# Navarra LAN Party
# Desarrollo de Alexa Skills

Augusto Mauch
Septiembre - 2022

# Agenda

- Introducción al taller
- Introducción teórica
  - Aplicaciones de voz
  - Tecnología
  - Alexa
- Desarrollo de una aplicación de voz para obtener información sobre películas
- Kahoot

# ¿Quién soy?

- Augusto Mauch Goya
- Ingeniería en Informática (UPNA)
- Team Lead en Openbravo
- Profesor de algoritmia en la UPNA
- Dirección de correo: augusto.mauch@openbravo.com

# Sobre Openbravo

**115+**
RETAILERS

**50+**
COUNTRIES

**60,000+**
POS

**18,000+**
Back office Users

**THE BUSINESS**
Software and integration for the retail sector

**THE STRATEGY**
Helping retailers to successfully implement their omnichannel strategies

**POS and retail management**

**A highly flexible & mobile technology platform**

**A dedicated and reliable cloud infrastructure**


https://greatplacetowork.es/openbravo

# Desarrollo de la sesión

- Vamos a hacer *live coding* para desarrollar una aplicación de voz utilizando el framework de Alexa
- Todo el código de la sesión está disponible en este repositorio: https://github.com/AugustoMauch/openbravo-movies-skill
- Todos los pasos que voy a realizar durante la sesión están escritos en detalle en esta presentación
- Al final de la sesión podéis participar en un Kahoot con preguntas sobre lo explicado. ¡La persona que obtenga mayor puntuación ganará un Alexa Echo Dot!

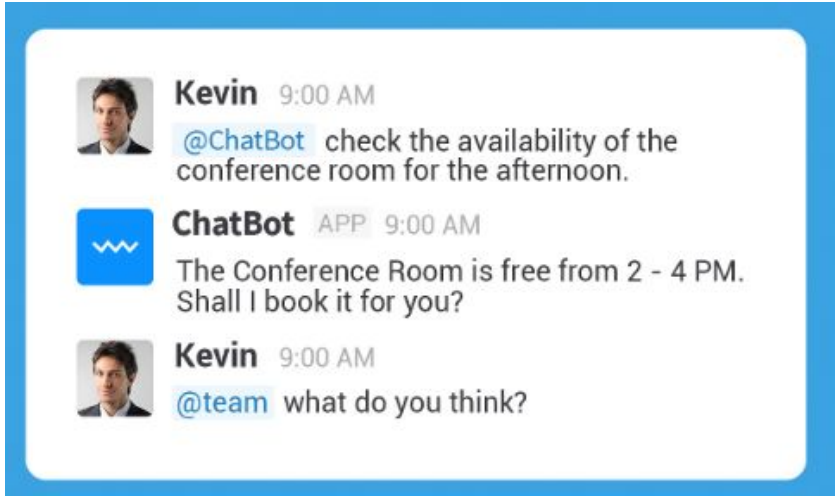
# Requisitos desarrollar Amazon Skills

- Portátil con conexión a internet
- Conocimientos de programación básicos en Javascript o Python
- Cuenta de Amazon Developer
    - https://developer.amazon.com/es-ES/docs/alexa/ask-overviews/create-developer-account.html
    - Si no la tenéis os costará 5 minutos crearla
- Para la demo basta con los recursos que proporciona la tier gratuíta

# Introducción a las aplicaciones de voz

# Introducción a las aplicaciones de voz

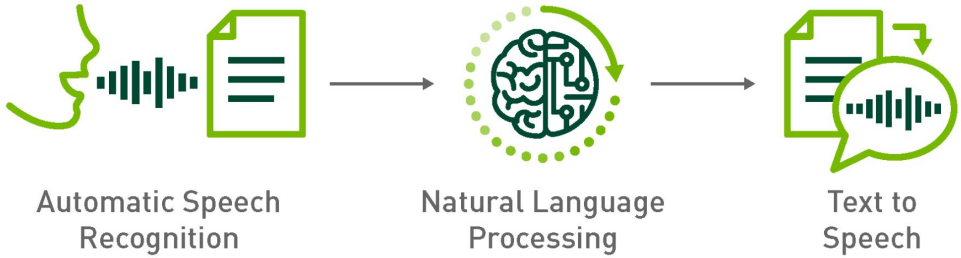
- Asistentes virtuales, evolución natural de los chatbots




Fuente: https://slack.com/apps/A7FTEHPEG-chatbot

- Voz en lugar de texto: Mayor accesibilidad al no requerir ni un teclado ni una pantalla

# Tecnologías sobre las que se fundamentan




Fuente: https://developer-blogs.nvidia.com/wp-content/uploads/2019/12/thumb.jpg

- **Automatic Speech Recognition (ASR)**: uso de IA/ML para convertir la voz humana a texto
- **Natural Language Processing (NLP):** hace un análisis sintáctico del texto para extraer información (esto es un verbo, esto es un color, esto es una fecha)
- **Text to speech (TTS):** transforma texto en voz (acentos, inflexiones de la voz, etc)

# Alexa - Historia

Amazon compra Ivona en 2013, saca la primera versión de Alexa en 2014

Inspirado en 2001: *Odisea en el espacio* y en *Star Trek*

Integración con su plataforma de servicios en la nube (AWS)

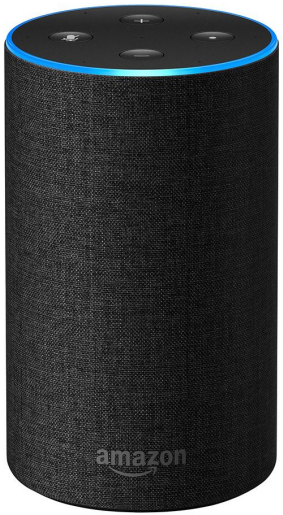
Inicialmente solo soporte en pocos idiomas, ahora disponible en 9 (Árabe, Inglés, Español, Alemán, Francés, Hindi, Japonés, Portugués)

Conferencias anuales: Alexa Live

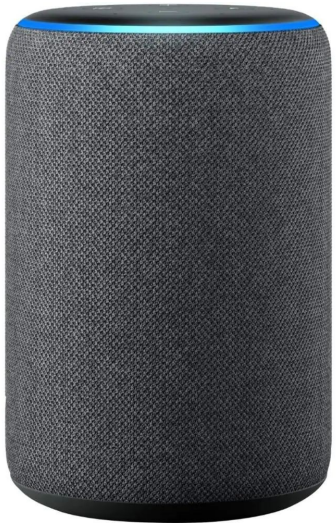
# Alexa - dispositivos

## Amazon Echo


2014




2016



2019

2020

# Alexa - dispositivos

Amazon Echo Dot

¡Kahoot winner!


2016


2016

2018

2020



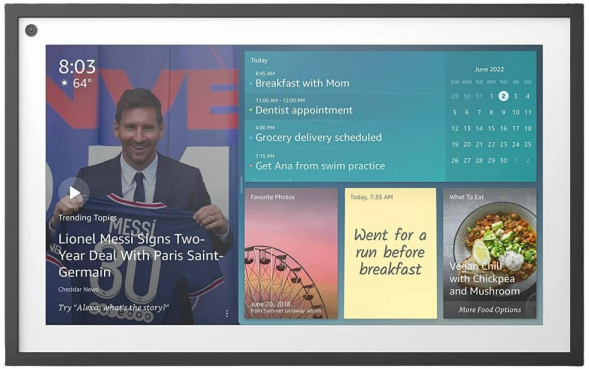
openbravo

# Alexa - dispositivos

## Amazon Echo Show


1st gen
(2015)


Echo Show 8
(2021)




Echo Show 15
(2021)

# Alexa - dispositivos

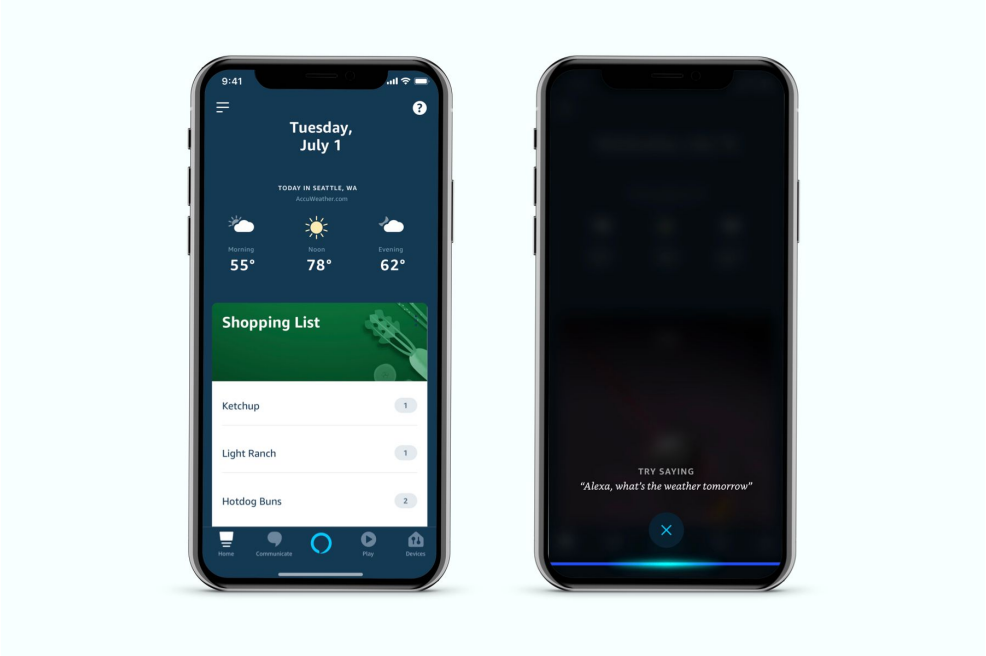
## Otros

Amazon Astro
(2021)


Echo Auto
(Alexa App)

# Alexa - dispositivos

## Otros

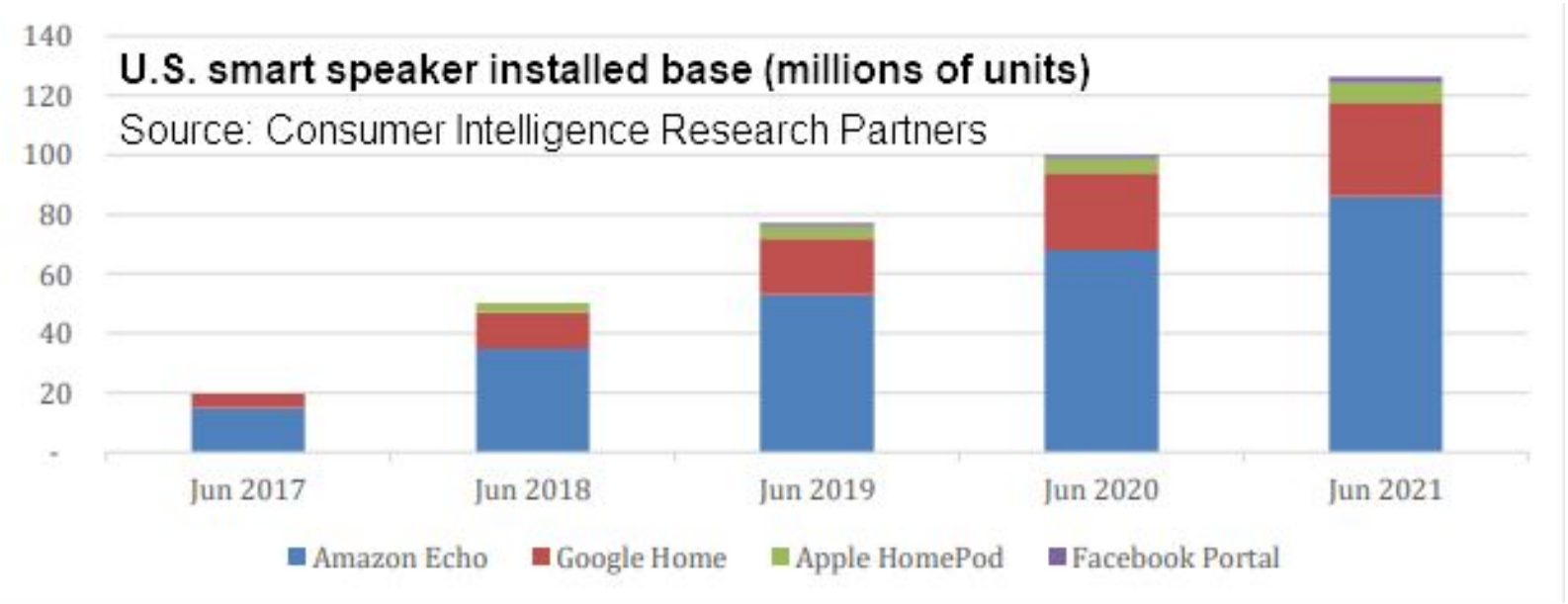



Dispositivos móviles
(Alexa App)



openbravo

# Alexa - predominancia en el mercado




https://www.geekwire.com/2021/amazon-maintains-big-lead-google-apple-u-s-smart-speaker-market-new-study-says/

# Desarrollo de aplicaciones con Amazon Alexa (Alexa Skills)

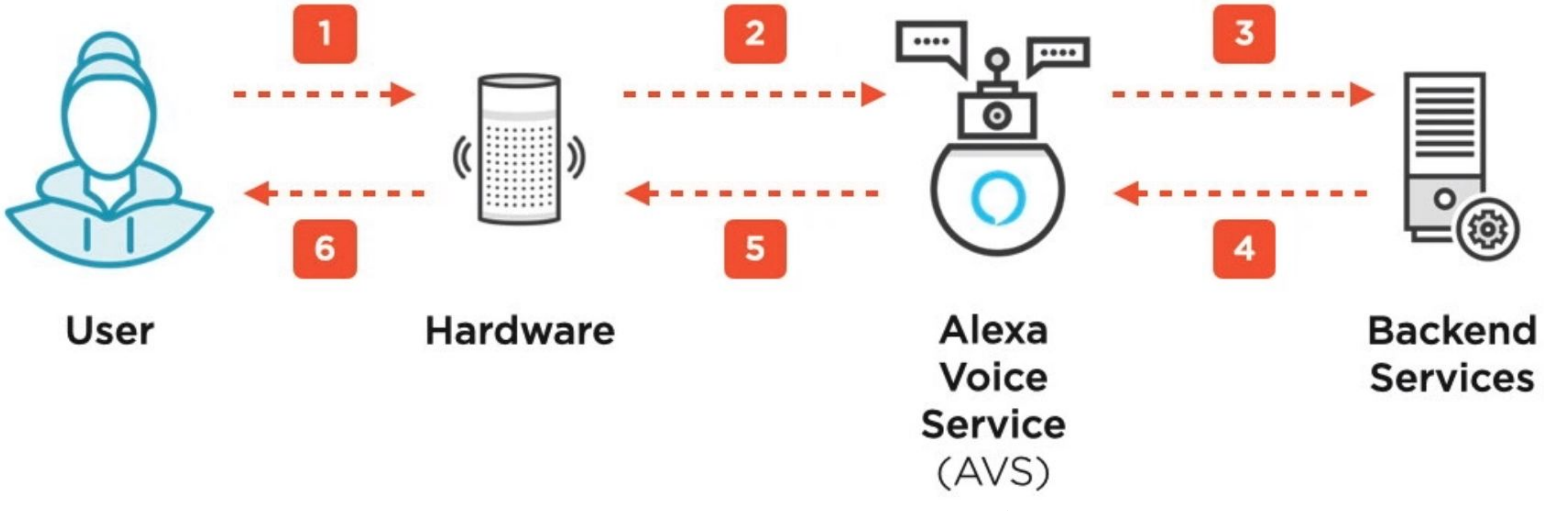
# Diseño de aplicaciones de voz - Frontend / Backend

- **Frontend:** parte de la aplicación con la que interactúa directamente el usuario. Genera peticiones que son enviadas al backend
- **Backend:** recibe peticiones del frontend, las procesa, genera una respuesta que es reenviada al frontend

En Alexa Skills:

- **Frontend:** escucha las **órdenes de voz** del cliente, utiliza el **Modelo de Interacción** para generar peticiones, las envía al backend
- **Backend:** recibe peticiones del frontend, las procesa, genera una respuesta que es reenviada al frontend **para que sea narrada**

# Alexa Skills Overview




- Automatic Speech Recognition (ASR)
- Natural Language Understanding (NLU)
- Text To Speech (TTS)

# Herramientas proporcionadas por Amazon

- **Skill Management API (SMAPI)**: API REST que permite crear una skill, actualizar su modelo de interacción y probarlo
- **SMAPI SDK**: Software development kit que facilita trabajar con el SMAPI (disponible para Node.js, Python y Java)
- **Alexa Skills Kit Command Line Interface (ASK CLI)**: Permite interactuar con el SMAPI desde el terminal

# Conceptos de Alexa

| | | |
|---|---|---|
| **Invocation Name** | **Utterance** | **Intent** |
| **Slot** | **Slot Type** | **Interaction Model** |

# Ejemplo real: Informativo de Ángel Martín

Link a skill en Amazon Marketplace, creada por Joaquín Engelmo (Kinisoftware)

**Alexa,** abre **el informativo de Ángel Martín** y **pon el resumen de ayer**

**Alexa,** abre **el informativo de Ángel Martín** y **pon lo del martes**

**Alexa,** pídele a **el informativo de Ángel Martín que ponga el resumen del pasado lunes**

**Alexa,** abre **el informativo de Ángel Martín** y **pon lo del 20 de Octubre**

**Wake word - Hace que Alexa empiece a escuchar**

**Invocation name - Determina la Alexa Skill que se va a invocar**

**Utterance - Determina la acción que se va a ejecutar en la skill**

**Slot value - Parámetros**

# Creación de una aplicación - Bravo Movies

- Intent:
  - información sobre películas

**Alexa**, pregunta a **Bravo Movies** cuáles son las películas más populares

**Alexa**, pregunta a **Bravo Movies** cuáles son las películas más populares en **2001**

**Alexa**, pregunta a **Bravo Movies** cuáles fueron las películas **de comedia** más populares en **1984**

**Wake word - Hace que Alexa empiece a escuchar**

**Invocation name - Determina la Alexa Skill que se va a invocar**

**Utterance - Determina la acción que se va a ejecutar en la skill**

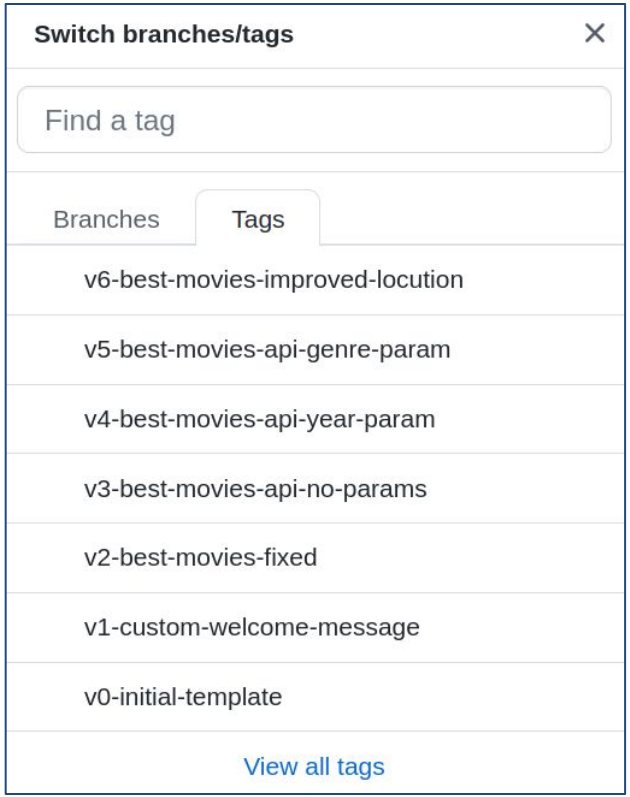
**Slot value - Parámetros**

# Creación de una aplicación

- Abrir https://developer.amazon.com/alexa/console/ask
- Hacer click en Crear Skill
  - **Skill Name:** Bravo Movies
  - **Primary Locale:** Spanish (ES)
  - **Model:** Custom
  - **Hosting:** Alexa Hosted (Node.js)
  - Hacer click en Crear Skill para continuar
- **Template:** Start from Scratch
- Click en Continue with Template (le cuesta ~1 minuto)

# Fases de la demo

0. Template inicial sin cambios
1. Modificar el mensaje de bienvenida
2. Crear nuevo Intent: BestMoviesIntent
3. Uso de la API sin parámetros
4. Añadir parámetro de fecha de estreno
5. Añadir parámetro de tipo de película
6. Mejorar la locución




En el repo hay un tag para cada fase, i.e.

https://github.com/AugustoMauch/openbravo-movies-skill/blob/v4-best-movies-api-year-param/lambda/index.js



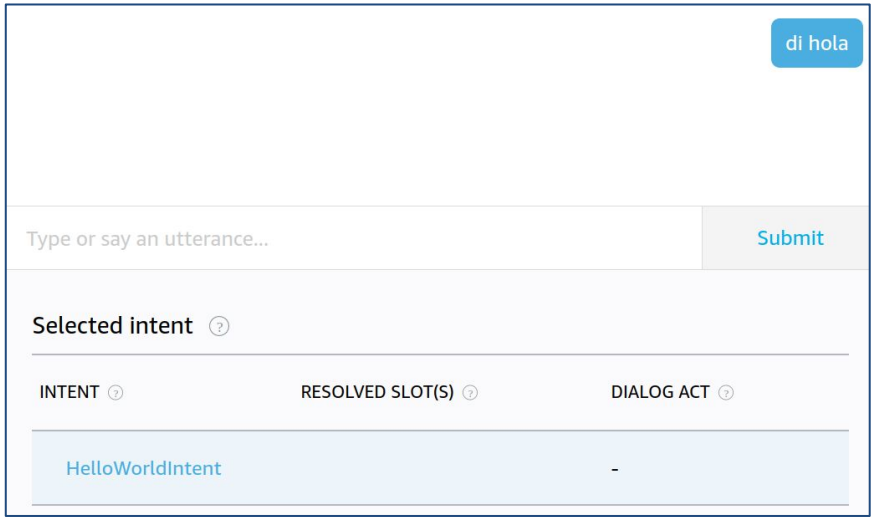
openbravo

# Creación de una aplicación

- Vamos a trabajar principalmente con tres de los tabs: Build, Code y Test
- Tab de BUILD. Lo usaremos para:
  - Editar el Interaction Model, tiene 5 intents predefinidos:
    - CancelIntent
    - HelpIntent
    - StopIntent
    - NavigateHomeIntent
    - HelloWorldIntent (revisar las 5 utterances definidas)
  - Probar el Interaction Model, comprobar que las requests se van a generar como esperamos
  - Definir los slots:
    - {year}: Amazon.FOUR_DIGIT_NUMBER
    - {genre}: MovieGenre (custom)

# Creación de una aplicación

- Tab de BUILD. Evaluate Model:
  - Lo usaremos para comprobar qué requests se generarían a partir de lo que el usuario haya pedido a la aplicación
  - Siempre devolverá un INTENT, opcionalmente incluirá información sobre los slots






openbravo

# Creación de una aplicación

Tab de CODE

- Implementación del backend en Node.js
- Ficheros principales:
  - package.json: gestión de dependencias
  - index.js: implementación del backend de la skill
- index.js -> Definición y registro de skill handlers
  - Definición: por cada skill hay que incluir dos funciones:
    - canHandle(handlerInput): dada una request, determina si este handler la tiene que procesar
    - handle(handlerInput): procesamiento de la request, determinar qué hay respuesta hay que dar al usuario en base a los parámetros de la request

# Creación de una aplicación

Tab de CODE - ejemplo de definición de un intent handler

```
const HelloWorldIntentHandler = {
    canHandle(handlerInput) {
        return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
            && Alexa.getIntentName(handlerInput.requestEnvelope) === 'HelloWorldIntent';
    },
    handle(handlerInput) {
        const speakOutput = 'Hello World!';

        return handlerInput.responseBuilder
            .speak(speakOutput)
            .getResponse();
    }
};
```

# Creación de una aplicación

Tab de CODE

- DynamoDB: base de datos no relacional autoaprovisionada
- S3: almacenamiento en disco autoprovisionado
- Cloudwatch: acceso al log de la aplicación
- Utilidades para descargar e importar el código

- Importante: Hay que guardar (SAVE) y desplegar (DEPLOY) para poder probarlo en el tab de TEST

# Creación de una aplicación

Tab de TEST


- Habilitar el test de la aplicación
- Lo utilizaremos para ver qué response devuelve nuestro backend a partir de un ejemplo de locución
- Ejemplos:
  - “Abre Bravo Movies”
  - “Pide a Bravo Movies que diga hola”
- JSON request
  - version, session, context, request (type, intent name)
- JSON response
  - body (version, response)
  - sessionAttributes

# Creación de una aplicación

Tab de TEST - Información de debug

- JSON request
  - version, session, context, **request (type, intent name)**


```
},
"request": {
    "type": "IntentRequest",
    "requestId": "amzn1.echo-api.request.3974b0a7-defa-422f-897b-1cd3ef62c8e8",
    "locale": "es-ES",
    "timestamp": "2022-08-28T16:24:07Z",
    "intent": {
        "name": "HelloWorldIntent",
        "confirmationStatus": "NONE"
    }
}
```

# Creación de una aplicación

Tab de TEST - Información de debug

- JSON response
  - body (version, response - outputSpeech)
  - sessionAttributes

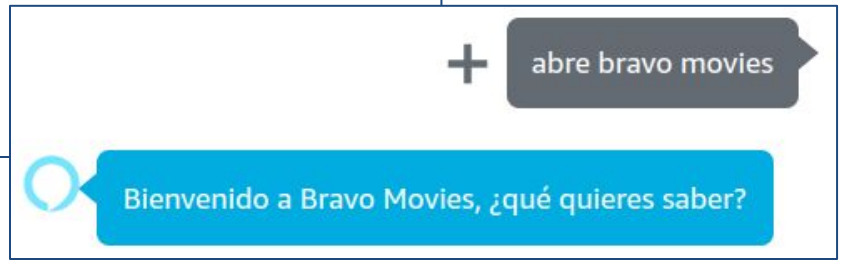

```
{
    "body": {
        "version": "1.0",
        "response": {
            "outputSpeech": {
                "type": "SSML",
                "ssml": "<speak>Hello World!</speak>"
            },
            "type": "_DEFAULT_RESPONSE"
        },
        "sessionAttributes": {},
        "userAgent": "ask-node/2.12.1 Node/v12.22.11 sample/hello-world/v1.2"
    }
}
```

# Creación de una aplicación

**v1:** Actualizar el texto que se narrará al abrir la aplicación

- Modificar la función handle del LaunchRequestHandler para que devuelva el nuevo texto.
- Importante: guardar y desplegar antes de probarlo

```
handle(handlerInput) {
    const speakOutput = 'Bienvenido a Bravo Movies, ¿qué quieres saber?';

    return handlerInput.responseBuilder
        .speak(speakOutput)
        .reprompt(speakOutput)
        .getResponse();
}
```



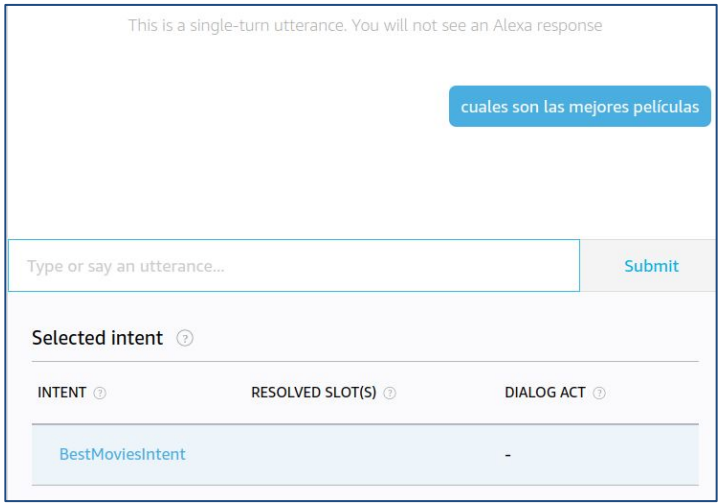



openbravo

# Creación de una aplicación

**v2:** Borrar HelloWorldIntent, crear BestMoviesIntent básico

- La primera versión devolverá un mensaje fijo
- Definición del Intent en el Interaction Model:


Evaluate Model



openbravo

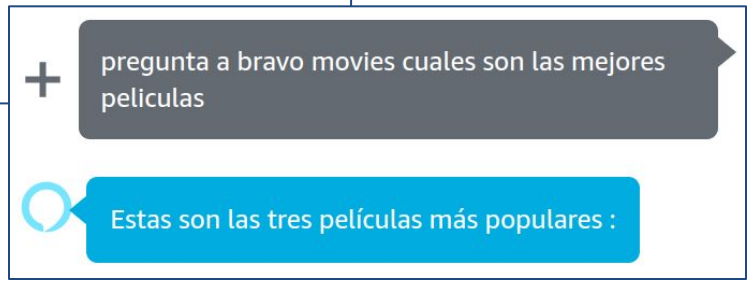
# Creación de una aplicación

**v2:** Borrar HelloWorldIntent, crear BestMoviesIntent básico

- La primera versión devolverá un mensaje fijo
- Implementación del intent handler en el backend:

```
const BestMoviesIntentHandler = {
    canHandle(handlerInput) {
        return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
            && Alexa.getIntentName(handlerInput.requestEnvelope) === 'BestMoviesIntent';
    },
    handle(handlerInput) {
        const speakOutput = 'Estas son las tres películas más populares:';

        return handlerInput.responseBuilder
            .speak(speakOutput)
            .getResponse();
    }
};
```

```
exports.handler = Alexa.SkillBuilders.custom()
    .addRequestHandlers(
        BestMoviesIntentHandler,
```






openbravo

# Información sobre las películas: themoviedb.org

- La información sobre las películas la vamos a obtener de themoviedb.org
- API pública, disponible solicitando una API Key siguiendo estos pasos
- API con endpoints que devuelven información sobre películas y series
- Endpoint para nuestra aplicación: listado de películas en orden de popularidad descendente, en un año determinado, de un género determinado

```
https://api.themoviedb.org/3/discover/movie
        ?sort_by=popularity.desc
        &primary_release_year=<year>
        &with_genres=<genreId>
        &api_key=<apiKey>
```

# Información sobre las películas: themoviedb.org

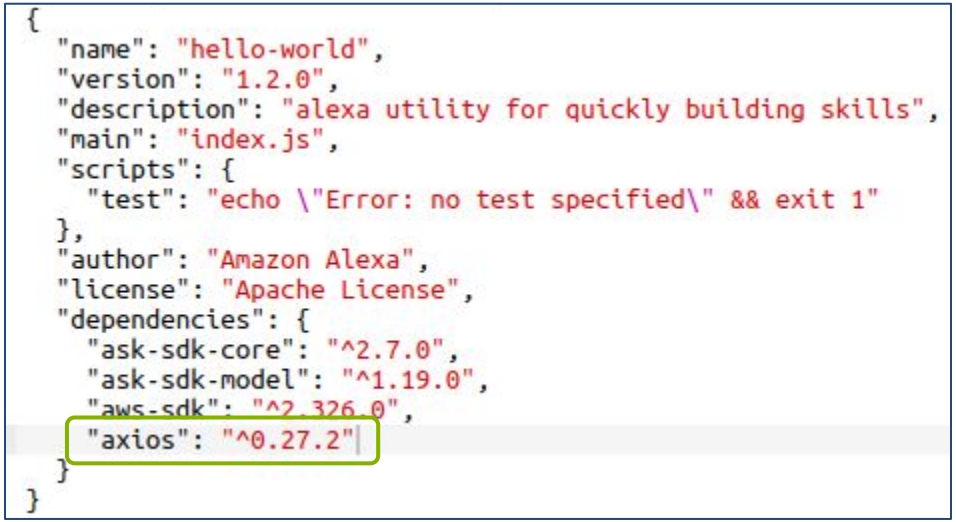
- Estructura de la respuesta. Para nuestra respuesta nos interesa el campo que contiene el título en inglés: “title”


```
{
    "page": 1,
    "results": [
        {
            "adult": false,
            "backdrop_path": "/g9ROrZey0JYp7kf8DoAyZtKnObj.jpg",
            "genre_ids": [
                27
            ],
            "id": 377,
            "original_language": "en",
            "original_title": "A Nightmare on Elm Street",
            "overview": "Teenagers in a small town are dropping like flies...",
            "popularity": 57.098,
            "poster_path": "/wGTpGGRMZmyFCcrY2YoxVTIBlli.jpg",
            "release_date": "1984-11-09",
            "title": "A Nightmare on Elm Street",
            "video": false,
            "vote_average": 7.3,
            "vote_count": 4094
        },
        ...
    ],
    "total_pages": 227,
    "total_results": 4529
}
```

# Creación de una aplicación

**v3:** Obtener listado de películas utilizando la API, sin filtros de fecha ni género

- Vamos a instalar axios como dependencia para facilitar comunicación con API. En el fichero package.json:

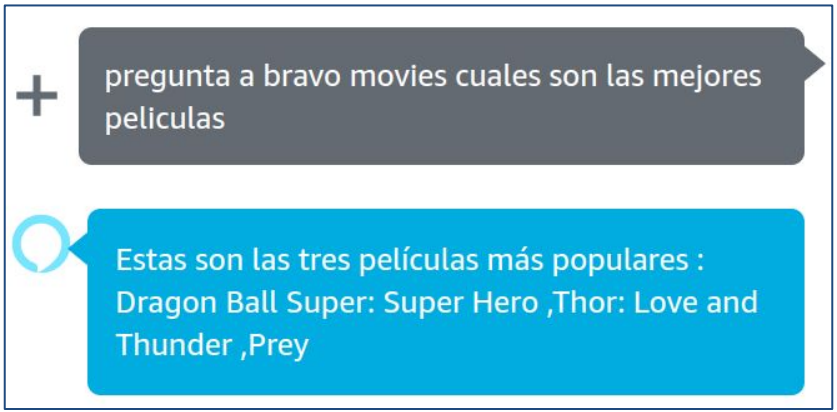


```
{
  "name": "hello-world",
  "version": "1.2.0",
  "description": "alexa utility for quickly building skills",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Amazon Alexa",
  "license": "Apache License",
  "dependencies": {
    "ask-sdk-core": "^2.7.0",
    "ask-sdk-model": "^1.19.0",
    "aws-sdk": "^2.326.0",
    "axios": "^0.27.2"
  }
}
```

# Creación de una aplicación

**v3:** Obtener listado de películas utilizando la API, sin filtros de fecha ni género

```
const response = await axios(
        `https://api.themoviedb.org/3/discover/movie?sort_by=popularity.desc&api_key=${API_KEY}`
        );
const filmTitles = response.data.results.slice(0, 3)
                                        .map(result => result.title)
                                        .join(',');
const speakOutput = `Estas son las tres películas más populares: ${filmTitles}`;
```



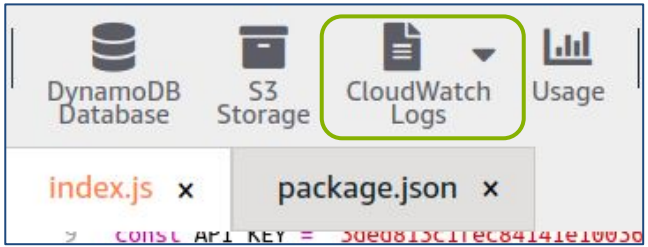



openbravo

# Creación de una aplicación

**Uso de logs para depurar:** lo que escribamos por la consola - console.log() - se guarda en el log de CloudWatch. Ejemplo:

```
const response = await axios(
        `https://api.themoviedb.org/3
console.log(response.data);
```




| Log stream | Last event time |
| --- | --- |
| 2022/08/28/[7]5da12612ce194244bfb5b6485ba12b7e | 2022-08-28 20:02:37 (UTC+02:00) |
| 2022/08/28/[6]2aa3f0e343f34be0bfdfd4004973e6e3 | 2022-08-28 19:40:06 (UTC+02:00) |

# Creación de una aplicación

**Uso de logs para depurar:** lo que escribamos por la consola - console.log() - se guarda en el log de CloudWatch. Ejemplo:

```
const response = await axios(
        `https://api.themoviedb.org/3
console.log(response.data);
```

```
2022-08-28T18:02:37.075Z        0597e948-8e4d-4125-a9bb-463901c70742    INFO    {
  page: 1,
  results: [
    {
      adult: false,
      backdrop_path: '/vvObT0eIWGlArLQx3K5wZ0uT812.jpg',
      genre_ids: [Array],
      id: 616037,
      original_language: 'en',
      original_title: 'Thor: Love and Thunder',
      overview: 'After his retirement is interrupted by Gorr the God Butcher, a ga
girlfriend Jane Foster, who now inexplicably wields Mjolnir as the Relatively Migh
vengeance and stop him before it's too late.',
      popularity: 7036.373,
      poster_path: '/pIkRyD18kl4FhoCNQuWxWu5cBLM.jpg',
      release_date: '2022-07-06',
      title: 'Thor: Love and Thunder',
      video: false,
      vote_average: 6.8,
      vote_count: 2067
    },
```

# Creación de una aplicación


Hay maneras de debugar mejores que usando console.log, ver:
https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-cli-using-debugging.html

# Creación de una aplicación

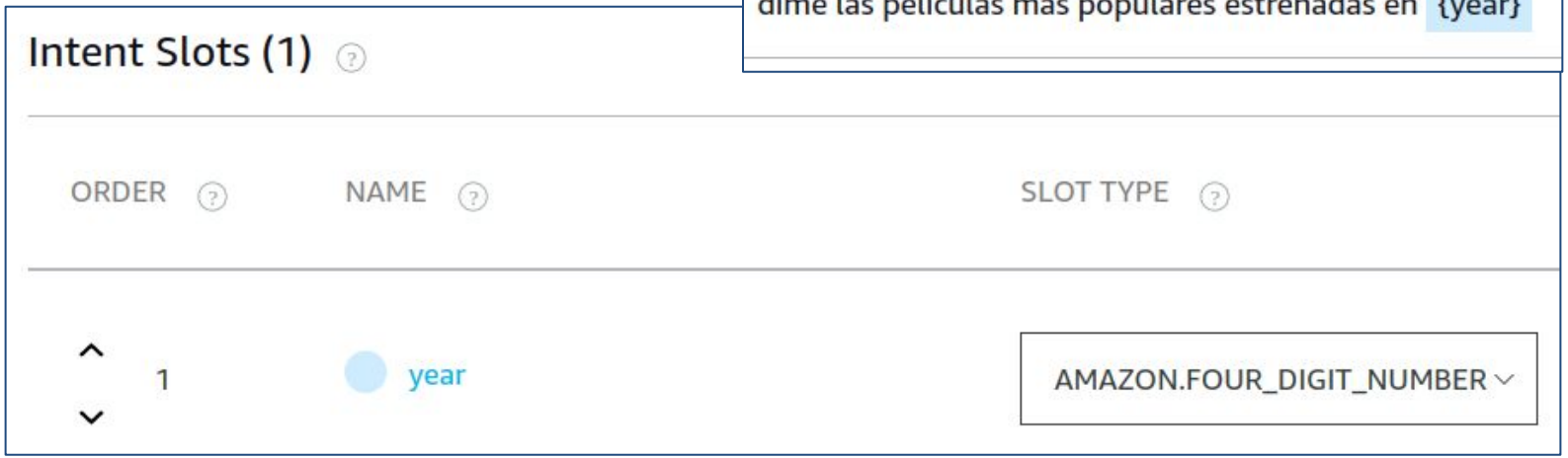
**v4:** filtro opcional de fechas - Evaluation Model

- Actualizar utterances para incluir un nuevo slot - {year}. Podemos utilizar un tipo predefinido (Amazon.FOUR_DIGIT_NUMBER)

devuelve las películas de más éxito de {year}

dame la lista de las mejores películas del año {year}

dime las películas más populares estrenadas en {year}



# Creación de una aplicación

**v4:** filtro opcional de fechas

- Podemos usar la función Evaluate Model para comprobar que la información del slot se mandará correctamente al backend:

# Creación de una aplicación

**v4:** filtro opcional de fechas - tab CODE

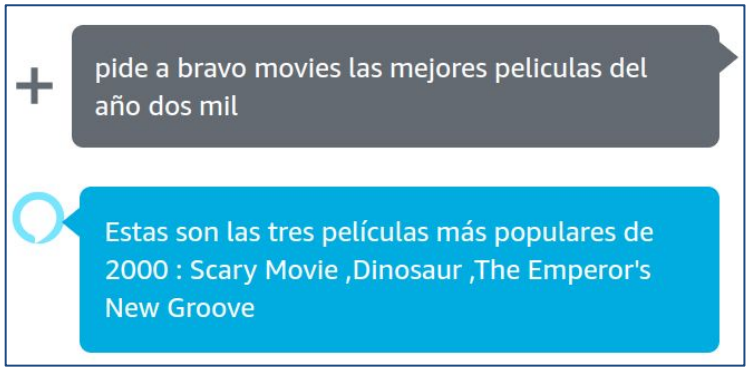
- Obtener el valor del parámetro usando la función Alexa.getSlotValue
- Usar el valor para incluir un parámetro en la request de la API de películas (&primary_release_year) y para incluir el año en la locución
- Importante: controlar el caso en el que el usuario no haya incluido un año

```
const yearValue = Alexa.getSlotValue(handlerInput.requestEnvelope, 'year');
const yearParam = yearValue ? `&primary_release_year=${yearValue}` : '';
const response = await axios(
        `https://api.themoviedb.org/3/discover/movie?sort_by=popularity.desc&api_key=${API_KEY}${yearParam}`);

const filmTitles = response.data.results.slice(0, 3).map(result => result.title).join(',');

const yearOutput = yearValue ? ` estrenadas en ${yearValue}` : '';
const speakOutput = `Estas son las tres películas más populares ${yearOutput}: ${filmTitles}`;
```

# Creación de una aplicación

**v4:** filtro opcional de fechas - Probar en tab de TEST


```
"request": {
    "type": "IntentRequest",
    "requestId": "amzn1.echo-api.request.b6477abb
    "locale": "es-ES",
    "timestamp": "2022-08-28T18:47:23Z",
    "intent": {
        "name": "BestMoviesIntent",
        "confirmationStatus": "NONE",
        "slots": {
            "year": {
                "name": "year",
                "value": "2000",
                "confirmationStatus": "NONE",
                "source": "USER",
                "slotValue": {
                    "type": "Simple",
                    "value": "2000"
                }
            }
        }
    }
}
```

# Creación de una aplicación

**v5:** filtro opcional tipo de película (género)

- Importante: el API no espera el nombre del género, sino su ID

| | | | |
|---|---|---|---|
| with_genres | string | Comma separated value of genre ids that you want to include in the results. | optional |

- Dos opciones:
  - Que el backend reciba el nombre y sepa qué ID corresponde a cada género
  - Que al backend le llegue tanto el nombre como el ID
- La segunda opción es posible si utilizamos slots custom

# Creación de una aplicación

**v5:** filtro opcional tipo de película (género)

- Actualizar utterances:

| |
|---|
| devuelve las películas de más éxito de {year} de {genre} |
| dame la lista de las mejores películas de {genre} del año {year} |
| dime las películas más populares de {genre} estrenadas en {year} |

- Crear un Slot Type personalizado:

# Creación de una aplicación

**v5:** filtro opcional tipo de película (género) - custom slot type

- Para cada género podemos indicar varios nombres (principal y sinónimos) y un ID

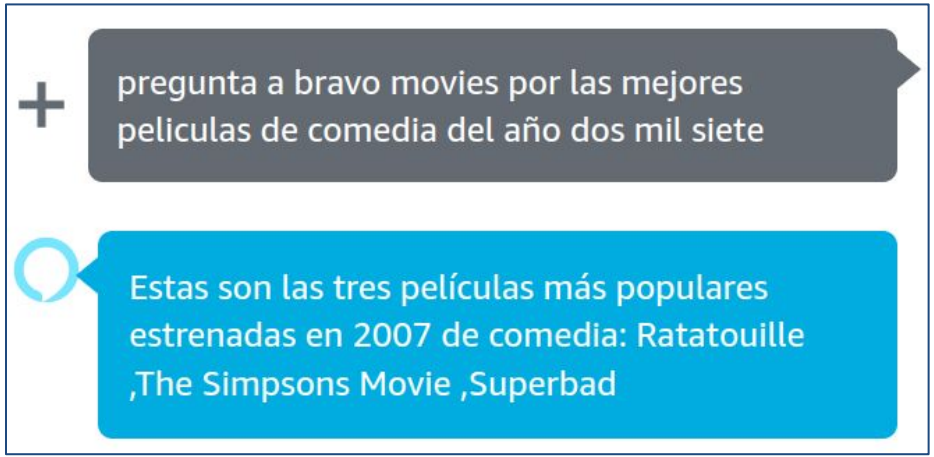
| VALUE | ID (OPTIONAL) | SYNONYMS (OPTIONAL) | |
|---|---|---|---|
| horror | 27 | Add synonym | + miedo × |
| comedia | 35 | Add synonym | + risas × |
| animación | 16 | Add synonym | + |

# Creación de una aplicación

**v5:** filtro opcional tipo de película (género) - tab CODE

- Cambios similares a los del filtro de fechas, la principal diferencia es la obtención del ID:

```
const genreSlot = Alexa.getSlot(handlerInput.requestEnvelope, 'genre');
const genreId = genreSlot && genreSlot.slotValue ? genreSlot.slotValue.resolutions
                                                   .resolutionsPerAuthority[0]
                                                   .values[0].value.id : '';
```



# Creación de una aplicación

**v6:** mejora de la locución. Podemos hacer dos mejoras sencillas. El Speech Synthesis Markup Language (SSML) nos permite configurarlas.

- El título de las películas nos viene dado en inglés, la respuesta sonaría mejor si esa parte fuera narrada por una voz con acento inglés
  - Tag voice

```
<voice name="Joanna">${filmTitles}</voice>`
```

- Añadir una pausa tras cada título de película ayudaría a que se entendiera mejor; tag break

```
results.slice(0, 3)
        .map(result => result.title)
        .map(title => `${title}<break time="1s"/>`)
        .join(',');
```

# Creación de una aplicación

**v6:** mejora de la locución. Se puede probar en el tab TEST, en la sección Voice & Tone

- Antes:

```
<speak>
      Estas son las tres películas más populares  de 2001  de comedia:
            Monsters, Inc.,
            Shrek ,
            Scary Movie 2
</speak>
```

- Después:

```
<speak>
      Estas son las tres películas más populares  de 2001  de comedia:
      <voice name=\"Joanna\">
            Monsters, Inc.<break time=\"1s\"/>,
            Shrek <break time=\"1s\"/>,
            Scary Movie 2 <break time=\"1s\"/>
      </voice>
</speak>
```

# Otras funcionalidades disponibles

- Gestión de credenciales - Account linking
- Gestión de atributos, atributos persistentes (sesión)
- Localización
- Gestión de alertas y notificaciones, (multicast / unicast)
- Respuesta visual, Alexa Presentation Response (APL), vídeo, blog
- Unit testing, TestFlow
- Visual Studio Code (I, II)
- ...

# Concurso, ¡gana un Amazon Echo Dot!

- Trivial de conceptos tratados en este taller usando Kahoot
- La persona que obtenga la mayor puntuación se llevará un Amazon Echo Dot
- Cuenta tanto la validez de las respuestas como la rapidez
- ¡Suerte!

# Join Openbravo!

We are always on the outlook of the brightest minds to join our **international team**!

We are open to any kind of collaboration (TFG - Internships etc.)

If you are interested in knowing more about a challenging and highly rewarding experience reach us at **careers@openbravo.com**
and we will find the way for you to become a very highly successful professional in our firm!

openbravo

# THE GLOBAL POS AND RETAIL MANAGEMENT CLOUD SOFTWARE VENDOR FOR INNOVATIVE AND AGILE RETAILING

## Notices