

La nostra Skill per Alexa

Il nostro progetto per il corso di *Human Computer Interaction* è consistito nello sviluppo di una *skill* per Alexa, chiamata Università di Pavia, in grado di fornire informazioni riguardanti gli orari di lezione, i corsi disponibili, i professori (come ad esempio i corsi tenuti dai singoli professori, gli orari di ricevimento e i laboratori di appartenenza) e la mensa. Per chiedere informazioni basta aprire la *skill* dicendo “*Chiedi a Università di Pavia*” e poi sottoporre ad Alexa la nostra domanda. Per esempio, per sapere il menù del giorno della mensa è possibile dire “*Alexa, chiedi a Università di Pavia che cosa c’è da mangiare oggi in mensa*”.

Lo scopo del nostro progetto è stato quello di sviluppare una *skill* per Alexa al fine di studiare alcune delle tecnologie che si celano dietro al funzionamento degli assistenti vocali, i quali attualmente ricoprono uno dei ruoli chiave per quanto riguarda gli strumenti di interazione uomo-macchina negli ambiti domestici, dell’industria e dell’automotive.

Tecnologie utilizzate

Per sviluppare la nostra *skill* abbiamo utilizzato le seguenti tecnologie:

- **Alexa Developer Console:** indispensabile per sviluppare la nostra *skill* lato frontend. Essa ci ha consentito di creare una nuova *skill*, di scegliere un nome, di stabilire una frase di invocazione, di definire gli *Intents* (di cui parleremo successivamente) e i tipi di variabile (Slot Type) da passare ai metodi lato backend.
- **Una connessione HTTPS con certificato SSL:** per permettere la comunicazione tra client e server mediante una connessione sicura. Per i nostri test ci siamo limitati all’utilizzo di *Ngrok*, ovvero un tool che ci consente di generare una connessione sicura in modo gratuito (stando nel limite delle 20 request al minuto). Tuttavia, prima della pubblicazione di una *skill*, è necessario registrare un dominio e affidarsi ad un servizio di hosting (generalmente a pagamento) in grado di permettere la registrazione di un certificato SSL. In alternativa è possibile anche sottoscrivere uno dei piani Amazon AWS (generalmente a pagamento se non limitatamente ad un periodo di tempo).

Flask-Ask: come framework per permettere al web server di interpretare le request provenienti dal client e di generare delle response appropriate

- **Python 3:** come linguaggio di programmazione
- **SQLAlchemy:** come ORM (Object-Relational Mapping) per consentire la comunicazione tra il web server e il database indipendentemente dal tipo di DBMS utilizzato
- **SQLite:** come DBMS

Features

Ecco un elenco delle informazioni che la nostra *skill* è in grado di fornire:

- l'indirizzo email di un professore (*"Chiedi a Università di Pavia l'indirizzo email del professor Mosconi"*)
- informazioni riguardanti l'apertura o la chiusura dell'Università (*"Chiedi a Università di Pavia se martedì l'università è aperta"*)
- i corsi tenuti da un professore (*"Chiedi a Università di Pavia che corsi tiene il professor Virginio Cantoni"*)
- chi è il professore di un determinato corso (*"Chiedi a Università di Pavia chi insegna mst"*)
- dove si trova l'ufficio di un professore (*"Chiedi a Università di Pavia dove si trova l'ufficio del professor Facchinetti"*)
- qual è il laboratorio di un professore (*"Chiedi a Università di Pavia qual è il laboratorio del professor Piastra"*)
- qual è l'orario di ricevimento di un professore e dove riceve (*"Chiedi a Università di Pavia quando riceve il professor Danese"*)
- qual è la data del prossimo esame di ogni corso (*"Chiedi a Università di Pavia quando sarà il prossimo appello di edi"*)
- quando saranno i prossimi appelli disponibili di ogni corso (*"Chiedi a Università di Pavia quando saranno i prossimi appelli di edi"*)
- l'attività in programma dati il giorno e il mese (*"Chiedi a Università di Pavia che attività ci sarà il 30 settembre"*)
- informazioni sull'orario delle lezioni di un determinato corso (*"Chiedi a Università di Pavia orario hci"*)
- informazioni sui corsi in programma in un dato giorno della settimana (*"Chiedi a Università di Pavia che cosa c'è venerdì"*)
- l'aula in cui si terrà la lezione di un corso in un dato giorno della settimana (*"Chiedi a Università di Pavia dove si terrà il corso di edi al mercoledì"*)
- che corso si tiene in un dato giorno ad una data ora (*"Chiedi a Università di Pavia che corso si terrà domani alle 9"*)
- il menù della mensa in un dato giorno della settimana (*"Chiedi a Università di Pavia cosa offre la mensa oggi"*)

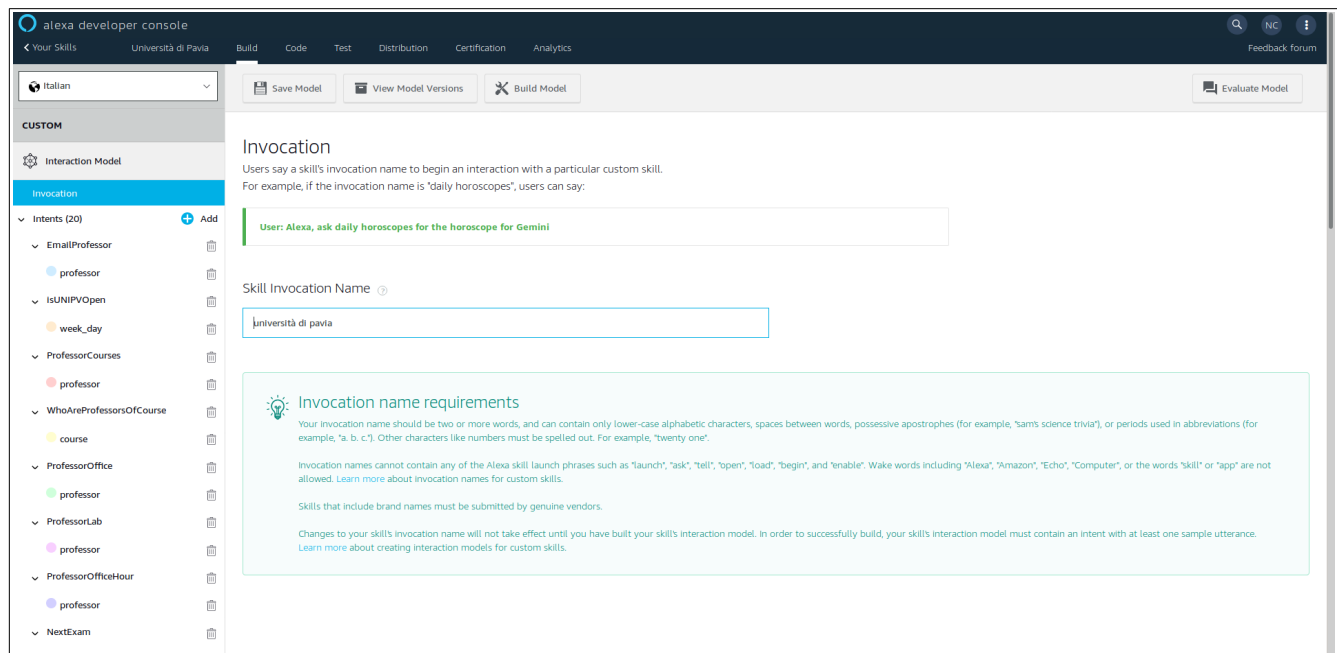
Inoltre è possibile creare un utente in cui viene specificato il nome e l'anno a cui è iscritto (*"Chiedi a Università di Pavia di aggiungere Nicolas secondo anno"*), cambiare l'utente in uso (*"Chiedi a*

Università di Pavia di cambiare l'utente in Marco”), aggiornare l'anno di corso (“Chiedi a Università di Pavia di cambiare anno di Marco in secondo anno”) e rimuovere un account (“Chiedi a Università di Pavia di rimuovere utente Marco”).

Sviluppo passo a passo

Prima di tutto è necessario registrarsi come **Amazon Developer** al fine di poter accedere alla **Alexa Developer Console**. A questo punto è possibile creare una nuova *skill* scegliendo un nome, un modello da aggiungere (nel nostro caso abbiamo scelto un modello *Custom*) e il modo in cui vogliamo hostare il backend, per esempio utilizzando i servizi di Amazon AWS oppure altri servizi a propria scelta. Noi, per i nostri test, abbiamo scelto di affidarci ad un web server generato con **Flask** e **Ngrok**. Quest'ultimo è un software che ci consente di rendere il nostro web server accessibile dall'esterno grazie alla generazione di un URL temporaneo. Tuttavia, nel momento in cui si decide di pubblicare la *skill*, è necessario configurare un web server con un servizio di hosting che permetta la registrazione di un certificato SSL (operazione che generalmente comporta dei costi).

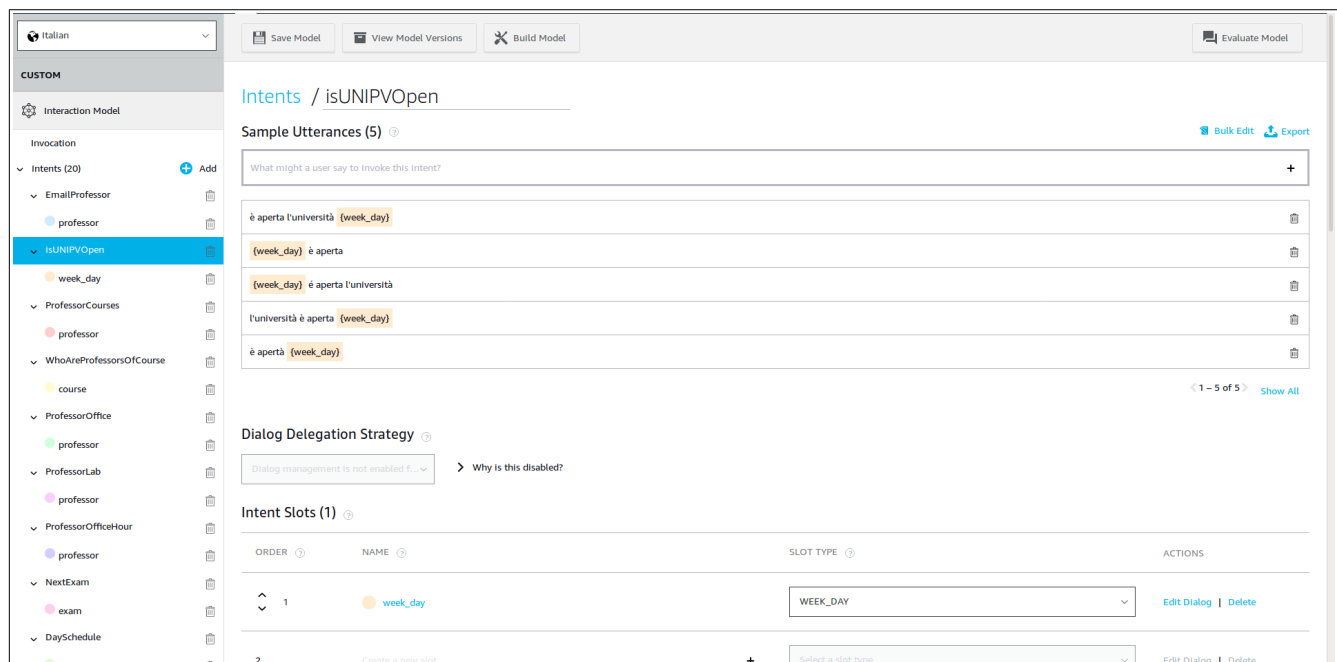
A questo punto possiamo decidere una frase con cui invocare *skill* semplicemente entrando nella sezione “*Invocation*” selezionabile nel menù a sinistra. Nel nostro caso abbiamo deciso di utilizzare “Università di Pavia” come **Skill Invocation Name**.



Successivamente possiamo passare alla definizione del nostro primo **Intent** premendo sul tasto “*Add*” nella sezione “*Intents*” del menù di sinistra. Per spiegare come si crea un *Intent*, vediamo come abbiamo creato l'*Intent* *isUNIPVOpen*. Dopo aver scelto un nome per il nuovo *Intent*, è necessario

definire le frasi necessarie per chiamarlo; tali frasi prendono il nome di **Sample Utterances**. Nel nostro caso, abbiamo inserito le seguenti frasi:

- *è aperta l'università {week_day}*
- *{week_day} è aperta*
- *{week_day} è aperta l'università*
- *l'università è aperta {week_day}*
- *è aperta {week_day}*



La parola tra le parentesi graffe è la **variabile** (o più precisamente **Intent Slot**) che nel nostro caso prenderà il valore di uno dei giorni della settimana oppure potrà contenere il valore oggi o domani. Nel momento in cui creiamo un nuovo *Intent* è necessario definire il tipo di ogni *Intent Slot* che può essere uno di quelli predefiniti come ad esempio `AMAZON.DayOfWeek`, oppure custom come nel nostro caso in cui abbiamo creato un tipo `WEEK_DAY` in modo tale da poter aggiungere anche i valori oggi e domani oltre ai giorni della settimana in italiano. Per creare un nuovo *Slot Type* è sufficiente premere sul tasto “Add” nella sezione “Slot Types” del menù di sinistra, scegliere un nuovo nome per il tipo e scegliere i valori che possono essere assunti dalle variabili dichiarate con il nuovo tipo.

Slot Types / WEEK_DAY

Slot Values (9)

Enter a new value for this slot type

VALUE	ID (OPTIONAL)	SYNONYMS (OPTIONAL)	
oggi	Enter ID	Add synonym	+
domani	Enter ID	Add synonym	+
domenica	Enter ID	Add synonym	+
sabato	Enter ID	Add synonym	+
venerdi	Enter ID	Add synonym	+
giovedi	Enter ID	Add synonym	+
mercoledi	Enter ID	Add synonym	+
martedi	Enter ID	Add synonym	+
lunedì	Enter ID	Add synonym	+

A questo punto possiamo salvare e generare il nuovo modello premendo rispettivamente su “Save Model” e “Build Model”. Se la procedura di costruzione del nuovo modello è avvenuta con successo riceveremo un feedback positivo.

Ora non ci resta che installare **Ngrok**, seguendo le istruzioni riportate sul [sito ufficiale](#), e passare allo sviluppo del backend. Quindi installiamo **Flask**:

```
$ pip install flask
```

e **Flask-Ask** (facendo il clone dal repository su *GitHub* e seguendo le indicazioni contenute nel file `README.md` nella sezione “Development”).

Creiamo un nuovo progetto *Python*, creiamo un nuovo file `app.py`, importiamo *Flask* e *Flask-Ask* e istanziamo `app` e `ask`:

```
from flask import Flask
from flask_ask import Ask, statement, question

app = Flask(__name__)
ask = Ask(app, '/')
```

Utilizzando il **decoratore** `@app.route('/')` definiamo la stringa che verrà stampata a schermo nel momento in cui visiteremo la *root directory* del nostro web server. Tale metodo è utile per verificare che il web server sia effettivamente raggiungibile.

```
@app.route('/')
def home():
    return 'Hi! I am UNIPV'
```

Ora creiamo un metodo che farà in modo che Alexa ponga una domanda all'utente nel momento in cui la nostra *skill* viene aperta utilizzando il decoratore `@ask.launch` per definire il metodo da eseguire all'avvio. Più precisamente il seguente metodo domanderà all'utente “*Bentornato Marco, in cosa posso esserti utile?*”, dove *Marco* è il nome dell'utente registrato. In tal modo la *skill* rimarrà in ascolto di un eventuale risposta (o meglio in attesa di una richiesta di informazione) da parte dell'utente.

```
@ask.launch
def start():
    user, year, flag = check_active_user()
    if not flag:
        speech_text = 'Per effettuare il primo accesso devi dirmi il tuo nome
e il tuo anno di corso oppure se sei già registrato rispondi con "cambia utente
in" seguito dal tuo nome'
        return question(speech_text)
    speech_text = f'Bentornato {user}, in cosa posso esserti utile?'
    return question(speech_text)
```

A questo punto utilizziamo il decoratore `@ask.intent("isUNIPVOpen")`, per specificare il metodo che dovrà essere eseguito nel momento in cui verrà invocato il nostro primo *Intent* che abbiamo chiamato `isUNIPVOpen`:

```
@ask.intent('isUNIPVOpen')
def unipv_open(week_day):
    return statement(is_unipv_open(week_day))
```

Tale metodo farà in modo di restituire una stringa di caratteri generata dal metodo `is_unipv_open()` che verrà pronunciata da Alexa prima di chiudere la *skill*.

Il metodo `is_unipv_open()` è il seguente:

```
def is_unipv_open(week_day):
    days_ita = {'0':'lunedì', '1':'martedì', '2':'mercoledì', '3':'giovedì',
                '4':'venerdì', '5':'sabato', '6':'domenica'}
    week_day = str(week_day).lower()
    if week_day == 'oggi':
        week_day = days_ita[str(datetime.datetime.today().weekday())]
    if week_day == 'domani':
        week_day = days_ita[str(datetime.datetime.today().weekday()+1)]
    if week_day == 'sabato' or week_day == 'domenica':
        speech_text = "È chiusa!"
    elif week_day in ['lunedì', 'martedì', 'mercoledì', 'giovedì', 'venerdì']:
        speech_text = "È aperta!"
    else:
        speech_text = "L'Università di Pavia è aperta dal lunedì al venerdì"
    return speech_text
```

dove `week_day` è una variabile che contiene il valore contenuto nell'*Intent Slot* `{week_day}` che abbiamo precedentemente definito grazie alla *Alexa Developer Console*.

Ora che abbiamo definito il nostro primo *Intent* e scritto il primo metodo che tale *Intent* dovrà invocare facendo una richiesta HTTP al web server, non ci resta che tornare alla *Alexa Developer Console* e

selezionare la voce “*Endpoint*” dal menù di sinistra. A questo punto dobbiamo specificare che vogliamo gestire il nostro *Endpoint* utilizzando il protocollo HTTPS; perciò dobbiamo specificare l’indirizzo HTTPS per ogni regione e dare informazioni sul dominio che ospita il backend.

Nel caso in cui si desidera semplicemente testare il funzionamento della *skill* con *Ngrok*, sarà sufficiente avviare il web server di *Flask*:

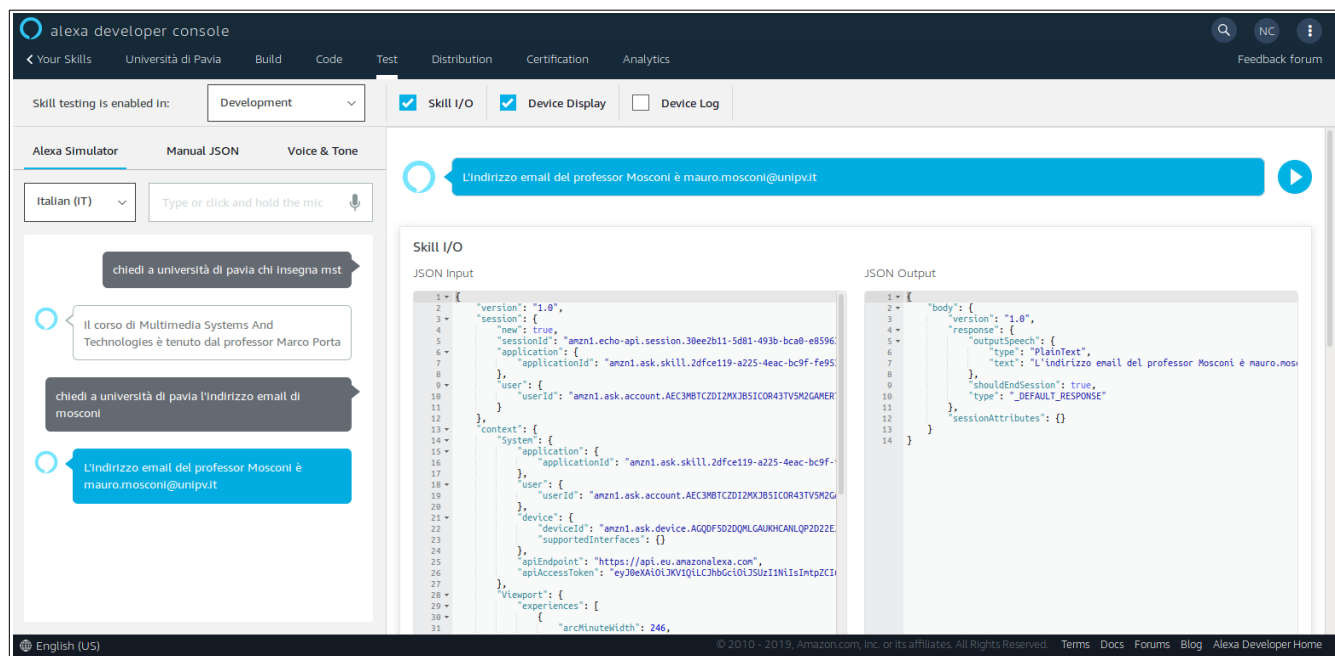
```
$ python app.py
```

e *Ngrok*:

```
$ ngrok http 5000
```

Successivamente specifichiamo l’indirizzo HTTPS dell’*Endpoint* (che apparirà sul terminale una volta avviato *Ngrok*) utilizzando la *Alexa Developer Console* e selezioniamo la voce “*My development endpoint is a sub-domain that has a wildcard certificate from a certificate authority*”.

A questo punto, andando nella sezione “*Test*” dell’*Alexa Developer Console*, è possibile utilizzare l’assistente vocale per testare il corretto funzionamento del nostro primo *Intent*. Proviamo a dire “*Chiedi a Università di Pavia se l’università è aperta*” e controlliamo che Alexa ci risponda correttamente.



The screenshot displays the Alexa Developer Console interface, specifically the 'Test' tab. The top navigation bar includes links for 'Your Skills', 'Università di Pavia', 'Build', 'Code', 'Test', 'Distribution', 'Certification', and 'Analytics'. The 'Test' section is active, showing a 'Skill testing is enabled in: Development' dropdown. Below this, there are tabs for 'Alexa Simulator', 'Manual JSON', and 'Voice & Tone'. The 'Voice & Tone' tab is selected, showing a voice input area with a microphone icon and a text input field. Below the input area, there are three sample intents: 'chiedi a università di pavia chi insegna mst', 'Il corso di Multimedia Systems And Technologies è tenuto dal professor Marco Porta', and 'chiedi a università di pavia l'indirizzo email di mosconi'. The 'Skill I/O' section displays the JSON input and output for the selected intent. The JSON input shows a request with session information and user details. The JSON output shows a response with an output speech containing the email address 'L'indirizzo email del professor Mosconi è mauro.mosconi@unipv.it'.

Inoltre, se si ha a disposizione un dispositivo Alexa dedicato, oppure uno smartphone o un tablet sul quale è installata l’app di Alexa, è possibile testare la *skill* direttamente sul dispositivo, a patto che quest’ultimo sia collegato all’account Amazon con cui ci siamo registrati come developer.

In modo analogo è possibile creare altri *Intents* che chiamano metodi in grado di interagire con un database. Per esempio, uno dei modi più semplice per gestire una base di dati con *Python* e *Flask* è quello di utilizzare *SQLite* come DBMS e *SQLAlchemy* come ORM.