AtAVi

Accoglienza tramite assistente virtuale

Università di Padova

Manuale Manutentore v2.00

answer.group17@gmail.com

Informazioni sul documento

Nome Documento:	*ManualeManutentore_v.2.00*
Versione:	2.00
Data di Creazione:	03 Aprile 2017
Data ultima Modifica:	11 Maggio 2017
Stato:	**Approvato**
Redazione:	Abdelilah Lahmer, Andrei Tabacariu, Alessandro Bari, Matteo Maran.
Verifica:	Francesca Del Nin
Approvazione:	Leonardo Zoso
Uso:	Esterno
Distribuzione:	*Answer*
Destinato a:	Prof. Vardanega, Prof. Cardin, *Zero12 s.r.l.*
Email di riferimento:	`answer.group17@gmail.com`

Sommario

Documento contenente il Manuale Manutentore per il progetto *AtAVi* del gruppo *Answer*.

Registro delle modifiche

Versione	Data	Autore	Ruolo	Descrizione
2.00	2017-05-12	Leonardo Zoso	*Responsabile di progetto*	Approvazione del documento. Causa consegna del documento.
1.06	2017-05-11	Francesca Del Nin	*Verificatore*	Verifica del documento. Causa modifiche effettuate.
1.05	2017-05-11	Abdelilah Lahmer	*Verificatore*	Aggiunti riferimenti ipertestuali. Causa incremento documento.
1.04	2017-04-27	Francesca Del Nin	*Verificatore*	Verifica del documento. Causa modifiche effettuate.
1.03	2017-04-26	Andrei Tabacariu	*Verificatore*	Modificato contenuto §7 Funzionalità. Causa resoconto consegna.
1.02	2017-04-25	Alessandro Bari	*Verificatore*	Aggiornato §10 Glossario interno. Causa resoconto consegna.
1.01	2017-04-24	Matteo Maran	*Verificatore*	Tolti i riferimenti ai documenti. Causa resoconto consegna.
1.00	2017-04-10	Abdelilah Lahmer	*Responsabile di progetto*	Approvazione del documento. Causa consegna del documento.

Versione	Data	Autore	Ruolo	Descrizione
0.09	2017-04-10	Matteo Maran	*Verificatore*	Verifica del documento. Causa cambiamenti effettuati.
0.08	2017-04-07	Eugen Saraci	*Verificatore*	Aggiunta sezione §10 Glossario interno. Causa incremento del documento.
0.07	2017-04-07	Leonardo Zoso	*Verificatore*	Aggiunta sezione §8 Persistenza Dati. Causa incremento del documento.
0.06	2017-04-06	Alessandro Bari	*Verificatore*	Finita sezione §7 Funzionalità. Causa incremento del documento.
0.05	2017-04-05	Alessandro Bari	*Verificatore*	Aggiunta sezione §7 Funzionalità. Causa incremento del documento.
0.04	2017-04-05	Andrei Tabacariu	*Verificatore*	Aggiunta sezione §?? Configurazione dell'ambiente di lavoro. Causa incremento del documento.
0.03	2017-04-04	Francesca Del Nin	*Verificatore*	Aggiunta sezione §2 Tecnologie Utilizzate. Causa incremento del documento.

Versione	Data	Autore	Ruolo	Descrizione
0.02	2017-04-03	Eugen Saraci	*Verificatore*	Aggiunta sezione §1 Introduzione. Causa incremento del documento.
0.01	2017-04-03	Alessandro Bari	*Verificatore*	Definizione struttura generale documento. Causa definizione del documento.

Indice

1 Introduzione

1.1 Scopo del Documento

Il documento rappresenta il *Manuale manutentore* per il software *AtAVi* sviluppato dal gruppo *Answer*. Esso ha lo scopo di illustrare al manutentore del software le modalità di installazione e utilizzo, i requisiti necessari per poterlo utilizzare e le librerie e frameworks$_{GI}$ esterni utilizzati per lo sviluppo dell'applicazione. Per ogni funzionalità offerta dall'applicazione viene presentata la lista dei packages$_{GI}$ e dei moduli che la costituiscono, così da aiutarlo in caso di anomalie.

1.2 Scopo del prodotto

Lo scopo del prodotto richiesto è implementare un assistente virtuale$_{GI}$ che svolga la funzione di accoglienza clienti. Il prodotto comprenderà:

- un'interfaccia Web$_{GI}$ di interazione con l'utente utilizzatore del sistema;
- gestione dei servizi *AWS Lambda*$_{GI}$ per l'interazione con le API$_{GI}$ dell'assistente virtuale;
- un interfacciamento con *Slack*$_{GI}$, in modo da poter notificare gli interessati sull'arrivo del cliente.

1.3 Glossario

Allo scopo di rendere più semplice e chiara la comprensione di alcuni termini tecnici, ambigui o abbreviazioni presenti nei documenti, viene resa disponibile §10 *Glossario interno*, nella quale verranno raccolte le varie spiegazioni. Per evidenziare un termine presente nel *Glossario interno* di questo documento, esso verrà marcato con il pedice $_{GI}$.

1.4 Riferimenti

1.4.1 Riferimenti informativi

- Materiale del corso di Ingegneria del Software:
  - documentazione: `http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L08.pdf`;

  - diagrammi delle classi:
    `http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/E02a.pdf`;
  - diagrammi dei packages:
    `http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/E02b.pdf`;
  - diagrammi di sequenza:
    `http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/E03a.pdf`.

- Documentazione del framework AngularJS$_{GI}$:
  - documentazione generica:
    `https://docs.angularjs.org/guide`;
  - documentazione servizio:
    `https://docs.angularjs.org/api/ng/service`.
- documentazione della piattaforma Node.js:
  `https://nodejs.org/api`.
- documentazione dei servizi AWS:
  - AWS Lambda:
    `https://aws.amazon.com/it/lambda`;
  - AWS DynamoDB:
    `https://aws.amazon.com/it/dynamodb`;
  - AWS Api Gateway:
    `https://aws.amazon.com/it/api-gateway/`.
- documentazione bootstrap:
  `http://getbootstrap.com/getting-started`.

2 Tecnologie Utilizzate

2.1 AngularJS

Per lo sviluppo della parte Front-End dell'applicazione si è scelto di usare il framework JavaScript AngularJS, che permette lo sviluppo di applicazioni in singola pagina. Esso consente di utilizzare HTML$_{GI}$ come linguaggio di template e permette di estenderne la sintassi per esprimere i componenti dell'applicazione in modo chiaro e conciso. Il *data binding*$_{GI}$ e il *dependency injection*$_{GI}$ di AngularJS eliminano gran parte del codice che altrimenti si sarebbe dovuto scrivere. Tutto ciò avviene all'interno del browser, che lo rende un partner ideale con qualsiasi tecnologia server.

2.2 Node.js

Per lo sviluppo della parte Back-End dell'applicazione si è deciso di utilizzare la piattaforma event-driven$_{GI}$ Node.js, basata sul motore JavaScript V8$_{GI}$. Esso permette di realizzare applicazioni Web, come appunto *AtAVi*, utilizzando il linguaggio JavaScript, tipicamente client-side, per la scrittura server-side. La caratteristica principale di Node.js risiede nella possibilità che offre di accedere alle risorse del sistema operativo in modalità event-driven non sfruttando il classico modello basato su processi o thread concorrenti, utilizzato dai classici web server. Il modello event-driven, o programmazione ad eventi, si basa su un concetto piuttosto semplice: si lancia una azione quando accade qualcosa. Ogni azione quindi risulta asincrona a differenza dei pattern di programmazione più comune in cui una azione succede ad un'altra solo dopo che essa è stata completata. Ciò garantisce una certa efficienza delle applicazioni grazie ad un sistema di callback$_{GI}$ gestito a basso livello a runtime.

2.3 AWS DynamoDB

Come DBMS$_{GI}$, per l'applicativo *AtAVi*, si è utilizzato DynamoDB. Amazon DynamoDB è un servizio di database NoSQL veloce e flessibile pensato per tutte le applicazioni che richiedono una latenza costante non superiore a una decina di millisecondi su qualsiasi scala. È un database cloud interamente gestito che supporta sia i modelli di document store sia quelli di tipo chiave-valore. Si tratta di un modello di dati flessibile in grado di assicurare prestazioni affidabili, perfetto per applicazioni Web o per dispositivi mobili, per videogiochi, per tecnologie pubblicitarie, per l'Internet of Things e per molti altri casi d'uso.

2.3.1 DynamoDB.DocumentClient

DocumentClient è una classe facente parte del dominio DynamoDB, utilizzata per la gestione dei dati e la comunicazione con il database.

2.4 Alexa Skills Kit

Alexa Skills Kit è un SDK$_{GI}$ che permette la creazione e lo sviluppo di una skill, tramite la definizione di Intent$_{GI}$ e Utterances$_{GI}$.

2.5 AWS Lambda

AWS Lambda è un servizio che consente di eseguire codice senza dover effettuare il provisioning$_{GI}$ né gestire server. Le tariffe sono calcolate in base ai tempi di elaborazione, perciò non viene addebitato alcun costo quando il codice non è in esecuzione. Con Lambda, può essere eseguito il codice per qualsiasi tipo di applicazione o servizio di back-end, senza alcuna amministrazione. Una volta caricato il codice, Lambda si prende carico delle azioni necessarie per eseguirlo e ricalibrarne le risorse con la massima disponibilità. Il codice può essere configurato in modo che venga attivato automaticamente da altri servizi AWS oppure in modo che venga richiamato direttamente da un qualsiasi app Web o mobile.

2.6 AWS API Gateway

Amazon API Gateway è un servizio completamente gestito che semplifica agli sviluppatori la creazione, la pubblicazione, la manutenzione, il monitoraggio e la protezione delle API su qualsiasi scala. Bastano pochi clic sulla Console di gestione AWS per creare un'API che agisca come porta d'entrata attraverso la quale le applicazioni possano accedere a dati, logica di business o funzionalità dei servizi di back-end, ad esempio i carichi di lavoro in esecuzione su AWS Lambda o le applicazioni Web. Amazon API Gateway gestisce tutte le attività di accettazione ed elaborazione relative a centinaia di migliaia di chiamate API simultanee, inclusi gestione del traffico, controllo di accessi e autorizzazioni, monitoraggio e gestione delle versioni delle API. Amazon API Gateway non prevede alcuna tariffa minima né costi di attivazione; saranno addebitati solamente le chiamate API ricevute e i volumi di dati trasferiti in uscita.

2.7 Bootstrap

Bootstrap è un framework per la creazione di siti e applicazioni per il Web. Esso contiene modelli di progettazione basati su HTML e CSS, sia per la tipografia, che per le varie componenti dell'interfaccia come moduli, pulsanti e navigazione, così come alcune estensioni opzionali di JavaScript.

2.8 SNS

SNS (Simple Notification Service) è un servizio di notifiche fornito da Amazon Web Services, utilizzato per gestire l'invio dei messaggi su Slack.

2.9 LWA

LWA (Login With Amazon) è un servizio Amazon che permette, una volta effettuata l'autenticazione, l'utilizzo del servizi AVS e l'utilizzo della Skill.

3 Prerequisiti

3.1 Account Amazon

È necessario avere un account Amazon per utilizzare AtAVi: Accoglienza tramite Assistente Virtuale; se non si è già in possesso di un account lo si può creare su
`https://www.amazon.com/ap/register`.

3.2 Installazione NodeJS

Per l'utilizzo dell'applicazione è necessaria l'installazione della piattaforma NodeJS e del Node Package Manager (npm). È possibile eseguire l'installazione seguendo le istruzioni su
`https://nodejs.org/it/download/`.

3.3 Installazione AWS CLI

Installare AWS CLI seguendo le istruzioni su
`https://aws.amazon.com/it/cli/`.

3.4 Configurazione Apex

Per permettere il deployment delle funzioni lambda è richiesto l'utilizzo di *Apex*.
Per l'installazione e la configurazione di *Apex* seguire la procedura su
`http://apex.run`.

4 Configuazione AtAVi

4.1 Configurazione AWS

Per configurare Amazon Web Services è possibile utilizzare una procedura automatica o una manuale.

4.1.1 Configurazione automatica

Per installare *AtAVi*:

1. Aprire il terminale all'interno della cartella *AtAVi-Software* e dare il comando

```
node aws-configure.js
```

2. Copiare l'ARN del Topic SNS ottenuto dal comando precedente ed incollarlo nella variabile presente in cima al file situato in:

   `AtAVi-Software/lambda/lambda-functions/skill-functions/functions/core/index.js`

3. Copiare l'ARN del ruolo ottenuto dallo script precedente ed incollarlo nella voce role del file json situato in:

   `AtAVi-Software/lambda/lambda-functions/admin-functions/auth/project.json`

   `AtAVi-Software/lambda/lambda-functions/admin-functions/rules/project.json`

   `AtAVi-Software/lambda/lambda-functions/util-functions/project.json`

   `AtAVi-Software/lambda/lambda-functions/skill-functions/project.json`

4. Continuare l'installazione proseguendo da §4.1.2.6.

4.1.2 Configurazione manuale

Aprire il terminale all'interno della cartella *AtAVi-Software*.

4.1.2.1 Creazione role

Creare un ruolo *invoke_lambda* con il documento *filepolicy.json* con il seguente comando

```
aws iam create-role --role-name invoke_lambda --assume-role-policy-
document file://filepolicy.json
```

Copiare l'ARN del ruolo ottenuto dal comando precedente ed incollarlo nella voce role del file json situato in:

```
AtAVi-Software/lambda/lambda-functions/admin-functions/auth/project.
json
```

```
AtAVi-Software/lambda/lambda-functions/admin-functions/rules/project.
json
```

```
AtAVi-Software/lambda/lambda-functions/util-functions/project.json
```

```
AtAVi-Software/lambda/lambda-functions/skill-functions/project.json
```

4.1.2.2 Aggiunta policy a role

Aggiungere le policy che permettono l'accesso e l'utilizzo di DynamoDB, Lambda e SNS con i seguenti comandi

```
aws iam attach-role-policy --role-name invoke_lambda
    --policy-arn
    arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess

aws iam attach-role-policy --role-name invoke_lambda
    --policy-arn arn:aws:iam::aws:policy/AWSLambdaExecute

aws iam attach-role-policy --role-name invoke_lambda
    --policy-arn
    arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole

aws iam attach-role-policy --role-name invoke_lambda
    --policy-arn
    arn:aws:iam::aws:policy/AWSLambdaInvocation-DynamoDB

aws iam attach-role-policy --role-name invoke_lambda
    --policy-arn arn:aws:iam::aws:policy/AmazonSNSFullAccess
```

4.1.2.3 Creazione Topic SNS Creare un topic SNS che invochi la funzione per l'invio di un messaggio su Slack con il comando

```
aws sns create-topic --name sendSlackMessage
```

Il comando ritornerà un ARN che va incollato nella variabile presente in cima al file situato in:

```
AtAVi-Software/lambda/lambda-functions/skill-functions/functions/
core/index.js
```

4.1.2.4 Creazione tabelle database Creare nel database le tabelle auth, curiosities, customer, customer_rules e employee_rules con i seguenti comandi

```
aws dynamodb create-table --table-name auth
    --attribute-definitions
    AttributeName=id,AttributeType=S --key-schema
    AttributeName=id,KeyType=HASH --provisioned-throughput
    ReadCapacityUnits=5,WriteCapacityUnits=5

aws dynamodb create-table --table-name curiosities
    --attribute-definitions
    AttributeName=id,AttributeType=S --key-schema
    AttributeName=id,KeyType=HASH --provisioned-throughput
    ReadCapacityUnits=5,WriteCapacityUnits=5

aws dynamodb create-table --table-name customer
    --attribute-definitions
    AttributeName=id,AttributeType=S --key-schema
    AttributeName=id,KeyType=HASH --provisioned-throughput
    ReadCapacityUnits=5,WriteCapacityUnits=5

aws dynamodb create-table --table-name customer_rules
    --attribute-definitions
    AttributeName=id,AttributeType=S --key-schema
    AttributeName=id,KeyType=HASH --provisioned-throughput
    ReadCapacityUnits=5,WriteCapacityUnits=5

aws dynamodb create-table --table-name employee_rules
    --attribute-definitions
```

```
AttributeName=id,AttributeType=S --key-schema
AttributeName=id,KeyType=HASH --provisioned-throughput
ReadCapacityUnits=5,WriteCapacityUnits=5
```

4.1.2.5 Struttura API Gateway

Importare la struttura di API Gateway facendo riferimento al file *API.json* con il comando

```
aws apigateway import-rest-api --body 'file://API.json' --region eu-
west-1
```

4.1.2.6 Configurazione Slack

1. Creare un *Incoming Webhook* su Slack collegandosi a
   `https://my.slack.com/services/new/incoming-webhook/`
   e incollarlo nella variabile presente in cima al file

   ```
   AtAVi-Software/lambda/lambda-functions/util-functions/functions/
   sendSlack/index.js
   ```

2. Cambiare, nel caso in cui si ritenga opportuno, il canale Slack di default (#general), sostituendo il valore della variabile in cima al file

   ```
   AtAVi-Software/lambda/lambda-functions/skill-functions/functions/
   core/index.js
   ```

4.1.2.7 Deployment funzioni

Per eseguire il deployment delle funzioni:

1. Aprire un terminale, nella directory `AtAVi-Software/lambda` e installare le dipendenze per utilizzare gli script con il comando

   ```
   npm install
   ```

2. Eseguire il comando

   ```
   npm run install-prod
   ```

3. Aprire un terminale, andare su `AtAVi-Software/lambda/lambda-functions/admin-funct`
4. Eseguire il comando

   ```
   apex deploy -E env.json
   ```

   `enj.json` contiene la stringa con cui vengono firmati i JWT.
5. Andare su `AtAVi-Software/lambda/lambda-functions/admin-functions/auth`
6. Eseguire il comando

   ```
   apex deploy -E env.json
   ```

   `enj.json` contiene la stringa con cui vengono firmati i JWT.
7. Spostarsi su `AtAVi-Software/lambda/lambda-functions/skill-functions`
8. Eseguire il comando

   ```
   apex deploy
   ```

9. Ripetere i punti 5 e 6 su `AtAVi-Software/lambda/lambda-functions/util-functions`

4.1.2.8 Configurazione API Gateway

Per il corretto funzionamento è necessario associare alle risorse di API Gateway le rispettive funzioni lambda.

1. Collegarsi a
   `https://eu-west-1.console.aws.amazon.com/apigateway/home?region=eu-west-1#/apis`
   e cliccare su *AtAVi*

Per ogni risorsa:

1. Cliccare su *Integration Request*
2. **Integration type**: Lambda Function
3. **Lambda Region**: eu-west-1
4. **Lambda Function**: selezionare la funzione lambda seguendo la tabella

Metodo	Risorsa	Funzione Lambda
/auth	POST	auth_login
/auth/changepsw	POST	auth_changePsw
/rules/customer	GET	rules_getAllCustomerRules
	POST	rules_addCustomerRule
/rules/customer/{id}	DELETE	rules_deleteCustomerRule
	GET	rules_getCustomerRule
	PUT	rules_updateCustomerRule
/rules/employee	GET	rules_getAllEmployeeRules
	POST	rules_addEmployeeRule
/rules/employee/{id}	DELETE	rules_deleteEmployeeRule
	PUT	rules_updateEmployeeRule

Figura 1: Associazione Metodo/Risorsa - Funzione Lambda

4.1.3 Popolamento database curiosità

Per popolare la tabella curiosità:

1. Collegarsi a
   `https://eu-west-1.console.aws.amazon.com/lambda/home?region=eu-west-1#/functions?display=list`
   o accedendo a Lambda dalla Management Console
2. Cliccare su util_addCur
3. Cliccare su *Test*

4.2 Configurazione Amministrazione

Per permettere al pannello di amministrazione di funzionare è necessario collegarsi a
`https://eu-west-1.console.aws.amazon.com/apigateway/home?region=eu-west-1#/apis`
cliccare su AtAVi e copiare l'ID dell'API, formato da un valore di 8 caratteri alfanumerici nel file

`AtAVi-Software/admin/site/modules/module.js`

sostituendolo a [YOUR_API_ID].

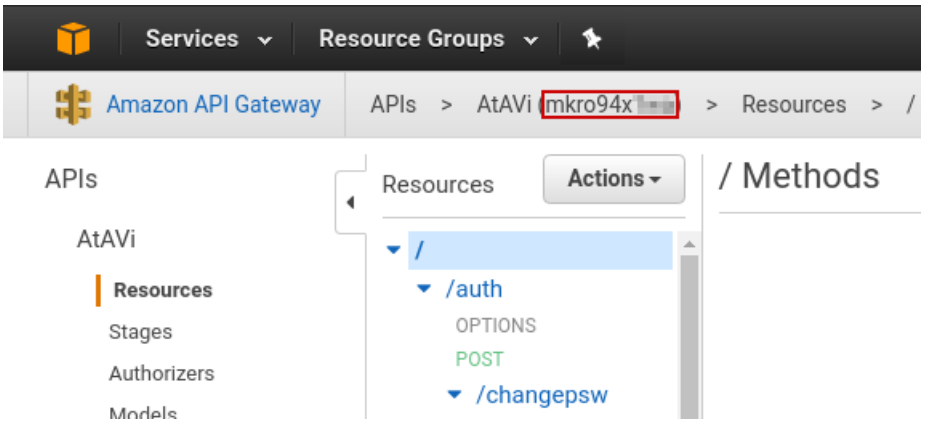

Figura 2: API Gateway - É evidenziato l'ID dell'API

4.3 Configurazione skill

È necessario configurare la skill per poter interagire con il sistema.
Per prima cosa è necessario prendere nota dell'ARN della lambda skill_core:

1. Collegarsi a
   `https://eu-west-1.console.aws.amazon.com/lambda/home?region=eu-west-1#/functions?display=list`
   o accedere al servizio Lambda dalla Management Console di AWS,
2. Cliccare sulla lambda

   `skill_core`

3. Annotare l'ARN posto in alto a destra

Figura 3: Lambda - ARN della funzione lambda skill_core

Configurare skill da Alexa Skills Kit:

1. Accedere a `https://developer.amazon.com/edw/home.html#/skills`,
2. Cliccare su *Add a New Skill* e compilare la sezione Skill Information:
   - **Skill type**: Custom,

- **Language**: English US,
- **Name**: AtAVi,
- **Invocation name**: hi,
- **Audio Player**: no(default)

3. Procedere con *Next*,
4. In *Interaction Model* compilare la sezione *Intent Schema* con il contenuto del file *intentSchema.json*,
5. In *Custom Slot Type* cliccare su *Add Slot Type* e compilare i campi nel seguente modo:
   Nomi italiani:

   - **Type**: first_name,
   - **Value**: Copiare e incollare i valori da *first-name-slot.txt*.

   Cognomi italiani:

   - **Type**: last_name,
   - **Value**: Copiare e incollare i valori da *last-name-slot.txt*.

   Impiegati:

   - **Type**: employee,
   - **Value**: Inserire nome e cognome degli impiegati dell'azienda

   Aziende:

   - **Type**: company,
   - **Value**: Inserire i nomi delle aziende

6. In *Sample Utterances* inserire il contenuto del file *utterances.txt*,
7. Procedere con *Next*,
8. In *Configuration* configurare le voci come segue:

   - **Service Endpoint Type**: AWS Lambda ARN,
   - **Geographical region**: Europe,
   - Inserire l'ARN della lambda *skill_core*

9. Cliccare su *Save*.

5 Testing

5.1 Installare tutte le dipendenze

Installare tutte le dipendenze necessario al testing aprendo un terminale nella directory `AtAVi-Software/lambda` e digitando i seguenti comandi

```
npm run install-all
```

```
sudo npm install -g mocha
```

```
sudo npm install -g istanbul
```

5.2 Lanciare DynamoDB in locale

1. Collegarsi a `http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DynamoDBLocal.html` e scaricare DynamoDB Local,
2. Scompattare l'archivio scaricato,
3. Spostarsi nella cartella dynamodb_local_latest,
4. Eseguire il comando

   ```
   java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -port 8000
   ```

Il processo deve restare in esecuzione durante tutta la fase di testing.
Per interrompere il processo premere CTRL + C nel terminale

5.3 Configurare il database locale

Lanciare un terminale nella directory `AtAVi-Software/lambda` ed eseguire il comando

```
npm run configure
```

5.4 Testing e code coverage

Per effettuare i test sulle funzioni lambda eseguire nella directory `AtAVi-Software/lambda` il comando

```
npm run test
```

Per verificare code/branch/function/statement coverage eseguire il comando

```
npm run cover
```

5.5 Eliminazione dipendenze testing

Per eliminare le dipendenze inutili al deployment è necessario:

1. Aprire un terminale su `AtAVi-Software/lambda` ed eseguire il comando

   ```
   npm run prune-dev
   ```

6 Reset AtAVi

Per rimuovere AtAVi è necessario effettuare il reset della configurazione di Amazon Web Services e la skill.
Per la rimozione da Amazon Web Services è disponibile uno strumento automatico. In alternativa è possibile effettuare manualmente il reset.

6.1 Reset

6.1.1 Reset automatico

Perché il reset vada a buon fine è necessario, prima di eseguirlo, riportare l'id dell'API e l'ARN del topic SNS nel file *aws-reset.js* presente nella cartella AtAVi-Software.
Per fare questo collegarsi a
`https://eu-west-1.console.aws.amazon.com/apigateway/home?region=eu-west-1#/apis`
cliccare su AtAVi e copiare l'ID dell'API, formato da un valore di 8 caratteri alfanumerici, in cima al file

`AtAVi-Software/aws-reset.js`

sostituendolo a [YOUR_API_ID].

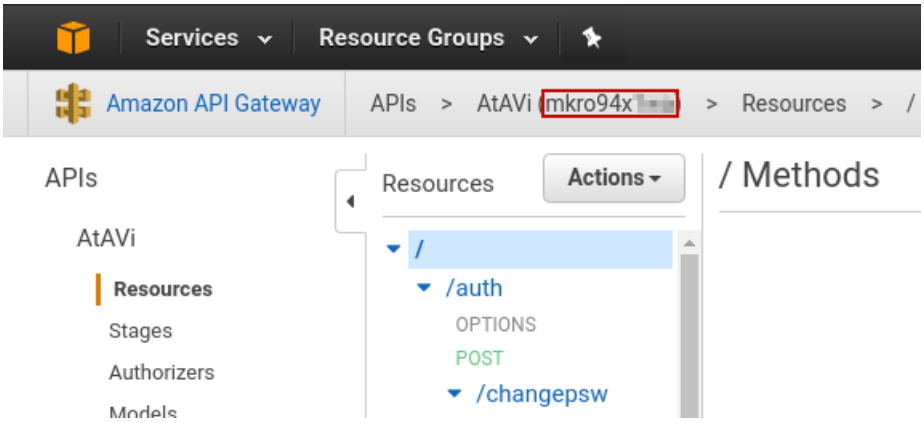

Figura 4: API Gateway - É evidenziato l'ID dell'API

Per l'ARN del topic SNS collegarsi a
`https://eu-west-1.console.aws.amazon.com/sns/v2/home?region=eu-west-1#/topics`
individuare il topic *sendSlackMessage* e copiare l'ARN corrispondente in cima al file

`AtAVi-Software/aws-reset.js`

sostituendolo a [YOUR_SNS_ARN].
Aprire il terminale dalla cartella *AtAVi-Software* e dare il comando

```
node aws-reset.js
```

Continuare il reset proseguendo da §6.1.2.5

6.1.2 Reset manuale

Aprire il terminale e seguire, nell'ordine, la seguente procedura.

6.1.2.1 Eliminazione API Per eliminare la struttura di API Gateway è prima necessario ricavare l'id dell'API.
Per fare questo collegarsi a
`https://eu-west-1.console.aws.amazon.com/apigateway/home?region=eu-west-1#/apis`
cliccare su AtAVi e prendere nota dell'ID dell'API, formato da un valore di 8 caratteri alfanumerico (vedere Figura 4).
Aprire un terminale ed eseguire

```
aws apigateway delete-rest-api --rest-api-id [YOUR_API_ID] --region eu-west-1
```

sostituendo [YOUR_API_ID] con l'id trovato.

6.1.2.2 Eliminazione database Per eliminare il database è necessario eliminare le tabelle create.
Aprire un terminale ed eseguire i comandi relativi all'eliminazione delle tabelle auth, curiosities, customer, customer_rules, employee_rules.

```
aws dynamodb delete-table --table-name auth
```

```
aws dynamodb delete-table --table-name curiosities
```

```
aws dynamodb delete-table --table-name customer
```

```
aws dynamodb delete-table --table-name customer_rules
```

```
aws dynamodb delete-table --table-name employee_rules
```

6.1.2.3 Eliminazione topic SNS Per eliminare il topic SNS è necessario avere l'ARN del topic.
Collegarsi a
`https://eu-west-1.console.aws.amazon.com/sns/v2/home?region=eu-west-1#/topics`
individuare il topic *sendSlackMessage* e prendere nota dell'ARN corrispondente.
Aprire un terminale ed eseguire il seguente comando

```
aws sns delete-topic --topic-arn [YOUR_ARN_SNS]
```

sostituendo [YOUR_SNS_ARN] con l'ARN trovato.

6.1.2.4 Eliminazione role Per eliminare un ruolo è necessario prima togliere tutte le policy associate.
Aprire un terminale e dare i seguenti comandi, che tolgono al ruolo invoke_lambda le policy relative all'utilizzo di DynamoDB, Lambda e SNS

```
aws iam detach-role-policy --role-name invoke_lambda
    --policy-arn
    arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess

aws iam detach-role-policy --role-name invoke_lambda
    --policy-arn arn:aws:iam::aws:policy/AWSLambdaExecute

aws iam detach-role-policy --role-name invoke_lambda
    --policy-arn
    arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole

aws iam detach-role-policy --role-name invoke_lambda
    --policy-arn
    arn:aws:iam::aws:policy/AWSLambdaInvocation-DynamoDB

aws iam detach-role-policy --role-name invoke_lambda
    --policy-arn arn:aws:iam::aws:policy/AmazonSNSFullAccess
```

Ora è possibile eliminare il ruolo eseguendo il comando

```
aws iam delete-role --role-name invoke_lambda
```

6.1.2.5 Eliminazione funzioni lambda Per eliminare le funzioni lambda eseguire i seguenti comandi

```
aws lambda delete-function --function-name util_addCur

aws lambda delete-function --function-name util_addCustomer

aws lambda delete-function --function-name util_checkConstraints
```

```
aws lambda delete-function --function-name skill_core

aws lambda delete-function --function-name rules_updateEmployeeRule

aws lambda delete-function --function-name rules_updateCustomerRule

aws lambda delete-function --function-name rules_getCustomerRule

aws lambda delete-function --function-name rules_getAllEmployeeRules

aws lambda delete-function --function-name rules_addCustomerRule

aws lambda delete-function --function-name rules_getEmployeeRule

aws lambda delete-function --function-name rules_deleteCustomerRule

aws lambda delete-function --function-name rules_deleteEmployeeRule

aws lambda delete-function --function-name rules_addEmployeeRule

aws lambda delete-function --function-name rules_getAllCustomerRules
```

6.1.2.6 Eliminazione skill Per eliminare la skill collegarsi a `https://developer.amazon.com/edw/home.html#/skills` cliccare su *Delete* in corrispondenza della skill AtAVi e confermare.

7 Funzionalità

Nella presente sezione vengono presentate le funzionalità offerte dall'applicazione. Ogni funzionalità viene prima descritta in una sottosezione *Comportamento generale* e successivamente, se necessario, vengono approfonditi gli aspetti che la compongono in altre sottosezioni. Ogni aspetto che lo richiede è accompagnato da una lista delle principali funzioni lambda e/o componenti di *AtAVi*$_{GI}$ che il manutentore deve conoscere, in modo da:

- garantire il loro funzionamento;
- modificarle in modo consono in caso di estensione delle funzionalità;
- non duplicare parte del prodotto in caso di aggiunta di nuove funzionalità.

Le funzionalità offerte dall'applicazione e descritte in seguito sono:

1. interazione vocale (§7.1);
2. notificazione automatica (§7.2);
3. intrattenimento dell'interlocutore (§7.3);
4. area di amministrazione (§7.4).

7.1 Interazione vocale

7.1.1 Comportamento generale

L'applicazione offre la funzionalità di interazione vocale tra l'interlocutore e *Alexa*, l'assistente virtuale di *AtAVi*, consentendo l'accoglienza dell'ospite in azienda in maniera automatizzata. L'interazione avviene seguendo le seguenti fasi:

- l'interlocutore avvia l'applicazione;
- l'applicazione avvia l'assistente virtuale *Alexa*;
- l'applicazione invoca la skill$_{GI}$ che gestisce gli intent$_{GI}$ dell'interazione;
- ogni funzione lambda si occupa dell'interazione con le altre componenti del sistema a seconda del loro contratto.

7.1.2 Skill

I packages AtAVi::AWS-Back-End::Skill e AtAVi::AWS-Back-End::Skill::Util si occupano della gestione dell'interazione, in particolare la skill ha questo comportamento:

- viene creata una nuova session$_{GI}$ per la gestione degli intent;
- ad ogni domanda posta dall'assistente virtuale corrisponde un LaunchRequest$_{GI}$ impostato nella skill;
- in base alle risposte dell'interlocutore la skill indirizza il sistema verso gli intent appositi.

Gli intent hanno lo scopo seguente:

- un intent può invocare un LaunchRequest$_{GI}$;
- un intent può invocare una o più funzioni lambda$_{GI}$ associate:

Le funzioni lambda hanno lo scopo seguente:

- una funzione lambda può includere dei moduli;
- una funzione lambda può comunicare con le componenti esterne.

7.1.2.1 Funzioni Lambda

7.1.2.1.1 util_addCustomer Questa funzione ha lo scopo di aggiungere un nuovo cliente al database. In essa sono definiti i vari campi dati che vengono richesti in input, in particolare:

```
{
   "id" : uuid(),
   "first_name": e.first_name,
   "last_name": e.last_name,
   "company": e.company,
   "last_employee":e.last_employee,
   "last_visit":date
 }
```

Se i campi richiesti per identificare un cliente cambiano nel tempo, il manutentore deve modificare questa funzione lambda.

7.1.2.1.2 util_getCustomer Questa funzione ha lo scopo di restituire i valori di alcuni campi dati di un dato cliente. In essa è definito di quali campi dati vengono restituiti i valori, in particolare sono:

```
{
   "first_name": e.first_name,
   "last_name": e.last_name,
   "company": e.company,
 }
```

Se il manutentore volesse estendere il numero di campi dati da restituire, deve modificare questa funzione lambda.

7.2 Notificazione automatica

7.2.1 Comportamento generale

L'applicazione offre la funzionalità di notificazione automatica sul canale *Slack* associato al dipendente cercato dall'interlocutore.
La notificazione automatica avviene seguendo queste fasi temporali:

1. l'interlocutore avvia l'applicazione;
2. l'interazione raggiunge il punto in cui viene fornito al sistema il nominitavo del dipendente cercato (**notificazione prioritaria**);
3. l'interazione raggiunge il punto in cui vengono fornite al sistema le risposte alle domande poste dall'assistente virtuale (**notificazione principale**);
4. l'interazione raggiunge il termine (**notificazione secondaria**).

7.2.1.1 Notificazione prioritaria Il package AtAVi::AWS-Back-End::Skill si occupa della gestione della notificazione prioritaria, in particolare la skill ha questo comportamento:

- viene richiesto il nominitavio del dipendente cercato, il sistema *AtAVi* distingue due casi:
  1. viene fornito il nominativo e il sistema imposta il dipendente come destinatario della notificazione;
  2. non viene fornito il nominativo e il sistema imposta il canale generale aziendale come destinatario della notificazione.
- il sistema *AtAVi* notifica automaticamente il destinatario impostato tramite il canale slack associato.

7.2.1.2 Notificazione principale Il package AtAVi::AWS-Back-End::Skill si occupa della gestione della notificazione principale, in particolare la skill ha questo comportamento:

- l'assistente virtuale pone delle domande all'interlocutore;
- il sistema *AtAVi* notifica automaticamente il destinatario impostato con le risposte alle domande poste.

7.2.2 Notificazione secondaria

Il package AtAVi::AWS-Back-End::Skill si occupa della gestione della notificazione secondaria, in particolare la skill ha questo comportamento:

- l'interazione arriva al termine;
- il sistema *AtAVi* notifica automaticamente il destinatario impostato inviando il resto dell'interazione.

7.2.2.1 Funzioni Lambda

7.2.2.1.1 util_sendSlack Questa funzione ha lo scopo di inviare un dato messaggio in un dato canale *Slack*. In essa è definita la struttura che il messaggio avrà quando il destinatario andrà a leggerlo, in particolare ogni messaggio sarà composto in questo modo:

AtAVi: testo del messaggio

Per modificare la struttura di visualizzazione dei messaggi, il manutentore deve modificare questa funzione lambda.

7.3 Intrattenimento dell'interlocutore

7.3.1 Comportamento generale

L'applicazione offre la funzionalità di intrattenimento dell'interlocutore. L'intrattenimento dell'interlocutore avviene seguendo queste fasi:

1. l'interlocutore avvia l'applicazione;
2. l'interazione raggiunge il punto in cui il sistema fornisce l'intrattenimento dell'interlocutore.

7.3.2 Intrattenimento

Il package AtAVi::AWS-Back-End::Skill si occupa della gestione dell'intrattenimento dell'interlocutore, in particolare la skill ha questo comportamento in maniera ricorsiva:

- viene richiamato l'intent adibito all'intratenimento;
- viene richiamata la funzione lambda che si occupa di prelevare casualmente dal database una curiosity$_{GI}$;
- lo stesso intent fornisce in output al sistema la curiosity;
- l'assistente virtuale pronuncia la curiosity.

7.3.2.1 Funzioni lambda

7.3.2.1.1 util_getCuriosity Questa funzione lambda ha lo scopo di selezionare in maniera casuale nel database una *curiosità* e restituirla; in particolare viene selezionata la curiosità il cui id è:

```
id = random.integer(0, 400);
```

Se il manutentore volesse cambiare l'algoritmo di scelta della curiosità deve modificare questa funzione lambda.

7.4 Area di amministrazione

7.4.1 Comportamento generale

L'applicazione offre alcune funzionalità di amministrazione, in particolare l'area di amministrazione è formata dalle seguenti componenti:

1. l'utente accede al pannello di autenticazione;
2. l'utente amministratore ottiene i permessi di amministrazione;
3. l'utente esce dall'area di amministrazione.

7.4.2 Autenticazione e log-in

Il package AtAVi::AWS-Back-End::Admin::Auth gestisce l'autenticazione e il log-in dell'amministratore. I permessi di accesso vengono concessi dopo l'autenticazione e hanno un tempo di validità di un'ora dopodiché sarà necessario rieffettuare il log-in.
Questa funzione è soddisfatta in queste fasi:

- l'utente accede al pannello di autenticazione;
- l'utente inserisce la password di amministrazione unica per l'intera azienda;
- il sistema *AtAVi* si occupa di controllare la corrispondenza tra la password inserita e quella contenuta nel database;
- l'utente amministratore cambia la password;
- l'utente amministratore inserisce la nuova password 2 volte;
- il sistema *AtAVi* si occupa di controllare la corrispondenza tra le due password inserite e cambia la password contenuta nel database ritornando un messaggio di successo.

7.4.2.1 Funzioni lambda

7.4.2.1.1 auth_login Questa funzione lambda si occupa di verificare i dati inseriti dall'utente tramite il seguente controllo:

```
hash256(e.password+db_salt) == db_password
```

dove:

- e.password è la password inserita dall'utente;
- db_salt è una stringa casuale.

L'hash a 256 bit di queste componenti deve corrispondere a db_password, cioè la password salvata nel database, per ottenere il token di accesso, altrimenti viene generato l'errore *Invalid password*.

7.4.2.1.2 auth_changePsw Questa funzione lambda si occupa di verificare i dati inseriti dall'utente tramite i seguenti controlli:

```
hash256(e.old_password+db_salt) == db_password
e.new_password != e.repeat_password
```

dove:

- e.old_password è la password che l'utente vuole cambiare;
- db_salt è una stringa casuale.

Per ottenere il permesso di cambiare la password l'hash a 256 bit di queste componenti deve corrispondere a db_password, cioè la password salvata nel database e e.new_password, la nuova password, deve corrispondere a repeat_password, dato inserito sempre dall'utente.

7.4.3 Area amministratore

Il package AtAVi::AWS-Back-End::Admin::Rules gestisce le funzionalità di amministrazione accessibili solo dall'utente amministratore.
Questa funzione è soddisfatta in queste fasi:

- l'utente amministratore si è identificato;
- l'utente amministratore aggiunge una nuova regola:
  - l'utente amministratore aggiunge una nuova regola riferita ai dipendenti;
  - l'utente amministratore aggiunge una nuova regola riferita ai visitatori.
- l'utente amministratore elimina una regola esistente;
  - l'utente amministratore elimina una regola esistente riferita ai dipendenti;
  - l'utente amministratore elimina una regola esistente riferita ai visitatori.
- l'utente amministratore modifica una regola esistente;
  - l'utente amministratore modifica una regola esistente riferita ai dipendenti;
  - l'utente amministratore modifica una regola esistente riferita ai visitatori.

7.4.4 Log-out

La funzionalità di logout dall'area di amministratore permette all'utente amministratore di revocare i propri permessi di accesso all'area di amministrazione. Se l'utente amministratore esce dall'area di amministrazione senza effettuare il log-out mantiene i permessi di accesso fino alla loro scadenza.
La funzione di log-out è soddisfatta in queste fasi:

- l'utente accede al pannello di autenticazione;

- l'utente amministratore si autentica;
- l'utente amministratore effettua il logout;
- il sistema *AtAVi* revoca i permessi di accesso all'utente amministratore che dovrà rieffettuare il log-in per successivi accessi all'area di amministrazione.

8 Persistenza Dati

AtAVi utilizza il database NoSql$_{GI}$ DynamoDB per salvare le regole di amministrazione e registrare le visite dei clienti. I dati inviati e ricevuti in risposta dal database sono in formato JSON. I dati sono memorizzati in tabelle contenenti item e attributi, ogni tabella ha una chiave primaria.

8.1 Tabelle

8.1.1 Tabella auth

La tabella auth contiene i dati per l'autenticazione dell'amministatore.

8.1.1.1 Attributi:

- **id**: stringa, è la chiave primaria;
- **password**: stringa, è la password dopo l'applicazione della funzione di hash;
- **salt**: stringa random, è usata come input in aggiunta alla password nella funzione di hash.

8.1.2 Tabella curiosities

La tabella curiosities contiene le informazioni che vengono usate da *AtAVi* per intrattenere l'utente.

8.1.2.1 Attributi:

- **id**: int, id incrementale, è la chiave primaria;
- **text**: string, è il testo che verrà proposto all'interlocutore.

8.1.3 Tabella customer

La tabella customer salva i dati dei clienti al loro arrivo in azienda.

8.1.3.1 Attributi:

- **id**: stringa, è la chiave primaria;
- **first_name**: stringa, è il nome del cliente;

- **last_employee**: stringa, è un id che identifica l'impiegato incontrato dal cliente durante l'ultima visita;
- **last_name**: stringa, è il cognome del cliente;
- **company**: stringa, è l'azienda del cliente;
- **last_visit**: number, numero che rappresenta l'ultima visita del cliente.

8.1.4 Tabella customer_rules

La tabella customer_rules contiene le regole che associano un cliente o un'azienda a un impiegato.

8.1.4.1 Attributi:

- **id**: stringa, è la chiave primaria;
- **company**: stringa, è il nome dell'azienda del cliente;
- **employee_id**: stringa, identifica l'impiegato;
- **first_name**: stringa, è il nome del cliente;
- **last_name**: stringa, è il cognome del cliente.

8.1.5 Tabella employee_rules

La tabella employee_rules contiene le regole che associano gli impiegati a un canale *Slack*.

8.1.5.1 Attributi:

- **id**: stringa, è la chiave primaria;
- **fullname**: stringa, è il nome e cognome dell'impiegato;
- **slack**: stringa, identifica il canale *Slack* associato all'impiegato.

9 Estensione delle funzionalità

Per ragioni di tempo e di competenze, alcune funzionalità del software *AtAVi* non sono state implementate. In questa sezione vengono elencati tutti i punti di possibile estensione delle funzionalità e un loro possibile sviluppo.

9.1 Aggiunta nuove attività

Le attività che l'interlocutore può scegliere sono state limitate alla richiesta di un caffè o di materiale aggiuntivo per la riunione. Il sistema *AtAVi* è stato realizzato in modo tale da consentire l'aggiunta di una nuova attività, in particolare deve essere modificato il file:

lambda-functions/skill-functions/functions/core/index.js

aggiungendo un nuovo $Statehandler_{GI}$ in un punto del flusso ritenuto opportuno. Non devono essere aggiunti nuovi Intent, in quanto la risposta attesa dall'interlocutore viene già gestita dal sistema.

9.2 Proposta automatica preferenze

Le attività scelte dall'interlocutore possono essere gestite in modo tale che, in caso lo stesso interlocutore ritorni in futuro, gli vengano proposte automaticamente le attività da lui scelte nelle precedenti interazioni. In particolare deve essere aggiunta una nuova funzione lambda:

lambda-functions/util-functions/functions/addPreferences/index.js

che ha lo scopo di aggiungere le preferenze alla tabella Customer del database. Il database è definito in modo tale da occuparsi automaticamente della creazione di un nuovo campo, se non è già presente, dove aggiungere la preferenza.
Deve essere poi modificato il file:

lambda-functions/skill-functions/functions/core/index.js

inserendo, nel punto in cui l'interlocutore accetta un'attività proposta, l'ivocazione della funzione lambda definita precedentemente.
Per ultimare la gestione di questa nuova funzionalità dovrà infine essere aggiunta una funzione lambda:

lambda-functions/util-functions/functions/getPreferences/index.js

che ha come scopo quello di leggere, dalla tabella Customer del database, le preferenze dell'interlocutore e darle in output; deve essere anche modificato il file:

lambda-functions/skill-functions/functions/core/index.js

inserendo, nel punto in cui l'interlocutore viene riconosciuto, la proposta di attività già scelte dallo stesso precedentemente.

9.3 Statistiche sugli arrivi

Può essere utile, per l'utente amministratore, conoscere il numero di visite di un interlocutore. In particolare deve essere aggiunta una nuova funzione lambda:

lambda-functions/util-functions/functions/addPresences/index.js

che ha lo scopo di incrementare il numero di visite di un interlocutore nella tabella Customer del database. Il database è definito in modo tale da occuparsi automaticamente della creazione di un nuovo campo, se non è già presente, dove aggiungere la nuova visita.
Deve essere poi modificato il file:

lambda-functions/skill-functions/functions/core/index.js

inserendo, nel punto in cui l'interlocutore viene riconosciuto, l'ivocazione della funzione lambda definita precedentemente.
Per ultimare la gestione di questa nuova funzionalità dovrà infine essere aggiunta una funzione lambda:

lambda-functions/util-functions/functions/getPresences/index.js

che ha come scopo quello di leggere, dalla tabella Customer del database, il numero di visite dell'interlocutore e darle in output.

9.4 Modifica della struttura di interazione

Per modificare la struttura di interazione deve essere poi modificato il file:

lambda-functions/skill-functions/functions/core/index.js

in modo da modificare il flusso come ritenuto opportuno. In caso serva definire nuovi Intent, devono essere aggiunti da:
`https://developer.amazon.com/edw/home.html` Una volta effettuato l'accesso alla propria skill, nella sezione 'Interaction Model', è possibile inserire in Intent Schema l'intent che si vuole aggiungere. Se questo fa parte della libreria interna di Alexa Skills Kit è necessario aggiungere:

```
{
   "intent":"AMAZON.$intentbuiltin"
}
```

dove $intentbuiltin è un intent pre-esistente tra quelli nella libreria, che possono essere trovati su
`https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/built-in-intent-ref/built-in-intent-library`
Se l'intent non fa parte della libreria interna di Alexa Skills Kit è necessario definire un nome all'intent e gli slot che lo caratterizzano:

```
{
   "intent": "$nomeintent",
   "slots":[
      {
         "name":"$nomeSlot",
         "type":"$tipoSlot"
      }
   ]
}
```

- $nomeintent rappresenta il nome che si vuole attribuire all'intent e che verrà utilizzato per definire il corpo dell'intent;
- $nomeSlot è il nome che si vuole attribuire allo slot e che verrà utilizzato per leggerne il valore;
- $tipoSlot è il tipo che contiene tutti i possibili valori che sono attribuibili allo slot. Può essere un tipo esistente nella libreria di Alexa Skills Kit, oppure un tipo personalizzato, che dovrà essere poi definito in Custom Slot Type, sotto a Intent Schema.

Una volta definito l'intent è necessario elencare quello che l'utente potrà dire per attivare l'intent. Questo dovrà essere definito nelle Sample Utterances. Ogni utterance è formata da $nomeintent seguita dalla frase che dovrà essere pronunciata. I valori degli slot che serviranno alla skill, devono essere scritti tra parentesi graffe utilizzando il valore definito nel campo name dello slot $nomeSlot.

10 Glossario interno

10.1 A

- **AngularJS**: è un framework web open source nato per affrontare le molte difficoltà incontrate nello sviluppo di applicazioni singola pagina;
- **API**: acronimo di Application Programming Interface ed indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma;
- **Apex**:apex è un programma che permette di gestire architetture serverless attraverso AWS Lambda. Apex permette la costruzione, il deployment e la gestione delle funzioni lambda in modo semplice;
- **Atavi**: acronimo di Accoglienza tramite Assistente Virtuale;
- **AVS**: Alexa Voice Service è un servizio di Amazon che consente di aggiungere controlli vocali intelligenti ad un prodotto connesso che abbia un microfono e delle casse audio. Viene utilizzato per gestire l'assistente virtuale Alexa;
- **AWS Lambda**: servizio di elaborazione serverless che esegue il tuo codice in risposta a determinati eventi e gestisce automaticamente le risorse di elaborazione sottostanti al tuo posto.

10.2 C

- **Callback**: callback è un equivalente asincrono di una funzione. Una funzione callback è chiamata al completamento del task assegnato. NodeJs usa cospicuamente le funzioni di callback, infatti tutte le APIs di Node sono scritte per supportarle;
- **Curiosity**: sono delle frasi che vengono usate dal sistema per l'intrattenimento dell'interlocutore.

10.3 D

- **Data binding**: il Data Binding è un meccanismo che consente di associare e sincronizzare una fonte dati agli elementi dell'interfaccia utente;
- **Database NoSql/Database non Relazionali**: database che non utilizzano il modello relazionale, solitamente usato dai database tradizionali quali SQL;
- **DBMS**: un Database Management System, abbreviato in DBMS o Sistema di gestione di basi di dati, è un sistema software progettato per consentire la creazione, la manipolazione e l'interrogazione efficiente di database;
- **Dependency injection**: design pattern della Programmazione orientata agli oggetti il cui scopo è quello di semplificare lo sviluppo e migliorare la testabilità di software di grandi dimensioni.

10.4 E

- **Event-driven**: paradigma di programmazione. Mentre in un programma tradizionale l'esecuzione delle istruzioni segue percorsi fissi, che si ramificano soltanto in punti ben determinati predefiniti dal programmatore, nei programmi scritti utilizzando la tecnica a eventi il flusso del programma è largamente determinato dal verificarsi di eventi esterni.

10.5 F

- **Framework**: architettura logica di supporto su cui un software può essere progettato e realizzato, facilitandone lo sviluppo da parte del programmatore.

10.6 H

- **HTML**: acronimo di Hyper Text Markup Language, è un linguaggio di markup utilizzato per la creazione di pagine web.

10.7 I

- **Intent**: la richiesta principale o azione associata con il comando dell'utente per una skill personalizzata. Ad esempio: Alexa, ask History Buff what happened on June third. In questo statement, what happened on June third indica a un intent specifico che può essere gestito da una particolare abilità di Alexa. Questo dice ad Alexa che l'utente vuole History Buff per ottenere informazioni storiche su una data specifica;
- **Interfaccia Web**: l'interfaccia web è una Graphical User Interface (GUI) utilizzabile all'interno di un browser.

10.8 L

- **LaunchRequest**: si tratta di un intent che consente di attuare richieste all'assistente virtuale *Alexa*.

10.9 P

- **Package**: un Package rappresenta una collezione di classi ed interfacce che possono essere raggruppate in base alla funzione comune da esse svolta;
- **Provisioning**: processo mediante il quale un amministratore di sistema assegna risorse e privilegi, non solo agli utenti di una rete ma anche a chi le utilizza da remoto (ad esempio i fornitori).

10.10 R

- **Refresh token**: stringa che permette di ottenere nuovamente l'accesso e la possibilità di utilizzo della skill e di Alexa Voice Service;

10.11 S

- **Session**: è un'unità interattiva, semi-permanente per il mantenimento degli storici della comunicazione tra l'utente e il sistema;
- **Skill**: una capacità o abilità di Alexa. Alexa mette a disposizione delle skill di default (come ad esempio la possibilità di riprodurre musica), inoltre i programmatori possono aggiungerne delle altre configurandone di nuove;
- **Slack**: è uno strumento di collaborazione basato sul cloud. Consente, tra le altre funzioni, di chattare e di condividere materiale con i membri del gruppo;
- **Statehandler**: variabile che permette di gestire lo stato della skill.

10.12 T

- **Token**: stringa che permette l'accesso e l'utilizzo della skill e di Alexa Voice Service;
- **Token JWT**: token utilizzati per la comunicazione sicura tra due parti.

10.13 U

- **Utterances**: frasi che possono essere comprese da *Alexa* a cui sono associati i corrispettivi Intent.