgen (ggf. sogar mit direkter Verbindungsmöglichkeit zum Gesprächspartner)
- Bestell- bzw. Prozessvorgänge (z.B. in der Logistik, bei Arbeitsprozessen)

Architektur

In unserer in diesem Dokument beschriebenen Beispielanwendung bedienen wir uns einer einfachen Herangehensweise, welche auch für ähnliche Anwendungen durchaus als Vorlage genutzt werden kann. Wie diese aufgebaut ist, wird in diesem Absatz beschrieben.

Für die Sprachsteuerung wird ein Amazon Echo Dot verwendet. Die Architektur besteht aus 3 Schichten: der Präsentationsschicht, der Logikschicht und der Datenhaltungsschicht. Der Anwender, der Amazon Echo Dot und der Alexa Skill dienen der Benutzerschnittstelle. In der Logikschicht befindet sich die Lambda Funktion, welche alle Bearbeitungsmechanismen und die Anwenderlogik enthält. Der Oracle Rest Data Service und die Oracle Datenbank mit Apex enthält alle Daten und ist verantwortlich für das Speichern und Laden der Daten.

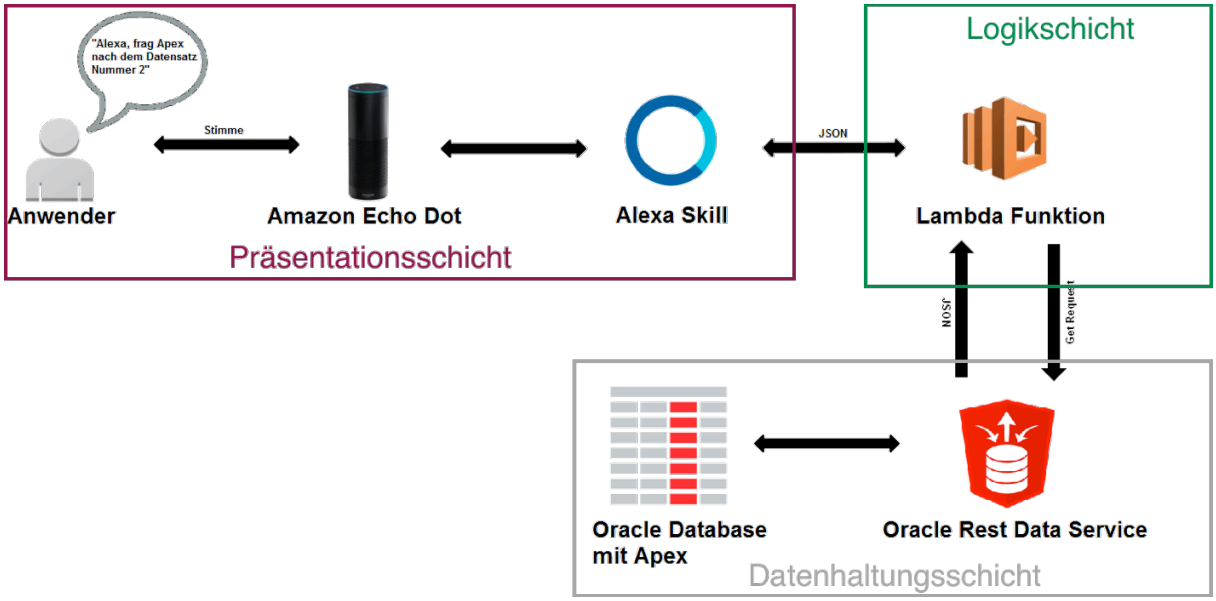


Abb. 1: Architektur Alexa mit Apex

Der **Anwender** aktiviert über seine Stimme den Amazon **Echo Dot** indem er zunächst das Wakeword, also z.B. „Alexa“ oder „Echo“ spricht und darauf folgend die Anweisung für die Anwendung wie z.B. „Frag Apex nach dem Datensatz mit der Nummer 12“. Diese Aussage wird vom Echo Dot interpretiert und in Text umgewandelt. Dabei werden sogar intelligente Algorithmen seitens Alexa bzw. Echo verwendet um den Wortlaut auch bei kleineren Abweichungen den zu erwartenen Befehlen anzupassen. So ist es beispielsweise problemlos möglich auch folgendes zu sagen mit exakt selbem Ergebnis: „Alexa, kannst Du Apex mal nach dem Datensatz mit der ähhh… Nummer 12 fragen?“. Diese Funktionalität bekommt man als Alexa Entwickler quasi geschenkt. Damit dies so gut funktioniert, ist es aber erforderlich Alexa vorher schon über die gleich folgende Alexa Skills App mitzugeben welche Befehle es gibt. Mehr dazu folgt im Interaction Model in diesem Dokument.

Die Alexa Skills App ist letztlich eine kleine benutzerdefinierte Anwendung, die man selbst anlegt und in der man letztlich grundlegende Informationen, sowie alle gewünschten Befehle der Anwendung hinterlegt. Diese Informationen gibt man an Amazon weiter, so dass diese Anwendung mit Alexa bzw. Echo verwendet werden kann. Eine essentielle Information ist hierbei auch welche API angesteuert

werden soll, wenn diese Anwendung über Alexa bzw. Echo einen Befehl erhält, nachdem der Benutzer seinen Wunsch geäußert hat. Hier gibt es grundsätzlich zwei Möglichkeiten die technisch dasselbe bedeuten. Nämlich die URL zu einem Web-Service. Und hierbei kommt nun der AWS Service Namens Lambda ins Spiel. Hiermit ist es möglich ein kleines Programm, z.B. in Node.JS geschrieben, in die Cloud zu AWS hochzuladen und die dazugehörige Web Service Url in dem Alexa Skill zu hinterlegen. Da beide Dinge von Amazon bzw. AWS angeboten werden ist hier die Integration besonders einfach und von daher auch für Anfänger zu empfehlen.

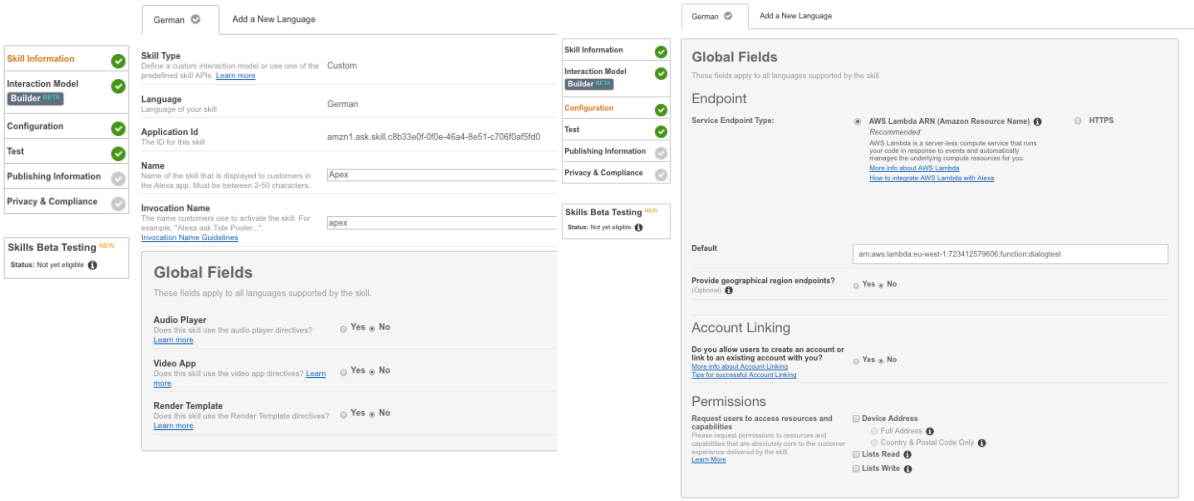


Abb. 2: Alexa Skill App: Allgemeine Einstellungen und Konfigurationen

In jedem Falle braucht man eine Anwendung um die API Aufrufe vom Alexa Skill entgegenzunehmen. Ob man dies nun mit einer separaten Anwendung z.B. mit **AWS Lambda** macht oder versucht dies direkt in Apex bzw. Oracle einzubauen bleibt dem Entwickler selbst überlassen. In unserem Modell haben wir uns aber für eine strikte Trennung dieser entschieden. Unsere Anwendung haben wir in Node.JS geschrieben und sie letztlich in Lambda hochgeladen. Unsere Anwendung macht nicht viel, beinhaltet aber dennoch eine grobe Steuerungslogik für unsere Nutzeranwendung und übersetzt die Befehle von Alexa/Echo kommend in Richtung Oracle Apex. So ist in dieser Anwendung klar erkennbar wo ein eingehender Befehl zum Lesen eines Eintrages eingeht und wie dieser verarbeitet wird, indem die Oracle Apex Anwendung dazu gefragt wird. Das Ergebnis von Oracle Apex wird in der Anwendung dann wieder als Resultat an Alexa zurückgegeben, um als Sprachausgabe wiedergegeben zu werden.

Damit dem Entwickler nicht langweilig wird, kann man natürlich in der Anwendung auch ein paar komplexere Fälle abbilden: Beispielsweise ist es auch möglich Alexa zu helfen etwaige Folgefragen an den Anwender automatisch zu klären. So ist es möglich, dass zum Abfragen eines Telefonbucheintrages immer ein Name und ein Ort erforderlich sind. Also zum Beispiel: „Alexa, gebe mit die Nummer von Tom“. Darauf könnte Alexa fragen: „In welcher Stadt lebt Tom?“. Sollte es aber im gesamten Telefonbuch nur einen Eintrag mit Tom geben, so könnten wir diese Tatsache bereits früh in unserer kleinen Anwendung feststellen und Alexa mitteilen, dass diese Folgefrage nicht mehr gestellt werden muss (oder technisch gesagt: bereits beantwortet ist). Mehr hierzu wieder im Interaction Model weiter unten.

Zu guter Letzt muss man nur noch dafür sorgen, dass die **Apex Anwendung** über eine einfach zu bedienende Webservice-API verfügt, die von unserer vorher genannten Anwendung benutzt werden kann. Das tolle an unserer Trennung der verschiedenen Komponenten/Schichten ist, dass diese API auf

Seite von Oracle Apex sehr generisch gebaut werden kann und sollte. Ob nun eine Anwendung für Alexa diese API benutzt oder ein Entwickler aus einem anderen Tool heraus spielt hierbei keine Rolle. Hierfür nutzt die Oracle Apex Anwendung die integrierten RestFul Services innerhalb von Apex.

Beispielanwendung Apex

Wie schon in den vorherigen Abschnitten angedeutet ist unsere Beispielanwendung eine einfache Anwendung zum Verwalten von Telefonnummern. Ein Datensatz in diesem Telefonbuch besteht immer aus Vorname, Stadt und Telefonnummer.

Es handelt sich um eine Standard Apex Anwendung. Sie besteht aus 3 Seiten. Die Eingabe Seite besteht aus einem Standard Formular auf Basis einer Tabelle und wurde mit dem Assistenten erstellt. Es können Daten inseriert, aktualisiert und gelöscht werden.

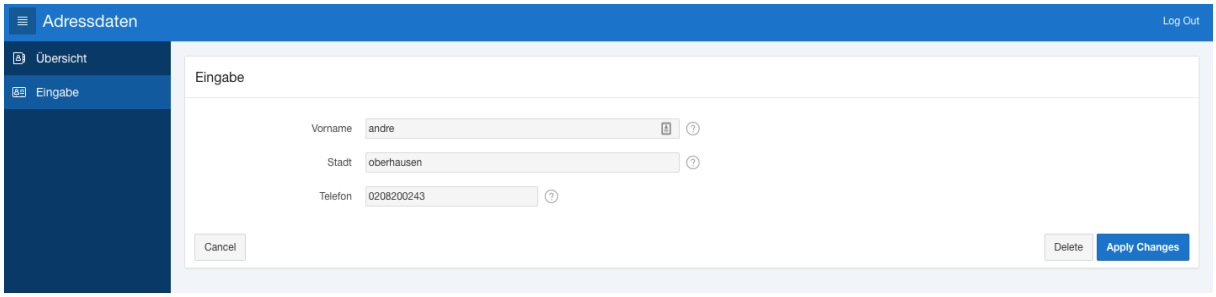


Abb. 3: Apex Anwendung: Datensatz ändern und speichern oder löschen

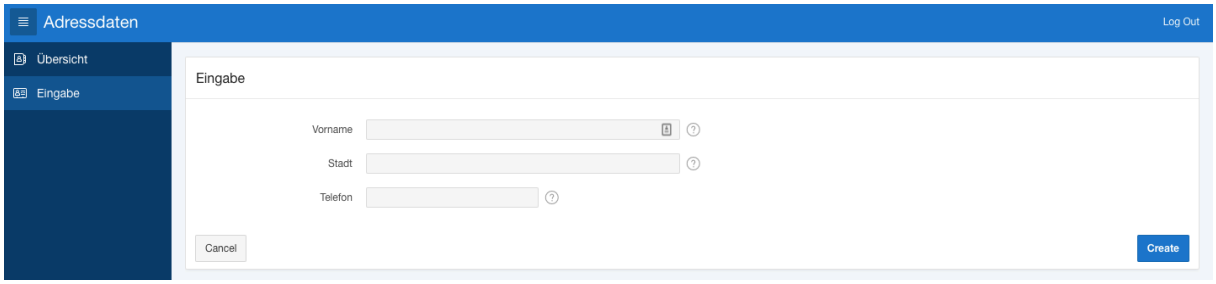


Abb. 4: Apex Anwendung: Datensatz hinzufügen

Die Übersichtsseite enthält einen Interaktiven Bericht, welcher alle Daten aus der Tabelle Adressen enthält. Über eine Link kann für jede Datenzeile das Eingabe Formular geöffnet werden, um den Datensatz zu löschen oder zu aktualisieren. Für die Neueingabe eines Datensatzes kann direkt auf Eingabe im Navigationsbereich geklickt werden.

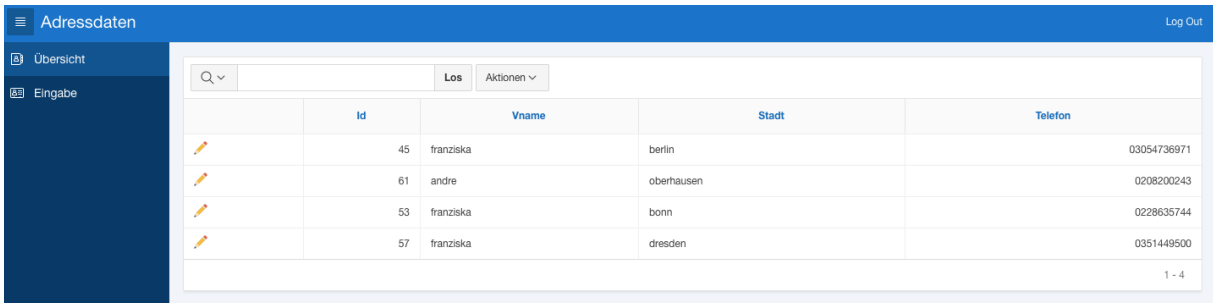


Abb. 5: Apex Anwendung: Übersichtseite mit Interaktivem Bericht

Die dritte Seite ist die Login Seite welche standardmäßig durch den Assistenten beim Erstellen der Anwendung angelegt wird. Es ist momentan kein Login mit Passwort notwendig (openDoor).

Das Datenmodell zu der Anwendung besteht nur aus einer Tabelle, der Tabelle Adressen. Diese Tabelle enthält alle Felder welche über das Apex Formular bearbeitet werden können. Es existieren noch zwei Eindeutige Indexe. Die Spalte Telefon muss eindeutig sein und es dürfen die Kombination aus Vname und Stadt jeweils nur einmal vorkommen.

ADRESSEN

Table | Data | Indexes | Model | Constraints | Grants | Statistics | UI Defaults | Triggers | Dependencies | SQL

Add Column | Modify Column | Rename Column | Drop Column | Rename | Copy | Drop | Truncate | Create Lookup Table

Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER	No	-	1
VNAME	VARCHAR2(100)	No	-	-
STADT	VARCHAR2(100)	No	-	-
TELEFON	VARCHAR2(50)	No	-	-

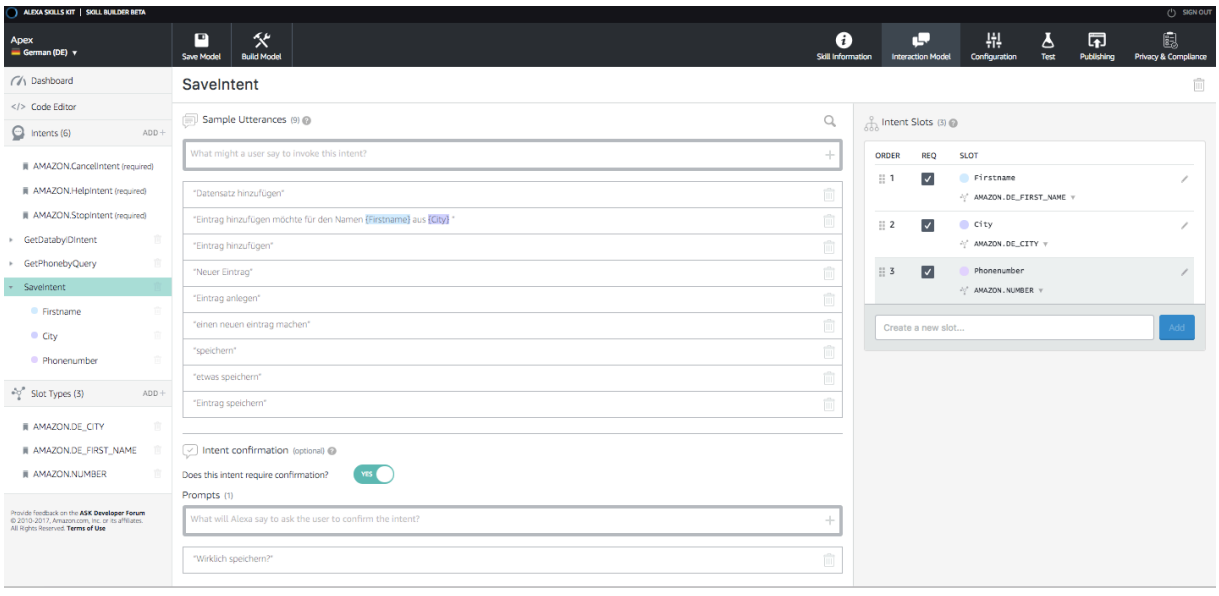
Abb. 6: Datenmodell der Apex Anwendung: Tabelle Adressen

Interaction Model

Was ist überhaupt ein **Interaction Model**?
Letztlich ist das Interaction Model so etwas wie eine große Konfigurationsdatei in JSON. Daraus ergibt sich welche Befehle Alexa für unseren Skill entgegennehmen kann, welche Rückfragen sich ergeben, welche Antworten zu erwarten sind vom Benutzer und sogar abweichende Fragestellungen zur gleichen Frage. Alexa benötigt dieses Interaction Model um zu wissen auf welche Fragestellungen oder Befehle des Benutzers sie reagieren muss.

Innerhalb eines Interaction Models können verschiedene **Intents** definiert werden. Ein Intent steht für eine Aktion die ausgeführt wird für eine bestimmte Anfrage des Benutzers. In unserem Fall gibt es zum Beispiel den „SaveIntent“, dieser ist in unserer Anwendung dafür da einen Datensatz hinzuzufügen. Er kann aktiviert werden durch bestimmte Wörter bzw. ganze Sätze die als sog. „**Sample Utterances**“ definiert wurden. Der Benutzer kann zum Beispiel sagen „Alexa sag Apex ich möchte einen Datensatz hinzufügen“. Letztlich also das Wakeword „Alexa“, gefolgt von dem Applikationsnamen „Apex“ und dem vordefinierten Satz „ich möchte einen Datensatz hinzufügen“ für den jeweiligen Intent. Hierbei ist es auch möglich mehrere Formulierungen bzw. „Sample Utterances“ für den selben Intent zu hinterlegen. Beispielsweise wäre es sinnvoll noch folgende hinzuzufügen zu unserem SaveIntent: „ich möchte einen Datensatz speichern“, „ich möchte einen Eintrag speichern“ usw.

Abb. 7 Interaction Model: SaveIntent



Es gibt auch eingebaute Intents für das Alexa Skills Kit. Diese benötigen keine „Sample Utterances“, es können aber welche zusätzlich definiert werden.

Für jeden Intent bzw. dessen „Sample Utterances“ können sogenannte **Slots** festgelegt werden. Slots sind sozusagen Parameter die einem Intent bzw. dessen Satz mitgegeben werden können. Beispielsweise könnte man für „Vornamen“ einen Slot definieren. So könnte man Alexa sagen: „Schaue nach ‚Hans‘ in meinem Telefonbuch“.

Für jeden Slot muss auch ein **Slot Type** definiert werden. Dieser definiert um welche Art von Slot es sich handelt. Diese können manuell angelegt werden oder aus bestehenden in Alexa ausgewählt werden. In diesem Fall wurde für den Slot Firstname eine bereits existierende Liste an deutschen Vornamen ausgewählt. Natürlich können diese auch jederzeit erweitert werden.

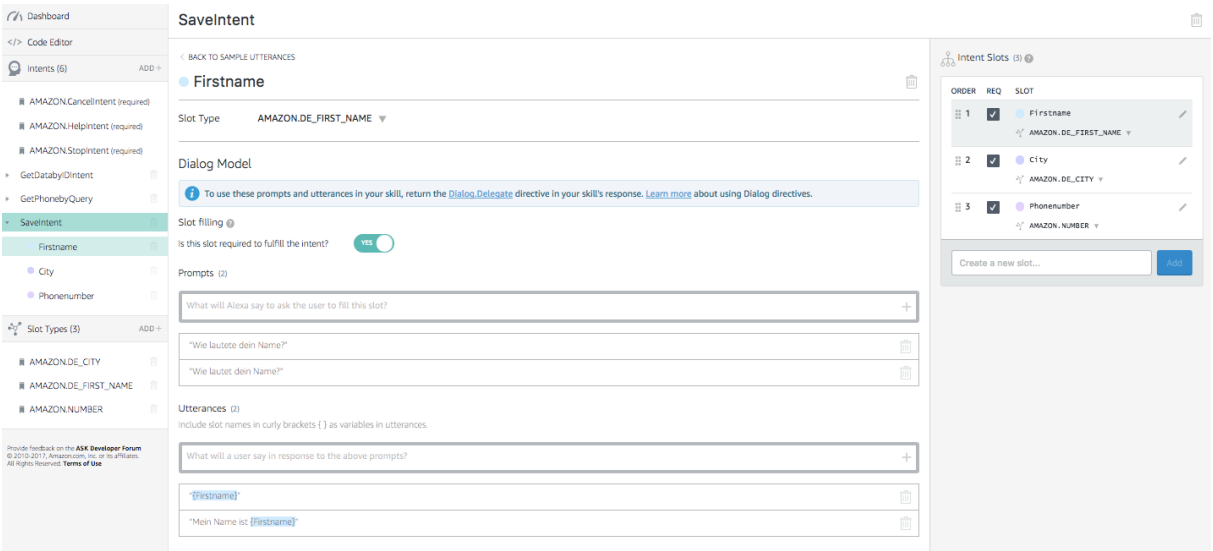
Es ist auch möglich mehrere Slots zu definieren um auch Sätze wie „Schaue nach ‚Hans‘ aus ‚Köln‘ in meinem Telefonbuch“ abbilden zu können. Zusätzlich lassen sich für den selben Intent auch Sätze ohne die Slots definieren und für jeden einzelnen Slot nochmal sogenannte **Prompts**. Also Fragen die Alexa dem Benutzer stellt wenn diese Slots noch nicht mitgegeben wurden. So könnte der Benutzer also sagen: „Schaue in mein Telefonbuch“. Daraufhin würde Alexa probieren die beiden Slots zu füllen und nachfragen: „Welcher Name?“ und danach „Welcher Stadt?“. In diesen Prompt Texten kann aber auch auf andere Slots zugegriffen werden. So kann für einen weiteren Slot der erforderlich ist die Frage entstehen: „Aus welcher Stadt kommt ‚Hans‘“.

Auch hier kann man sich als Entwickler austoben und in diese Frage-/Antwortspiel mit seiner eigenen Anwendung eingreifen. Alexa meldet jeden Schritt der Anwendung und diese kann sich auch beispielsweise dafür entscheiden einen Slot bzw. die Nachfrage bspw. nach dem Ort selbständig zu beantworten.

Für ganze Intents können auch Prompts angelegt werden. Diese sind dafür da, dass der Benutzer nochmal gefragt wird ob man den Intent schlussendlich wirklich ausführen möchte. Solche Nachfragen werden ausgeführt, nachdem alle Fragen zu den einzelnen Slots vom Benutzer beantwortet wurden.

In unserem Fall haben wir für den SaveIntent einen Slot der Firstname heißt. Dieser soll für die Aufnahme von Vornamen sein. Alexa kann den Benutzer nach Aktivierung des Intents also fragen „Wie lautet dein Vorname“.

Abb. 8 Interaction Model: SaveIntent Slot Firstname



In unserer Beispielanwendung gibt es noch zwei weitere Intents:

GetPhoneByQuery:
Sample Utterances: „Telefonnummer von ‚Hans'", „Suche die Telefonnummer von ‚Hans'", usw.

GetDataById:
Sample Utterances: „Eintrag mit der Id ‚12'", „Datensatz mit der Id ‚12'", usw.

SaveIntent: wie oben beschrieben

Lambda

Wie bereits erwähnt ist der einfachste Weg um einen Cloud basierten Service für einen Alexa Skill zu bauen, die Verwendung von **AWS Lambda**, da man somit nicht überprüfen muss, ob Anfragen vom Alexa-Service selbst kommen und die Verbindung zwischen dem Alexa Skill und der Lambda im Hintergrund von AWS verläuft. Der Zugriff auf die Ausführung unserer Funktion wird so durch Berechtigungen innerhalb von AWS gesteuert.

Es handelt sich bei AWS Lambda um ein Angebot von **Amazon Web Services**, das den Code nur bei Anforderung ausführt und automatisch skaliert. Dadurch entfällt die Notwendigkeit einen Server bereitzustellen, also ist die Verwendung von AWS Lambda somit „Serverless". Auch die Kosten für den Lambda Service sind absolut übersichtlich, denn es wird nur für die Rechenzeit bezahlt, die auch verbraucht wird. Wenn der Code nicht ausgeführt wird entstehen auch keinerlei Kosten. Es gibt ein kostenloses Kontingent für Lambda, welches 1 Million Anforderungen pro Monat umfasst. Erst wenn dieses Kontingent überschritten wird, kostet der Service pro Anforderung gemäß der Preistabelle von AWS.

Um einen Lamdba Service für unseren Alexa Skill zu nutzen, muss eine **Lambda Funktion** erstellt werden. Der Code der in AWS Lamdba später ausgeführt werden soll, wird als Lamdba Funktion hochgeladen. Der Code kann in Node.js, Python, Java oder C# geschrieben sein und es können auch noch eigene Bibliotheken dazu hochgeladen werden.

In unserer Beispielanwendung ist die Lambda Funktion bzw. unsere Anwendung auf der Logikschicht die Schnittstelle zwischen Apex, dem Interaction Model und damit Alexa. Es reagiert auf die verschiedenen Intents und Anfragen von Alexa, nimmt die dazugehörigen Slots an und formt diese Anfragen um in API-Calls in Richtung Apex.

Den Source-Code unserer Beispielanwendung habe ich in einem öffentlichen Repository auf GitHub zur Verfügung gestellt. Dort sind auch weitere Informationen zum Aufbau und den verwendeten Libraries zu finden.

Das Repository findet man hier: https://github.com/FranziskaHoecker/doag17-alexa-demo

Webservices Apex

In Apex gibt es die Möglichkeit RESTful Services einzurichten. Dafür muss als erstes ein RESTful Modul erstellt werden. Für dieses Modul können dann die einzelnen URI Templates mit Ihren Handlern definiert werden. Über den Namen des URI Templates kann auch ein benötigter Parameter mitgegeben werden. Außerdem kann so auch relativ schnell gesehen werden welche Anforderungen es gibt.
Für den Handler eines URI Templates kann dann die Methode gewählt werden, in unserem Fall wurden nur GET und POST verwendet. Außerdem kann festgelegt werden in welchem Format die Ausgabe sein soll und wie die Daten verarbeitet bzw. geholt werden sollen, also z.B. per PL/SQL oder eine einfache Query etc.

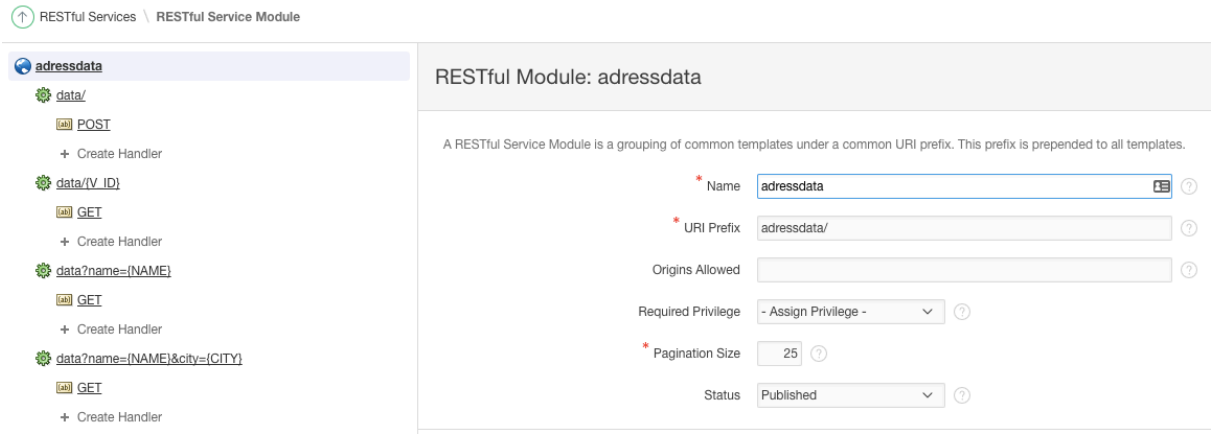


Abb. 9 Webservices Apex: Restful Module adressdata mit den verschiedenen Routen

Es gibt insgesamt 4 Routen, die wie folgt funktionieren:

Zum Anlegen eines Eintrages in der Datenbank.

```
POST /data/
{
    "vname"   : "Hans",
    "stadt"   : "Bonn",
    "telefon" : "03055533990"
}
```

Zum Abfragen eines Eintrages anhand der ID, es wird immer nur 1 Eintrag zurückkommen, daher wurde auch als Source Type die Query One Row ausgewählt.

```
GET /data/{ID}
{
    "id"      : "53",
    "vname"   : "Hans",
    "stadt"   : "Bonn",
    "telefon" : "03055533990"
}
```

Zum Abfragen nach einer Telefonnummer anhand eines Namens und einer Stadt, es wird immer nur 1 Eintrag zurückkommen, daher wurde auch als Source Type die Query One Row ausgewählt.

```
GET /data?name={NAME}&city={CITY}
{
    "id"      : "53",
    "vname"   : "Hans",
    "stadt"   : "Bonn",
    "telefon" : "03055533990"
}
```

Zum Abfragen nach einer Telefonnummer anhand eines Namens, es können hier auch mehrere Einträge existieren. Daher wurde als Source Type die Query gewählt.

```
GET /data?name={NAME}
items:
[
      {
            "id"      : "53",
            "vname"   : "Hans",
            "stadt"   : "Bonn",
            "telefon" : "03055533990"
      },
      {
            "id"      : "54",
            "vname"   : "Hans",
            "stadt"   : "Hamburg",
            "telefon" : "04066633880"
      },
]
```

Bei einfachen Get Prozessen wird kein Fehlerhandling benötigt, da bei nicht vorhandenen Daten ein 404 HTTP Fehlercode zurückgeliefert wird. Dieser wird dann innerhalb der Lambda Funktion abgefangen und entsprechend ausgewertet.
Für unseren SaveIntent, zum Abspeichern eines Datensatzes, kann es schon vorkommen das bestimmte Fehler abgefangen werden müssen, da der Webservice sonst die Standard Oracle Fehlermeldung mitschickt, welche nicht besonders Anwenderfreundlich ist. Wir haben zB. zwei Eindeutige Indexe auf der Tabelle Adressen, welche innerhalb des PL/SQL Codes abgefangen werden und dann einen definierten Fehlertext zurückliefern. Der Webservice antwortet in diesem Fall mit einem 500 HTTP Fehlercode und dem definierten Fehltertext. Dieser wird dann innerhalb der Lambda Funktion für Alexa weiterverwendet.

Tipps & Tricks

Zum Bearbeiten des Lambda Codes empfehle ich unbedingt einen lokalen Editor für Javascript zu haben, da das Editieren direkt in AWS Lambda sehr unvorteilhaft ist. Ein freier Editor für Javascript ist beispielsweise Atom.

Damit man zum Speichern nicht die ganze Zeit den Code per Copy und Paste aus bzw. nach AWS Lambda bringen/holen muss, macht es Sinn sich einem Hilfsframework wie z.B. Claudia.JS zu bedienen. Dieses kann über eine Konfiguration direkt den Lambda Code nach AWS hochladen. Außerdem können mit Claudia auch noch eigene Bibliotheken hochgeladen werden (zB. Request Library und Alexa-Sdk).

Ausblick

Um schnell Änderungen am Datenmodell durchzuführen und diese dann über den Alexa Skill nutzen zu können, könnte man einen Builder für das Interaction Model bauen. Für diesen könnte man die Fragestellungen nach den Slots aus den Hilfe-Texten für ein Item aus Apex nehmen.
Diese würde man über einen Webservice abfragen. Damit wären jederzeit Änderungen der Spalten auf Tabellenebene möglich.

Es könnten noch weitere Intents eingebaut werden für die restlichen Standard Funktionen in Apex (Aktualisieren und Löschen). Sicherlich gibt es auch noch mehr Abfragen die man als Intent anlegen könnte.

Grundsätzlich ist aufgefallen das bei Nutzung vom kostenlosen Oracle Apex in der Cloud der Apex Webservice manchmal sehr lange braucht bis eine Antwort kommt, daher wäre es sinnvoll den Timeout hoch zu setzen für Webservice Anfragen.

Kontaktadresse:
Franziska Höcker
DB Systel
Caroline-Michaelis-Straße, 5-11
D-10115 Berlin

Telefon: +49 0163-6327934
E-Mail f.hoecker87@web.de
Internet: https://www.dbsystel.de