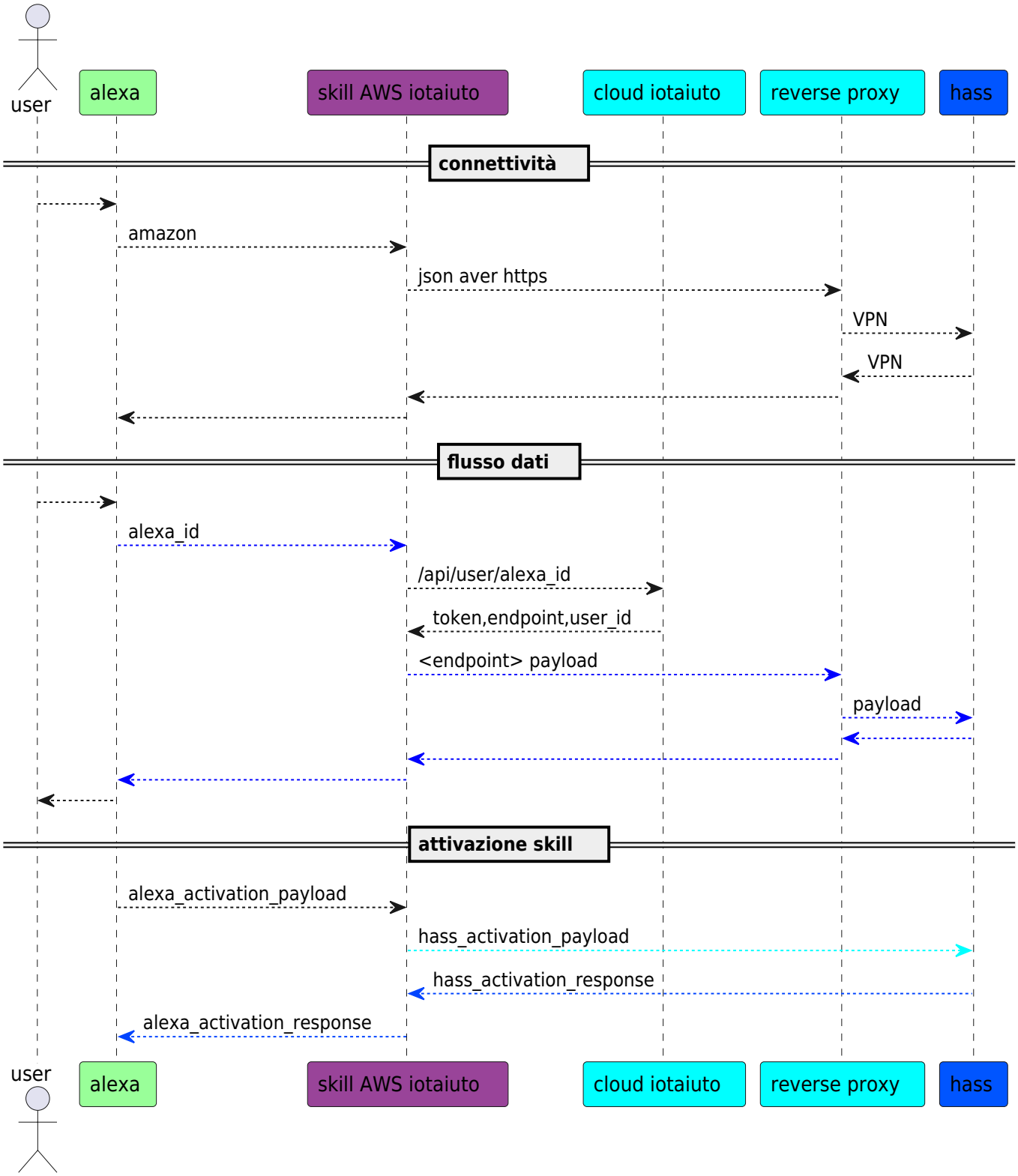
iotaiuto skill



Flusso dati

ogni comando che arriva da alexa si traduce in una richiesta verso la skill 'iotaiuto' (metodo event_handler) il cui payload è:

- event
- context

in particolare event contiene aws_token (estratto dalla funzione get_aws_token)

esempio di event

```
TODO
```

amazon api https://api.amazon.com/user/profile:

- parametro ingresso aws_token
- risposta

```
{'user_id': 'amzn1.account.AGH7EXZ2ZZMPAUDFRBZQQFK7HDQQ', 'name': 'Stefano
Scipioni', 'email': 'stefano.scipioni@csgalileo.org'}
```

Cloud iotaiuto

Questo webservice REST offre i parametri di funzionamento del servizio lambda AWS. In particolare alla richiesta di uno **user_id** risponde con i parametri necessari ad alexa per instaurare una comunicazione con l'endpoint homeassistant dell'utente.

test funzionali (cd /home/iotaiuto):

- make test-users

AWS

- AWS account root user: amazon@csgalileo.org
- IAM user: https://049753375854.signin.aws.amazon.com/console
  - ID: 049753375854
  - username: stefano.scipioni
  - region: Europa (Irlanda) eu-west-1
  - create security credential
- Alexa developer console: https://developer.amazon.com/alexa/console/ask
- lambda service: iotaiuto

Creare una skill iotaiuto

Stiamo modificando haaska per permettere la gestione di più utenti. La skill chiamerà un service di galileo che, dato l'id dell'utente chiamante, fornirà il giusto endpoint di home assistant.

La creazione di questa skill è analoga a quella di haaska e la repository è:
https://git.csgalileo.org/iotaiuto.git

Per far si che ogni utente venga associato al proprio homeassistant, aggiungere a **config.json** questa riga:

```
"api_users": "https://ha.csgalileo.org/api/users"
```

Quando si arriva alla parte di test della **funzione lambda**, inserire questo codice:

```
{
  "directive": {
    "header": {
      "payloadVersion": "3",
      "correlationToken": "12345",
      "namespace": "Alexa.PowerController",
      "name": "TurnOff",
      "messageId": "abcd"
    },
    "endpoint": {
      "scope": {
        "token": "access-token-from-skill",
        "type": "BearerToken"
      },
      "cookie": {},
      "endpointId": "abcd"
    },
    "payload": {}
  }
}
```

N.B. Questo test restituirà un errore se il “config.json” avrà l'attributo “api_users” settato, perché avendo un token fake non riuscirà ad autenticarsi per ricevere lo “user_id”.

Pubblicare su AWS

- https://aws.amazon.com → my account → console → IAM user

[DEVELOP] Go to developer console adn login with amazon account:

- click to login with Amazon
- creare “security profile” (http://wiki.csgalileo.org/tips:android:gianomobile as privacy URL)
- get client id and secret from manage “security profile” –> web settings
  - –> client id: xxxx
  - –> client secret: yyyy
  - freeze browser tab [DEVELOP.1]

[ALEXA] Go to amazon alexa console:

- create skill with name, default language and smart home model
- get **skill ID**
- freeze browser tab [ALEXA.1]

[AWS] Go to AWS console

- open services → IAM
  - roles → create role → AWS service and Lambda → Next: permissions → DatabaseAdministrator → role name=lambda_iotaiuto
- open services → lambda → [Ireland]
- create function from scratch with:
  - function name 'iotaiuto'
  - runtime python 3.7
  - choose execution role, select lambda_iotaiuto and “create function”
  - add trigger “Alexa Smart Home” with **skill ID**
  - get **ARN** from top/rigth of page

In [ALEXA.1]:

- set ARN into default endpoint and into Europe/India region
- save
- setup account linking
- Authorization URI: https://www.amazon.com/ap/oa
- Access Token URI: https://api.amazon.com/auth/o2/token
- **client id** and **client secret** from [DEVELOP.1]
- Client Authentication Scheme: http basic
- add scope named “profile”
- save and get 3 **redirect urls**

In [DEVELOP.1] compile 3 redirect urls

In [DEVELOP.1] disable **Send Alexa Events** in Permission tab

Create access key from [AWS] → “username on top” → “my security credential”:

- “create new access key”

Login in aws console environment with “aws configure”

From linux console upload skill with “make deploy”

From linux run “make log” to see runtime errors

Modificare e testare una funzione lambda

Le funzioni lambda sono codice che viene eseguito dalle skill custom Andare su https://console.aws.amazon.com e cliccare su Lambda.
Selezionare la skill che si desidera modificare/testare

Per **modificare una funzione**, scorrere giù fino alla sezione Function Code e modificare il codice, poi premere su Save.
In alternativa nel campo “Code entry type” si può selezionare “Upload a .zip file” per ricaricare la skill
Per inserire dei log in python è sufficiente usare la funzione print
Dopo ogni modifica premere su Test per assicurarsi che non ci siano errori nel codice

Per **testare una funzione**, selezionare una funzione di test e premere Test.
Nella parte alta della pagina compariranno i dati restituiti dalla funzione, in formato json, e il Log output.
Il Log output contiene, oltre a id e orario di richieste e risposte, tutti gli output delle print presenti nel codice.

Per **visualizzare i log** di una richiesta proveniente da un dispositivo come Echo dot,
andare nella tab Monitoring e cliccare View Logs in CloudWatch e cliccare sul Log Stream più recente.
Cambiare la modalità di visualizzazione da Row a Text potrebbe rendere più chiara la lettura.

Cose da fare

Creare un servizio all'indirizzo https://ha.csgalileo.org/api/users
Il servizio riceverà il parametro “user_id” in una richiesta post e restituirà qualcosa del genere:

```
{
    "endpoint": "https://ha.csgalileo.org/ha-12/api/",
    "bearer_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1..."
}
```