

# Beyond Web Interfaces

Fabrizio Ciacchi / Spryker Systems GmbH.



# I'm Fabrizio



Software Engineer at Spryker

<http://ciacchi.it> - <https://spryker.com>

## What is this talk about?

- Give you insights so you can start to look into the best articles, tools and books;
- Tell you real examples of (chat/voice) bots. How to implement them, so to solve real problems;
- Discuss solutions and ideas on how bots should solve existing problems.

# It's the next Far West



Google

Google Search

I'm Feeling Lucky



“Alexa, add soap to my shopping cart”



“I would like to see who sells shoes nearby”

# It's the next Far West



**IBM**  
Watson & Cloud

<https://www.ibm.com/cloud/ai>

**Facebook**  
Messenger / WhatsApp  
PyTorch / Wit.ai  
Marketplace

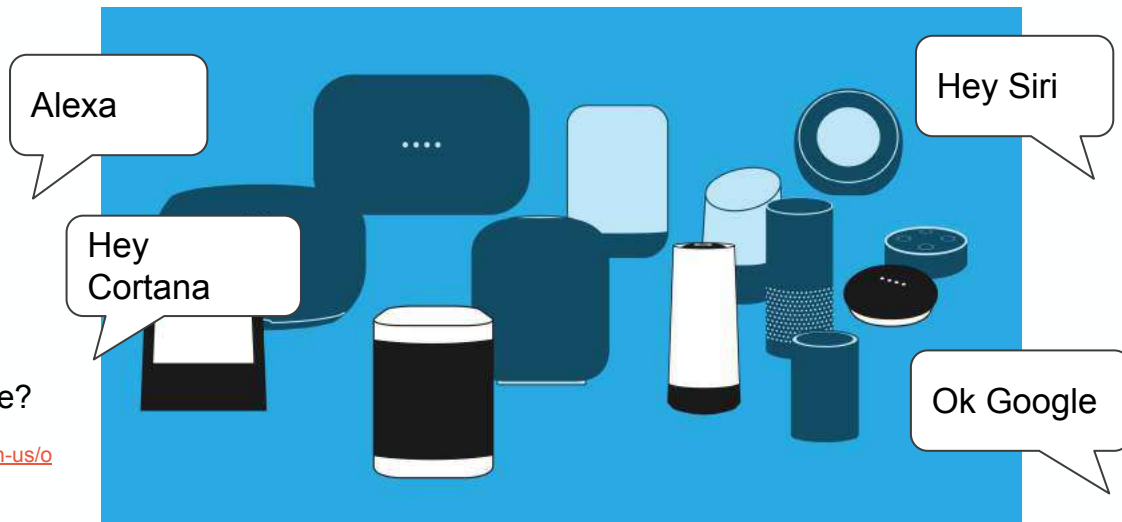
<https://pytorch.org/>

Cloud (AWS)  
AI (Lex, Polly)  
Machine Learning  
Alexa (Voice-bot)

<https://aws.amazon.com/machine-learning/>

Cloud (Azure)  
AI & ML  
Any Windows device?

<https://azure.microsoft.com/en-us/overview/ai-platform/>



Apple Watch  
Siri/Assistant  
Shortcuts

(not to mention Apple Car)

Google Cloud  
Google Home  
Tensorflow  
Dialogflow

<https://dialogflow.com/>

# User Journey



10 results

300-500 msec

1-2 steps



Search - 50 results

Response - 2 sec

Checkout - 4/5 steps



3 results

100 msec\*

1 click

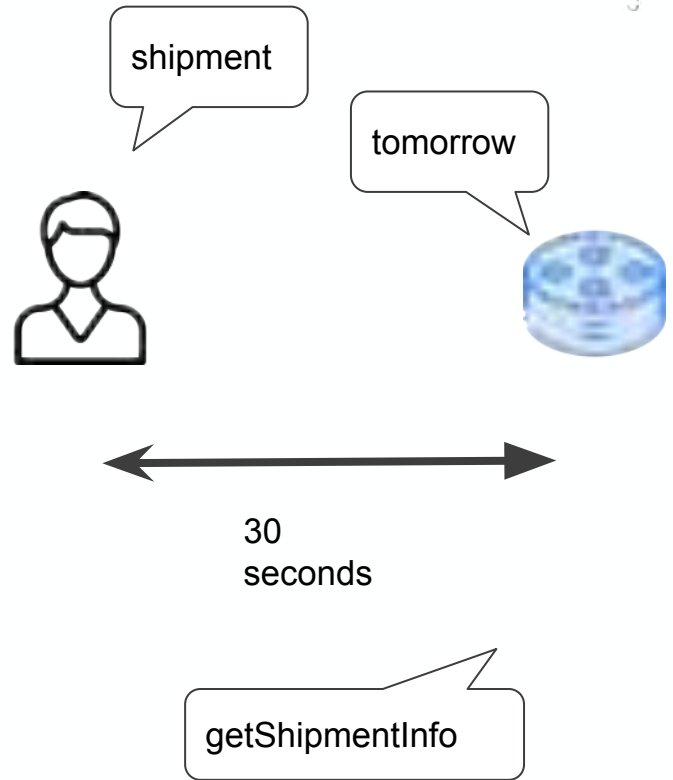


# Chat VS Voice



# Similarities

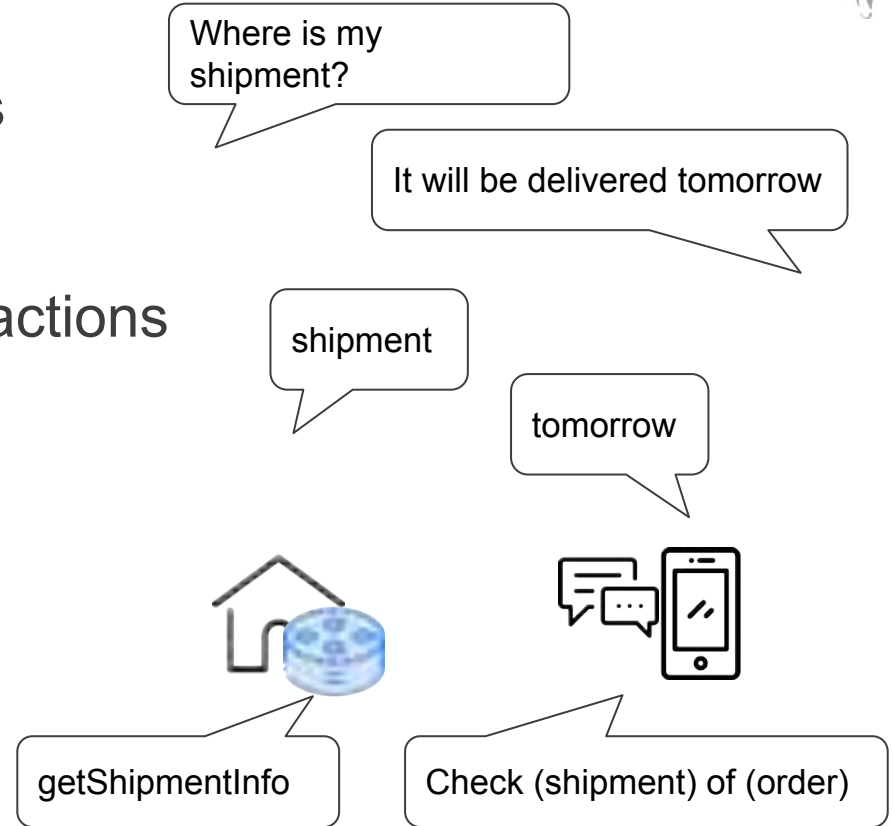
- Convey small information
- Users want to ask for info
- Time interaction is short
- Narrow down to intent/response



# Differences

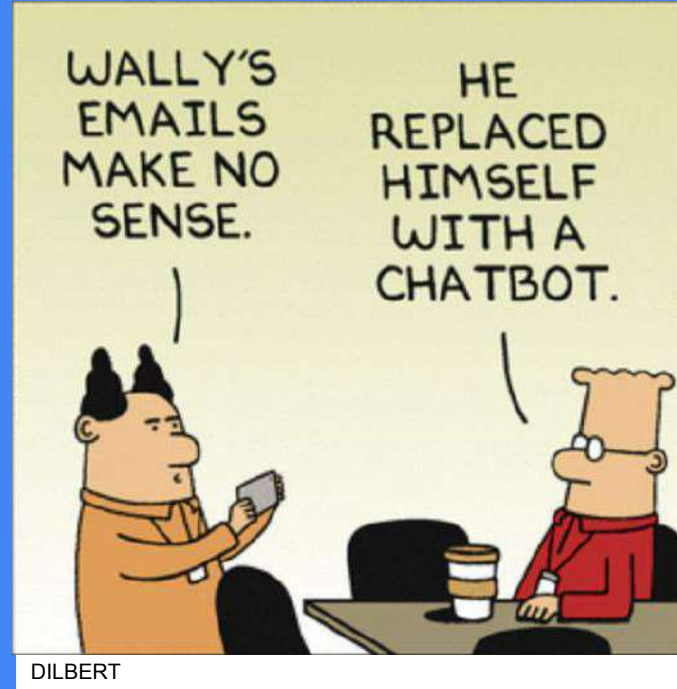


- Voice better for informative skills
- Voice has longer message/interactions
- Use cases are different
- Different technologies





Tell me more



# Start simple



- My example uses a Simple PHP Application:  
Can be Symfony, Spryker or any other framework

- Botman with Regular Expressions

<https://botman.io/>

Laravel module - Open source - Works also in Symfony

<https://gist.github.com/fciacchi/30bcff6d77ff40ae811740efa338adf0>

- Use Docker & Ngrok

<https://gist.github.com/fciacchi/0a85530efb0f05396c21778af22ff1cc>

<https://ngrok.com/>

```
$ ./ngrok http -subdomain=yousubdomain -region eu 192.168.99.100:80
```

<http://localhost:4040>

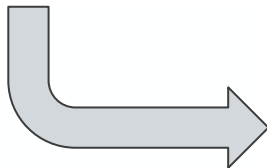
# Conversations in PHP



- Then you can start grouping the intents and the conversations

<https://botman.io/2.0/conversations>

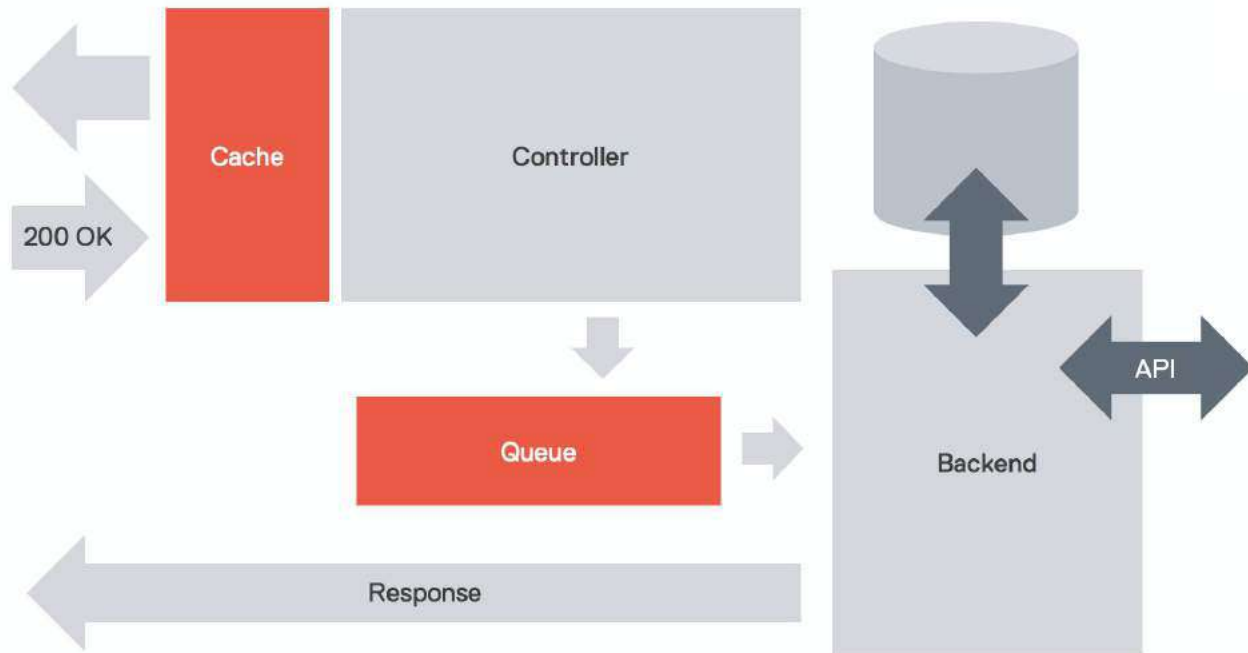
```
$botman->hears('Hello', function($bot) {  
    $bot->startConversation(new OnboardingConversation);  
});
```



Let's take a look at the `OnboardingConversation` class:

```
class OnboardingConversation extends Conversation  
{  
    protected $firstname;  
  
    protected $email;  
  
    public function askFirstname()  
    {  
        $this->ask('Hello! What is your firstname?', function(Answer $answer) {  
            // Save result  
            $this->firstname = $answer->getText();  
  
            $this->say('Nice to meet you ' . $this->firstname);  
            $this->askEmail();  
        });  
    }  
  
    public function askEmail()  
    {  
        $this->ask('One more thing - what is your email?', function(Answer $answer) {  
            // Save result  
            $this->email = $answer->getText();  
  
            $this->say('Great - that is all we need, ' . $this->firstname);  
        });  
    }  
  
    public function run()  
    {  
        // This will be called immediately  
        $this->askFirstname();  
    }  
}
```

# Backend Design



## wit.ai



{ REST }

Build a Facebook Bot: <https://blog.spryker.com/how-to-build-bot-5-steps>



## CONFIDENCE

Where can I find a restaurant in **Berlin**?



myIntent

getRestaurants

0.990

myCity

Berlin

0.988

myContext

restaurant

0.989

mySubContext

venues

0.930

myOneWord

restaurant

0.592

myDefaultAnswer

Berlin is a city that never slee...

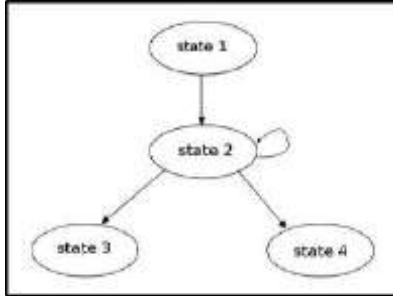
0.974

+ Add a new entity

✓ Validate



# And a Conversation manager

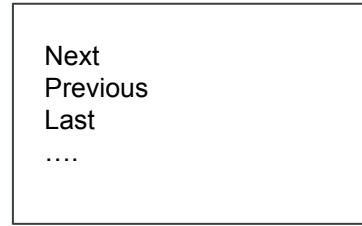


We start with a State machine

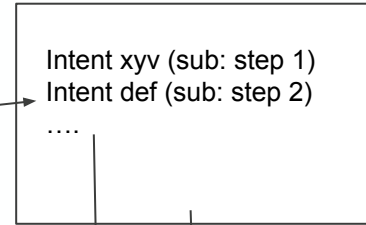
## SESSION

Last Intent: getRestaurants  
Context: context-2  
Item selected: 3  
Location: Berlin  
Filter:  
- type: sushi  
- price: \$\$

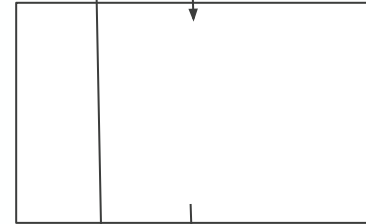
navigation



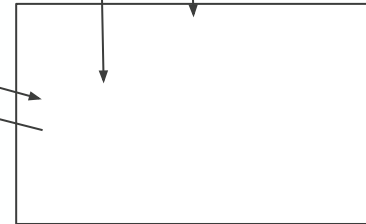
context-1



context-2



context-3



# Example Location



Where can I find a restaurant in Berlin?

If you share your location I can be more accurate. Do you want to?

Yes

Please share your location...



Your location  
Tap to view on map



**Zet: Das Restaurant**  
Julie-Wolfthorn-Str. 1, Berlin

[Website](#)

Where can I find a restaurant nearby?

Where can I find a restaurant in Amsterdam?

## SESSION

City: Berlin  
Airport: TXL, SXF

Zip Code: 10117  
Address: Friedrichstraße 43

Lat: 52.520450  
Lon: 13.407320

1d

3h

# Speaking of Alexa





# Alexa Skill creator

The screenshot shows the Alexa Skill creator interface for the skill 'getCalendarByDay'. The left sidebar contains a navigation menu with the following items:

- Dashboard
- Code Editor
- Intents (6) ADD +
  - addItemTodo
  - AMAZON.CancelIntent (required)
  - AMAZON.HelpIntent (required)
  - AMAZON.StopIntent (required)
  - getCalendarByDay** (selected)
  - day
  - getTodoList
- Slot Types (2) ADD +
  - AMAZON.DATE
  - text

The main content area is titled 'getCalendarByDay' and contains the following sections:

- Sample Utterances (4)**: A list of sample utterances for the intent, each with a trash icon for deletion.
  - What might a user say to invoke this intent? +
  - \*do I have any appointment on {day}\* 🗑️
  - \*which are my appointments for {day}\* 🗑️
  - \*how does my calendar look like {day}\* 🗑️
  - \*what do I have to do {day}\* 🗑️
- Intent confirmation (optional)**: A toggle switch labeled 'NO' for 'Does this intent require confirmation?'.
- Prompts (0)**: A text input field for 'What will Alexa say to ask the user to confirm the intent?'.

The screenshot shows the 'Intent Slots (1)' configuration panel. It features a table with columns for 'ORDER', 'REQ', and 'SLOT'. The table contains one slot configuration:

ORDER	REQ	SLOT
1	<input type="checkbox"/>	day AMAZON.DATE

Below the table is a text input field labeled 'Create a new slot...' and a blue 'Add' button.

<https://developer.amazon.com/designing-for-voice/>

# Get the Skill's JSON

The screenshot shows the Amazon Skill Builder interface for a skill named "RemindMe" in English (US). The top navigation bar includes "Save Model", "Build Model", "Skill Information", and "Interaction Model". The left sidebar shows a "Code Editor" view with a list of intents: "additemTodo", "AMAZON.CancelIntent (required)", "AMAZON.HelpIntent (required)", "AMAZON.StopIntent (required)", "getCalendarByDay", "day", "getTodoList", "Slot Types (2)", "AMAZON.DATE", and "text". The main code editor displays the following JSON:

```
3 {
4   "name": "additemTodo",
5   "samples": [
6     "I want to add a task (todo)"
7   ],
8   "slots": [
9     {
10      "name": "todo",
11      "type": "text",
12      "samples": []
13    }
14  ],
15  },
16  {
17    "name": "AMAZON.CancelIntent",
18    "samples": []
19  },
20  {
21    "name": "AMAZON.HelpIntent",
22    "samples": []
23  },
24  {
25    "name": "AMAZON.StopIntent",
26    "samples": []
27  },
28  },
29  {
30    "name": "getCalendarByDay",
31    "samples": [
32      "what do I have to do {day}",
33      "how does my calendar look like {day}",
34      "which are my appointments for {day}",
35      "do I have any appointment on {day}"
36    ],
37    "slots": [
38      {
39        "name": "day",
40        "type": "AMAZON.DATE",
41        "samples": []
42      }
43    ]
44  }
45 }
```

<https://skillinator.io/>

The screenshot shows the Skillinator v1.0.3 interface. It features two main panels: "Interaction Model JSON" and "Lambda Template". A "Generate" button is located in the top right corner of the Lambda Template panel. The "Interaction Model JSON" panel displays the following JSON:

```
{
  "synonyms": []
},
{
  "id": null,
  "name": {
    "value": "Call mama",
    "synonyms": []
  }
},
{
  "id": null,
  "name": {
    "value": "Call electrician",
    "synonyms": []
  }
}
}
```

The "Lambda Template" panel displays the following code:

```
AMAZON.CancelIntent: function () {
  speechOutput = "";
  this.emit('tell', speechOutput);
},
AMAZON.StopIntent: function () {
  speechOutput = "";
  this.emit('tell', speechOutput);
},
SessionEndedRequest: function () {
  speechOutput = "";
  //this.emit('saveState', true); //uncomment to save attributes to db on session end
  this.emit('tell', speechOutput);
},
additemTodo: function () {
  var speechOutput = "";
  //any intent slot variables are listed here for convenience
  var todoSlotRaw = this.event.request.intent.slots.todo.value;
  console.log(todoSlotRaw);
  var todoSlot = resolveCanonical(this.event.request.intent.slots.todo);
  console.log(todoSlot);

  //Your custom intent handling goes here
```

# NodeJS function 1/2

---

<https://console..amazon.com/console/home>

<https://eu-west-1.console.aws.amazon.com/lambda/home?region=eu-west-1#>

```
"getCalendarByDay": function () {  
    var speechOutput = "I didn't understand. Can you repeat?";  
    var self = this;  
    var daySlotRaw = this.event.request.intent.slots.day.value;  
    console.log(daySlotRaw);  
    var daySlot = resolveCanonical(this.event.request.intent.slots.day);  
    console.log(daySlot);  
  
    var url = 'https://flybybot.eu.ngrok.io/ikarus/alexa/calendar?day='+daySlot;
```

# Lambda function 2/2

---

```
...
https.get( url, function( response ) {

    var str = "";

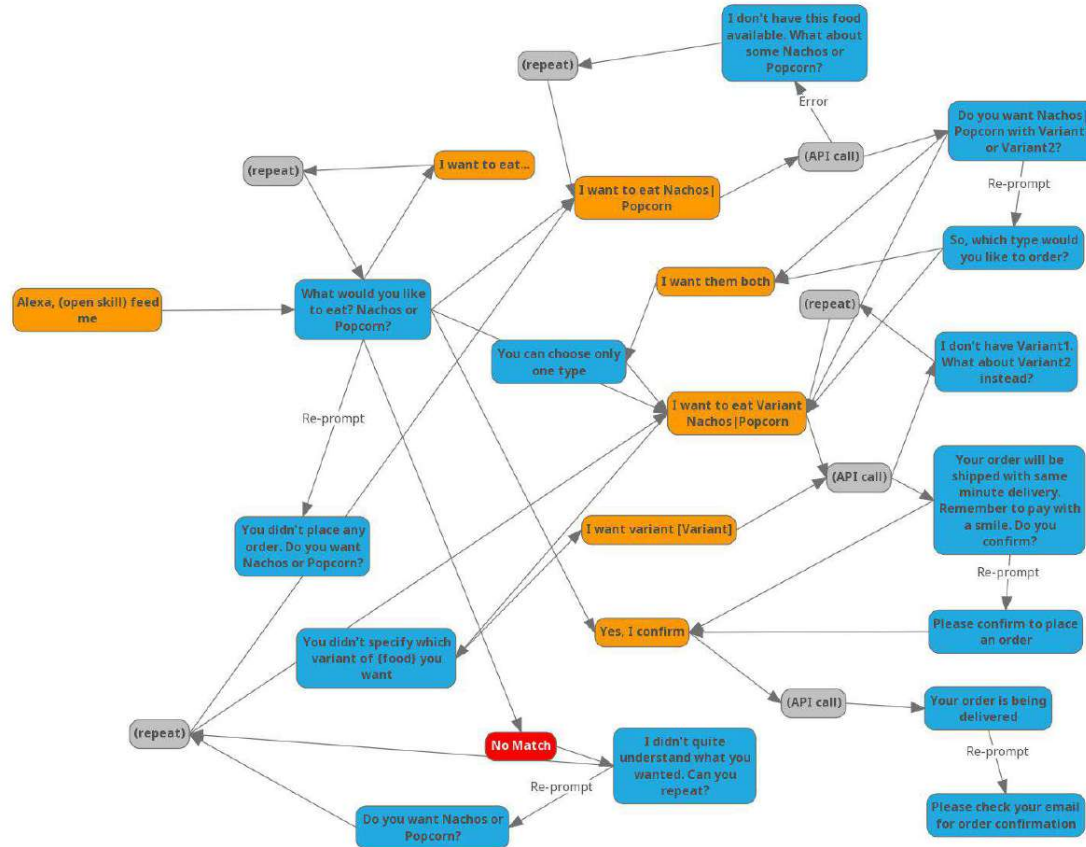
    response.on('data', function (chunk) {
        str += chunk;
    });

    response.on('end', function () {
        jsonResponse = JSON.parse(str);
        console.log("RESPONSE ---> " + jsonResponse.response);
        self.emit(':ask', 'On ' + daySlot + ' ' + jsonResponse.response);
    });
});
},
```

Model: <https://gist.github.com/fciacchi/61d42439f413d8faffd4a284b2aeee99>

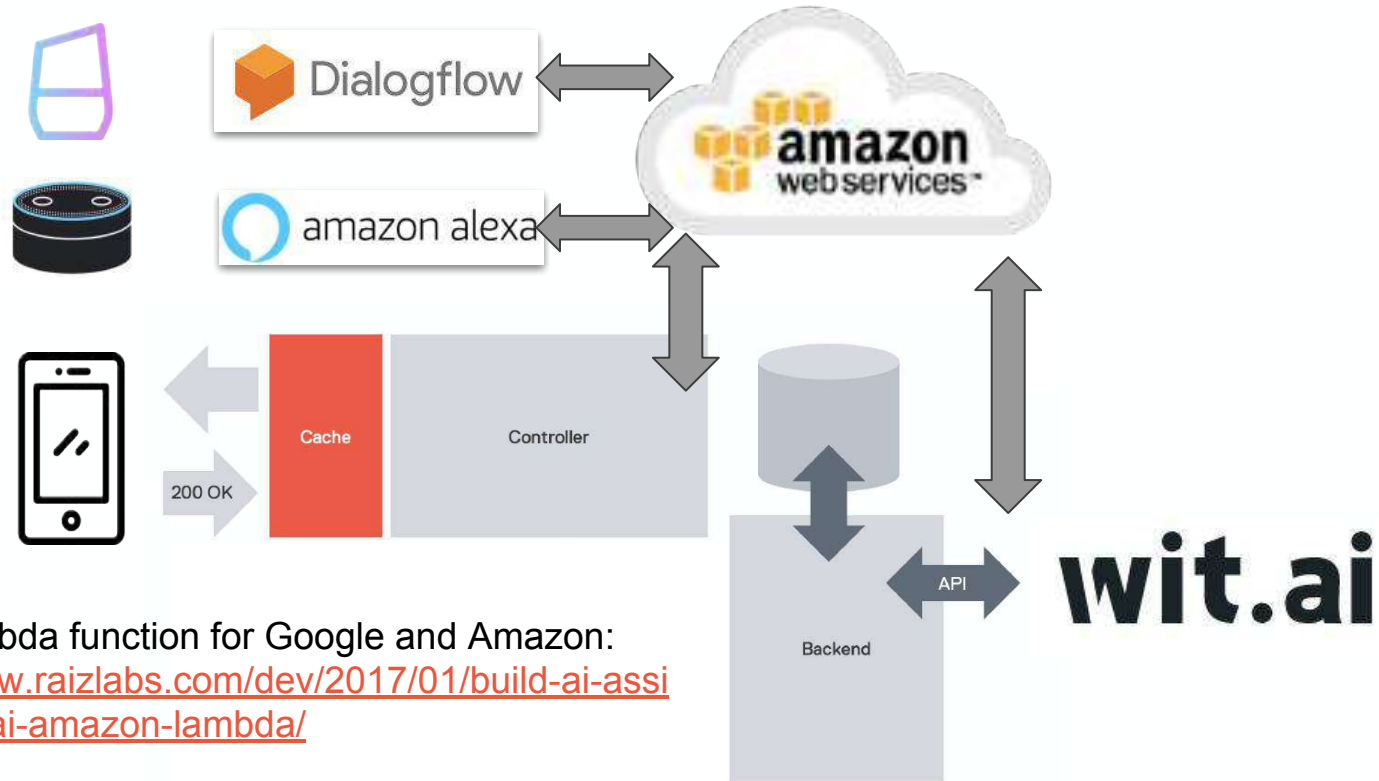
Lambda: <https://gist.github.com/fciacchi/4415a55b3dbc5eeede3bdccf76fc663e>

# A simple Alexa skill dialog





# Design to a new level



Build Lambda function for Google and Amazon:  
<https://www.raizlabs.com/dev/2017/01/build-ai-assistant-api-ai-amazon-lambda/>

# API Design





# Use HTTP verbs



- There are many HTTP verbs and you can apply them on both collections or elements. Make it simple and stick to it.

	Safe	Idempotent	Possible on	Success Response
GET	Yes	Yes	Collections Elements	200
POST	No	No	Collections*	201 + Location
PUT	No	Yes	Elements**	204
DELETE	No	Yes***	Elements**	204/404***

- HTTP verbs should be mapped 1-to-1 to CRUD operations. (but use encoded IDs/structure)

Operation	SQL	HTTP
Create	INSERT	POST
Read (Retrieve)	SELECT	GET
Update (Modify)	UPDATE	PUT
Delete (Destroy)	DELETE	DELETE

# Use HTTP response codes



- There are many HTTP response codes, use it!
  - 200 (GET)
  - 201 + Location (POST)
  - 204 (PUT/DELETE)
  - 301 Moved Permanently
  - 404 Resource not found (/customers/5 -> does not exist)
  - 405 Method not allowed (should tell which verbs are allowed)
  - 409 Conflict (try to insert twice the same resource)
  - 415 Unsupported media type
  - 422 Unprocessable entity (wrong payload POST)
  - 500 Managed Exceptions

# DDD Approach



- Use the 'Domain' definition as in the Domain Driven Design to define your API:
  - Customer is one domain
  - Wishlist is one domain
  - Cart is one domain

carts		▼
GET	/carts/{cartId} Retrieve a cart.	🔒 ↩
DELETE	/carts/{cartId} Delete cart by id.	🔒 ↩
GET	/carts Retrieve list of all customers carts	🔒 ↩
POST	/carts Create cart.	🔒 ↩

items		▼
POST	/carts/{cartId}/items Adds an item to the cart.	🔒 ↩
PATCH	/carts/{cartId}/items/{itemId} Update cart item quantity.	🔒 ↩
DELETE	/carts/{cartId}/items/{itemId} Remove item from the cart.	🔒 ↩

Interesting Reading:

<https://herbertograca.com/2017/11/16/explicit-architecture-01-ddd-hexagonal-onion-clean-cqrs-how-i-put-it-all-together/>

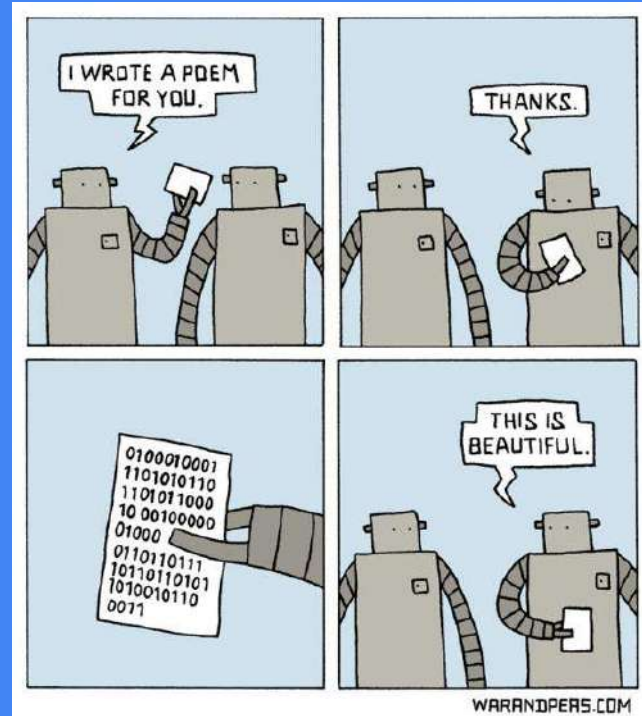
## Why design good API?

30%

Saved in Wrong API Calls

- JSON API:  
<http://jsonapi.org/>
- JWT token:  
<https://jwt.io/>
- REST API Refactor:  
<https://www.slideshare.net/ciacchi/rest-api-a-real-case-scenario>
- Build REST API with Symfony:  
<https://williamdurand.fr/2012/08/02/rest-api-with-symfony2-the-right-way/>

# Design Tips



# Design tips

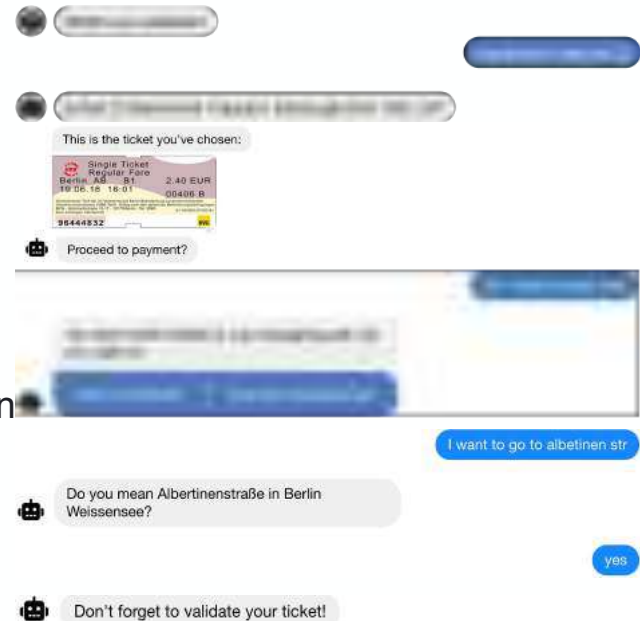


Ulrike Eisengräber @ [Rasa.com](https://Rasa.com)

<http://blog.rasa.com/13-rules-for-cui-design/>

- Rule n.3: Offer Informative Feedback.
- Rule 12: Make Use Of Familiar Concepts.
- Rule 8: Reduce Short-term Memory Load.
- Rule 5: Prevent Errors & Rule 11: Have A Fallback Option
- Rule 4: Design Dialogs To Provide Closure
- But you can also try new things...

When is the next train? 🇮🇹  
What time is my appointment? 🐾



# VisualBots



Welcome

Question 1

Question 2

Question 3

...

Question 9

DROP

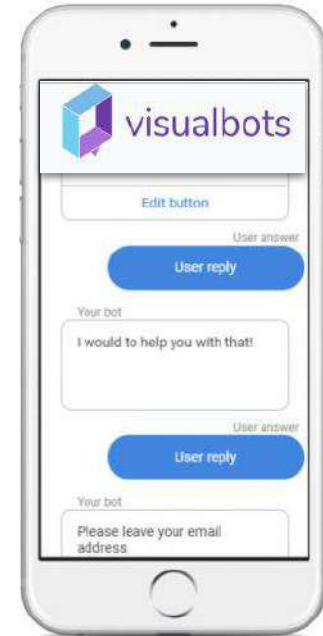
-37%

-4%

0%

0%

0%



[Livio Marcheschi](#)

<https://www.visualbots.io/>

# Latest News



## New Alexa Devices



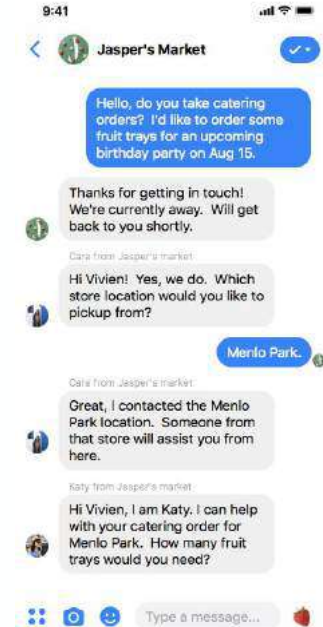
<https://techcrunch.com/2018/09/20/the-long-list-of-new-alexa-devices-amazon-announced-at-its-hardware-event/>

## Portal FB/Alexa



<http://bit.ly/fb-portal-alexa>

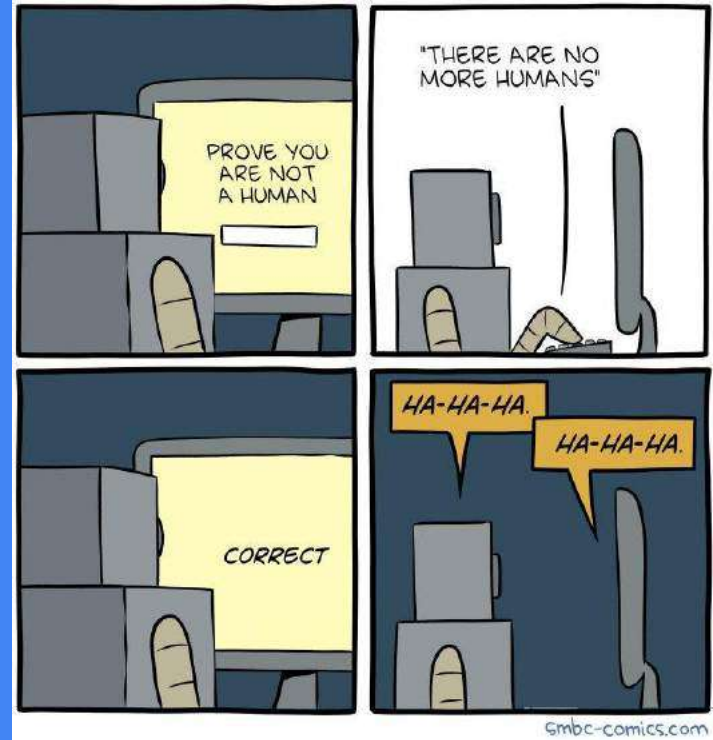
## FB Mess. Personas



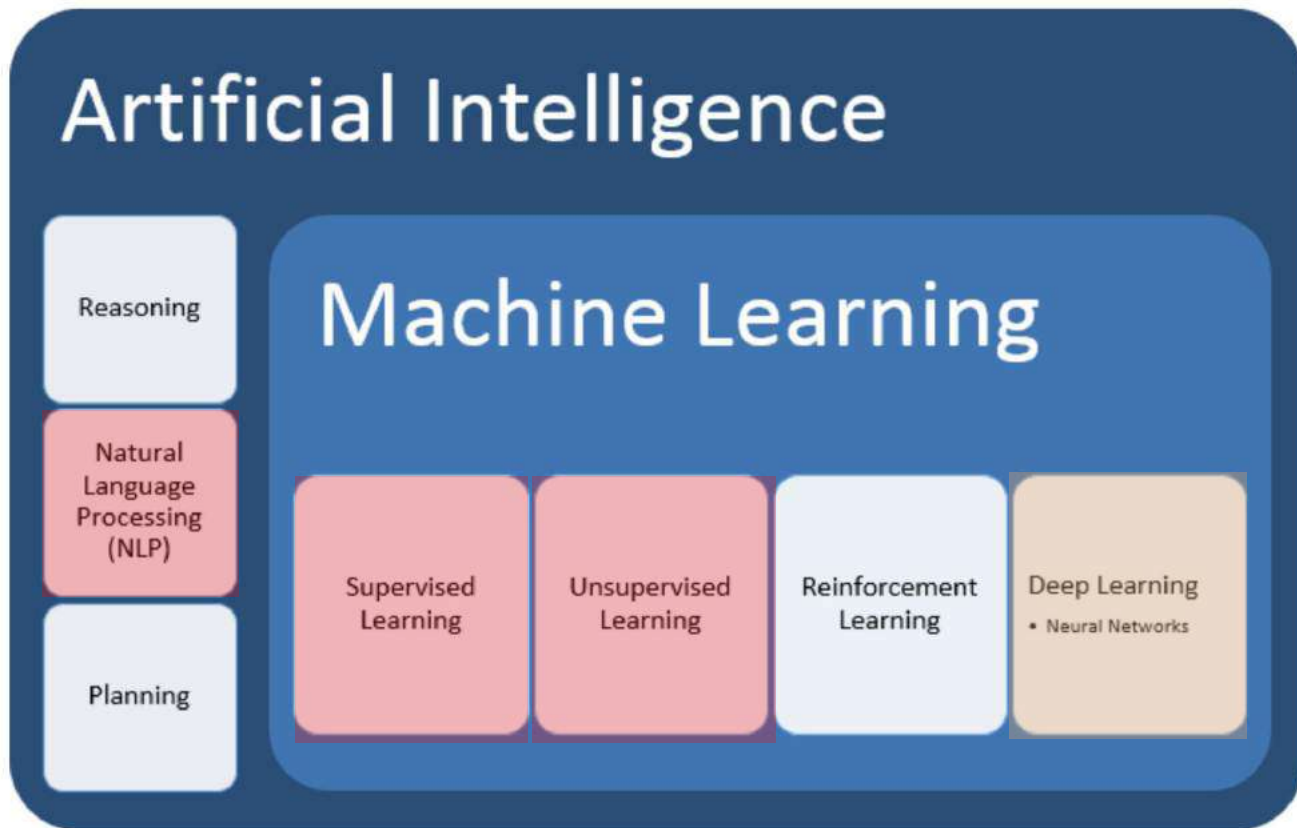
<https://developers.facebook.com/docs/messenger-platform/send-messages/personas/>



# Machine Learning



# Machine Learning

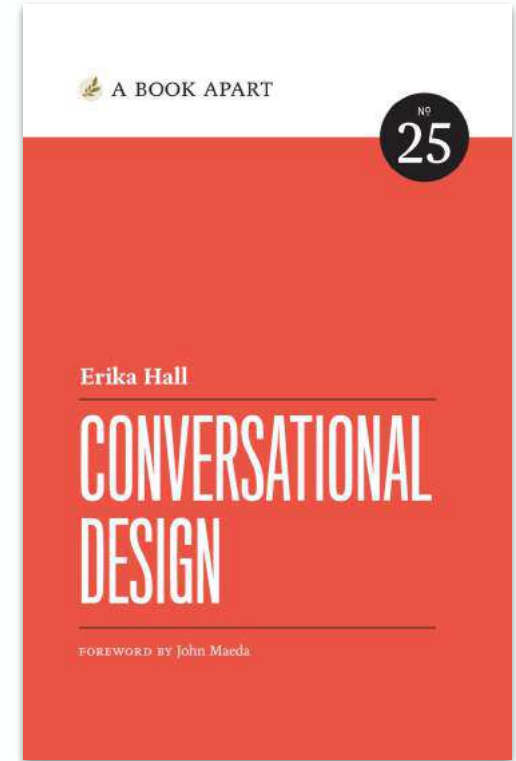
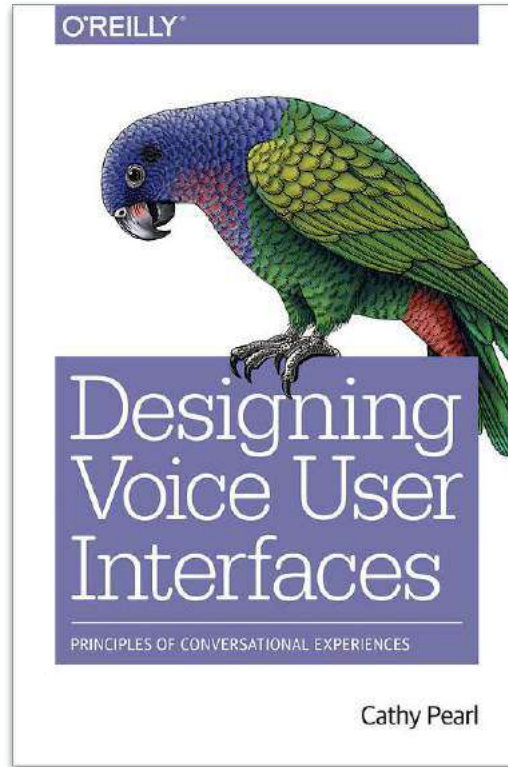
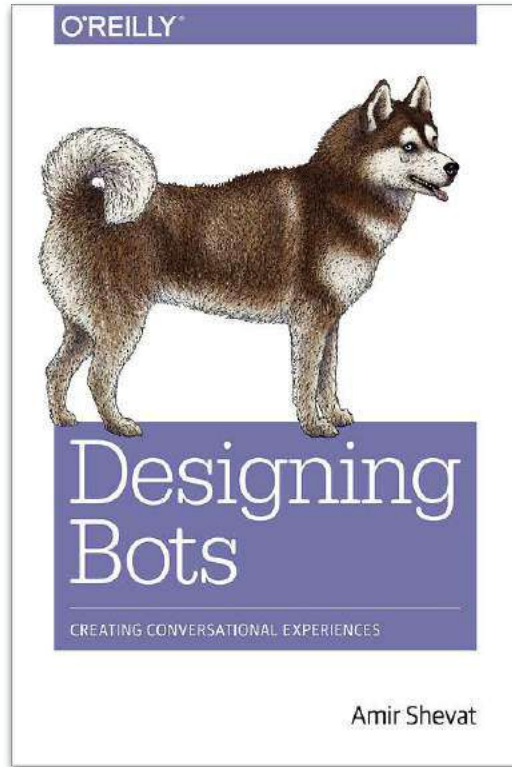


# Links to Learn



- **Facebook Developer Circle**  
<https://www.facebook.com/groups/DevCBerlin/>
- **Session 1: Intro to Machine Learning**  
Wen-Ru Sheu / Backend Engineer @Visual Meta GmbH  
<http://bit.ly/ml-session-1>  
<https://tinyurl.com/machinelearning-intro-1>
- **Session 2: Neural Networks**  
Christopher Lennan / Senior Data Scientist @Idealo.de  
<http://bit.ly/ml-session-2>  
<https://tinyurl.com/machinelearning-intro-2>
- **Session 3: Natural Language Processing** (Lennan + Sheu)  
<http://bit.ly/ml-session-3>  
<https://tinyurl.com/machinelearning-intro-3>
- **ML 101:**  
<http://bit.ly/machine-learning-101>
- **Build Intelligent Bots in Python:**  
<https://speakerdeck.com/kprzystalski/building-intelligent-bots-in-python>
- **Start With NLP:**  
<https://towardsdatascience.com/an-easy-introduction-to-natural-language-processing-b1e2801291c1>

# Books



– <https://oreil.ly/2pDdHYH>

<https://oreil.ly/2G9ZR7d>

<http://bit.ly/2G7Ggo4>

# Some Examples



<https://youtu.be/elqsMF2Bd4k>



<https://youtu.be/WLu6K8w8P1Q>



[https://youtu.be/O5Jz\\_67iN30](https://youtu.be/O5Jz_67iN30)

# Beyond Web Interfaces

Fabrizio Ciacchi



Thanks



<http://ciacchi.it/CodeteCon-v4-2018-Ciacchi-Bot.pdf>

[spryker.com](http://spryker.com) / [ciacchi.it](http://ciacchi.it)

[fabrizio.ciacchi@gmail.com](mailto:fabrizio.ciacchi@gmail.com)

twitter [@sprysys](https://twitter.com/sprysys) [@ciacchi](https://twitter.com/ciacchi)

Icons designed by [Freepik](#) from [Flaticon](#)