**Sviluppare un’applicazione LUIS ed integrarla con attuatore/sensore**

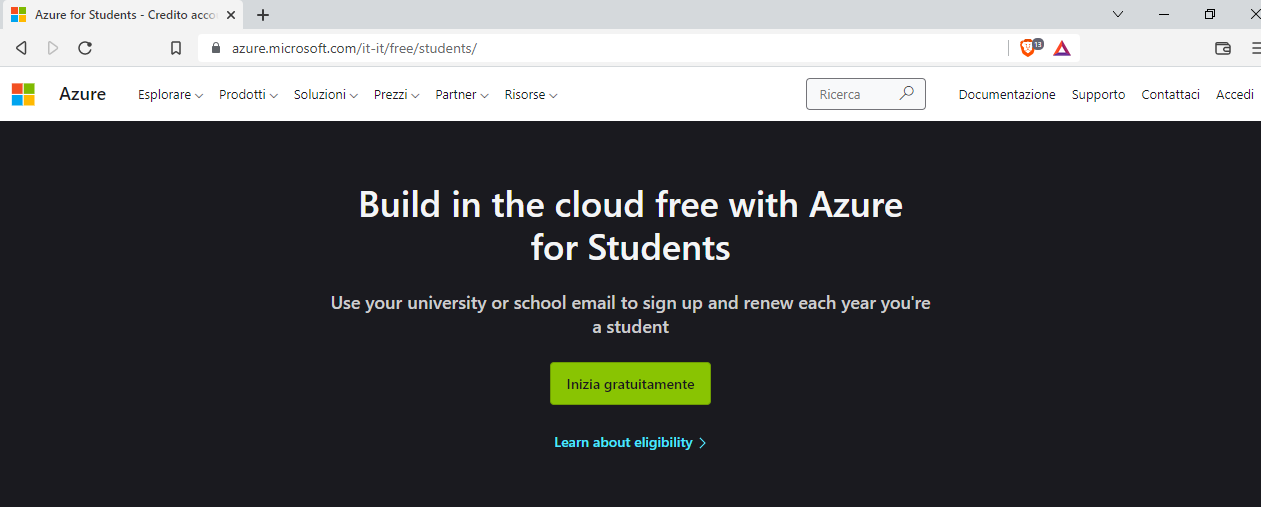
*Lo scopo di questa attività è controllare l’attuatore mediante comandi vocali analizzati da LUIS ed ottenere i dati del sensore mediante sintesi vocale, interagendo con un server Flask o MQTT installato su Raspberry.*

*Nella sezione A verrà illustrato come configurare LUIS, nella sezione B come sviluppare un programma Python (sotto Windows) che interagisce con LUIS e nella sezione C come utilizzare la sintesi vocale. Nella sezione D, infine, verranno date indicazioni su come integrare il tutto con l’attuatore ed il sensore.*

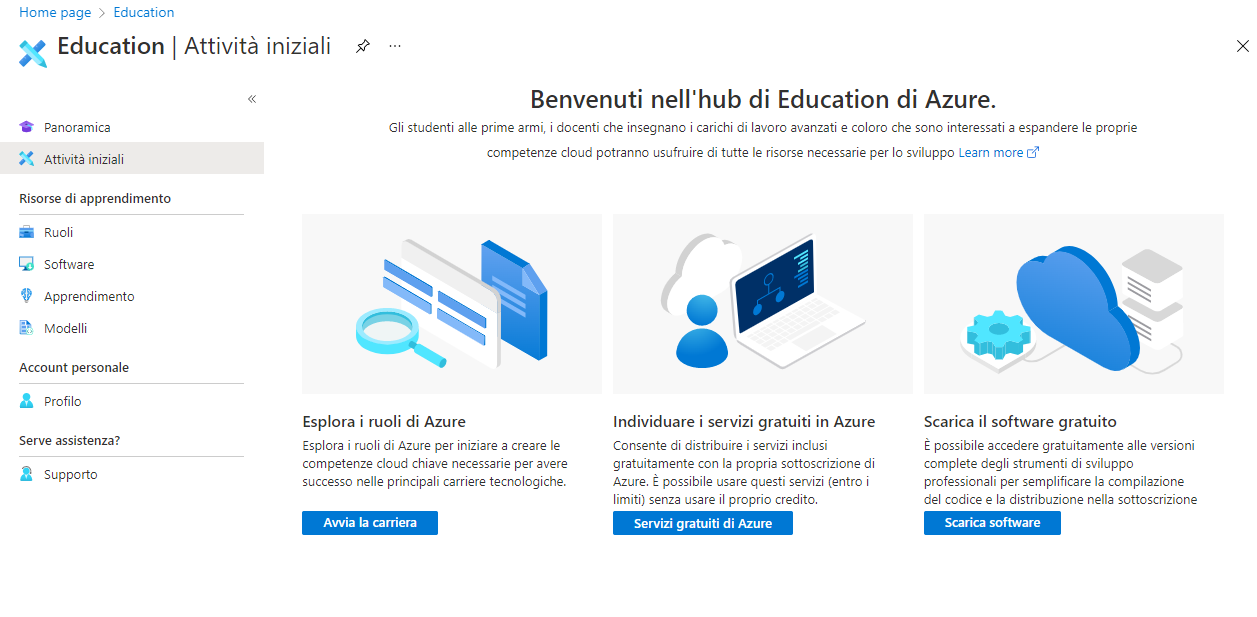
**Sezione A: configurazione LUIS**

**Parte 1: creazione di un account Azure e creazione dell’applicazione LUIS**

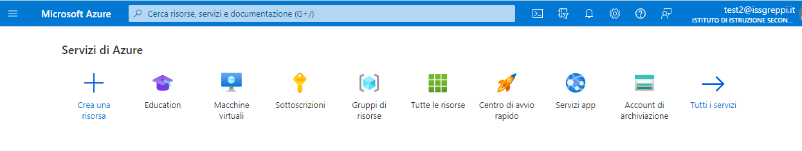
1. Accedere a <https://azure.microsoft.com/it-it/free/students/>



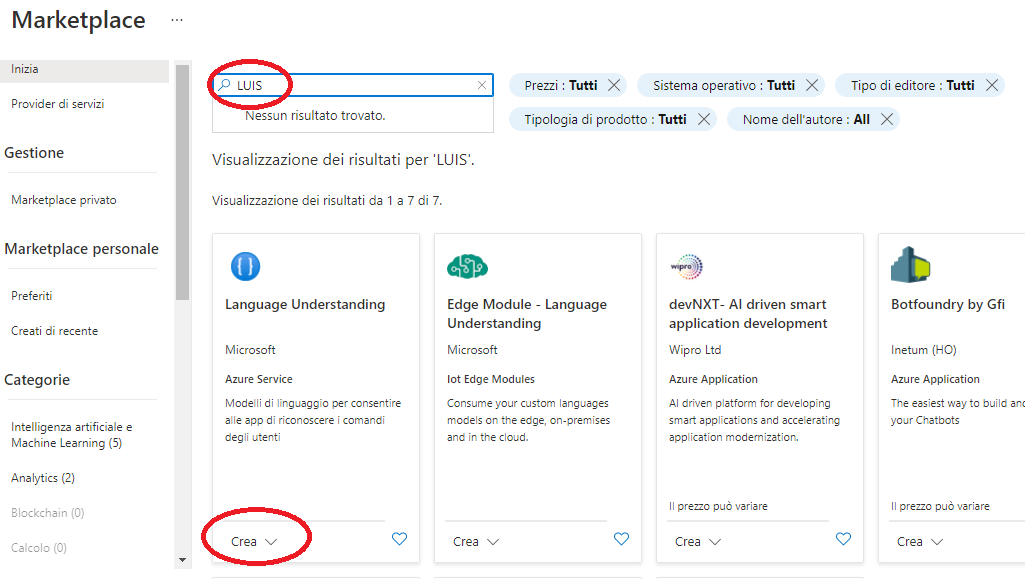
1. Scegliere *”Inizia gratuitamente”,* loggarsi con l’account scolastico ed eseguire la procedura di registrazione



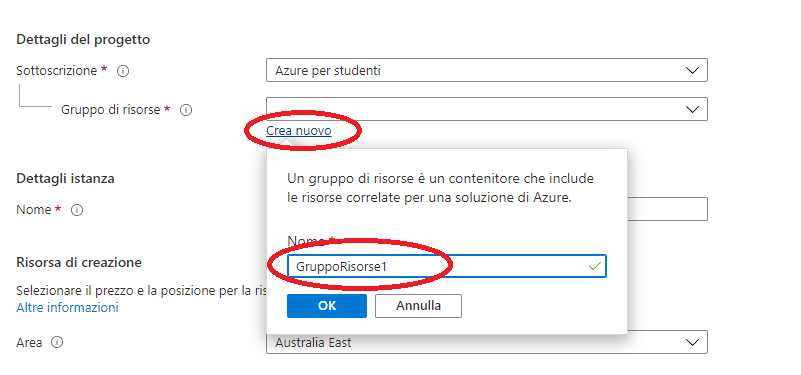
1. Scegliere *“Home page”* (in altro a sinistra) e selezionare *“Crea una risorsa”*



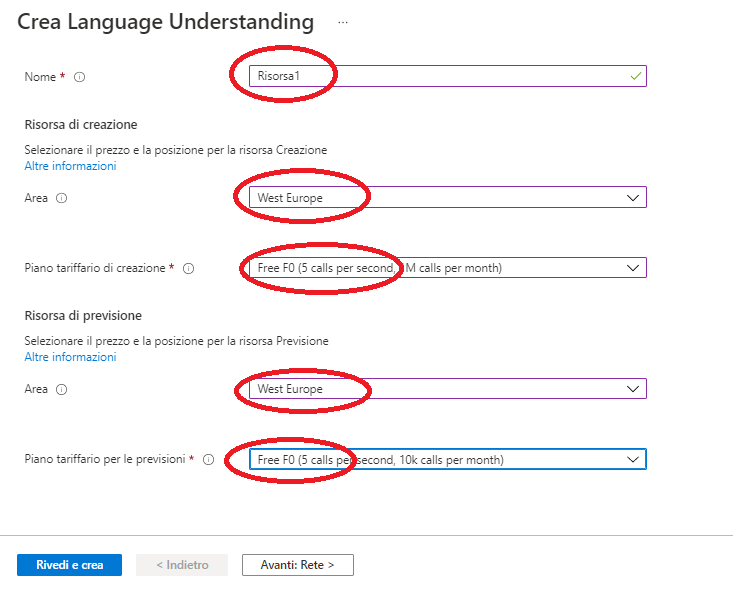
1. Cercare *“LUIS”* e poi scegliere *“Crea”* nella finestra di *“Language understanding”*



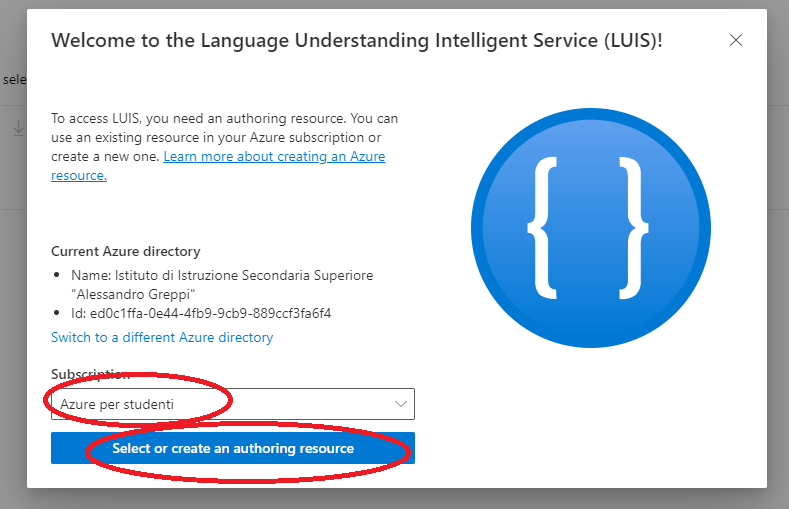
1. Creare un gruppo di risorse con nome a piacere (ad esempio *GruppoRisorse1)*



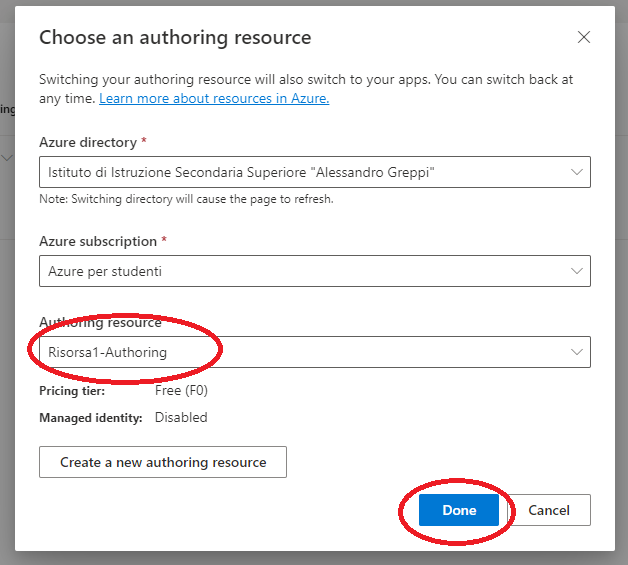
1. Assegnare un nome alla risorsa (ad esempio *Risorsa1) ,* scegliere zona (*West Europe)* e piano tariffario (*F0*) e clicca su *“Rivedi e crea”*



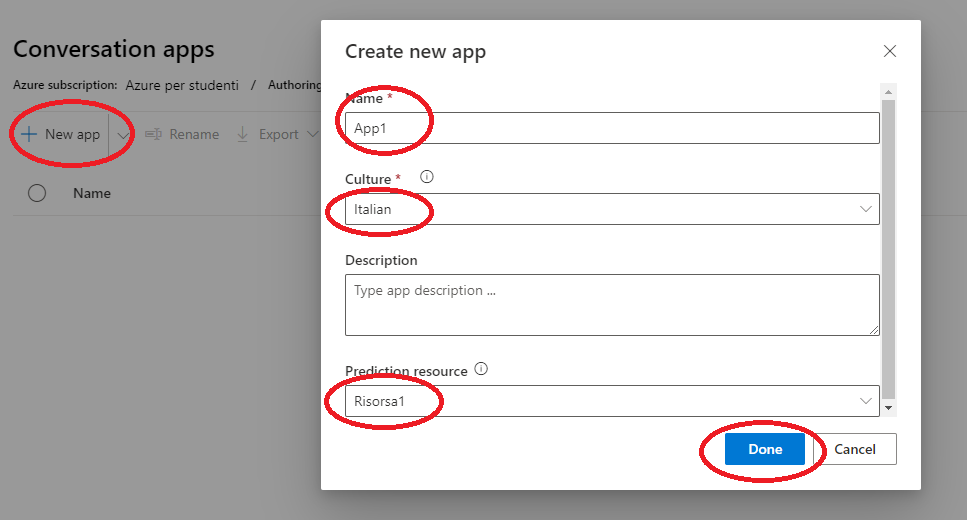
1. Clicca su *“Crea”* e attendi qualche istante; scegliere poi *“Vai al gruppo di risorse”* dove si noterà che sono state create le risorse *Risorsa1* e *Risorsa1-Authoring*
2. Accedere poi a <https://www.luis.ai/> e loggarsi con le proprie credenziali scolastiche



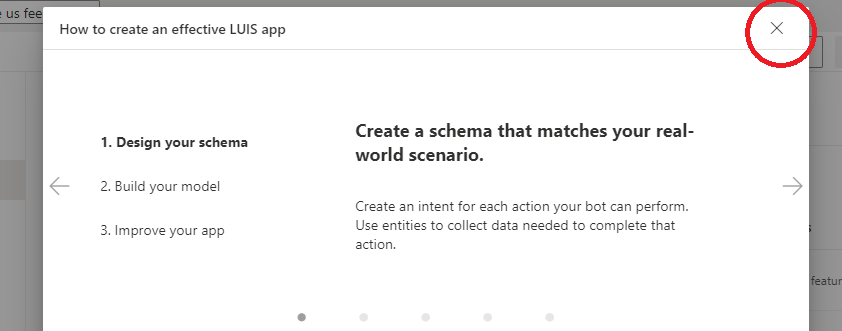
1. Specificare la risorsa di Authoring



1. Scegliere *“New App”*, assegnare un nome all’applicazione (ad esempio *App1*) e scegliere la risorsa creata precedentemente.

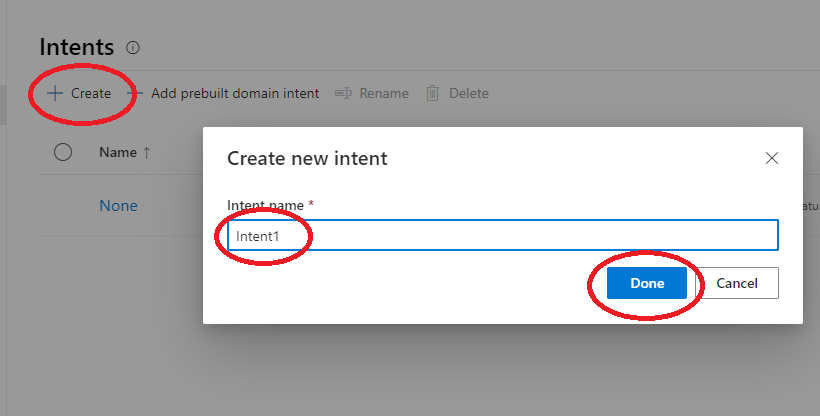


1. Chiudere la finestra di help che appare e passare alla fase successiva di configurazione.

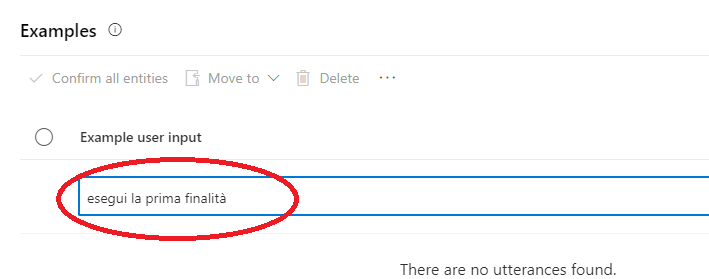


**Parte 2: definizione del modello**

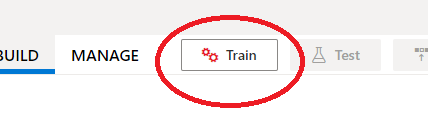
1. Creare la prima ***finalità***(*intent*) ed assegnare un nome (ad esempio *Intent1*)

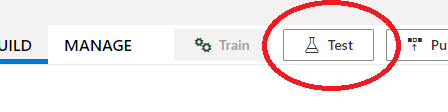


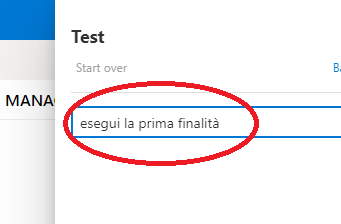
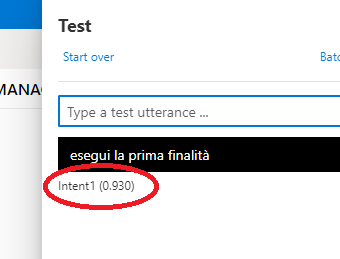
1. Definire la prima ***espressione di riconoscimento*** (*utterance*), ad esempio: *“esegui la prima finalità”* e premere invio



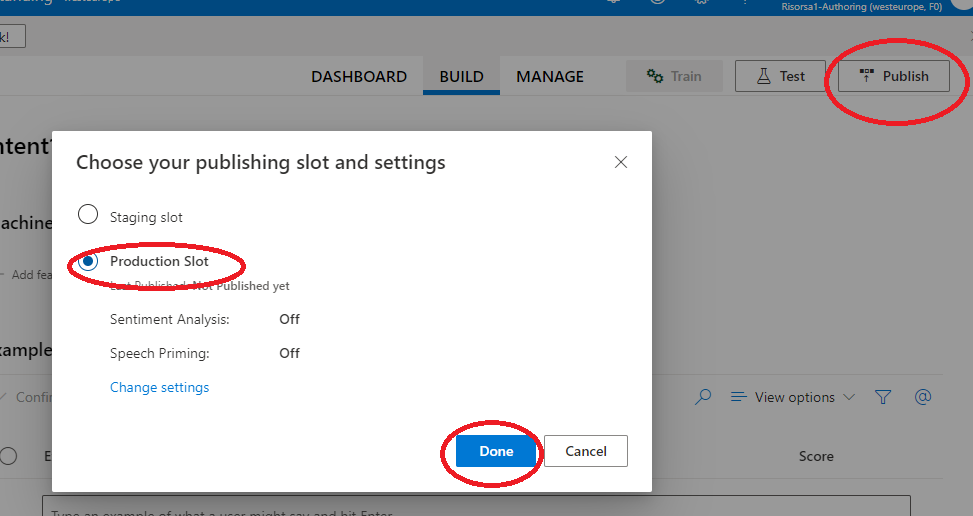
1. Eseguire il training dell’applicazione e poi passare al test, fornendo l’utterance definito e premendo invio; verificare il corretto riconoscimento dell’intent.



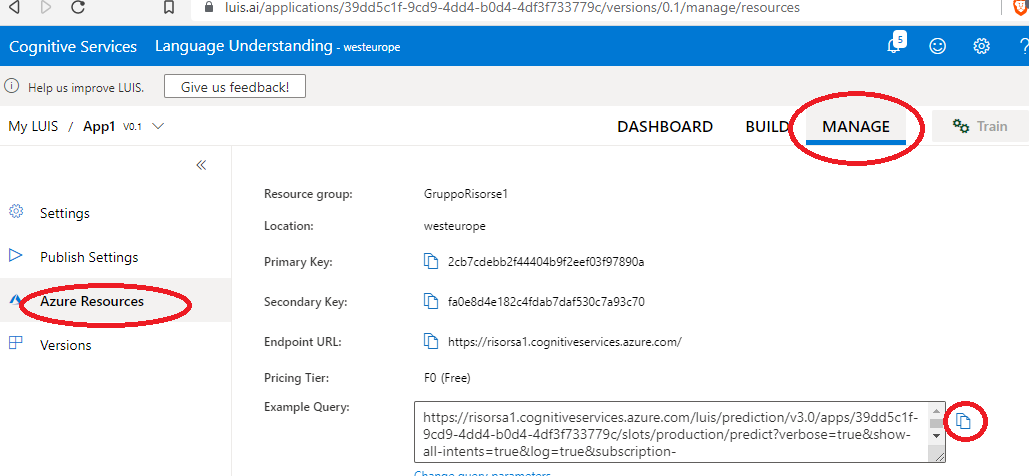


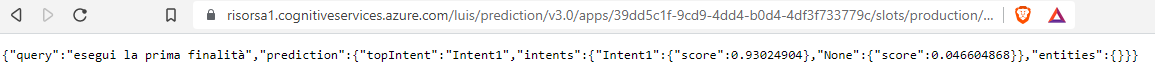
1. Pubblicare l’applicazione



1. Recuperare la stringa di query nella sezione *“Manage” – “Azure Resource”*, provarla nel browser indicando  *“esegui la prima finalità”*  come parametro e verificare il file JSON ricevuto



https://risorsa1.cognitiveservices.azure.com/luis/prediction/v3.0/apps/39dd5c1f-9cd9-4dd4-b0d4-4df3f733779c/slots/production/predict?verbose=true&show-all-intents=true&log=true&subscription-key=2cb7cdebb2f44404b9f2eef03f97890a&query=esegui la prima finalità



1. Recuperare il codice applicazione (in giallo) e il codice sottoscrizione (in verde)

**Sezione B: programma Python di esempio**

1. Installare la libreria

Py –m pip install azure-cognitiveservices-speech

1. Provare il programma di esempio che produrrà il seguente output:

IntentRecognitionResult(result\_id=683ebc97b7d3451d917889fb78a41364, text="Esegui la prima finalità.", intent\_id=Intent1, reason=ResultReason.RecognizedIntent)

import time

import azure.cognitiveservices.speech as speechsdk

#-------------------------------------------------------------

# costanti

#-------------------------------------------------------------

SUBSCRIPTION\_KEY="2cb7cdebb2f44404b9f2eef03f97890a"

APPLICATION\_ID="39dd5c1f-9cd9-4dd4-b0d4-4df3f733779c"

REGION="westeurope"

#-------------------------------------------------------------

# configurazione (proxy opzionale)

#-------------------------------------------------------------

intent\_config = speechsdk.SpeechConfig(

subscription=SUBSCRIPTION\_KEY,

region=REGION)

#intent\_config.set\_proxy("proxy.intranet",3128,"","")

intent\_config.speech\_recognition\_language="it-IT"

#-------------------------------------------------------------

# creazione riconoscitore vocale

#-------------------------------------------------------------

intent\_recognizer = speechsdk.intent.IntentRecognizer(speech\_config=intent\_config)

model = speechsdk.intent.LanguageUnderstandingModel(app\_id=APPLICATION\_ID)

intent\_recognizer.add\_all\_intents(model)

#-------------------------------------------------------------

# funzione di callback chiamata dopo ogni riconoscimento

#-------------------------------------------------------------

def riconosciuto(evt):

print(evt.result)

#-------------------------------------------------------------

# attivazione del riconoscitore

#-------------------------------------------------------------

intent\_recognizer.recognized.connect(riconosciuto)

intent\_recognizer.start\_continuous\_recognition()

#-------------------------------------------------------------

# ciclo di attesa dei riconoscimenti

#-------------------------------------------------------------

while True:

time.sleep(0.5)

#-------------------------------------------------------------

# terminazione del riconoscimento

# (mai attivata in questo esempio)

#-------------------------------------------------------------

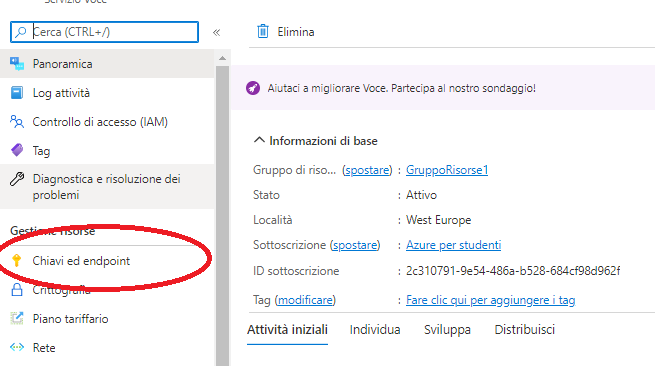
intent\_recognizer.stop\_continuous\_recognition()

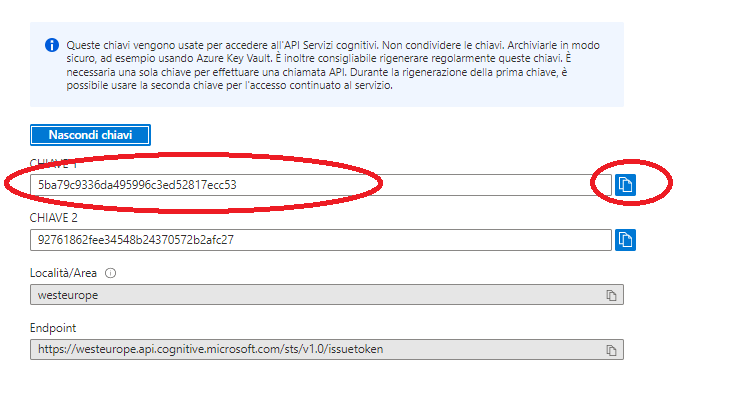
**Parte 3: sintesi vocale**

1. Accedere al portale Azure per la creazione della risorsa di sintesi vocale (usare il link diretto seguente):

<https://ms.portal.azure.com/#create/Microsoft.CognitiveServicesSpeechServices>

1. Creare alla risorsa (ad esempio *Sintesi1*) e, dopo aver completato la creazione, accedere alla sezione *“Chiavi ed endpoint”* e copiare la prima chiave:





1. Provare il programma di esempio:

import time

import azure.cognitiveservices.speech as speechsdk

#-------------------------------------------------------------

# costanti

#-------------------------------------------------------------

SUBSCRIPTION\_KEY="5ba79c9336da495996c3ed52817ecc53"

REGION="westeurope"

#-------------------------------------------------------------

# configurazione (proxy opzionale)

#-------------------------------------------------------------

speech\_config = speechsdk.SpeechConfig(

subscription=SUBSCRIPTION\_KEY,

region=REGION)

speech\_config.speech\_synthesis\_language="it-IT"

#speech\_config.set\_proxy("proxy.intranet",3128,"","")

#-------------------------------------------------------------

# creazione sintetizzatore vocale

#-------------------------------------------------------------

speech\_synthesizer = speechsdk.SpeechSynthesizer(speech\_config=speech\_config)

#-------------------------------------------------------------

# sintesi

#-------------------------------------------------------------

text="ciao, come va?"

result = speech\_synthesizer.speak\_text\_async(text).get()

print(result)

**Parte 4: integrazione con il sensore e l’attuatore**

*L’SDK di Azure gira sotto Windows e Debian 10. Esistono quindi diverse possibilità di integrazione:*

* *Installare la versione Debian 10 per Raspberry, configurare un microfono USB e permettere all’applicazione Python-Azure di interagire via RF24 con il sensore e l’attuatore*
* *Installare l’applicazione Python-Azure sotto Windows, un server Flask-ngrok su Raspberry ed interagire via richieste http-JSON (come già fatto in un’attività precedente).*
* *Installare l’applicazione Python-Azure sotto Windows, un server Mosquitto-ngrok su Raspberry ed interagire via MQTT (come già fatto in un’attività precedente).*