

# UD8 INTERFACES NATURALES.

## Servicios conversacionales para

### ChatRote



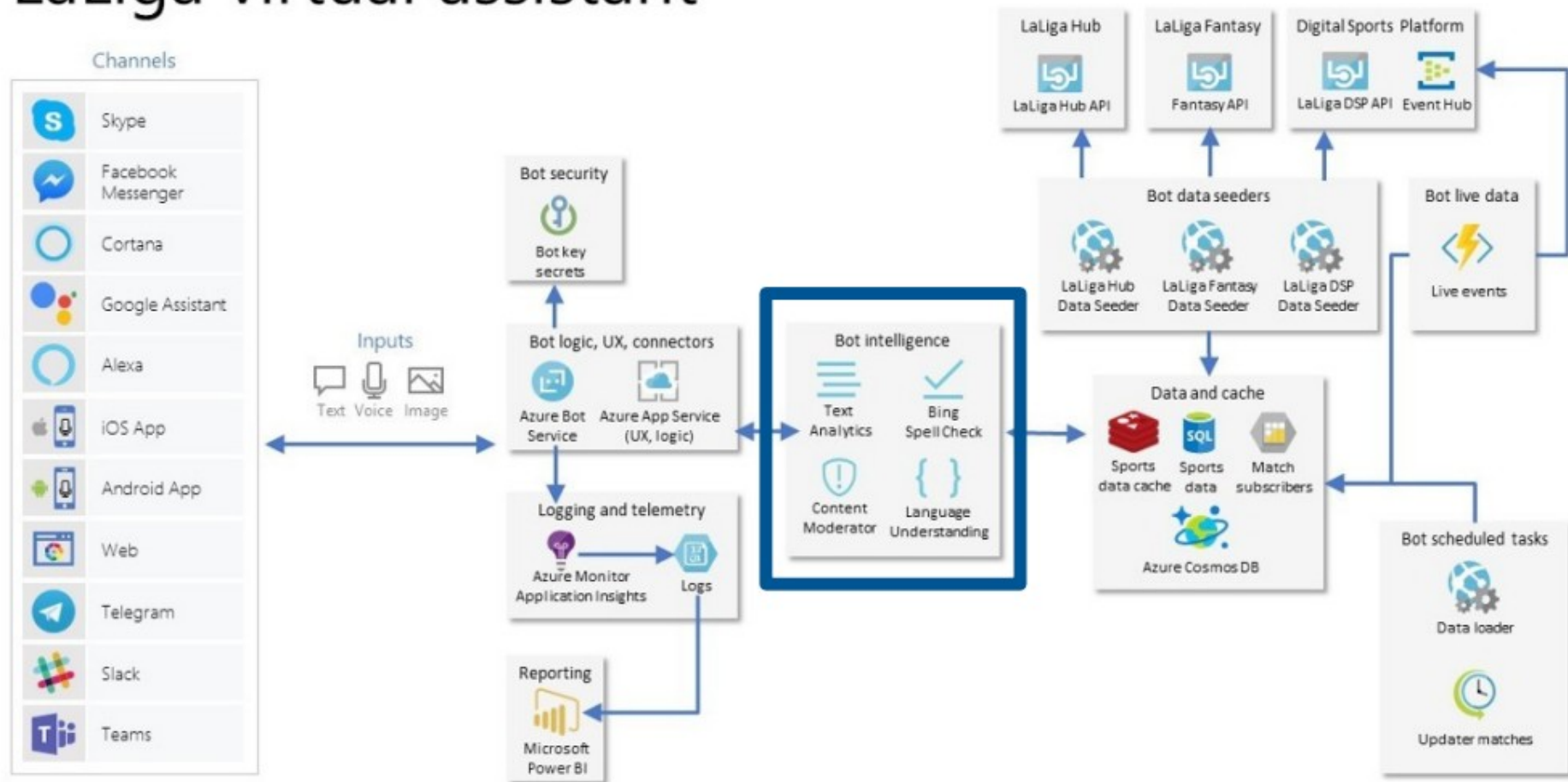
Ciclo: Desarrollo de Aplicaciones Multiplataforma  
Módulo: Desarrollo de interfaces. Curso 2021-22



# Arquitectura de un ChatBot

- Cuando se diseña la arquitectura de un bot conversacional o ChatBot, se pueden distinguir dos partes bien diferenciadas:
  - **Interfaz con los usuarios:** Incluye toda la infraestructura necesaria para conectar el bot con los diferentes canales de comunicación. El servicio Azure Bot Service de Azure se utiliza con este propósito.
  - **Lógica del bot:** Una vez que la entrada del usuario llega a la aplicación del bot, debe procesarse para proporcionar una respuesta y realizar las acciones pertinentes. En este procesamiento suelen jugar un papel importante diferentes servicios cognitivos de los que hemos visto en otros temas, como Text Analytics, Speech o Computer Vision. Pero hay dos servicios basados en procesamiento del lenguaje natural especialmente diseñados para ello: QnA Maker y LUIS (Language Understanding Intelligent

# LaLiga virtual assistant



# QnA Maker

- QnA Maker es un servicio conversacional basado en una colección de preguntas y respuestas, conocida como base de conocimiento. En la base de conocimiento se pueden asociar múltiples preguntas a una misma respuesta. Una vez que la base de conocimiento está creada, el servicio ofrecerá la respuesta más adecuada ante una nueva pregunta (aunque ésta no coincida totalmente con una de las preguntas presentes en la base de conocimiento).
- QnA Maker permite la edición directa de pares pregunta-respuesta, pero una de las principales funcionalidades de este servicio es la extracción automática de conocimiento de fuentes de información semiestructurada, como por ejemplo:
  - Páginas web de preguntas frecuentes (FAQ) o soporte técnico.
  - Documentos como manuales o guías de usuario (en formato PDF o DOC).
  - Hojas de cálculo o ficheros de texto plano.

- Además, el servicio ofrece la posibilidad de incluir en la base de conocimiento un lote de preguntas-respuesta para dar una personalidad base al bot, pudiendo elegir entre las siguientes personalidades:
  - Profesional (Professional)
  - Amigable (Friendly)
  - Ingenioso (Witty)
  - Bondadoso (Caring)
  - Entusiasta (Enthusiastic)
- Otra funcionalidad interesante que ofrece este servicio es la de las conversaciones multiturno, en las que ante una pregunta del usuario la respuesta incluye a su vez referencias a otras preguntas que pueden completar o aclarar la respuesta.

# 1 Creación de la base de conocimiento

<https://www.qnamaker.ai>

Authoring Key





Cognitive Services | QnA Maker | My knowledge bases | Create a knowledge base

Design sophisticated multi-turn conversations easily with follow-up prompts. [Learn more.](#)

My knowledge bases

Microsoft | documentationteam | All services

Knowledge base name	Last modified	Last published	Sample Code	Azure service name
 My Sample QnA KB	3/18/2020, 10:54:49 AM	None	<a href="#">View Code</a>	qnamaker-14kb
 Multi-turn example	3/17/2020, 10:55:00 AM	3/17/2020, 10:55:20 AM	<a href="#">View Code</a>	qnamaker-14kb

# 2 Consulta a la base de conocimiento



Endpoint Key

# Utilización de QnA Maker

- Para utilizar el servicio QnAMaker tendremos que realizar dos pasos:
  1. Creación de la base de conocimiento.
  2. Consulta a la base de conocimiento.

# Creación de la base de conocimiento

- La primera tarea que tenemos que realizar para utilizar QnA Maker es la creación y configuración de la base de conocimiento. Este paso implica tanto suministrar las fuentes de las que el servicio obtendrá los pares pregunta-respuesta (páginas web, documentos, personalidades base,...) como la creación directa de preguntas y sus correspondientes repuestas.
- La creación de la base de conocimiento se puede realizar desde el portal <https://qnamaker.ai> o utilizando la API de QnA Maker con la clave de edición (authoring key), obtenida del portal de Azure.
- Una vez que la base de conocimiento está preparada es necesario entrenarla y, posteriormente, publicarla para que sea accesible desde la API REST de consulta. El entrenamiento y la publicación se pueden realizar desde el portal de QnA Maker o utilizando la API directamente.



# Consulta a la base de conocimiento

- Una vez entrenada y publicada la base de conocimiento, podremos usar el servicio QnA Maker para obtener una respuesta a una pregunta de un usuario. Para ello utilizaremos la API de consulta (QnA Maker – Runtime – Generate Answer).
- La petición al servicio REST será de tipo POST, y se configurará de la siguiente forma:
  - URL: {RuntimeEndpoint}/qnamaker/knowledgebases/{kbId}/generateAnswer
    - RuntimeEndpoint: se obtiene tras la publicación de la base de conocimiento en el portal de QnA Maker. No coincide con el endpoint que se obtiene en el portal de Azure.
    - kbId: el identificador de la base de conocimiento, que también obtendremos del portal de QnA Maker.

– Cabeceras:

- Authorization: clave para acceder a la API de consulta, que obtendremos también del portal de QnA Maker.

– Cuerpo:

- La pregunta que se realiza a la base de conocimiento en formato JSON, por ejemplo:

```
{"question": "qna maker and luis"}
```

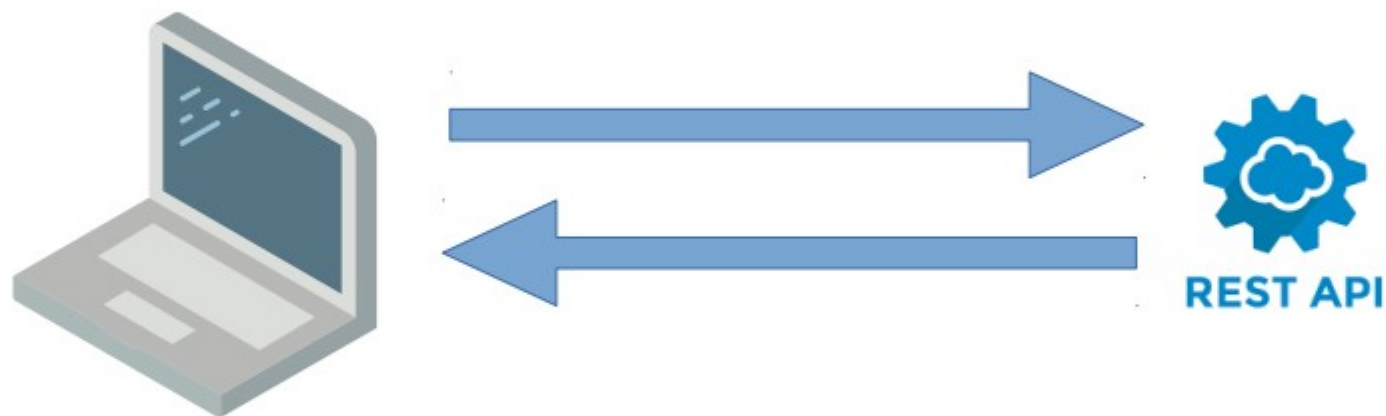
- O bien el identificador de la pregunta (cuando utilicemos la funcionalidad de conversación multiturno).

```
{"qnaId": 15}
```

- En el cuerpo de la respuesta recibiremos una lista con las diferentes respuestas encontradas para la pregunta, con una puntuación asociada a cada una de ellas (entre 0 y 100).

```
{
  "answers": [
    {
      "score": 28.54820341616869,
      "id": 20,
      "answer": "There is no direct integration of LUIS with QnA  
Maker. But, in your bot code, you can use LUIS and  
QnA Maker together.",
      "source": "Custom Editorial",
      "questions": [
        "How can I integrate LUIS with QnA Maker?"
      ],
      "metadata": [
        {
          "name": "category",
          "value": "api"
        }
      ]
    }
  ]
}
```

Parámetros	<i>kbld</i> (identificador de la base de conocimiento)
Cabeceras	<i>Authorization</i> (Endpoint Key)
Cuerpo	Pregunta que se quiere realizar, en formato JSON



<i>answer</i>	Respuesta localizada en la base de conocimiento para la pregunta enviada.
<i>score</i>	Puntuación asociada a la respuesta (entre 0 y 100)

# Lenguaje Natural (LUIS)

- LUIS (Language Understanding Intelligent System) es un servicio de procesamiento de lenguaje natural que permite a una aplicación entender la intención de un usuario a partir de una orden en lenguaje natural.
- A diferencia de QnA Maker, LUIS está orientado a aplicaciones en las que la interacción del usuario está encaminada a realizar una acción, no solo a obtener una respuesta.
- Entre los tipos de aplicaciones que pueden utilizar este servicio se encuentran, evidentemente, los ChatBots, pero también cualquier tipo de aplicación cuya interfaz sea con lenguaje natural.

## Language Understanding

Social Media  
Feedback  
Monitoring &  
Response



Speech enabled  
Intelligent desktop  
app for order  
tracking



Bot servicing product  
queries and feedback

- Dentro de este servicio se pueden destacar dos conceptos fundamentales:
  - Intenciones (Intents): representan las acciones que el usuario puede hacer en una aplicación.
  - El sistema se entrena mediante ejemplos (posibles solicitudes del usuario), y se recomienda un número de ejemplos entre 15 y 30 por intención, siempre equilibrando el número de ejemplos de las diferentes intenciones.
  - Además, los ejemplos deberían ser variados (en cuanto al número de palabras, palabras usadas o puntuación).
  - También es importante destacar que existe una intención por defecto (None) que se devolverá por el servicio cuando no se detecte ninguna de las intenciones, y que también debe ser entrenada con ejemplos (se recomienda que aproximadamente acapare el 10% de los ejemplos).

- Entidades (Entities): las entidades representan la información relacionada con la intención. No son necesarias para utilizar el servicio, pero sí muy recomendables. LUIS permite entidades formadas por otras entidades, y definir entidades de distintos tipos:
  - Aprendizaje automático: el servicio aprenderá de forma automática a reconocer dichas entidades.
  - Listas: se definen listas de términos asociadas a la entidad.
  - Patrones: entidades en las que puede resultar difícil determinar los límites de la entidad.
  - Expresiones regulares: permiten reconocer entidades con una estructura predecible.
  - Predefinidas: entidades creadas previamente en LUIS, pertenecientes a distintos dominios.
- El servicio LUIS, a partir de una orden del usuario en lenguaje natural, extraerá la intención del usuario y las entidades asociadas a dicha intención.



# Utilización de LUIS

- Para utilizar el servicio LUIS tendremos que realizar dos pasos:
  1. Creación de la aplicación.
  2. Predicción para una acción.

## Chat bot

I want to call my HR rep.

You

Your HR rep is John Smith [johns@contoso.com](mailto:johns@contoso.com) 212-555-1212

bot at 1:28:00 PM



Type your message...



Request



Response

## 2 Language Understanding (LUIS)

```
{
  "query": "I want to call my HR rep.",
  "prediction": {
    "topIntent": "HRContact",
    "intents": {
      "HRContact": {
        "score": 0.921233
      }
    }
  },
  "entities": {
    "Contact Type": [
      "call"
    ]
  }
}
```

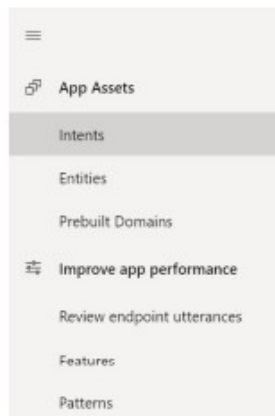
# Creación de la aplicación

- En LUIS, las intenciones y entidades relacionadas con un dominio se engloban en una aplicación. La creación de la aplicación implica la definición de intenciones y entidades, y el suministro de ejemplos para las distintas intenciones (marcando en cada ejemplo la presencia de entidades).
- La creación de la aplicación se puede realizar desde el portal <https://luis.ai> o utilizando la API de creación de LUIS con la clave del recurso de edición (authoring key), obtenida del portal de Azure.
- Una vez que la aplicación está preparada es necesario entrenarla, y posteriormente publicarla para que sea accesible desde la API REST de consulta. El entrenamiento y la publicación se pueden realizar desde el portal de LUIS o mediante la API REST.
- La publicación de una aplicación se puede realizar en dos ranuras (slots) diferentes, pruebas (staging) y producción (production). Ambas ranuras están disponibles al mismo tiempo a través de la API de predicción. Además de esta distinción, LUIS permite gestionar distintas versiones de una misma aplicación, de forma que podríamos tener publicada en cada ranura una versión diferente.

# 1 Creación de la aplicación

<https://www.luis.ai>

## Authoring Resource



## Intents

+ Create + Add prebuilt domain intent Rename Delete

Name ↑	Examples	Features
None	0	+ Add feature

# 2 Predicción para una acción



## Prediction Resource

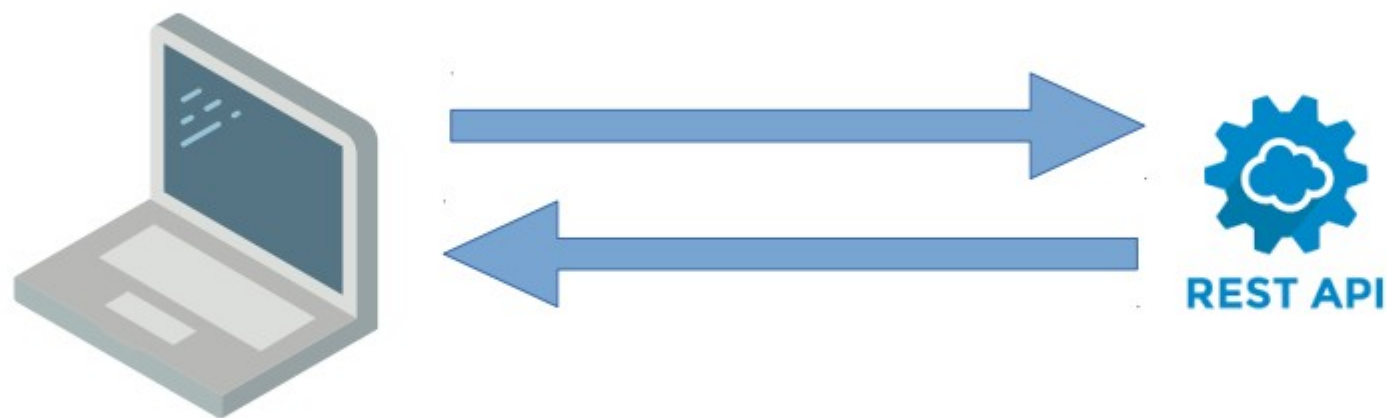
# Predicción para una acción

- Una vez entrenada y publicada la aplicación, podremos usar el servicio LUIS para obtener la intención y las entidades a partir de una orden del usuario. Para ello utilizaremos la API de predicción (LUIS – Endpoint). La API tiene métodos diferentes para dirigir la consulta a un slot (pruebas o producción) o a una versión de la aplicación. Nosotros utilizaremos la opción de consulta a un slot (Get published slot predictions).
- La petición al servicio REST será de tipo GET, y se configurará de la siguiente forma:
  - URL: `https://{endpoint}/luis/prediction/v3.0/apps/{appId}/slots/{slotName}/ predict`
    - endpoint: se puede obtener del portal de Azure (del recurso de predicción) o del portal de LUIS.
    - appId: el identificador de la aplicación, que obtendremos del portal de LUIS.
    - slotName: Staging o Production.

- Parámetros:
  - query: la consulta del usuario que queremos analizar.
- Cabeceras:
  - Ocp-Apim-Subscription-Key: clave para acceder a la API de predicción, que obtendremos del portal de Azure o del portal de LUIS.
- En el cuerpo de la respuesta recibiremos la intención con mayor puntuación (topIntent), una lista con las posibles intenciones y la puntuación asociada (intents) y, por último, la lista de entidades reconocidas en la consulta (entities).

```
{
  "query": "añade dos litros de leche a la lista",
  "prediction": {
    "topIntent": "AddItem",
    "intents": {
      "AddItem": {
        "score": 0.99503726
      }
    },
    "entities": {
      "Quantity": ["dos litros"],
      "Item": ["leche"]
    }
  }
}
```

Parámetros	<i>appld, query, slot</i>
Cabeceras	<i>Ocp-Apim-Subscription-Key</i>
Cuerpo	



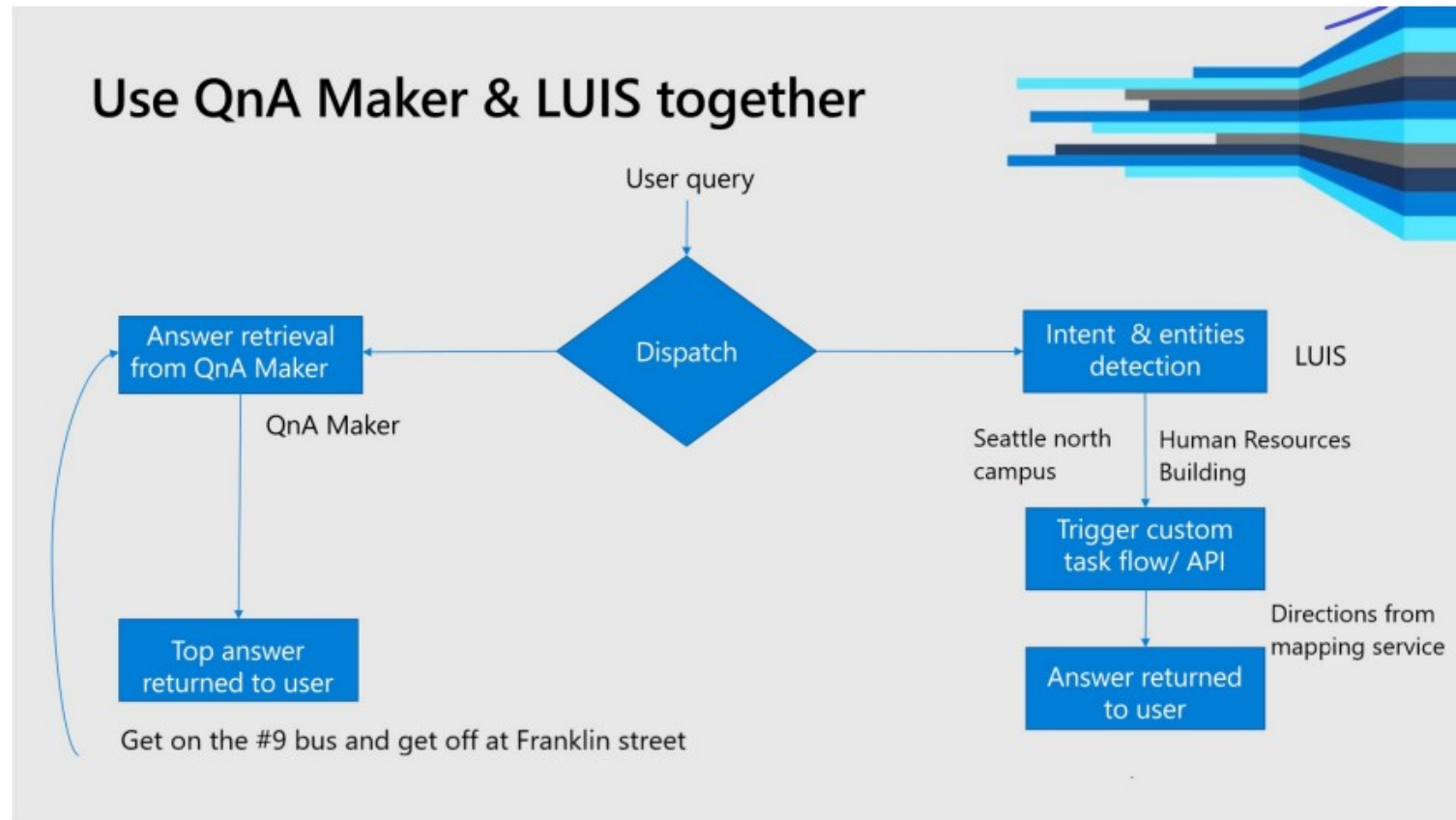
<i>topIntent</i>	Intención con mayor puntuación.
<i>entities</i>	Entidades reconocidas en la orden.

# Integración de QnA Maker y LUIS

- Es muy habitual en el desarrollo de aplicaciones con interfaz conversacional utilizar estos dos servicios de forma conjunta, ya que se complementan muy bien. Se pueden seguir dos estrategias diferentes:
  - Lanzar la consulta en primer lugar a QnA Maker y, si la respuesta tiene una confianza aceptable (por encima del umbral que se defina) devolver esa respuesta al usuario. En caso de que no se obtenga una respuesta aceptable, se puede pasar la consulta a LUIS para ver si el usuario tiene alguna intención reconocida por nuestra aplicación.
  - Otro enfoque sería preguntar primero a LUIS por la intención del usuario, y si no se obtiene una intención con una determinada puntuación, o se obtiene la intención None, entonces lanzar la consulta a QnA Maker.
- La elección de uno u otro enfoque dependerá del tipo de aplicación. Por ejemplo, en una aplicación de reserva de vuelos con interfaz conversacional seguramente será más adecuado el segundo planteamiento. Pero en un chat de consulta a un servicio técnico, es probable que se opte por el primero.



## How do I get to the Human Resources building on the Seattle North campus?



# Recursos

- Página principal de QnA Maker

<https://azure.microsoft.com/es-es/services/cognitive-services/qna-maker>

- QnA Maker en Microsoft Docs

<https://docs.microsoft.com/es-es/azure/cognitive-services/qnamaker>

- Referencia de la API QnA Maker

<https://docs.microsoft.com/en-us/rest/api/cognitiveservices-qnamaker/qnamaker4.0/runtime/generateanswer>

- Portal de QnA Maker

<https://www.qnamaker.ai>

- ChitChats en GitHub

<https://github.com/microsoft/botframework-cli/blob/main/packages/qnamaker/docs/chit-chat-dataset.md>

- Página principal de LUIS

<https://azure.microsoft.com/es-es/services/cognitive-services/languageunderstanding-intelligent-service>

- LUIS en Microsoft Docs

<https://docs.microsoft.com/es-es/azure/cognitive-services/luis>

- Referencia de la API LUIS - Predicción

<https://westus.dev.cognitive.microsoft.com/docs/services/luis-endpoint-api-v3-0/operations/5cb0a91e54c9db63d589f433>

- Portal de LUIS

<https://www.luis.ai>

- Ejemplos en GitHub

<https://github.com/Azure-Samples/cognitive-services-quickstart-code>

# Creación del recurso QnA Maker en el portal de Azure

- Para poder utilizar el servicio de preguntas y respuestas deberemos crear un recurso de tipo QnA Maker en el portal de Azure. La creación de este recurso implica la creación de otros recursos asociados en Azure, que se desplegarán de forma automática a la vez que el recurso principal de QnA Maker:
  - QnA Maker: el recurso principal.
  - Search: recurso que se utiliza para almacenar la información y realizar las búsquedas.
  - App Service: recurso que alojará la API REST a la que realizaremos las peticiones de consulta.
  - Plan de App Service: servicio que define el plan de tarifa del recurso App Service.
  - Application Insights: servicio de telemetría para las peticiones de consulta. Este recurso es opcional, y nosotros no lo utilizaremos.

- En la ventana de creación del recurso se solicitará la siguiente información:
  - Administrado (versión preliminar): dejar esta opción sin marcar.
  - Suscripción: deberemos escoger nuestra suscripción de Azure para estudiantes.
  - Grupo de recursos: es un contenedor que permite agrupar recursos de Azure relacionados entre sí. Se puede crear un único grupo de recursos para todos los recursos que iremos creando en el curso, o uno para cada uno de los temas.
  - Nombre: el nombre que le queremos dar al recurso. Este nombre también se utilizará para formar la URL del punto de conexión a la API.
  - Plan de tarifa: determina el nivel de servicio y la facturación asociada. En nuestro caso, seleccionaremos el plan gratuito (F0).

- Azure Search
  - Ubicación: determina la región en la que se creará el recurso. Podemos dejar la región por defecto.
  - Plan de tarifa: seleccionaremos Gratis (F).
- App Service
  - Nombre de la aplicación: nombre que daremos a este recurso. Podemos dejar el nombre propuesto.
  - Ubicación: determina la región en la que se creará el recurso. Podemos dejar la región por defecto.
- App Insights
  - Conclusiones de la aplicación: seleccionaremos Deshabilitar.
- Es importante destacar el mensaje de aviso que aparece asociado al recurso App Service:



El plan de App Service actualmente usa el nivel Estándar de forma predeterminada (S1; [precios](#)). Para modificarlo, visite la página de recursos del plan de App Service después de crear el recurso.

- Este mensaje nos indica que el Plan de App Service que se va a crear por defecto es de tipo S1, que no es gratuito. Es importante que, una vez creados todos los recursos asociados a QnA Maker, cambiemos el Plan de App Service a uno de tipo gratuito (tipo F1).
- Una vez creados los recursos y modificado el Plan de App Service, ya podremos utilizar el portal de QnA Maker para crear la base de conocimiento, entrenarla y publicarla. En dicho portal obtendremos todos los datos necesarios para realizar consultas a la base de conocimiento desde nuestra aplicación.
- Por último, indicar que si queremos modificar la respuesta por defecto que ofrece QnA Maker, lo podremos hacer desde la sección Configuración del recurso de tipo App Service. Ahí encontraremos una variable de configuración llamada DefaultAnswer.

# Creación del recurso LUIS en el portal de Azure

- Para poder utilizar el servicio de procesamiento del lenguaje natural deberemos crear un recurso de tipo LUIS en el portal de Azure.
- En la ventana de creación del recurso se solicitará la siguiente información:
  - Opciones de creación: seleccionaremos Ambos (para crear un recurso de creación y otro de predicción).
  - Suscripción: deberemos escoger nuestra suscripción de Azure para estudiantes.
  - Grupo de recursos: es un contenedor que permite agrupar recursos de Azure relacionados entre sí. Se puede crear un único grupo de recursos para todos los recursos que iremos creando en el curso, o uno para cada uno de los temas.
  - Nombre: el nombre que le queremos dar al recurso. Este nombre también se utilizará para formar la URL del punto de conexión a la API.
-

- Recurso de creación
  - Región: establece en cuál de las regiones de Azure se creará el recurso.
  - Plan de tarifa: determina el nivel de servicio y la facturación asociada. En nuestro caso, seleccionaremos el plan gratuito (F0).
- Recurso de predicción
  - Región: establece en cuál de las regiones de Azure se creará el recurso. Se recomienda utilizar la misma región que en el recurso de creación.
  - Plan de tarifa: determina el nivel de servicio y la facturación asociada. En nuestro caso, seleccionaremos el plan gratuito (F0).
- Al finalizar, se crearán dos recursos diferentes en Azure (uno de creación y otro de predicción). En ese momento ya podremos utilizar el portal de LUIS para crear la aplicación, entrenarla y publicarla. En dicho portal obtendremos todos los datos necesarios para realizar consultas a la aplicación desde nuestra aplicación.



# SDK de QnA Maker para C#

- Para poder utilizar el SDK de QnA Maker en nuestro proyecto será necesario añadir el paquete NuGet `Microsoft.Azure.CognitiveServices.Knowledge.QnAMaker`.
- Añadiremos la versión 2.0.1 (que es la que utiliza la versión 4.0 de la API de QnA Maker, la última estable).
- Una vez añadido el paquete, haremos referencia a los siguientes espacios de nombres en nuestro código:

```
using Microsoft.Azure.CognitiveServices.Knowledge.QnAMaker; using  
Microsoft.Azure.CognitiveServices.Knowledge.QnAMaker.Models;
```

- Las clases principales del SDK son las siguientes:
  - QnAMakerRuntimeClient: la utilizaremos como punto de conexión a la API de consulta de QnA Maker. Requiere la URL y la clave obtenidas del portal de QnA Maker. Contiene el método GenerateAnswer para obtener una respuesta de la base de conocimiento (a partir de una pregunta, o de un identificador de pregunta).
  - QueryDTO: representa una consulta a la base de conocimiento. Se puede configurar con una pregunta del usuario (propiedad Question) o con un identificador de pregunta (propiedad Qnald).
  - QnASearchResultList: es el tipo de retorno del método de consulta a la base de conocimiento. Contiene una lista con las posibles respuestas encontradas (Answers), estando en la posición inicial la respuesta con mejor puntuación. Para cada respuesta, contiene el texto de la respuesta (propiedad Answer) y, si los hay, la lista de temas relacionados (en la propiedad Context.Prompts).

# SDK de LUIS - Runtime para C#

- Para poder utilizar el SDK de predicción de LUIS añadiremos el paquete NuGet Microsoft.Azure.CognitiveServices.Language.LUIS.Runtime. Una vez añadido, haremos referencia a los siguientes espacios de nombres en nuestro código:

```
using Microsoft.Azure.CognitiveServices.Language.LUIS.Runtime;  
using Microsoft.Azure.CognitiveServices.Language.LUIS.Runtime.Models;
```

- Las clases principales del SDK son las siguientes:
  - LUISRuntimeClient: la utilizaremos como punto de conexión a la API de predicción de LUIS. Requiere la URL y la clave del recurso de predicción, obtenidas del portal de LUIS. Contiene el método GetSlotPrediction para obtener la predicción para una orden del usuario, dirigida a un slot (Staging o Production).
  - PredictionRequest: representa una petición de predicción al servicio. Con la propiedad Query indicaremos la orden del usuario.
  - PredictionResponse: es la respuesta del servicio. Contiene la intención con mayor puntuación (TopIntent), una lista de todas las intenciones detectadas con sus puntuaciones (Intents), y una lista de las entidades detectadas (Entities).