

## Alexa Skill - 2

### Juego de preguntas y respuestas

---

Puedes encontrar Cursos completos y gratuitos sobre Alexa y otras tecnologías en español en:

<https://plataforma.keepcoding.io/courses?query=alexa>


### Creamos Skill Hola Mundo

#### 1. Usaremos esta skill como base

Skill name

0/50 characters

Default language

Spanish (ES) 

More languages can be added to your skill after creation

##### 1. Choose a model to add to your skill

There are many ways to start building a skill. You can design your own custom model or start with a pre-built model. Pre-built models are interaction models that contain a package of intents and utterances that you can add to your skill.

<b>Custom</b> Design a unique experience for your users. A custom model enables you to create all of your skill's interactions.	<b>Flash Briefing</b> Give users control of their news feed. This pre-built model lets users control what updates they listen to.  "Alexa, pon el resumen de noticias."	<b>Smart Home</b> Give users control of their smart home devices. This pre-built model lets users turn off the lights and other devices without getting up.  "Alexa, enciende las luces de la cocina"	<b>Video</b> Let users find and consume video content. This pre-built model supports content searches and content suggestions.  "Alexa, pon Interstellar"
--	--	--	--

##### 2. Choose a method to host your skill's backend resources

You can provision your own backend resources or you can have Alexa host them for you. If you decide to have Alexa host your skill, you'll get access to our code editor, which will allow you to deploy code directly to AWS Lambda from the developer console.

<b>Alexa-hosted (Node.js)</b> Alexa will host skills in your account up to the AWS Free Tier limits and get you started with a Node.js template. You will gain access to AWS Lambda endpoints in all Alexa service regions, a DynamoDB table for data persistence, and S3 for media storage. <a href="#">Feedback</a> <a href="#">Learn more</a>	<b>Alexa-hosted (Python)</b> Alexa will host skills in your account up to the AWS Free Tier limits and get you started with a Python template. You will gain access to AWS Lambda endpoints in all Alexa service regions, a DynamoDB table for data persistence, and S3 for media storage. <a href="#">Learn more</a>	<b>Provision your own</b> Provision your own endpoint and backend resources for your skill. This is recommended for skills that have significant data transfer requirements. You will not gain access to the console's code editor.
---	--	--

A continuación Amazon ofrece la posibilidad de varios templates

### Choose a template to add to your skill

Select a skill template from the list below or import a skill shared by the Alexa community as a public Git repository.

Import skill

Continue with template

#### Start from Scratch

This skill gets you started with the required intents and with code demonstrating "Hello World" functionality if you are building an Alexa-hosted skill. [Learn more](#)

By [Alexa](#)

SELECTED

#### Fact Skill

Build an engaging fact skill about any topic. Alexa will select a fact at random and share it with the user when the skill is invoked. [Learn more](#)

Includes: custom intents

By [Alexa](#)

#### Scheduling Skill

Build a skill to allow users to schedule appointments on your calendar, receive email confirmations and reminders. [Learn more](#)

Includes: voice permissions, reminders, API calls, session persistence

By [Dabble Lab](#)

#### Survey Skill

Build a stand-up or survey skill that uses passcodes to allow only authorized users to provide updates and respond to questions. [Learn more](#)

Includes: using passcodes, API calls, session persistence

By [Dabble Lab](#)

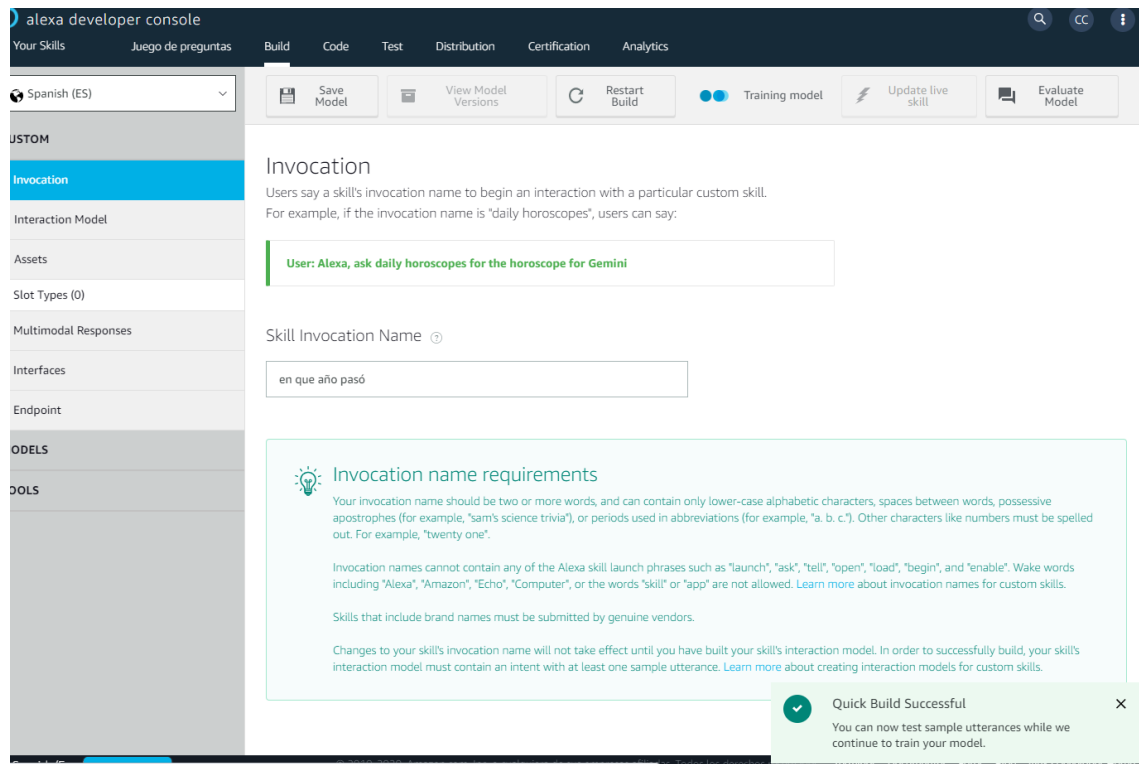
**Start from Scratch:** crea un template de un hola mundo.

**Fact Skill:** crea un template de un juego de curiosidades en el espacio con internacionalización.

**Scheduling Skill:** crea un template con una skill de programación de citas con email y teléfono.

**Survey Skill:** crea un template para una skill para hacer encuestas.

## Iniciando la Skill



The screenshot shows the Alexa Developer Console interface. The top navigation bar includes 'Your Skills', 'Juego de preguntas', 'Build', 'Code', 'Test', 'Distribution', 'Certification', and 'Analytics'. The left sidebar lists various skill components: 'CUSTOM', 'Invocation' (selected), 'Interaction Model', 'Assets', 'Slot Types (0)', 'Multimodal Responses', 'Interfaces', 'Endpoint', 'MODELS', and 'TOOLS'. The main content area is titled 'Invocation' and explains that users say a skill's invocation name to begin an interaction. It provides an example: 'User: Alexa, ask daily horoscopes for the horoscope for Gemini'. Below this, there is a text input field for the 'Skill Invocation Name' with the value 'en que año pasó'. A light blue box contains 'Invocation name requirements', detailing rules for naming conventions. A green success message at the bottom right states 'Quick Build Successful' and 'You can now test sample utterances while we continue to train your model.'

## Actualizando el front-end - Intents



Añadimos un intent para preparar la skill para recibir preguntas relacionadas con años, para ello seleccionamos un Slot, y lo creamos un slot “numberSlot” con AMAZON\_FOUR\_DIGIT\_NUMBER



Updates to sample utterances qualify for instant live updates. [Learn more](#) about live updates to your skill.

### Intents / AnswerIntent

Sample Utterances (0) ⓘ

 Bulk Edit  Export

What might a user say to invoke this intent?



This intent has no sample utterances

A sample utterance is a phrase a user might speak to invoke the intent.

◀ 0 - 0 of 0 ▶

Dialog Delegation Strategy ⓘ

Finalmente añadimos las posibles respuestas asociadas a ese año, por ejemplo:

La respuesta es el {numberSlot}

Año {numberSlot}

En el año {numberSlot}

En {numberSlot}

Es {numberSlot}

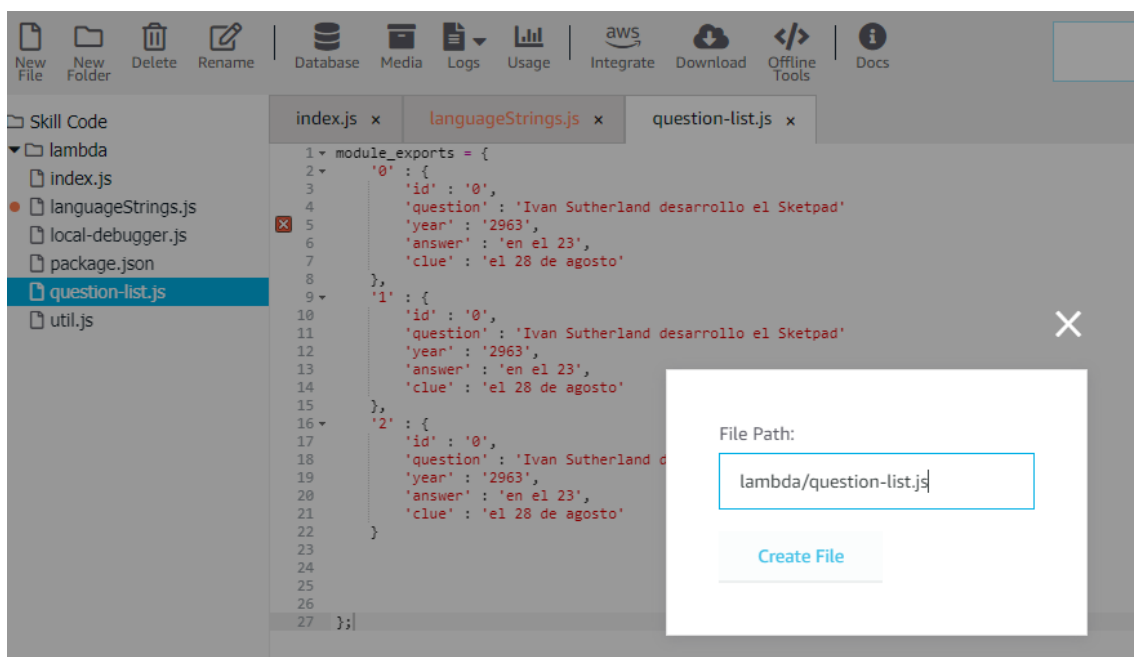
La respuesta es {numberSlot}

Para desarrollar el back-end, que va a hacer Alexa cuando reconozca una frase dentro de tu skill. El primer intent necesario será uno que responda cuando se hace la llamada de invocación.

Este intent puede ser “LaunchRequestHandler”

```
const LaunchRequestHandler = {  
  canHandle(handlerInput) {  
    return Alexa.getRequestType(handlerInput.requestEnvelope) === 'LaunchRequest';  
    //LaunchRequest indica que es la petición de lanzamiento  
  },  
  handle(handlerInput) {  
    const speakOutput = "Bienvenido";  
    return handlerInput.responseBuilder  
      .speak(speakOutput)    //responde y  
      .reprompt(speakOutput) //se queda esperando una respuesta  
      .getResponse();  
  },  
};
```

Creemos un nuevo fichero question-list.js que añadiremos con la información de las preguntas que hay dentro del juego, así como las respuestas y pistas.



```
module.exports = {  
  
  '0': {  
  
    'id': '0',  
  
    'question': 'Ivan Sutherland desarrollo el Sketpad',  
  
    'year': '2963',  
  
    'answer': 'en el 23',  
  
    'clue': 'el 28 de agosto'  
  
  },  
  
  '1': {  
  
    'id': '0',  
  
    'question': 'Ivan Sutherland desarrollo el Sketpad',  
  
    'year': '2963',  
  
    'answer': 'en el 23',  
  
    'clue': 'el 28 de agosto'  
  
  }  
  
};
```



```
1 module_exports = {  
2   '0': {  
3     'id' : '0',  
4     'question' : 'Ivan Sutherland desarrollo el Sketpad'  
5     'year' : '2963',  
6     'answer' : 'en el 23',  
7     'clue' : 'el 28 de agosto'  
8   },  
9   '1': {  
10    'id' : '0',  
11    'question' : 'Ivan Sutherland desarrollo el Sketpad'  
12    'year' : '2963',  
13    'answer' : 'en el 23',  
14    'clue' : 'el 28 de agosto'  
15  },  
16  '2': {  
17    'id' : '0',  
18    'question' : 'Ivan Sutherland desarrollo el Sketpad'  
19    'year' : '2963',  
20    'answer' : 'en el 23',  
21    'clue' : 'el 28 de agosto'  
22  }  
23  
24  
25  
26  
27 };
```

**Nota:** Siempre que se modifica algo se debe grabar para no perder los datos si cerramos sin querer la pestaña y construir (solo si se va a probar).

A continuación tendremos que desarrollar el código para que nuestra skill seleccione una pregunta aleatoria y la pregunte.

```
const questionlist = require('./question-list'); //ruta a las preguntas
```

```
var currentIndex = null
```

```
function getRandomItem (obj) { //función para selección aleatoria de preguntas
```

```
  if (Object.keys(obj).length === 0){
```

```
    return null;
```

```
  }
```

```
  currentIndex = obj[Object.keys(obj)[Math.floor(Math.random()*Object.keys(obj).length)]];
```

```
  return currentIndex;
```

```
}
```

## Construimos la pregunta

Modificamos la función de lanzamiento para que haga la pregunta

```
const LaunchRequestHandler = {
```

```
  canHandle(handlerInput) {
```

```
    //const request = handlerInput.requestEnvelope.request;
```

```
    return Alexa.getRequestType(handlerInput.requestEnvelope) === 'LaunchRequest';
```

```
  },
```

```
  handle(handlerInput) {
```

```
    const questionText = getQuestion(); //AÑADIMOS LA PREGUNTA
```

```
    //const AnswerValue = handlerInput.requestEnvelope.request.intent.slots.numberSlot.value;
```

```
    const speakOutput = "Bienvenido al juego más divertido, empecemos. " + questionText;
```

```
    return handlerInput.responseBuilder
```

```
      .speak(speakOutput)
```

```
      .reprompt(speakOutput)
```

```
      .getResponse();
```

```
  },
```

```
};
```

Hasta este punto la skill es capaz de presentarse y lanzar una pregunta, esperar por la respuesta del usuario y reconocer dicha respuesta. Faltaría comprobar si la respuesta es o no correcta.

```
const AnswerIntentHandler = {  
  canHandle(handlerInput) {  
    const request = handlerInput.requestEnvelope.request;  
    return (request.type === 'IntentRequest'  
      && request.intent.name === 'AnswerIntent');  
  },  
  handle(handlerInput) {  
    const AnswerValue = handlerInput.requestEnvelope.request.intent.slots.numberSlot.value;  
    let speakOutput = "Respondiste " + AnswerValue;  
    if (AnswerValue === currentIndex.year){  
      speakOutput += '. Respuesta correcta, el año ' + currentIndex.year + ' es verdadero porque'  
      + currentIndex.answer;  
    } else {  
      speakOutput += '. Respuesta incorrecta, el año correcto es ' + currentIndex.year + ' porque'  
      + currentIndex.answer;  
    }  
  
    return handlerInput.responseBuilder  
      .speak(speakOutput)  
      .reprompt(speakOutput)  
      .getResponse();  
  },  
};
```

Comprobamos si la respuesta es correcta o no y de esta forma sabremos si el usuario ha acertado o ha fallado



Ahora se añaden posibilidades al juego, repetir pregunta, una pista, número de respuestas correctas, etc...

Añadimos variables que usaremos para el juego

```
//Declaramos variables, la añadimos en este lugar porque es donde
//estamos metiendo código, se pueden añadir al principio del fichero también.
const questionList = require('./question-list');
var currentIndex = null;
var currentStatus = null;
var count = 0;
var hits = 0;
var pending = null;
```

Actualizamos AnswerIntentHandler para que sea capaz de detectar si continua el juego o no.

```
const AnswerIntentHandler = {
  canHandle(handlerInput) {
    const request = handlerInput.requestEnvelope.request;
    return (request.type === 'IntentRequest'
      && request.intent.name === 'AnswerIntent');
  },
  handle(handlerInput) {
    const AnswerValue = handlerInput.requestEnvelope.request.intent.slots.numberSlot.value;
    let speakOutput = ''; // "Respondiste " + AnswerValue;
    if (currentStatus === 'Continue'){
      speakOutput += 'Responde si o no';
    }else{
      if (AnswerValue === currentIndex.year){
        speakOutput += '. Respuesta correcta, el año ' + currentIndex.year + ' es verdadero porque ' + currentIndex.answer;
        hits++;
      } else {
        speakOutput += '. Respuesta incorrecta, el año correcto es ' + currentIndex.year + ' porque ' + currentIndex.answer;
      }
    }
    currentIndex=null; // inicializamos la variable
    speakOutput += "...Continuamos? ";
    currentStatus = 'Continue'; //podemos estar en estado de continuar o de pregunta. se añade el status en cada momento

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

Volvemos al Front-End y añadimos intents nuevos

-Añadimos el YesIntent, NoIntent, RepeatIntent .



## Add Intent

An intent represents an action that fulfills a user's spoken request. [Learn more](#) about intents.

☐ Create custom intent <sup>?</sup>

Enter name for intent


Create custom intent

☒ Use an existing intent from Alexa's built-in library <sup>?</sup>

[Learn more](#) about using built-in intents.

yes

1/26 built-ins

Name	Description
 AMAZON.YesIntent	<a href="#">+ Add Intent</a>

## Add Intent

An intent represents an action that fulfills a user's spoken request. [Learn more](#) about intents.

☐ Create custom intent <sup>?</sup>

Enter name for intent


Create custom intent

☒ Use an existing intent from Alexa's built-in library <sup>?</sup>

[Learn more](#) about using built-in intents.



repe



1/26 built-ins

Name	Description
 AMAZON.RepeatIntent	<a href="#">added</a> <a href="#">view</a>

y añadimos utterrances personalizados

Sample Utterances (5) 


 Bulk Edit  Export



What might a user say to invoke this intent?	+
otra vez	
repite de nuevo	
repite	
dime otra vez	
vuelve a decirlo	

◀ 1 – 5 of 5 ▶ [Show All](#)

Añadimos un intent para las pistas

Intents / ClueIntent

Sample Utterances (6) 

 Bulk Edit  Export

What might a user say to invoke this intent?	+
ni idea	
más información	
dime una pista	
una pista	
dame una pista	

◀ 1 – 5 of 6 ▶ [Show All](#)

Añadimos el intent para la siguiente pregunta.

## Add Intent

An intent represents an action that fulfills a user's spoken request. [Learn more](#) about intents.

☐ Create custom intent <sup>?</sup>

Enter name for intent


Create custom intent

☒ Use an existing intent from Alexa's built-in library <sup>?</sup>

[Learn more](#) about using built-in intents.

next



1/26 built-ins






Name	Description
 AMAZON.NextIntent	<div>added</div> <div>view</div>

Y lo personalizamos

## Intents / AMAZON.NextIntent

Sample Utterances (5) <sup>?</sup>

 Bulk Edit  Export





What might a user say to invoke this intent?	+
otra	
paso palabra	
dime la siguiente	
siguiente pregunta	
siguiente	

< 1 – 5 of 5 > [Show All](#)

Añadimos un intent para las preguntas pendientes

Sample Utterances (4) ⓘ

 Bulk Edit  Export

What might a user say to invoke this intent?	+
cual era la pregunta pendiente	
dime la pregunta pendiente	
dime la pendiente	
pendiente	

◀ 1 – 4 of 4 ▶

Volvemos al Back-End y añadimos las funciones para controlar las nuevas funcionalidades.

```
const YesIntentHandler = {  
  canHandle(handlerInput) {  
    const request = handlerInput.requestEnvelope.request;  
    return request.type === 'IntentRequest'  
      && request.intent.name === 'AMAZON.YesIntent';  
  },  
  handle(handlerInput) {  
    const requestAttributes = handlerInput.attributesManager.getRequestAttributes();  
    const speakOutput = getQuestion();  
    currentStatus = 'Question';  
    return handlerInput.responseBuilder  
      .speak(speakOutput)  
      .reprompt(speakOutput)  
      .getResponse();  
  },  
};
```

Modificamos el Exit handler para controlar el no y mostrar el número de aciertos

```
const ExitHandler = {  
  canHandle(handlerInput) {  
    const request = handlerInput.requestEnvelope.request;  
    return request.type === 'IntentRequest'  
      && (request.intent.name === 'AMAZON.CancelIntent'  
        || request.intent.name === 'AMAZO.NoIntent' // Añadimos el no intent  
        || request.intent.name === 'AMAZON.StopIntent');  
  },  
  handle(handlerInput) {  
    const requestAttributes = handlerInput.attributesManager.getRequestAttributes();  
    const speakOutput = 'Has conseguido acertar ' + hits + ' de ' + count + ' preguntas, ... hasta luego!';  
    return handlerInput.responseBuilder  
      .speak(speakOutput)//requestAttributes.t('STOP_MESSAGE')  
      .getResponse();  
  },  
};
```

Añadimos la función para el control de la pista

```
const ClueIntentHandler = {  
  canHandle(handlerInput) {  
    const request = handlerInput.requestEnvelope;  
    return Alexa.getRequestType(request) === 'IntentRequest'  
      && Alexa.getIntentName(request) === 'ClueIntent';  
  },  
  handle(handlerInput) {  
    let speakOutput="";  
    if (currentStatus === 'Question'){  
      let speakOutput = 'Ahí va la pista ' + currentIndex.clue + ' te la vuelvo a preguntar ' +  
        getQuestion(false);  
    }  
  }  
};
```



```
} else if (currentStatus === 'Continue'){  
    speakOutput += 'Responde Si o No.';  
}  
  
return handlerInput.responseBuilder  
    .speak(speakOutput)  
    .reprompt(speakOutput)  
    .getResponse();  
},  
};
```

Y la función para repetir

```
const RepeatIntentHandler = {  
    canHandle(handlerInput) {  
        const request = handlerInput.requestEnvelope;  
        return Alexa.getRequestType(request) === 'IntentRequest'  
            && Alexa.getIntentName(request) === 'RepeatIntent';  
    },  
    handle(handlerInput) {  
        let speakOutput="";  
        if (currentStatus === 'Question'){  
            let speakOutput = 'Repetimos!... ' + getQuestion(false);  
        } else if (currentStatus === 'Continue'){  
            speakOutput += 'Continuamos? ';  
        }  
  
        return handlerInput.responseBuilder  
            .speak(speakOutput)  
            .reprompt(speakOutput)
```

```
.getResponse();  
  
},  
  
};
```

Añadimos la función NextIndex

```
const NextIntentHandler = {  
  canHandle(handlerInput) {  
    const request = handlerInput.requestEnvelope;  
    return Alexa.getRequestType(request) === 'IntentRequest'  
      && Alexa.getIntentName(request) === 'NextIntent';  
  },  
  handle(handlerInput) {  
    let speakOutput="";  
    if (pending !== null){  
      speakOutput = 'Alcanzaste el máximo de preguntas pendientes de responder, vamos a por  
ella de nuevo... + getQuestion(false)';  
      const tmpIndex = pending;  
      currentIndex = pending;  
      pending = tmpIndex;  
      speakOutput+= getQuestion(false);  
    }  
    else{  
      speakOutput ='Gardamos esta pregunta para después, vamos con la siguiente! ...';  
      pending = currentIndex;  
      speakOutput += getQuestion();  
    }  
    currentStatus = 'Question';
```



```
return handlerInput.responseBuilder

    .speak(speakOutput)

    .reprompt(speakOutput)

    .getResponse();

},

};

//función para recuperar una pregunta pendientes

const PendingIntentHandler = {

    canHandle(handlerInput) {

        const request = handlerInput.requestEnvelope;

        return Alexa.getRequestType(request) === 'IntentRequest'

            && Alexa.getIntentName(request) === 'PendingIntent';

    },

    handle(handlerInput) {

        let speakOutput="";

        if (pending === null){

            if (currentIndex !== null && currentStatus === 'Question'){

                speakOutput += 'Hemos dejado esta pregunta sin responder, la guardamos para despues... ';

                pending = currentIndex;

            }

            speakOutput = 'no tienes preguntas pendientes! ... Quieres continuar con una pregunta? ';

            currentStatus = 'Continue';

        }else{

            if(currentIndex !== null && currentStatus === 'Question'){

                let tmpIndex = currentIndex;

                currentIndex = pending;

                pending = currentIndex;

            }

        }

    }

}
```



speakOutput += 'Hemos dejado esta pregunta sin responder, la guardamos para  
despues ...';

```
    } else {  
        currentIndex = pending;  
        pending = null;  
    }  
    speakOutput = 'Vamos con la pregunta que teníamos pendiente! ... ' + getQuestion(false);  
    currentStatus = 'Question';  
}  
return handlerInput.responseBuilder  
    .speak(speakOutput)  
    .reprompt(speakOutput)  
    .getResponse();  
},  
};
```

Añadimos los Handler que se han añadido

```
exports.handler = skillBuilder  
    .addRequestHandlers(  
        LaunchRequestHandler,  
        AnswerIntentHandler,  
        //GetNewFactHandler,  
        HelpHandler,  
  
        RepeatIntentHandler,  
        NextIntentHandler,  
        ClueIntentHandler,  
        PendingIntentHandler,
```

ExitHandler,

FallbackHandler,

SessionEndedRequestHandler,

)

Actualizamos la función getQuestion

```
function getQuestion (random = true) {  
  let speechText = "";  
  if (random){  
    speechText = getRandomItem(questionList);  
    if (currentIndex === null && pending === null){  
      return 'ya respondiste las preguntas! ... Has acertado '+hits+' de '+ count + 'preguntas. ';  
    }else if ( currentIndex === null){  
      return 'ya no te quedan más preguntas nuevas, pero si te queda una pendiente, vamos a por ella. ¿En que año '+ speechText.question + '?';  
    }  
    delete questionList[currentIndex.id];  
    count++;  
  } else {  
    speechText = currentIndex;  
  }  
  const speakOutput = '¿En qué año ' + speechText.question + '?';  
  return speakOutput;  
}
```