



Natürliche Benutzerschnittstellen mit Amazon Echo

Ein Praxisbericht

Heidelberg, 05.05.2017
Moritz Kammerer

Über den Autor

- 2016 Bachelor-Thesis über NUIs
 - 2016 Entwicklung eines Kundenprojektes mit Amazon Echo
 - 2017 Diverse Skills im Alexa Skill Store publiziert
-
- GitHub: @phxql / Twitter: @phxql
 - Blog: <https://www.mkammerer.de/blog>
-
- Sourcecode: <https://github.com/qaware/building-iot-2017>



Was ist eine Amazon Echo?

- Digitaler Assistent von Amazon
- Bietet eine Audioschnittstelle
- Werbe-Video: <https://www.youtube.com/watch?v=KkOCeAtKHlc>
- Genau genommen:
 - Echo ist die Hardware
 - Alexa ist die “Künstliche Intelligenz”

Was ist das Tolle daran?

- Alexa hat eingebaute Fähigkeiten wie Wetter, Fakten, Nachrichten, Musik...
- Und ist auch durch sogenannte Skills erweiterbar
- In diesem Talk: Entwicklung eines Skills für ein Lagerverwaltungssystem

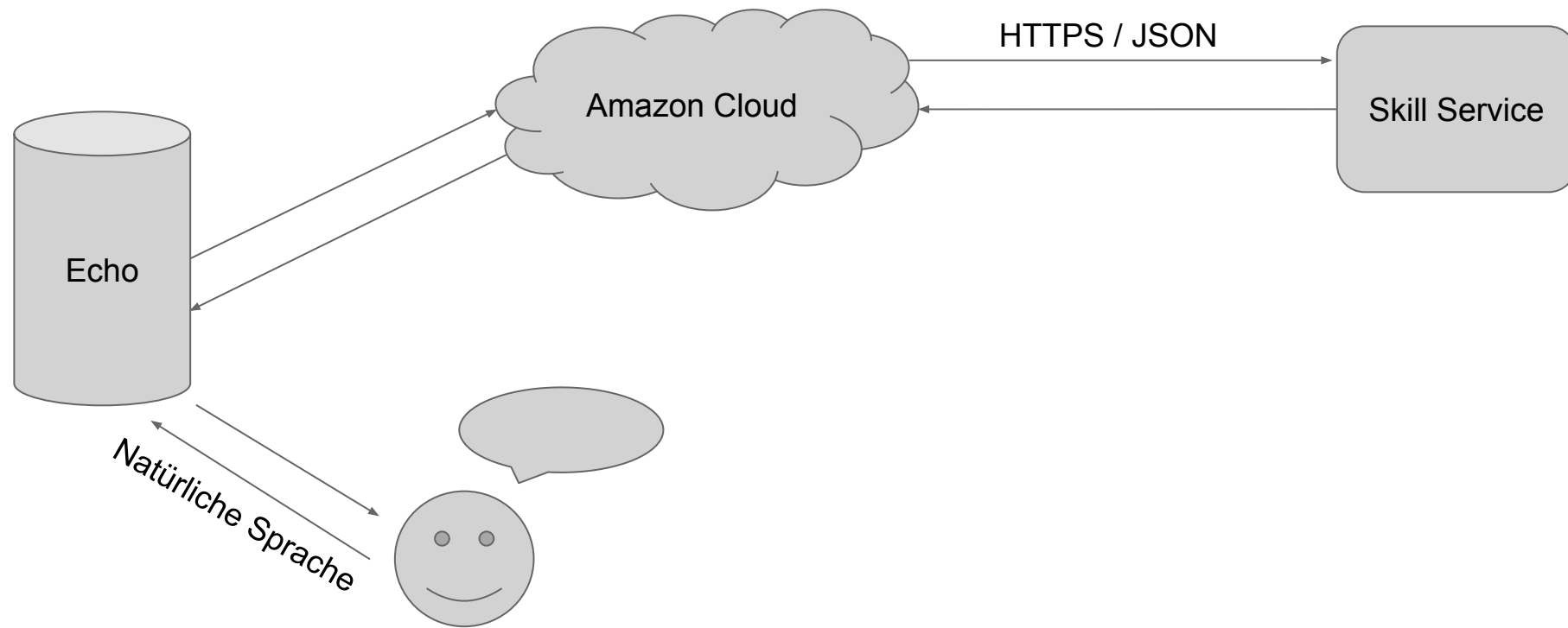
Demo

- “Wie viele Schrauben haben wir noch?”
- “Bestelle mir neue Schrauben”
- “In welchem Regal befinden sich die Schrauben?”

Wie funktioniert das?

- Skill im Amazon Skill Store registrieren
- Anwendung mit einer HTTP-Schnittstelle entwickeln
- Skill in der Echo aktivieren

Okay, aber wie funktioniert das wirklich?



Entwicklung eines Skills, Schritt 1

- Amazon Developer Console / Alexa / Alexa Skill Kit [7] / Add a new skill

Skill Type
Define a custom interaction model or use one of the predefined skill APIs. [Learn more](#)

☒ Custom Interaction Model
☐ Smart Home Skill API
☐ Flash Briefing Skill API

Language
Language of your skill

English (U.S.) ▼

Name
Name of the skill that is displayed to customers in the Alexa app. Must be between 2-50 characters.

Invocation Name
The name customers use to activate the skill. For example, "Alexa ask Tide Pooler...".
[Invocation Name Guidelines](#)

Global Fields
These fields apply to all languages supported by the skill.

Audio Player
Does this skill use the audio player directives?
[Learn more.](#)

☐ Yes ☒ No

Skill-Typen

- SmartHome Skill API: An- und Ausschalten von Lampen, Heizungssteuerung, ...
- Flash Briefing Skill API: Nachrichten als Audio- oder Text-Feed
- Custom Interaction Model: Eigene Utterances, am flexibelsten
 - Das werden wir verwenden

Entwicklung eines Skills, Schritt 2

German

Add New Language

Intent Schema
The schema of user intents in JSON format. For more information, see [Intent Schema](#).
Also see [built-in slots](#) and [built-in intents](#).

```
1 {  
2   "intents": [  
3     {  
4       "intent": "QueryInventory",  
5       "slots": [  
6         {  
7           "name": "ware",  
8           "type": "LIST_OF_WARES"  
9         }  
10      ]  
11    },  
12  ]  
13 }
```

Custom Slot Types (Optional)
Custom slot types to be referenced by the Intent Schema and Sample Utterances.
For general information about custom slots, see [Custom Slot Types](#).
Example: TOPPINGS - cheese | onions | ham (note: newlines displayed as | for brevity)

Add Slot Type

Type	Values
LIST_OF_WARES	Schrauben Winkel

Edit

Sample Utterances
These are what people say to interact with your skill. Type or paste in all the ways that people can invoke the intents. [Learn more](#)
Up to 3 of these will be used as Example Phrases, which are hints to users.

```
1 QueryInventory Wie viele {ware} haben wir noch  
2 OrderWare Bestelle mir neue {ware}  
3 LocateWare In welchem Regal befinden sich die {ware}  
4 LocateWare Wo sind die {ware}  
5 Quit Beenden  
6 Quit Abbrechen  
7 Quit Nein  
8
```

Was sind Intents?

- “an intent represents an action that fulfills a user’s spoken request”
- “Intent schema” ist eine JSON formatierte Liste von Intents

```
{  
  "intent": "QueryInventory",  
  "slots": [ ← Slots (Parameter) des Intents  
    {  
      "name": "ware",  
      "type": "LIST_OF_WARES" ← Typ des Slot  
    }  
  ]  
}
```

Intents unseres Skills

- QueryInventory (ware)
 - Wie viele Stücke der Ware sind noch im Lager?
- OrderWare (ware)
 - Bestellt neue Waren
- LocateWare (ware)
 - Findet eine Ware im Lager
- AMAZON.CancelIntent, AMAZON.StopIntent
 - Bricht den Dialog ab

Typen der Intent-Slots

Custom Slot Types (Optional)

Custom slot types to be referenced by the Intent Schema and Sample Utterances

For general information about custom slots, see [Custom Slot Types](#).

Example: TOPPINGS - cheese | onions | ham (note: newlines displayed as | for brevity)

Add Slot Type

Editing slot type

Enter Type

LIST_OF_WARES

Enter Values

Values must be line-separated

- 1 Schrauben
- 2 Winkel

Delete

Cancel

Save

Tipp: Es gibt vordefinierte Slot-Typen für Datum, Dauer etc.: [1]

Achtung: Alexa versucht, die gesprochenen Wörter auf die Liste zu matchen. Es werden aber auch andere Wörter an den Skill gesendet!

Utterances - Kombiniert die Intents mit Sätzen

Diagram illustrating the structure of utterances, combining intents and user sentences.

Name des Slots (points to the slot name {ware})

Name des Intents (points to the intent name QueryInventory)

Satz vom Benutzer (points to the user sentence)

Sample Utterances
These are what people say to interact with your skill. Type or paste in all the ways that people can invoke the intents. [Learn more](#)
Up to 3 of these will be used as Example Phrases, which are hints to users.

Intent	Utterance
QueryInventory	Wie viele {ware} haben wir noch
OrderWare	Bestelle mir neue {ware}
LocateWare	In welchem Regal befinden sich die {ware}
LocateWare	Wo sind die {ware}
Quit	Beenden
Quit	Abbrechen
Quit	Nein

Tipp: Best practices und ein Handbuch für das Design von utterances: [2], [3]

Konfiguration des Skills

Man kann auch AWS Lambda verwenden

HTTPS Endpunkt
unseres Skills

OAuth2, falls ein externer
Account angebunden werden
soll

German ☒ Add New Language

Global Fields

These fields apply to all languages supported by the skill.

Endpoint

Service Endpoint Type:

☐ AWS Lambda ARN (Amazon Resource Name) ⓘ ☒ HTTPS

Recommended

AWS Lambda is a server-less compute service that runs your code in response to events and automatically manages the underlying compute resources for you.

[More info about AWS Lambda](#)

[How to integrate AWS Lambda with Alexa](#)

Pick a geographical region that is closest to your target customers: ⓘ

☐ North America ☒ Europe

Europe


`https://ec2-35-157-19-115.eu-central-1.compute.ama`

Account Linking

Do you allow users to create an account or link to an existing account with you? ☐ Yes ☒ No

[Learn more](#)

Konfiguration von SSL

German 

Add New Language

Global Fields

These fields apply to all languages supported by the skill.

To protect your security and the security of end users, we require that you use a certificate while developing an Alexa skill. For more information, see [Registering and Managing Alexa Skills - About SSL Options](#).

Certificate for EU Endpoint:

Please select one of the three methods below for the web service:

- ☐ My development endpoint has a certificate from a trusted certificate authority
- ☐ My development endpoint is a sub-domain of a domain that has a wildcard certificate from a certificate authority
- ☒ I will upload a self-signed certificate in X.509 format. [Learn how to create a self signed certificate.](#)

Skill-Konfiguration abgeschlossen, Zeit für Code!

- Wir verwenden Spring Boot für unseren Skill-Service
- Spring Initializr [6] öffnen, Dependencies: nur “web”
- Alexa Skills Kit for Java [4] in die Maven POM eintragen:

```
<dependency>  
    <groupId>com.amazon.alexa</groupId>  
    <artifactId>alexa-skills-kit</artifactId>  
    <version>1.2</version>  
</dependency>
```

Das Speechlet implementieren

```
com.amazon.speech.speechlet.SpeechletV2
```

- `void onSessionStarted(...)`
 - Wird aufgerufen, wenn die Session gestartet wird
- `SpeechletResponse onLaunch(...)`
 - Wird aufgerufen, wenn der Benutzer eine Unterhaltung startet
- `SpeechletResponse onIntent(...)`
 - Wird aufgerufen, wenn der Benutzer einen Intent aufruft
- `void onSessionEnded(...)`
 - Wird aufgerufen, wenn die Session beendet wird

Unsere Speechlet-Implementierung

- onSessionStarted: benötigen wir nicht
- onLaunch:
 - Wird ausgeführt, wenn der Benutzer eine Unterhaltung möchte
 - Vermerken wir in der Session

```
requestEnvelope.getSession().setAttribute("conversation", "true");
```

 - Wenn onLaunch nicht ausgeführt wird, möchte der Benutzer keine Unterhaltung (one-shot Intent)
- onSessionEnded: benötigen wir nicht

Die Logik befindet sich in onIntent

- onIntent: Name des Intents auslesen und bearbeiten

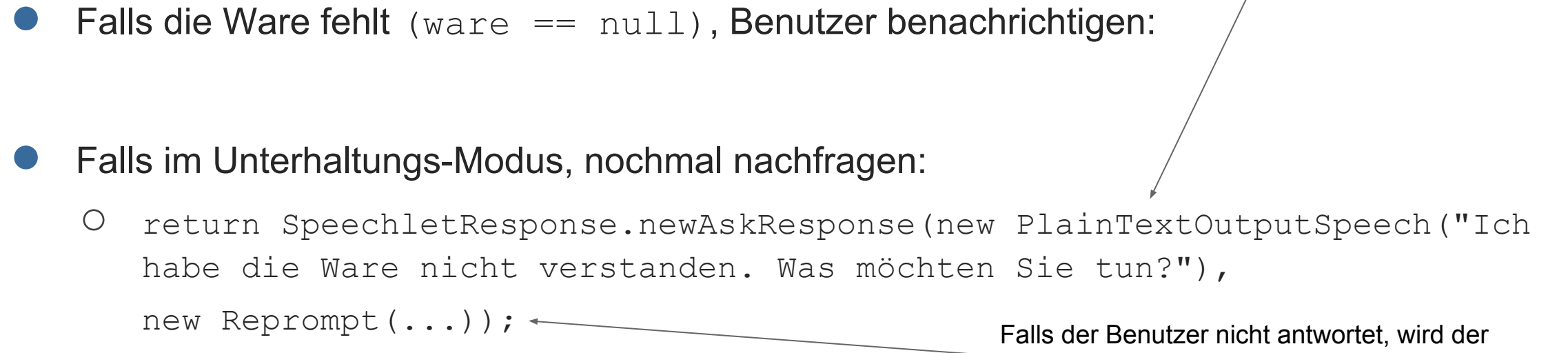
```
Intent intent = requestEnvelope.getRequest().getIntent();
switch (intent.getName()) {
    case "QueryInventory":
        return handleQueryInventory(requestEnvelope);
    ...
}
```

Behandlung des “QueryInventory” Intents

- Slot “ware” auslesen:

```
Slot wareSlot = intent.getSlot("ware");  
String ware = wareSlot == null ? null : wareSlot.getValue();
```

Behandlung des “QueryInventory” Intents

- Falls die Ware fehlt (`ware == null`), Benutzer benachrichtigen:
- Falls im Unterhaltungs-Modus, nochmal nachfragen:
 - `return SpeechletResponse.newAskResponse(new PlainTextOutputSpeech("Ich habe die Ware nicht verstanden. Was möchten Sie tun?"),
new Reprompt(...));`
- Falls nicht im Unterhaltungsmodus (one-shot intent):
 - `return SpeechletResponse.newTellResponse(new PlainTextOutputSpeech("Ich habe die Ware nicht verstanden."));`

Behandlung des “QueryInventory” Intents

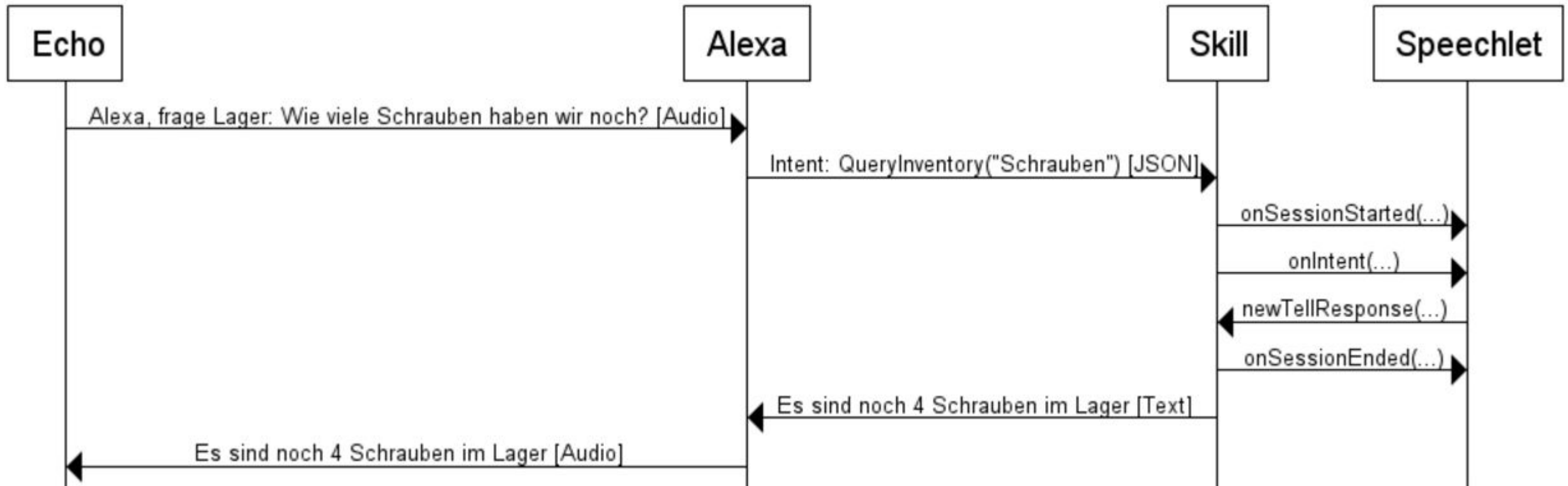
- Anzahl der Ware im Lager finden:

```
int amount = warehouseService.getAmount(ware);
```

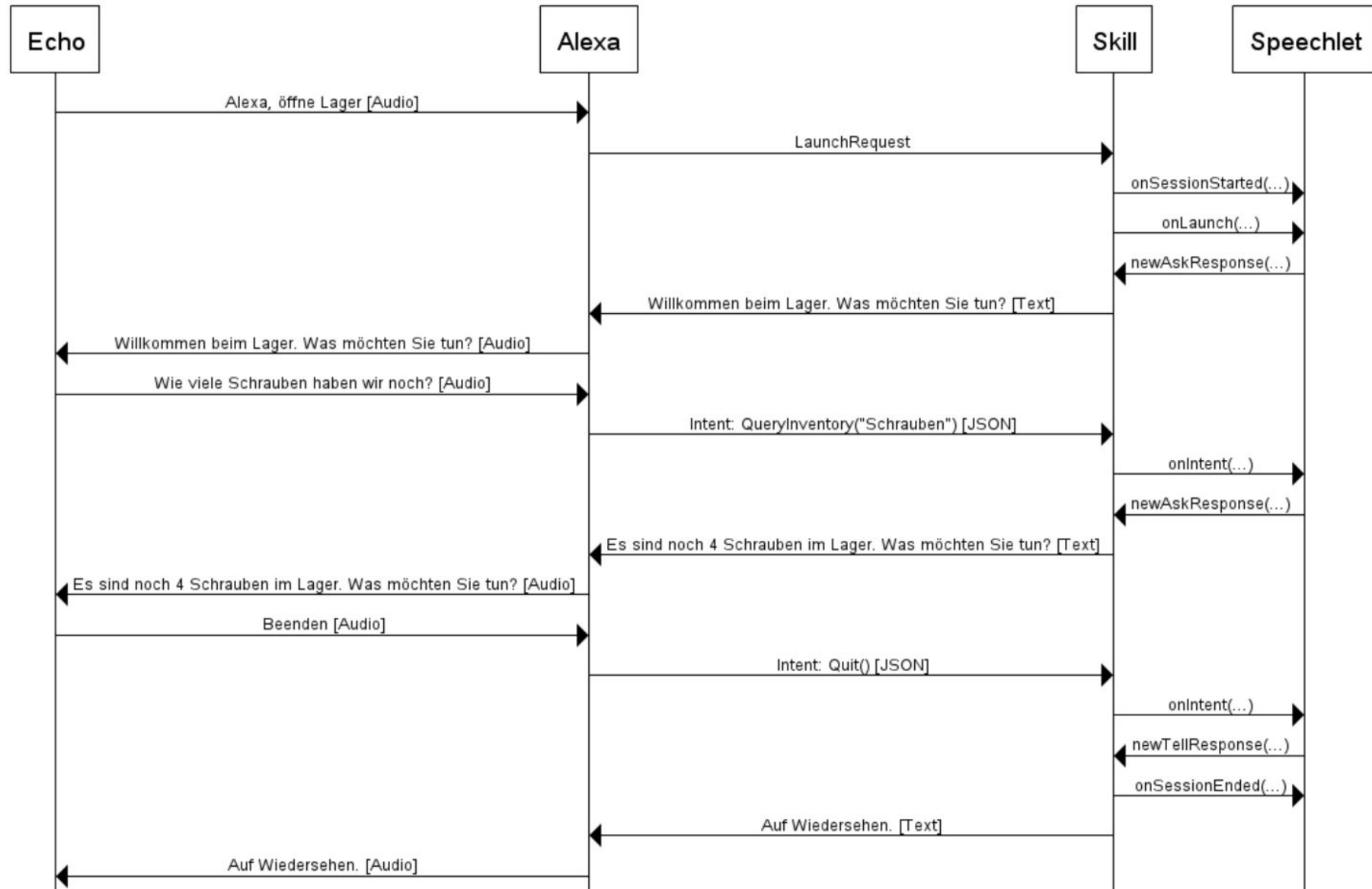
- Und dem Benutzer mitteilen:

```
return SpeechletResponse.newTellResponse(new PlainTextOutputSpeech(  
    String.format("Es sind noch %d %s im Lager.", amount, ware)  
));
```

Sequenzdiagramm des One-shot



Sequenzdiagramm der Unterhaltung




Das Speechlet in Spring Boot registrieren

@Bean

```
public ServletRegistrationBean alexaServlet(WarehouseSpeechlet speechlet) {  
    SpeechletServlet speechServlet = new SpeechletServlet();  
    speechServlet.setSpeechlet(speechlet);  
  
    ServletRegistrationBean servlet = new ServletRegistrationBean(speechServlet, "/alex");  
    servlet.setName("alex");  
  
    return servlet;  
}
```

Speechlet ist unter /alex zu erreichen



Properties für Speechlet setzen

```
// Signatur-Prüfung für Entwicklung deaktivieren
System.setProperty(Sdk.DISABLE_REQUEST_SIGNATURE_CHECK_SYSTEM_PROPERTY, "true");

// Alle Skill-IDs für Entwicklung zulassen
System.setProperty(Sdk.SUPPORTED_APPLICATION_IDS_SYSTEM_PROPERTY, "");

// Zeitstempel-Validierung für Entwicklung deaktivieren
System.setProperty(Sdk.TIMESTAMP_TOLERANCE_SYSTEM_PROPERTY, "");
```

Diese ID befindet sich auf der Skill-Konfigurations-Seite bei Amazon

Skill implementiert, wie testen?

Text hier eintragen

Dieser Request
wird zum Skill
gesendet

The screenshot displays the AWS Lambda console's testing interface for an Alexa skill. At the top, there are two tabs: 'Text' (selected) and 'JSON'. Below the tabs is a text input field labeled 'Enter Utterance' containing the text 'Wie viele Schrauben haben wir noch?'. Below this input are two buttons: 'Ask Lager' and 'Reset'. The interface is divided into two main sections: 'Service Request' and 'Service Response'. The 'Service Request' section shows a JSON payload, and the 'Service Response' section shows an error message.

Service Request

```
1 {
2   "session": {
3     "sessionId": "SessionId.81beaac3-2e82-4aa
4     "application": {
5       "applicationId": "amzn1.ask.skill.f8237
6     },
7     "attributes": {},
8     "user": {
9       "userId": "amzn1.ask.account.AEYN77EGL5
10    },
11    "new": true
12  },
13  "request": {
14    "type": "IntentRequest",
15    "requestId": "EdwRequestId.f0639f6a-5a4f-
16    "locale": "de-DE"
```

Service Response

```
1 The remote endpoint could not be called, or t
```

At the bottom right of the 'Service Response' section, there is a 'Listen' button with a play icon.

Tipp: JSON kopieren und mit Tool der Wahl (curl, Postman, etc.) an /alexa POSTen

Und jetzt mit Sprache...

- Amazon zwingt den Entwickler zu SSL
- Also... Spring Boot mit SSL konfigurieren
 - Neuen Keystore anlegen
 - Neues Keypair anlegen, CN muss dem DNS-Namen des Servers entsprechen
 - SSL in Spring Boot aktivieren:

```
server.port: 443
server.ssl.key-store: classpath:keystore.jks
server.ssl.key-store-password: ""
server.ssl.key-store-type: jks
server.ssl.key-alias: ec2-35-157-19-115.eu-central-1.compute.amazonaws.com
```

Und jetzt mit Sprache...

- Zertifikat als X.509 exportieren (-----BEGIN CERTIFICATE-----)
- X.509-Zerifikat in die Skill-Konfiguration unter “SSL Certificate” einfügen
- Applikation auf einen öffentlich erreichbaren Server (z.B. EC2) installieren
- URL zur Applikation in Skill-Konfiguration eintragen
- Applikation starten
- Echo unter dem Account registrieren, der auch den Skill erstellt hat
- Skill in der Alexa App aktivieren
- Nun kann man den Skill mit Sprache aufrufen

Ich wünschte, ich hätte das vorher gewusst...

- TLS: **Muss** auf Port 443 sein, CN **muss** dem DNS-Namen des Servers entsprechen
- Typen des Slot sind keine Enums, nur eine Empfehlung für die Spracherkennung
- Slots können nicht vorhanden sein (z.B. “Bestelle mir neue “)
- Die User-ID im Request ändert sich, wenn der Benutzer den Skill entfernt und neu hinzufügt
- Alexa Skill Kit for Java: `log4j` und `slf4j-log4j12` exkludieren
- Verwende Alexa App beim Testen des Skills: Die Cards enthalten nützliche Debug-Informationen
- Lokal entwickeln und mit Sprache testen mit SSH remote port forwarding:

```
ssh -R 443:localhost:8443 root@server
```

Was haben wir gelernt?

- Wir kennen die verschiedenen Skill-Typen
- “Interaction model” entwickeln:
 - Intents, Slots und Slot-Typen
 - Utterances
- Wie man das Alexa Skill Kit verwendet und ein Speechlet erstellt
 - Wann `onSessionStarted()`, `onLaunch()`, `onIntent()` und `onSessionEnded()` aufgerufen werden
 - Was der Unterschied zwischen `newAskResponse()` und `newTellResponse()` ist
- Wie man TLS besiegt

Fazit

- Pro:

- Alexa kann durch eigene Skills erweitert werden
- Skills sind sehr flexibel
- Das Alexa Skills Kit abstrahiert Request- und Response-Handling
- Amazon kümmert sich um Speech-To-Text und Text-To-Speech

- Contra:

- Keine semantische Analyse des gesagten, nur exaktes Matchen auf die Utterances
- Man muss “Alexa, sag [skill]...” oder ähnliche Kommando-Sprache verwenden
- TLS ist nervtötend, schlecht dokumentiert und stört beim Skill entwickeln

Source Code:

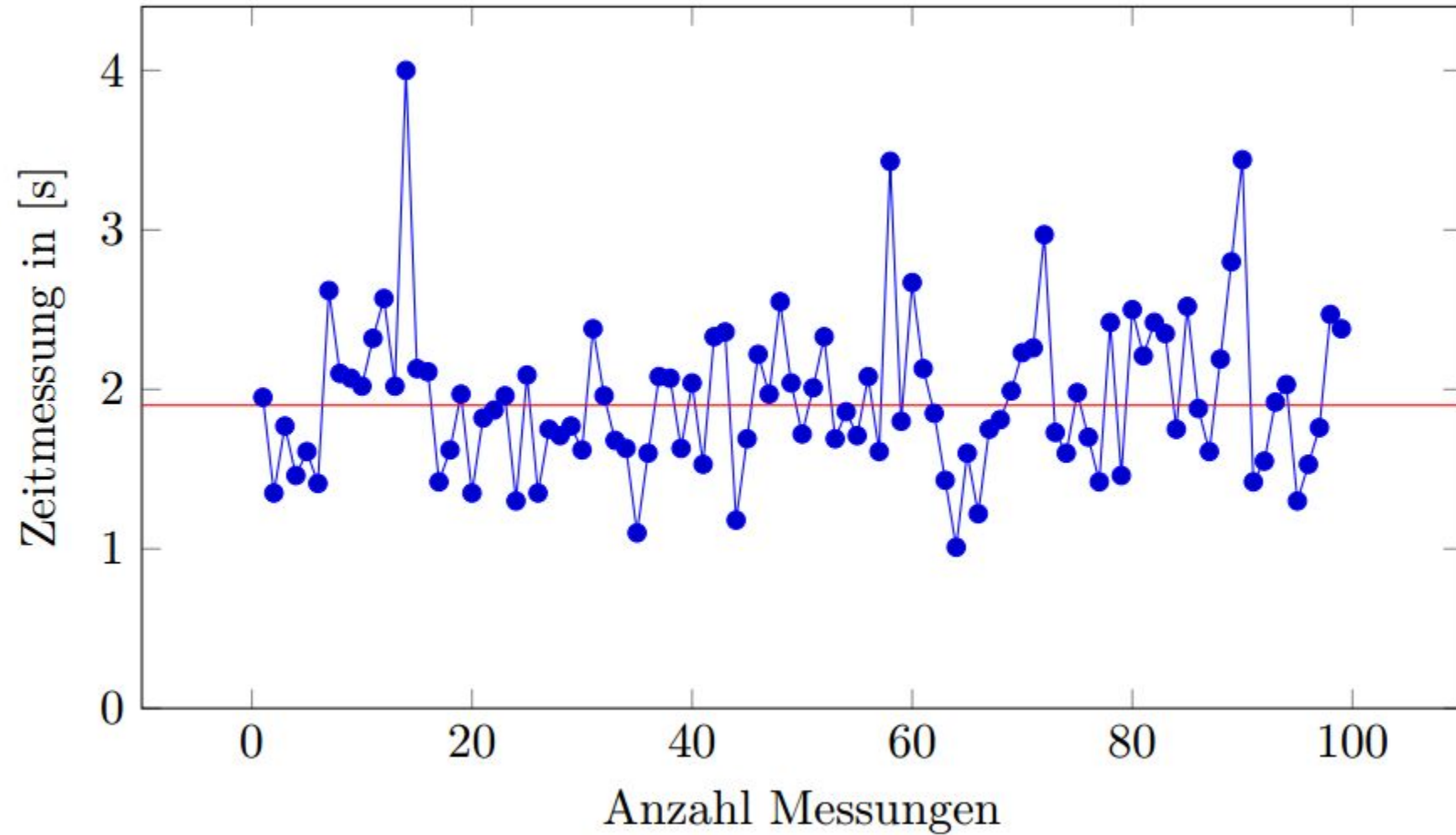
<https://github.com/qaware/building-iot-2017>

Referenzen / Literatur

- [1] <https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/built-in-intent-ref/slot-type-reference>
- [2] <https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/alexa-skills-kit-voice-design-best-practices>
- [3] <https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/alexa-skills-kit-voice-design-handbook>
- [4] <https://github.com/amzn/alexa-skills-kit-java>
- [5] <https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/speech-synthesis-markup-language-ssml-reference>
- [6] <http://start.spring.io/>
- [7] <https://developer.amazon.com/edw/home.html#/skills/list>

Appendix

Zeitmessungen



Fragen? Anmerkungen?