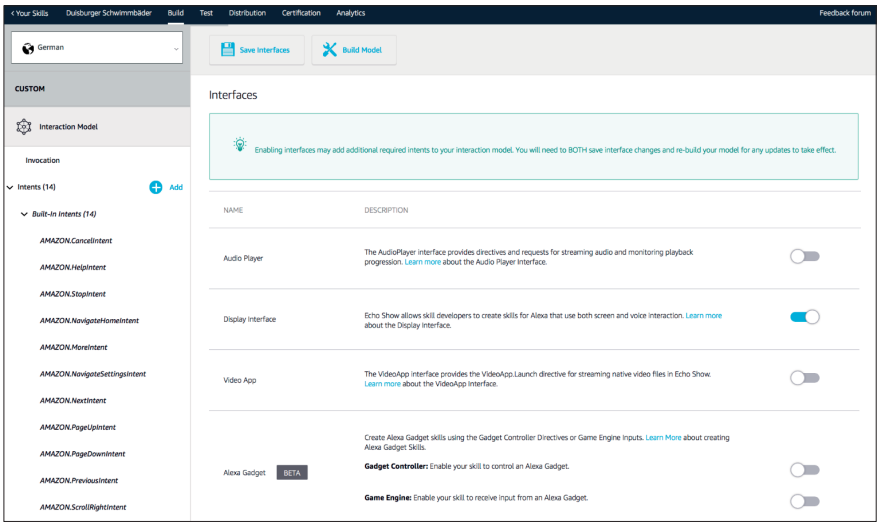
GRAFISCHE EIN- UND AUSGABE MIT ALEXA

Visuelle Rückmeldung

Mit dem Echo Show und Echo Spot existieren Alexa Endgeräte mit integriertem Bildschirm.

Das Display-Interface ermöglicht einem Skill nicht nur eine weitere Ausgabe, sondern auch ein alternative Eingabe zur Sprache. Ursprünglich waren die Echo Geräte auf Audio-Ein- und Ausgabe beschränkt. Die Card erschien ausschließlich in der App und später auch auf dem FireTV. Mit Echo Spot und Echo Show gibt es nun Endgeräte mit integriertem Bildschirm auf welchem zumindest auch die Card angezeigt wird. Mit der Integration des Bildschirmes erweiterte Amazon auch das SDK und so wurde analog zum Audioplayer Interface auch ein Interface für das Display hinzugefügt.



Aktivierung des Display Interfaces in der Skill-Konfiguration (Bild 1)

Display-Interface

Um das Display-Interface im Skill nutzen zu können, muss es in der Skill-Konfiguration zunächst aktiviert werden (Bild 1). Hierdurch werden automatisch 10 ver-pflichtende Intents hinzugefügt. Von diesen müssen nur zwei (*AMAZON.PreviousIntent* und *AMA-ZON.NextIntent*) vom Skill selbst verarbeitet werden. Die restlichen Intents werden vom Endgerät selbst verarbeitet, um den Bildschirminhalt per Sprache zu steuern (*Scroll Up / Left / Down / Right; Page Up / Down; More; NavigateHome; NavigateSettings*).

Bevor ein Skill in seiner Rückmeldung die Information zur Displaynutzung (eine Direktive vom Typ *Dis-play.RenderTemplate*) enthält, muss geprüft werden, ob das jeweilige Endgerät diese auch unterstützt. Ein Endgerät, welches die Displaynutzung nicht unterstützt, ignoriert diese nämlich nicht automatisch, sondern wird mit einem Fehler das Skill abbrechen:

```
if (isset($alexa_request->
context->System->
device->supportedInterfaces->Display))
{
//Display.RenderTemplate
}
```

Obwohl sich die Art der Datenanzeige auf dem kleinen runden Bildschirm des Echo Spot erheblich vom größeren rechteckigen Echo Show unterschiedet, war es lange Zeit nicht möglich, den Display-Typ zu erfahren. Während dieser Beitrag entstand, führte Amazon im Request innerhalb des *Context*-Objektes das Objekt *Viewport* ein – ohne es bisher in der Dokumentation des *Context*-Objekts zu beschreiben. Sein Attribut *shape* kann die Werte *RECTANGLE* (zum Beispiel FireTV und Echo Show) und *ROUND* (zur Zeit nur Echo Spot) annehmen, sodass ab sofort zwischen den Display-Typen wie folgt unterscheiden werden kann:

```
if ($alexa_request->context->
Viewport->shape == "ROUND")
{
// Code für Echo Spot
} else {
// Code für rechteckige Displays
// wie Echo Show oder FireTV
}
```

Daneben gibt es noch weitere Attribute unter anderem *pixelWidth* und *pixelHeight*, sodass man bei den rechteckigen Displays auch deren Größe erfährt und darauf reagieren könnte (Listing 1).

Reagieren bedeutet im ersten Schritt, welches der sieben Templates gewählt wird. Diese Templates agieren als Schablone und ordnen die möglichen Elemen-

Listing 1: Viewport-Objekt

```
"Viewport": {
"experiences": [{
"arcMinuteWidth":246,
"arcMinuteHeight":144,
"canRotate":false,
"canResize":false
}],
"shape":"RECTANGLE",
"pixelWidth":1024,
"pixelHeight":600,
"dpi":160,
"currentPixelWidth":1024,
"currentPixelHeight":600,
"touch":["SINGLE"],
"keyboard":[]
}
```

te auf dem Display unterschiedlich an. Aufgrund der extrem kleinen Darstellungsfläche des Echo Spots erhalten die Templates dort teilweise ein ganz anderes Aussehen. Bisher musste man sich für ein Template entscheiden, welches für einen Use-Case auf beiden Display-Typen akzeptabel war. Nun kann man das optimale Template für den Use-Case je Display-Typ ausspielen.

Einfache Schablone vs. Liste

Als erstes muss anhand des Use-Case entschieden werden, ob eine einfache Anordnung von Informationen oder eine Auflistung von Einträgen, welche selbst wieder eine Anordnung von Informationen sind, genutzt werden soll.

Bei dieser Entscheidung ist zu berücksichtigen, dass Amazon in der Zertifizierung davon ausgeht, dass die Einträge der Liste selektierbar sind. Dies führt dazu, dass mit dem Response die Session nicht geschlossen werden darf, damit der Nutzer per Sprache eine weitere Eingabe tätigen kann. Hiermit greift dann die nächste Zertifizierungsregel: Wenn die Session offen gelassen wird, muss der letzte Satz der Alexa Rückmeldung eine Frage sein.

Somit übersteht eine Liste als finale Rückmeldung (zum Beispiel als Auflistung der Zutaten für ein Koch-Rezept) die Zertifizierung nicht. Entweder muss noch ein Dialog folgen, mit welchem zu jedem Listeneintrag Detail-Informationen erfragt werden können, oder die Auflistung muss als formatierter Text in einem normalen Template zurückgeliefert werden.

Allgemeine Elemente eines Templates

Auch wenn die Nummerierung etwas anderes vermuten lässt, so gibt es zur Zeit nur fünf normale Templates. Hinzu kommen noch zwei Templates für Auflistungen, deren Elemente sich im Inhaltsbereich wiederholen. Diese Templates enthalten überwiegend die gleichen Elemente:

Das Skill-Icon wird aus der Skill-Konfiguration übernommen und auf einem rechteckigen Display in der rechten oberen Ecke eingeblendet. Auf einem Echo Spot befindet es sich mittig am oberen Rand. Der Titel wird außer in *BodyTemplate6* (und dem *List-Templat2* auf einem Echo Spot) in allen Templates verwendet und verhält sich gleich. Auf rechteckigen Displays wird dieser in der obersten Zeile von links nach rechts neben dem Skill-Icon ausgegeben. Auf dem Echo Spot wird er zentriert unter dem Skill-Icon angezeigt. Beim Überschreiten der maximal anzeigbaren Länge werden die letzten beiden anzeigbaren Zeichen durch drei Punkte ersetzt (Tabelle 1).

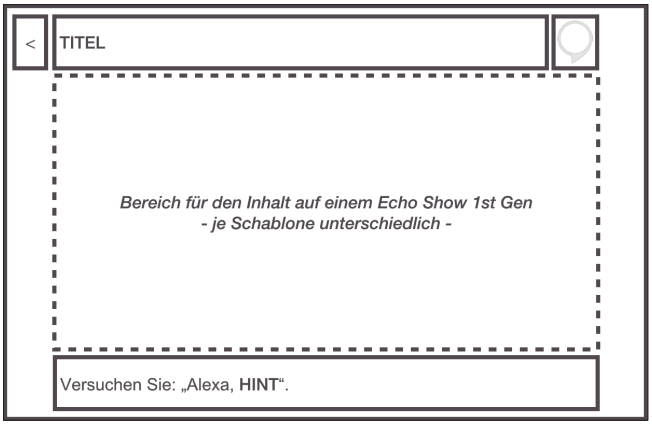
Tabelle 1: Titellänge

Gerät	Titel
Simulator – small	bis zu 2 Zeilen circa 38 Zeichen
Simulator – medium	1 Zeile circa 44 Zeichen
Echo Show (1. Gen)	1 Zeile circa 47 Zeichen
Echo Show (2. Gen)	1 Zeile circa 63 Zeichen
Echo Spot	bis zu 2 Zeilen circa 32 Zeichen
Fire TV	1 Zeile circa 53 Zeichen

Links neben dem Titel ist ein kleiner Freiraum. Dieser wird für den Back-Button bereitgehalten. Er erscheint ab dem zweiten Template, um zum vorherigen zurückzukehren. Dabei wird allerdings keine Aufruf im Backend vorgenommen und es erfolgt auch keine erneute Sprachausgabe, sondern nur die Anzeige der vorherigen Schablone. Das Einblenden des Back-Buttons kann durch Setzen des optionalen Attributs *backButton* auf den Wert *HIDDEN* unterdrückt werden.

Bei einem Echo Show erstreckt sich der für den Back-Button reservierte Freiraum über die gesamte Höhe des Templates als linksseitige Begrenzung für die weiteren Inhalte. Auf der rechten Seite wird symmetrisch ein gleicher Bereich freigehalten, wobei sich das Skill-Icon noch über dem Inhaltsbereich befindet (Bild 2).

Bei einem FireTV ist die Kopfzeile nach rechts verschoben. Das bedeutet, der Platz für den Back-Button ist nicht mehr über dem Freiraum am linken Rand, sondern über dem In- ▶



Aufteilung der Schablonen auf Echo Show 1st Generation (Bild 2)

Aufteilung der Schablonen auf einem FireTV (Bild 3)

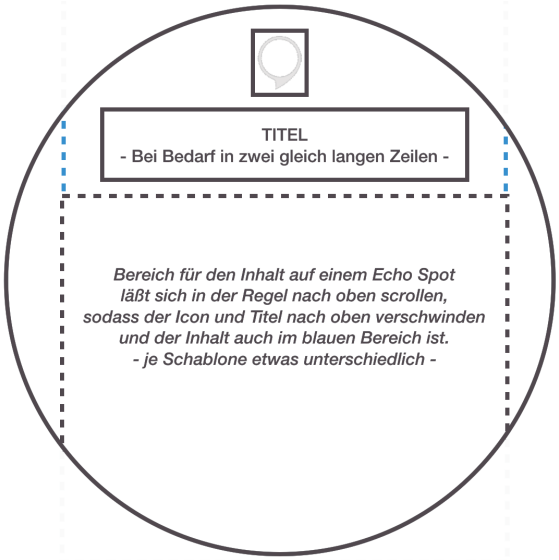
haltsbereich. Allerdings wird ein Back-Button zur Zeit auf dem FireTV nicht angezeigt, selbst wenn *backButton* explizit auf *VISIBLE* gesetzt wird. Der Effekt des Back-Buttons kann jedoch über die Fernbedienung mit der Zurück-Taste ausgelöst werden – selbst wenn der Backbutton explizit auf *HIDDEN* gesetzt ist. Während bei einem Echo Show, Echo Spot und FireTV der Rücksprung zum vorherigen Template die Sprachausgabe des aktuellen Templates bricht, wird die Sprachausgabe des bisherigen Templates auf dem FireStick fortgesetzt. Das Skill-Icon ist nun nicht mehr über dem Inhaltsbereich, sondern befindet sich über dem Freiraum am rechten Rand (Bild 3).

Kein Back-Button

Bei einem Echo Spot wird kein Back-Button angezeigt (Bild 4). Der Nutzer kann mit einem langen Wisch vom linken Rand den Effekt des Back-Buttons auslösen; entgegen der Amazon-Dokumentation auch dann, wenn der Back-Button eigentlich nicht aktiviert ist. Neben dem Titel, welcher die obere Begrenzung eines Templates darstellt, unterstützten einige Templates auf einem rechteckigen Display den optionalen Hint. Dies ist eine Beispiel-Phrase, die vom Skill als nächste Anfrage durch den Nutzer empfohlen wird und am unteren Rand das Template begrenzt.

Listing 2: DisplayTemplate Directive

```
{
"version": "1.0",
"response": {
"outputSpeech": {
    "type": "SSML",
    "ssml": "<speak>Das ist zu hören.</speak>"
  },
  "directives": [
    {
      "type": "Display.RenderTemplate",
      "template": {
      "type": "BodyTemplate2",
      "backButton": "HIDDEN",
      "title": "Hier ist Oben"
      }
    },
    {
      "type": "Hint",
      "hint": {
      "type": "PlainText",
      "text": "Und hier ist unten"
      }
    }
    ],
"shouldEndSession": false
    },
"sessionAttributes": {}
}
```



Bei einem Echo Spot wird kein Backbutton angezeigt (Bild 4)

Allerdings hat Amazon dieses Bildschirm-Element nicht als Bestandteil der Directive eines Display-Templates definiert, sondern als eigenständige Directive, die jedoch nur greift, wenn auch eine Display-Template-Directive in der Response vorhanden ist (Listing 2). Entgegen der Dokumentation wurde der Hint beim Testen auf einem Echo Show und FireTV nur bei *BodyTemplate2* und *BodyTemplate6* angezeigt. Zusätzlich ist zu beachten, dass der Hint bei der Aus-gabe in einen Satz eingebettet wird: »Versuchen Sie: Alexa, xxx.«

Dies reduziert die Anzahl der möglichen Zeichen deutlich. Bei Überschreitung erfolgt kein Zeilenumbruch und der gesamte Satz wird auf die darstellbaren Zeichen reduziert, sodass die drei Punkte am Ende auch das hochgestellte Anführungszeichen und den Satzpunkt entfernen. Die Mengenangaben in Tabelle 2, beziehen sich auf die vom Entwickler veränderbaren Zeichen nach dem Komma im Satz.

Inhalt eines Templates

Im Inhaltsbereich liegt der eigentliche Unterschied zwischen den verschiedenen Templates, da hier Text, Bild und/oder Hintergrundbild anders anordnet werden.

Der Text wird in dem Objekt *textContent* übergeben, welches selbst wieder bis zu drei Objekte (*primaryText, secondaryText* und *tertiaryText*) enthalten kann. Diese enthalten im Attribut *text* den eigentlichen Text, welcher jeweils auf maximal 8.000 Zeichen limitiert ist. Im zweiten Attribut *type* wird angeben, ob es sich um einfachen (PlainText) oder formatier-

Tabelle 2: Hint – Anzahl Zeichen

Gerät	Hint
Simulator – medium	circa 29 Zeichen
Echo Show (1. Gen)	circa 26 Zeichen
Echo Show (2. Gen)	circa 38 Zeichen
Fire TV	circa 62 Zeichen

ten (RichText) Text handelt (Listing 3). Der Typ kann für jedes der drei Objekte unterschiedlich gesetzt werden. Auch welches der Objekte verwendet wird, ist nicht eingeschränkt. So wird auch ein einzelnes *tertiaryText*-Objekt ohne die vorherigen beiden angezeigt.

Wird mehr als eines dieser Objekte verwendet, so werden die Objekte gemäß Ihrer namentlichen Reihenfolge mit einem Zeilenumbruch verbunden.

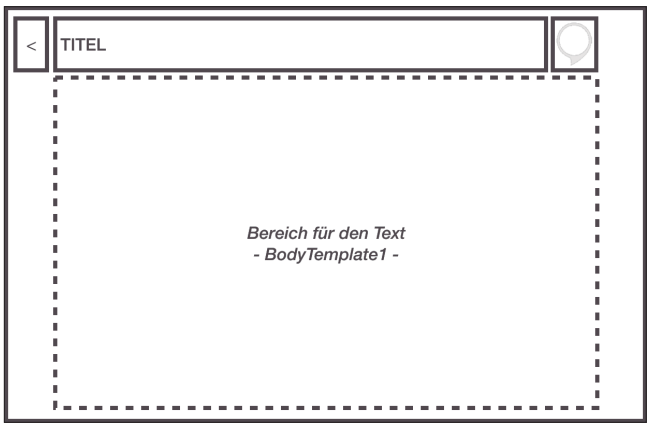
Neben Text können auch Bilder an ein Template übergeben werden, wobei zwischen einem Bild im Vordergrund und einem Hintergrundbild unterschieden wird. Beide Objekte (*image* und *backgroundImage*) haben die gleiche Struktur. Das optionale Attribut *contentDescription* enthält eine Beschreibung, welche für Screenreader zur Verfügung gestellt werden kann. Allerdings wird es von Amazon Echo-Endgeräten zur Zeit nicht verwendet. In dem Array-Attribute *sources* muss mindestens ein Objekt enthalten sein, welches eine Bildquelle definiert. Bei mehreren kann das Endgerät die beste Bildquelle für das genutzte Endgerät selbst aussuchen.

Bildquellen-Objekte

Diese Bildquellen-Objekte müssen mindesten ein Attribut *url* enthalten. Die genutzt URL muss eine öffentlich erreichbare HTTPS-Adresse eines JPG- oder PNG-Bildes sein. Daneben können die Attribute *size, widthPixels* und *heightPixels* Hilfestellungen geben, anhand derer das passende Bild ausgewählt wird. Das *size*-Attribut kann die folgenden Werte annehmen: *X_SMALL, SMALL, MEDIUM, LARGE, X_LARGE* (Listing 4). Ohne diese Angaben wird automatisch die erste Bildquelle genutzt. Wenn solche Angaben verwendet werden, führt es allerdings zu einem Fehler, wenn mehr als eine Bildquelle den gleichen Wert für das *size*-Attribut hat. Bei den Pixelangaben können die gleichen Werte genommen werden, allerdings werden diese auch nicht zur Auswahl der Bildquelle genutzt – selbst ohne *size*-Attribut. Templates, die eines der Bilder nicht unterstützten, ignorieren deren Angabe im JSON und führen nicht zu einem Fehler.

Listing 3: TextContent Objekt eines Templates

```
"textContent": {
"primaryText": {
    "type": "PlainText",
    "text": "Hier steht normaler Text!"
  },
"secondaryText": {
    "type": "RichText",
    "text": "Aber auch <b>fett</b> oder <i>kursiv
    </i> oder <u>unterstrichen</u> sind möglich."
  },
"tertiaryText": {
    "type": "PlainText",
    "text": "Und hier kommt nochmal etwas Text."
  }
}
```



BodyTemplate1 auf einem recht-eckigem Bildschirm (Bild 5)

Das Template *BodyTemplate1* ist fokussiert auf den *textContent*, welcher über den gesamten Inhaltsbereich ausgegeben wird. Weder ein Bild im Vordergrund noch ein Hint werden unterstützt, sodass der Inhaltsbereich bis zum unteren Bildschirmrand genutzt wird (Bild 5).

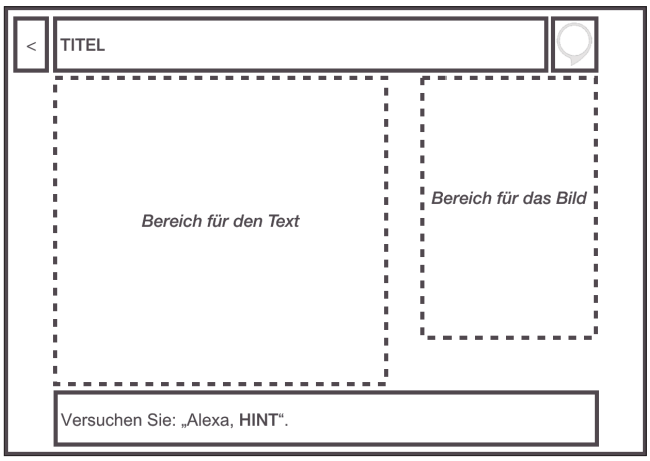
Hintergrundbild wird skaliert

Das Scroll-Verhalten ist in der Dokumentation von Amazon nicht eindeutig definiert, beim Testen stellt man aber fest, dass auf einem rechteckigen Bildschirm der Bereich um den Titel fixiert ist und der Text dahinter verschwindet. Auf einem runden Bildschirm hingegen wird der Bereich des Titels und das Skill-Icon aus dem sichtbaren Bereich geschoben. Ein eventuell vorhandenes Hintergrundbild wird auf die gesamte Fläche des Bildschirmes skaliert, was zu Verzerrungen führen kann. Bei runden Bildschirmen bedeutet dies, dass Höhe und Breite des Bildes auf den Durchmesser angepasst, und ▶

Listing 4: Image Objekt eines Templates

```
"image": {
"contentDescription" : "Auf dem Bild ist nicht zu
sehen.",
  "sources": [
    {
      "url": "https://woauchimmer.de/B-xs.png",
      "size" : "X_SMALL"
    },
    {
      "url": "https://woauchimmer.de/B-xl.png",
      "size" : "X_LARGE"
    },
    {
      "url": "https://woauchimmer.de/B-m.png",
      "widthPixels": 640,
      "heightPixels": 480
    }
  ]
}
```

die Ecken nicht angezeigt werden. Das Hintergrundbild ist fixiert und wird damit nicht beim Scrollen verschoben. Bei allen getesteten Endgeräten wird für das Hintergrundbild die *X_LARGE*-Bildquelle bevorzugt (Tabelle 3). Das Template *BodyTemplate2* unterteilt den Inhaltsbereich zu circa 2/3 links für den Text und auf der rechten Seite für das Vordergrundbild. Zwischen den beiden Bereichen ist eine Freifläche von der Breite des linken beziehungsweise rechten Randes. Auch für das Vordergrundbild wird die Bildquelle der Größe *X_LARGE* bevorzugt. Wenn es zu groß für den Bildbereich ist, wird es proportional verkleinert. Kleinere Bilder werden nicht vergrößert. Der Bereich für das Bild ist in der Regel fest reserviert, mit der Ausnahme des Echo Shows 1st Generation. Dort ist die Breite des Bildbereiches dynamisch und orientiert sich an der genutzten Breite für das Bild.



BodyTemplate2 auf einem recht-eckigem Bildschirm (Bild 6)

Zentrierung im Bildbereich

Das Bild wird eigentlich im Bildbereich zentriert. Dabei fällt jedoch auf, dass sich der Bildmittelpunkt etwas oberhalb des Mittelpunktes des freien Bereiches befindet. Bei der Verwendung eines Bildes, welches die maximale Höhe ausnutzt, sieht man dann, dass sich unterhalb des Bildbereichs – hin zum Bereich des Hints – noch eine Freifläche befindet (Bild 6). Auf einem Echo Show endet ein solches Bild in der Mitte der Zeile 9 und beim FireTV nach der Zeile 17. Beim Echo Show wird das Bild zwar horizontal zentriert, aber an der oberen Kante des Bildbereiches ausgerichtet.

Wird kein Vordergrundbild verwendet, so verschwindet der Bildbereich beim Echo Show, wobei der Trennungsbereich verbleibt, sodass der rechte Rand doppelt so breit wirkt. Beim Medium Simulator und dem FireTV bleibt der Bildbereich auch ohne Bild reserviert.

Laut Amazon Dokumentation ist nur auf einem Echo Spot der Text scrollbar; dies trifft jedoch auch auf den Echo Show zu. Ebenfalls ist die Aussage aus der Amazon-Dokumentation, dass dieses Template auf einem EchoSpot keinen Text enthalten würde, sowohl auf einem echten Spot als auch auf einem Small Simulator (zur Zeit) falsch. Dies trifft auch auf die Aussage zu, dass auf einem Spot das Hintergrundbild nur angezeigt wird, wenn kein Vordergrundbild verwendet wird. Zwar wird ein Vordergrundbild bei diesem Template als Hintergrundbild verwendet, wenn aber kein Vordergrundbild verwendet wird, zeigen weder der Echo Spot noch der Small Simulator das eigentliche Hintergrundbild an.

Bei rechteckigen Bildschirmen verhält sich das Hintergrundbild wie beim *BodyTemplate1* und der Bereich für den Hint bleibt auch dann reserviert, wenn keiner genutzt wird (Tabelle 4).

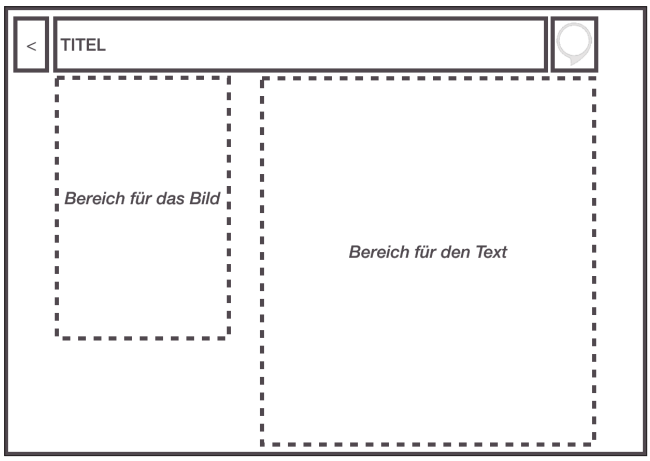
Das Template *BodyTemplate3* unterteilt den Inhaltsbereich antiproportional zum *BodyTemplate2*, sodass sich der circa 2/3 große Bereich für den Text nun rechts und das Vordergrundbild sich links befindet. Zwischen den beiden Bereichen ist weiterhin eine Freifläche von der Breite des linken

Tabelle 3: TextContent im BodyTemplate1

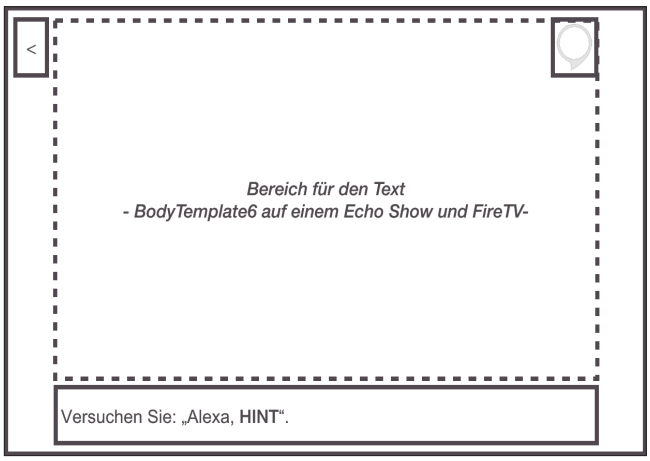
Gerät	Länge
Simulator – small	circa 4 Zeilen (nicht scrollbar) circa 13 Zeichen
Simulator – medium	12 Zeilen (es sei denn eine Zeile enthält ein Wort mit mehr als 46 Zeichen, dann werden nur die ersten 3 Zeilen angezeigt) 46 Zeichen
Echo Show (1. Gen)	13 Zeilen 46 Zeichen
Echo Show (2. Gen)	18 Zeilen 53 Zeichen
Echo Spot	circa 5 Zeilen (Anfangs, ohne Titel und Icon circa 7) circa 13 Zeichen
Fire TV	12 Zeilen 42 Zeichen

Tabelle 4: TextContent im BodyTemplate2

Gerät	Länge
Simulator – small	circa 7 Zeilen – nicht scrollbar circa 20 Zeichen
Simulator – medium	11 Zeilen (es sei denn eine Zeile enthält ein Wort mit mehr als 24 Zeichen, dann werden nur die ersten 3 Zeilen angezeigt) – nicht scrollbar 24 Zeichen
Echo Show (1. Gen)	circa 11 Zeilen (der Bereich für den Hint schneidet circa 25% der elften Zeile ab) – scrollbar 28 – 43 Zeichen
Echo Show (2. Gen)	circa 15 Zeilen (der Bereich für den Hint schneidet circa 90% der sechzehnten Zeile ab) – scrollbar 35 – 55 Zeichen
Echo Spot	circa 7 Zeilen (Anfangs, ohne Titel und Icon circa 11) – scrollbar circa 19 Zeichen
Fire TV	18 Zeile – nicht scrollbar 51 Zeichen



BodyTemplate7 auf einem runden Bildschirm (Bild 7)



BodyTemplate6 auf einem recht-eckigem Bildschirm (Bild 8)

bzw. rechten Randes. Allerdings wird bei diesem Template kein Hint unterstützt. Der Bereich für den Text nutzt dies aus, wodurch seine untere Kante bis zur unteren Bildschirmkante reicht. Der Bereich für das Bild hingegen bleibt in seiner Höhe unverändert, sodass der Unterschied bei voller Ausnutzung beider Bereiche besonders auffällig ist (Bild 7).

Alle im *BodyTemplate2* erwähnten Verhaltensweisen zu Vor- und Hintergrundbildern treffen auch hier zu. Auch die erwähnten Abweichungen zwischen echten Geräten und Dokumentation sind bei diesem Template vorhanden. Davon ausgenommen ist das Scrolling, welches auch in der Dokumentation für dieses Template bei allen Endgeräten aktiv ist. Im Medium Simulator tritt der 3-Zeilen-Fehler bei zu langen Wörtern allerdings nicht auf (Tabelle 5).

Das Template *BodyTemplate6* verzichtet auf den Titel. Dafür wird der Text am unteren Bildschirmrand ausgerichtet. Trotzdem ist es mit einem langen Text möglich, den ganzen Bildschirm zu füllen und dann passiert etwas Merkwürdiges: Zum einen erscheint er im Kopfbereich und sogar über dem Skill-Icon, zum anderen wächst der Text vom unteren Rand zum oberen. Wenn er diesen aufgrund seiner Größe erreicht, wird weiterer Text vom Hint-Bereich wie gewohnt abgeschnitten (Bild 8). Auf einem Echo Spot erscheint der Text nicht über dem Icon, sondern einfach ohne Freifläche für einen Titel direkt unterhalb des Icons.

Das Template zeigt wie gewohnt – sofern vorhanden – ein Hintergrundbild an, ein Vordergrundbild hingegen wird nicht unterstützt (Tabelle 6).

Das letzte normale Template *BodyTemplate7* dient der großflächigen Anzeige eines Bildes. Dabei wird das Hintergrundbild wie gehabt über die gesamte Bildschirmfläche gelegt, während das Vordergrundbild im vergleichbaren Inhaltsbereich wie bei *BodyTemplate1* proportional skaliert und mittig ausgerichtet wird. So nutzt eine Kante des Bildes die maximale Breite beziehungsweise Höhe und die andere wird mittig justiert.

Ein Text oder ein Hint werden nicht unterstützt. Da das Bild vollständig sichtbar ist, wird auch Scrollen nicht unter- ▶

Tabelle 5: TextContent im BodyTemplate3

Gerät	Länge
Simulator – small	circa 7 Zeilen – nicht scrollbar circa 20 Zeichen
Simulator – medium	13 Zeilen (kein Fehler bei zu längen Wörtern wie bei BodyTemplate3) – nicht scrollbar 24 Zeichen
Echo Show (1. Gen)	13 Zeilen 28 – 43 Zeichen
Echo Show (2. Gen)	18 Zeilen 35 – 55 Zeichen
Echo Spot	circa 7 Zeilen (Anfangs, ohne Titel und Icon circa 11) circa 19 Zeichen
Fire TV	circa 23 Zeilen (der Bildschirm schneidet circa 30 Prozent der 23 Zeile ab) – scrollbar 51 Zeichen

Tabelle 6: TextContent im BodyTemplate6

Gerät	Länge
Simulator – small	circa 6 Zeilen – nicht scrollbar circa 11 Zeichen
Simulator – medium	6 Zeilen (Fehler bei zu längen Wörtern wie bei BodyTemplate3) – nicht scrollbar 20 Zeichen
Echo Show (1. Gen)	13 Zeilen – nicht scrollbar 46 Zeichen
Echo Show (2. Gen)	18 Zeilen (19 halb abgeschnitten) – nicht scrollbar 58 Zeichen
Echo Spot	circa 6 Zeilen (Anfangs, ohne Icon circa 7) circa 12 Zeichen
Fire TV	circa 17 Zeilen (der Bildschirm schneidet circa 20 Prozent der 17 Zeile ab) – nicht scrollbar 58 Zeichen

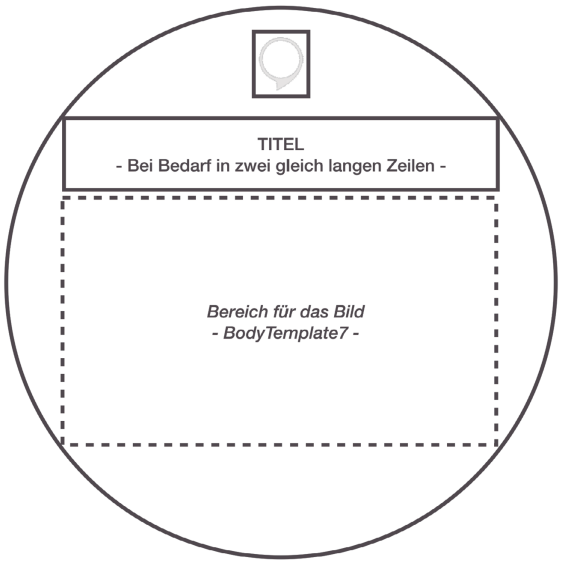
stützt. Die Verschiebung des Kopfbereiches auf einem FireTV findet auch bei diesem Template statt. Auf einem Echo Spot ist der Bereich für den Titel etwas vergrößert, da seine Breite der des Inhaltsbereiches entspricht. Es werden bis zu 44 Zeichen angezeigt (Bild 9).

Anordnung der List-Templates

Es existieren zusätzlich noch zwei Templates zur Auflistung von Informationen. Das Template *ListTemplate1* listet die Einträge vertikal auf und nutzt als Bereich für die Liste einen vergleichbaren Inhaltsbereich wie *BodyTemplate1*. Das Template *ListTemplate2* ordnet die Einträge horizontal in einem etwas kleineren Inhaltsbereich an, da darunter noch ein Hint angezeigt werden kann.

Tabelle 7: TextContent im ListTemplate1

Gerät	Länge
Simulator – small	1,5 Einträge sichtbar (scrollbar) Primärer Text: 19 Zeichen Sekundärer Text: 14 Zeichen Tertiärer Text: max 14 Zeichen Alle drei werden untereinander angezeigt
Simulator – medium	3 Einträge sichtbar (scrollbar) Primärer Text: 14 Zeichen Sekundärer Text: 34 Zeichen Tertiärer Text: 11 Zeichen Wenn TertiärerText verwendet wird, reduzieren sich Primärer Text auf 7 Zeichen und Sekundärer Text auf 18 Zeichen
Echo Show (1. Gen)	3,5 Einträge sichtbar (scrollbar) Primärer Text: 16 Zeichen Sekundärer Text: 26 Zeichen Tertiärer Text: 28 Zeichen Bei Kollision wird nur ein Text angezeigt, in folgender Präferenz Primär, Tertiär, Sekundär
Echo Show (2. Gen)	5 Einträge sichtbar (scrollbar) Primärer Text: 22 Zeichen Sekundärer Text: 51 Zeichen Tertiärer Text: 37 Zeichen Bei Kollision wie Echo Show 1 Gen, aber primärer und sekundärer Text werden ohne tertiären Text zusammen angezeigt Bei sekundärem und tertiärem Text ohne primären Text erfolgt keine Anzeige
Echo Spot	3 Einträge (scrollbar, aber Skill-Icon und Titel bleiben fixiert) Primärer Text: 8 Zeichen Sekundärer Text: 14 Zeichen Tertiärer Text: 17 Zeichen Bei Kollision wie Echo Show 1 Gen
Fire TV	5 Einträge sichtbar (scrollbar) Primärer Text: 33 Zeichen Sekundärer Text: 65 Zeichen Tertiärer Text: 37 Zeichen Bei Kollision wie Echo Show 1 Gen, aber primärer und sekundärer Text werden ohne tertiären Text zusammen angezeigt. Tertiärer Text dehnt sich bis in die Freifläche für Bilder aus.

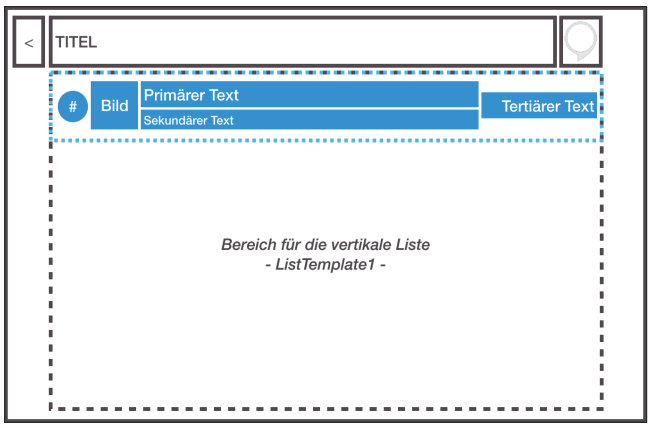


BodyTemplate7 auf einem runden Bildschirm (Bild 9)

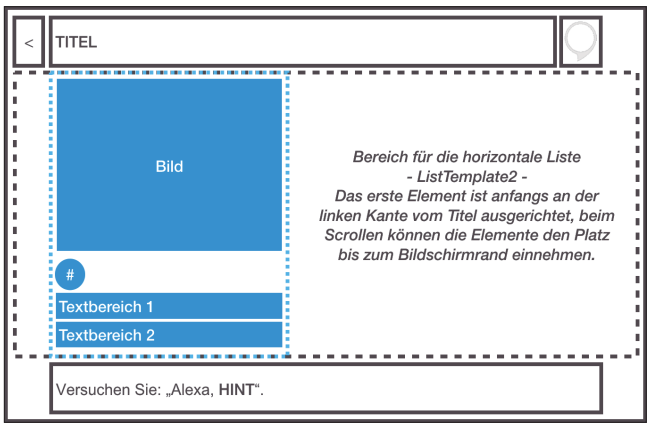
Um eine Auflistung zu ermöglichen, wird das Vordergrundbild und das *textContent*-Objekt in einem List-Item zusammengefasst, welches in einem Array *listItems* aufgezählt wird (Listing 5). Dieses Array muss mindestens ein Element enthalten. Text und Vordergrundbild direkt im Template-Objekt

Listing 5: Listeneinträge eines ListTemplates

```
"listItems": [
{"token": "MeinToken",
"textContent": {
"primaryText": {
      "type": "PlainText",
      "text": "Frühstück"
    }
  }
  "image": {
"sources": [{
      "url": "https://woauchimmer.de/bild1.png"
    }],
"contentDescription": "Eier und Speck"
  }
},{
  "token": "MeinToken2",
  "textContent": {
  "primaryText": {
  "type": "PlainText",
  "text": "Mittagessen"
  }
}
"image": {
"sources": [{
      "url": "https://woauchimmer.de/bild2.png"
      }],
      "contentDescription": "Schnitzel und Pommes"
  }
}]
```



ListTemplate1 auf einem rechteckigen Bildschirm (Bild 10)



ListTemplate2 auf einem rechteckigen Bildschirm (Bild 11)

werden ignoriert. Hintergrundbild und Titel werden auf den rechteckigen Bildschirmen wie gewohnt verwendet.

Im Template *ListTemplate1* werden die Inhalte gemäß Bild 10 angeordnet. Dabei beginnt die Zeile links mit einer automatisch generierten Nummer, gefolgt von einer quadratischen Freifläche für ein optionales Bild. Dieses wird proportional skaliert und an der kleineren Kante mittig ausgerichtet. Dann folgen der primäre und sekundäre Text, welche übereinander angeordnet werden. Die Zeile schließt mit dem rechtsbündig ausgerichteten tertiären Text ab. Bei kürzeren Texten ist dies sehr übersichtlich, während sich bei zu langen Texten Überlappungen ergeben, die von den verschieden Endgeräten unterschiedlich aufgelöst werden (Tabelle 7).

Proportionale Skalierung

Im Template *ListTemplate2* werden die Inhalte auf einem rechteckigen Bildschirm gemäß Bild 11 angeordnet. Im Zentrum steht bei einem Eintrag die große quadratische Freifläche für ein eventuelle vorhandenes Bild. Diese wird proportional skaliert und die kleinere Kante mittig ausgerichtet. Darunter folgt in einer Zeile linksbündig die automatisch generierte Nummer des Eintrages.

Unter dieser recht leeren Zeile folgen dann zwei Zeilen. Wenn ein primärer Text angegeben ist, so wird dieser in der ersten Zeile ausgegeben. In der zweiten wird der sekundäre Text – sofern vorhanden – dem tertiären vorgezogen. Ohne primären Text wird nur die erste Zeile genutzt. Auch dann gilt, dass der sekundäre Text – wenn vorhanden – dem tertiären vorgezogen und dieser wiederum nicht angezeigt wird, wobei die zweite Zeile leer bleibt. Auf einem runden Bildschirm wird bei diesem Template nur ein Eintrag auf einmal angezeigt. Ein Hintergrundbild wird ignoriert und das Bild des Eintrages wird als Hintergrundbild genutzt. Oben erscheint das Skill-Icon und der Titel wird ignoriert. Am unteren Bildschirmrand werden bis zu zwei Textbereiche angezeigt, die genauso belegt wer-den, wie beim rechteckigen Bildschirm. Unter dem Textbereich des Eintrages steht noch die Nummer des aktuellen Eintrages, getrennt mit einem | von der Gesamtanzahl der Einträge der Liste (Tabelle 8).

Auswählen eines Eintrages

Wie bereits erwähnt, darf die Session bei einem Listen-Template nicht geschlossen werden, damit der Nutzer die Möglichkeit hat, per Sprache einen Eintrag auszuwählen. Da die Bildschirme aber auch eine Auswahl per Touch zulassen, muss dies ebenfalls unterstützt werden. Hierfür erhält jeder Listeneintrag ein Attribut *token*.

Dies ist ein frei definierbarer String, über den das Skill die Auswahl erkennt. Ein- und derselbe String können dabei öfter verwendet werden, sind dann aber nicht mehr unter- ▶

Tabelle 8: TextContent im ListTemplate2

Gerät	Länge
Simulator – small	1 Eintrage sichtbar (nicht scrollbar) Primärer Text: 25 Zeichen Sekundärer Text: 18 Zeichen Tertiärer Text: max 18 Zeichen
Simulator – medium	1,8 Einträge sichtbar (scrollbar) Primärer Text: 25 Zeichen Sekundärer Text: 34 Zeichen Tertiärer Text: 34 Zeichen In der ersten Zeile wird nur der primäre Text ausgegeben und bleibt ohne einen solchen frei In der zweiten wird der sekundäre einem tertiären vorgezogen
Echo Show (1. Gen)	2,8 Einträge sichtbar (scrollbar) Primärer Text: 15 Zeichen Sekundärer Text: 21 Zeichen Tertiärer Text: 21 Zeichen
Echo Show (2. Gen)	2,8 Einträge sichtbar (scrollbar) Primärer Text: 22 Zeichen Sekundärer Text: 31 Zeichen Tertiärer Text: 31 Zeichen
Echo Spot	1 Entrag sichtbar (scrollbar) Primärer Text: 24 Zeichen Sekundärer Text: 37 Zeichen Tertiärer Text: 37 Zeichen
Fire TV	4,2 Einträge sichtbar (scrollbar) Primärer Text: 17 Zeichen Sekundärer Text: 24 Zeichen Tertiärer Text: 24 Zeichen

scheidbar und führen zum gleichen Ergebnis. Der ausgelöste Aufruf hat aber nicht den Type IntentRequest, sondern Display.ElementSelected und erhält als Attribut den token auf das entsprechend Listing 6 zugegriffen werden kann. Auf diesen Aufruf wird mit dem gleichen JSON geantwortet, wie bei einem Aufruf eines Intents. Wichtig ist jedoch, dass eventuelle Card-Informationen bei dieser Rückmeldung ignoriert und nicht in der Alexa-App angezeigt werden.

Formatierter Text

Alle bisher ermittelten Zeichenlängen bezogen sich auf Text vom Typ *PlainText* mit seiner vorgegeben Schriftgröße. Beim Typ *RichText* kann für die mögliche Zeichenlänge keine Aussage getroffen werden, da die Formatierung hier zu viele Möglichkeiten zulässt.

Die interessanteste Variante ist der Action-Tag. Durch diesen wird ermöglicht, dass – angelehnt an eine HTML-Verlinkung – der entsprechende Text zu einem *Display.ElementSelect*-Aufruf führt:

```
Der <action token="MeinToken">aktuelle Speiseplan
</action> ist verfügbar.
```

Neben den Tags für fett, kursiv und unterstrichen existieren noch Tags für den Zeilenumbruch und die Schriftgröße:

```
<font size="2">kleiner Text</font><br/>
<font size="7">großer Text</font>
```

Wobei letztere nur die Größen 2, 3, 5 und 7 unterstützt – welche ihrerseits auf den verschiedenen Endgeräten zu passenden absoluten Größen interpretiert werden. Zu guter Letzt existiert noch ein Tag zum Einfügen eines Bildes, wobei dieses dann laut Dokumentation auf die Zeilenhöhe des Textes skaliert werden sollte und nicht mit einem Vordergrundbild sondern eher mit einem Custom-Emoji vergleichbar ist:

```
Das Product verdient eine <img src=
'https://www.woauchimmer.de/4Sterne.jpg' width='500'
height='100' alt='test image' /> Bewertung.
```

Allerdings wird ein Bild ohne Größenangaben nur im Simulator (dort aber in voller Größe) angezeigt. Alle echten Gerät zeigen das Bild nur an, wenn Höhe und Weite angegeben werden, und dann wird das Bild entsprechend diesen Angaben skaliert – ohne Rücksicht auf die Zeilen-Höhe des Textes.

Listing 6: Touchauswahl-Auswertung

```
// Den Datenstream der Anfrage erhalten
$postPlain = file_get_contents( 'php://input' );
// Decode the JSON into a stdClass object
$alexa_request = json_decode( $postPlain );
if ($alexa_request->request->type ==
"Display.ElementSelected")
   {
   if ($alexa_request->request->token == "MeinToken")
      {
       // Was bei der Auswahl von MeinToken
       // passieren soll.
            }
   }
```

Links zum Thema

- Beta Dokumentation – Alexa Presentation Language (APL) – Viewport Property
  https://developer.amazon.com/de/docs/alexa-presentation-language/apl-viewport-property.html
- Display Template Reference
  https://developer.amazon.com/de/docs/custom-skills/display-template-reference.html
- Markup Referenz für RichText im DisplayTemplate
  https://developer.amazon.com/de/docs/custom-skills/display-interface-reference.html#supported-markup
- Hint Definition
  https://developer.amazon.com/de/docs/custom-skills/display-interface-reference.html#hint-directive
- 3rd Party (Jovo) Editor zum schnellen validieren des Layouts von Display Templates
  https://www.jovo.tech/display-template-editor

Fazit

Amazon hat mit Cards von Anfang an neben der Sprach- auch Zusatzinformationen für Bildschirme unterstützt. Mit dem Display-Template wurden die Möglichkeiten der Ein- und Ausgabe direkt auf dem Bildschirm des Sprachassistenten stark erweitert. Bei der Geschwindigkeit mit der Amazon neue APIs/SDK-Erweiterungen veröffentlicht beziehungsweise bestehende Implementierungen anpasst, fällt es nicht nur uns Entwicklern schwer mitzuhalten, sondern der offiziellen Dokumentation anscheinend auch (alle im Artikel aufgeführten Fehler wurden an Amazon gemeldet). Mit dem aktuellen Beta-Test der APL (Amazon Presentation Language) zeichnet sich bereits die dritte Evolutionsstufe für multimodale Skills ab, die den Freiheitsgrad bei der Gestaltung von Bildschirmausgaben deutlich erhöhen wird. ■

Emil Thies
ist in der IT Abteilung der Deutschen Telekom angestellt und probiert auch nebenberuflich neue Technologien aus, indem er eigene Apps, Skills und Actions programmiert und veröffentlicht.