Universitatea Alexandru Ioan Cuza, Iași

Facultatea de informatică

Bob-jocul de cultură generală cu interacțiune vocală

Autor: Filimon Marta-Diana

Coordonator științific: Conf. Dr. Adrian Iftene

Iulie 2017

Universitatea Alexandru Ioan Cuza, Iași

Facultatea de informatică

Bob-jocul de cultură generală cu interacțiune vocală

Autor: Filimon Marta-Diana

Coordonator științific: Conf. Dr. Adrian Iftene

Iulie 2017

DECLARAȚIE PRIVIND ORIGINALITATE ȘI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de Licență cu titlul *Bob-jocul de cultură generală cu interacțiune vocală* este scrisă de mine și nu a mai fost prezentată niciodată la o altă facultate sau instituție de învățământ superior din țară sau din străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și dețin referința precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alți autori deține referința precisă;
- codul sursă, imagini etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor și dețin referințe precise;
- rezumarea ideilor altor autori precizează referința precisă la textul original.

Iași,
Absolvent Filimon Marta-Diana

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul *Bob-jocul de cultură generală cu interacțiune vocală*, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea Alexandru Ioan Cuza Iași să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași,

Absolvent Filimon Marta-Diana

Cuprins

.

1 Introducere

Lucrarea prezintă un joc de cultură generală implementat ca o abilitate pentru *Alexa*, acesta se numește *Bob*. Bob este capabil să adreseze utilizatorilor întrebări de cultură generala din domeniul geografiei. Totodata acest joc va clasifica jucătorii, iar la cerere va posta numărul de puncte împreuna cu locul pe *Twitter*.

Alexa este interfața vocală implementată de către *Amazon*. Ea este integrată cu multiple difuzoare inteligente, dar și alte dispozitive precum frigider inteligent sau mașini. Figura 1.1 oferă o imagine de ansamblu asupra modului în care utilizatorii folosesc *Alexa*. Aparent aceasta devine un vector important în dezvoltarea conceptului de casă inteligentă, ea făcându-și simțită prezența în din ce în ce mai multe case după cum arată și creșterea vânzărilor de *Amazon Echo*.

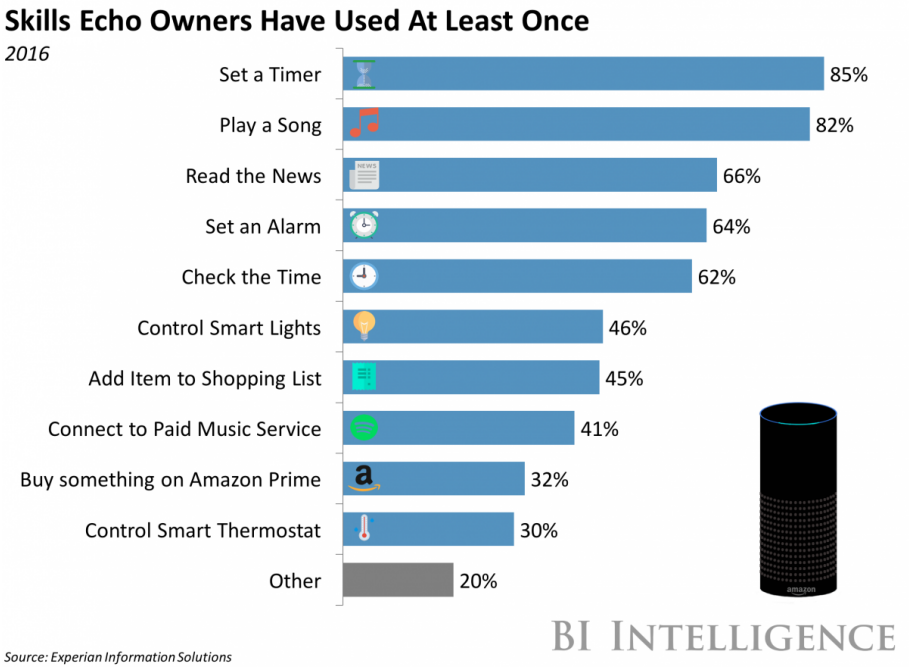

Figura 1.1: Abilitățile puse la dispoziție de *Amazon Ehho* pe care utilizatorii săi le-au folosit măcar o data (http://uk.businessinsider.com/)

Amazon Ehho este utilizată și pentru divertisment (vezi figura 1.1). Locurile 2 și 3 din topul celor mai folosite aplicații sunt ocupate de abilități făcând parte din acestă categorie. În cazul altor dispozitive (telefoane mobile, tablete, computere personale) divertismentul este unul dintre activitățile pentru care acestea sunt utilizate cel mai mult, deși destinația inițială a acestora nu a fost divertismentul. De exemplu, 49% dintre cei care dețin un dispozitiv mobil în *Statele Unite ale Americii* le utilizează pentru a se juca jocuri video. Ceea ce deschide posibilitatea ca în viitor

Amazon Echo, sau dispozitivele care sunt integrate cu *Alexa*, să devină o unealtă prin intermediul cărora utilizatorii să se joace.

Lucrarea este structurată în patru capitole principale care vor ajuta la înțelegerea structurii aplicației, prezentarea detaliilor legate de implementare și introducerea în tehnologiile folosite.

Primul capitol va prezenta dispozitivul *Amazon Echo*.

Al doilea capitol va expune conceptele similare atât la nivel *hardware*, cât și *software*. În prima parte va prezenta dispozitive similare cu *Amazon Echo* și diferențele dintre acestea, iar în a doua parte se vor fi prezentate trei aplicații care fac parte din aceeași categorie cu aplicația *Bob*.

Al treilea capitol va contura conceptele și tehnologiile necesare pentru dezvoltarea aplicației. Au fost expuse noțiuni pe care un dezvoltator trebuie să le aibă atunci când dezvoltă o abilitate pentru Alexa, dar și elemente care țin de elemente utilizate pentru dezvoltarea abilității Bob.

Al patrulea capitol va prezenta arhitectura aplicației, dar și elemente care țin crearea mediului de lucru pentru o abilitate. Prezentarea se va concentra asupra serviciilor utilizate, modului de utilizare, dar totodată va avea în vedere și motivarea alegerilor arhitecturale făcute.

În încheiere se vor prezenta concluziile deduse în urma lucrului la această aplicație, posibilele direcții de dezvoltare pentru viitor, dar și îmbunătățirea celor existente.

2 Amazon Echo

2.1 Introducere

Amazon Echo este un difuzor inteligent dezvoltat de către *Amazon*. Dezvoltarea acestuia a început în anul 2010, conform unor surse neoficiale, dar lansarea a fost făcută în luna iunie a anului 2015, în Statele Unite ale Americii. Dispozitivul a fost pentru prima dată pus la dispoziția publicului european la mai mult de un an diferență, în luna septembrie a anului 2016.

2.2 Dispozitive asemănătoare

Pe lângă *Amazon Echo* au mai fost dezvoltate și difuzoarele inteligente *Amazon Echo Dot* și *Amazon Tap* (vezi figura 3.1). Cele două îndeplinesc aceleași funcționalități ca și Amazon Echo, doar că au un public țintă diferit. *Amazon Tap* este destinat pentru utilizatorii care doresc portabilitate. *Amazon Echo Dot* peste destinat celor care doresc să aibă acces la toate funcționalitățile lui *Amazon Echo* însă consideră că nu au nevoie de un difuzor performant, sau celor care doresc să controleze cât mai multe spații cu asemenea dispozitive.

Figura 2.1: Difuzoarele inteligente produse de *Amazon*

2.3 Funcționalități

Integrarea cu *Alexa* care permite utilizatorilor accesarea a numeroase aplicații ce oferă posibilitatea de de a pune alarme, de a adăuga lucruri în coșul de cumpărături de pe *Amazon*, de a afla starea vremii sau traficului, de a asculta muzică sau audiobook-uri etc.

2.4 Sistemul de microfoane

Unul dintre lucrurile pentru care este sunt cunoscute cele trei dispozitive de la *Amazon* este performața sistemului de microfoane. Acestea impresionează și prin distanța de la care poți interacționa cu ele, chiar și în prezența obstacolelor. În figura 3.2 este o imagine cu dispunerea microfoanelor pentru pentru cele trei dispozitive.

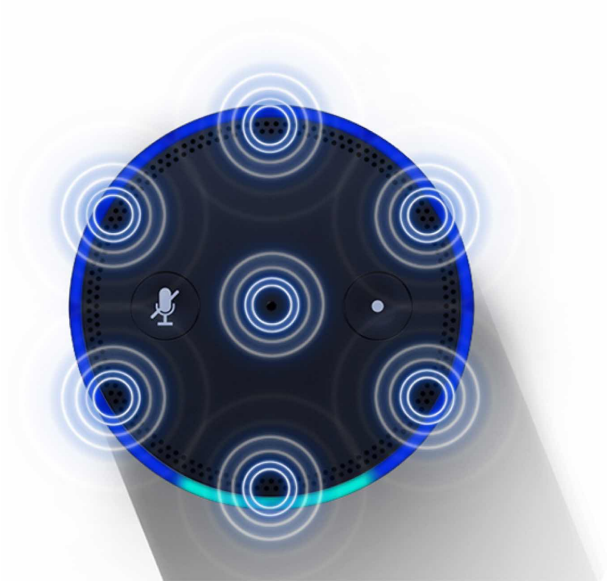

Figura 2.2: Dispunerea microfoamenlor *Amazon Echo* (https://sixcolors.com/)

Amazon a pus la dispoziția dezvoltatorilor un serviciu prin intermediul cărora aceștia pot integra interfața vocală *Alexa* cu aplicațiile pe care aceștia le dezvoltă, iar în anul curent (2017) sistemul de microfoane a fost și el făcut disponibil pentru terțele părți. Această nouă lansare are beneficii atât în rândul dezvoltatorilor de hardware și software cât și în rândul utilizatorilor obișnuiți care vor putea comunica cu orice dispozitiv ce are integrat sistemul de microfoane în aceleași condiții în care o fac și cu *Amazon Echo*.

2.5 Concluzii

În această secțiune a fost prezentat difuzorul inteligent *Amazon Echo*, câteva elemente legate de istoria dezvoltării acestuia, cazuri de utilizare și sistemul de microfoane care îl particularizează pe acesta în industrie.

3 Concepte similare

Care este diferența dintre Amazon Echo și Alexa? *Amazon Echo* este un dispozitiv fizic, iar *Alexa* este partea software a acestuia. Între cele două concepte este exact aceeași diferență ca între un telefon care suportă sistemul de operare *Android* și sistemul de operare.
Capitolul de față își propune să prezinte în prima parte dispozitivele care îndeplinesc funcții similare cu *Amazon Echo*. A doua parte va conține o prezentare a aplicațiilor asemănătoare cu cea propusă în continuare puse deja la dispoziția utilizatorilor de către *Alexa*.

3.1 Dispozitive asemănătoare cu *Amazon Echo*

Dispozitivele similare cu Amazon Echo pot fi împărțite în două categorii: cele care folosesc *Alexa* și cele care folosesc alt asistent virtual.

3.1.1 Dispozitive care folosesc Alexa

Dispozitivele care folosesc *Alexa* sunt fie cele trei propuse de către *Amazon* (*Amazon Echo*, *Amazon Tap*, *Amazon Dot* și *Fire TV*) sau cele propuse de către alte firme (*Triby Kitchen Speaker*, *Lenovo Smart Assistant*, *Vobot*, *FABRIQ Portable Smart Speaker* și *JAM Portable Speaker*). O parte dintre acestea se regăsesc în figura 4.1.

Figura 3.1: Dispozitive care au integrată interfața vocală Alexa

Toate dispozitivele enumerate anterior îndeplinesc aceleași funcții. Singurele lucruri care le

deosebesc sunt din punct de vedere hardware. Din punct de vedere hardware cea mai importantă componentă este sistemul de microfoane care este esențial pentru comunicarea utilizatorului cu dispozitivul. Sistemul de microfoane al *Amazon Echo* este dezvoltat în secțiunea 3.4.

3.1.2 Dispozitive care nu folosesc Alexa

Dispozitivele care nu folosesc *Alexa* se pot împărți în mai multe categorii în funcție de asistentul pe care acestea îl folosesc. În continuare sunt enumerați cei mai cunoscuți asistenți virtuali împreună cu difuzoarele inteligente cu care aceștia sunt integrați. Cei trei asistenți virtuali un utilizator îi poate găsi integrați si cu alte dispozitive precum telefoane mobile, laptop-uri etc. În figura 4.2 sunt imagini difuzoarele inteligente care au integrate cele mai cunoscute interfețe vocale diferite de *Alexa*.

- Asistentul propus de către *Google* este folosit de către dispozitivul *Google Home*. *Google Home* este unul dintre cele mai cunoscute difuzoare inteligente alături de *Amazon Echo*. Acesta este recunoscut pentru numărul mare de întrebări de cultură generală la care poate răspunde.
- *Mycroft* este un asistent virtual open-source, iar acesta este integrat în dispozitivul cu același nume, care a fost dezvoltat utilizându-se *Raspberry Pi* și *Arduino*.
- Cortana este asistentul produs de către cei de la Microsoft care va fi integrat cu un difuzor inteligent produs de către Harman Kardon. Acest nou produs va fi lansat în anul acesta (2017).

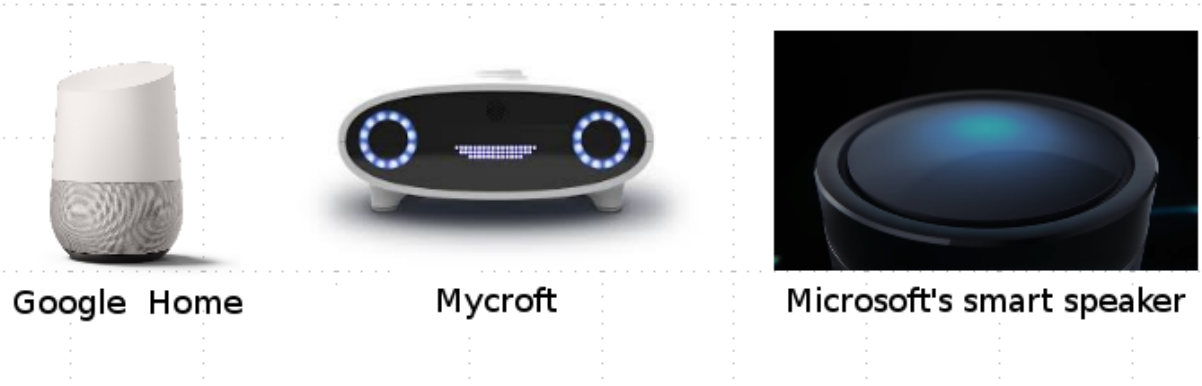

Figura 3.2: Difuzoare inteligente care **nu** au integrată interfața vocală *Alexa*

Diferențele dintre dispozitivele de mai sus sunt majore la nivel sofware. Deoarece funcțiile pe care acestea le îndeplinesc pot diferi în funcție de asistentul pe care acestea îl utilizează. O altă

importantă diferență este din perspectiva utilizatorului expert pentru care dezvoltarea de abilități noi ale sistemelor diferă de la asistent la asistent.

3.2 Aplicații Similare

Aplicațiile ce vor fi prezentate mai jos sunt disponibile atunci când utilizăm Alexa. Pentru a le activa trebuie să utilizăm interfața grafică disponibilă la adresa www.alexa.amazon.com sau aplicația pentru telefoanele mobile pe care o putem descărca din magazinul fiecărui sistem de operare în parte.

3.2.1 Question of the day

Aplicația *Question of the day*[1] va fi activată atunci când utilizatorul va rosti ”Alexa, ask question of the day.”, ”Alexa, play question of the day.” sau ”Alexa, play the game question of the day”. În fiecare zi *Question of the day* îți va propune o nouă întrebare la care să răspunzi, de la întrebări din artă sau divertisment până la întrebări de literatură sau știință. Question of the day este o metodă bună pentru a-ți testa și a-ți îmbunătăți cultura generală. Colectează puncte și află cum au răspuns alții (http://alexa.amazon.com, 2017). Pentru a ajuta utilizatorul să răspundă la întrebări aplicația va sugera patru variante de răspuns. După ce utilizatorul a dat un răspuns, în caz că acesta este greșit, acesta poate reîncerca să rezolve provocarea zilei. La final aplicația îți va comunica cât la sută dintre jucători au răspuns corect la întrebării.
De exemplu întrebarea de pe data de 24 aprilie 2017 a fost următoarea: *Ce dramaturg american faimos a scris piesa de teatru ”Tramvaiul numit dorință”?*

Figura 3.3: Interfața de activare a unei abilități *Question of the day*

[1] https://www.alexaskillstore.com/Question-of-the-Day/42441

3.2.2 Amazing Word Master Game

Aplicația *Amazing Word Master Game*[2] va fi activată atunci când utilizatorul va rosti *"Alexa, ask Word Master to play a game."*, *"Alexa, ask Word Master to start with ≪ Apple ≫* sau *"Alexa, open Word Master"*.

De fiecare dată când aplicația este activată *Alexa* va rosti un cuvânt, iar utilizatorul trebuie să rostească un cuvânt care începe cu ultima literă a cuvântului spus de *Alexa*. *Alexa* va calcula scorul atât pentru ea cât și pentru utilizator, iar acesta va fi egal pentru fiecare jucător cu suma lungimilor cuvintelor date drept răspuns. După fiecare răspuns al utilizatorului dispozitivul va comunica jucătorului scorul obținut de ambii jucători în etapa curentă. Oricând utilizatorul poate afla punctajele jucătorilor spunând *"score"*, poate opri jocul spunând *"stop"* sau poate afla definiția cuvântului spus de *Alexa*.

Acest joc este asemănător jocului *Fazan* și este o unealtă care te poate ajuta utilizatorii să își îmbunătățească vocabularul în limba engleză.

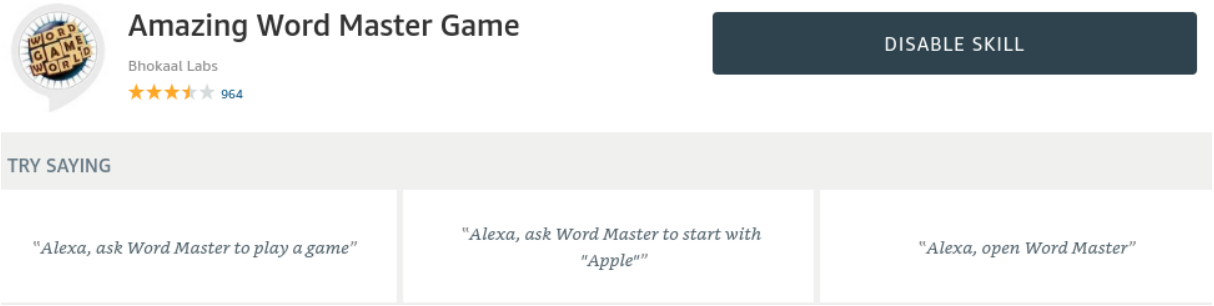

Figura 3.4: Interfața de activare a unei abilități *Amazing Word Master Game*

3.2.3 Tricky Genie

Aplicația *Tricky Genie*[3] va fi activată atunci când utilizatorul va rosti *"Alexa, start tricky genie."*, *"Alexa, launch tricky genie."* sau *"Alexa, ask tricky genie for a story"*.

Alexa va prezenta o poveste și va provoca utilizatorul să se pună în pielea unui personaj aflat într-o situație limită. Acest personaj este pus sa facă o alegere de către *Genie*. El trebuie să aleagă între trei saci în care se află soluții necunoscute jucătorului. *Genie* va dezvălui conținutul a doi dintre cei trei saci pe care îi va alege jucătorul pe rând. După ce alege primul sac și primește prima soluție el o poate alege sau poate sa renunțe la ea și să mai aleagă un sac din cei doi rămași. La final utilizatorul va afla dacă alegerea făcută este optimă și va mai primi doua întrebări la care răspunsul va fi da sau nu. Întrebările vor avea legătură cu povestea spusă de către *Alexa*. Jocul

[2]https://www.alexaskillstore.com/Amazing-Word-Master-Game/730
[3]https://www.alexaskillstore.com/Tricky-Genie/38177

conține doar doisprezece povești.

Acest joc este foarte potrivit pentru a-ți îmbunătăți abilitatea de a înțelege un text în engleză, dar si de a analiza cât de bună este o decizie.

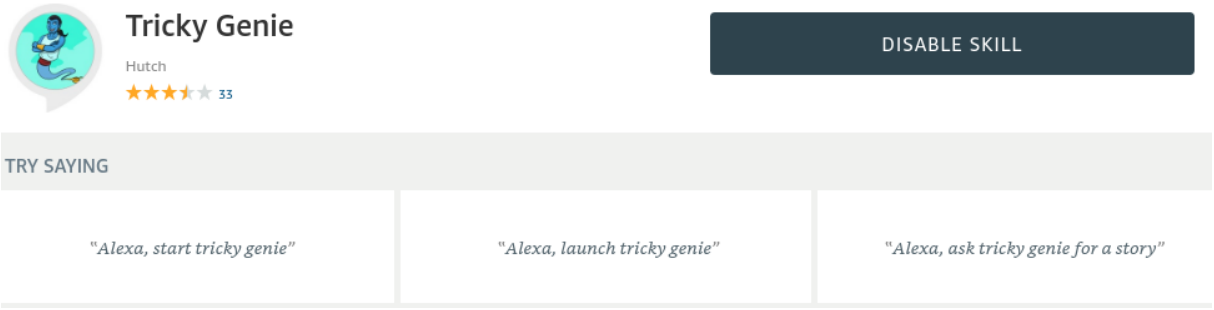

Figura 3.5: Interfața de activare a unei abilități *Tricky Genie*

3.3 Concluzii

În această secțiune a fost prezentată diversitatea de difuzoare inteligente care a apărut pe piață după lansarea *Amazon Echo* și o parte dintre aplicațiile asemănătoare cu jocuri disponibile prin intermediul interfeței vocale *Alexa* .

4 Tehnologii utilizate

4.1 AWS Lambda

Platform as a service (PaaS) este o platforma de calcul care asigură abstractizarea infrastructurii, sistemului de operare și a middleware-ului, astfel oferind suport pentru creșterea productivității dezvoltatorilor. (Alboaie, 2016). La acest nivel unul dintre serviciile pe care *Amazon Web Services* (AWS) le pune la dispoziție este *AWS Lambda*.

AWS Lambda ca orice serviciu computațional la nivelul PaaS ajută dezvoltatorii să nu se mai preocupe de administrarea resurselor hardware sau a sistemelor de operare, de probleme precum scalarea, performanța, securitatea sau monitorizarea. Acesta este utilizat pentru a rula codul acestora drept răspuns pentru evenimente precum: încărcarea de obiecte în *Amazon Simple Storage Service* (*Amazon S3*) sau modificări în tabelele din *DynamoDB*. După rularea codului o serie de metrici și log-uri sunt publicate în *Cloud Watch*. Fiecare dezvoltator poate să își creeze propriile metrici pe care să le utilizeze pentru a monitoriza părțile de interes din sistemul creeat.

4.2 DynamoDB

Not Only SQL sau NoSQL este un termen utilizat pentru a descrie o bază de date nerelațională. Această idee a luat amploare o data cu creșterea marilor furnizori de servicii (*Amazon*, *Facebook* și *Google*) care trebuie să proceseze volume imense de date. Pentru reprezentarea datelor NoSQL utilizează o varietate de modele precum documente, grafuri sau perechi chei valoare. Bazele de date nerelaționale sunt cunoscute pentru performanța demonstrată la scalarea pe orizontală, rezistența la erori și pentru disponibilitate.

DynamoDB este baza de date nerelațională pusă la dispoziție de către AWS. Aceasta suportă două tipuri de reprezentări ale datelor: documente și perechi cheie valoare. Această bază de date este renumită pentru flexibilitate, consistența datelor și faptul că interogările se realizează foarte rapid atât la scară mică cât și la scară mare. Cea din urmă caracteristică o recomandă pentru a fi folosită într-o varietate mare de aplicații precum aplicații pentru telefon, web, jocuri, internet of things etc. Câteva informații utile pentru dezvoltatorii care decid să utilizeze *DynamoDB* ar putea fi:

- Trigger-ele care sunt neobișnuite pentru o bază de date. Atunci când o schimbare apare într-o tabelă o Lambda funcție (serviciu prezentat în secțiunea 4.2) va fi apelată.
- Este integrată cu *AWS Identity and Access Management* (IAM), ceea ce ajută la asigurarea confidențialității datelor, deoarece doar un proces poate opera asupra unei baze de date

doar dacă acestă are permisiunile necesare.

- Consistența datelor la operațiile de citire este foarte bună, ceea ce asigură faptul că aplicația dezvoltată va răspunde clienților cu ultima valoare introdusă în baza de date.

4.3 JavaScript Object Notation

JavaScript Object Notation (JSON) este un format de transfer de date ușor, bazat pe text, independent de limbă. Acesta definește un set mic de reguli de formatare pentru reprezentarea portabilă a datelor structurate (RFC7159).

O reprezentare de tipul JSON poate fi formată dintr-un tip de date clasic sau un obiect. Tipurile de date clasice sunt un număr (natural, întreg sau real), o valoare de adevăr, un șir de caractere sau un vector, iar obiect care este format dintr-o colecție neordonată de perechi cheie valoare, unde cheile sunt în general șiruri de caractere, iar valorile sunt o reprezentare de tip JSON. Un exemplu de astfel de reprezentare se poate vedea în figura 5.1. Nume, prenume și ocupație sunt cheile acestuia. Se observa că valorile pot avea tipuri diferite (șiruri de caractere, numere naturale sau chiar alte obiecte) totodată se mai observă că putem folosi caractere în unicode.

```
{
    "nume": "Popescu",
    "prenume" : "Maria",
    "ocupa\u021Bie": {
        "titlu": "student",
        "institu\u021Ba": "Universitatea Alexandru Ioan Cuza"
    },
    "age" : 18
}
```

Figura 4.1: Exemplu de reprezentare JSON

Acest format este ușor de citit și de scris de către oameni, dar și ușor de procesat de către mașini.

4.4 Alexa

Alexa este serviciul vocal dezvoltat de către Amazon care permite utilizatorilor să interacționeze cu dispozitive într-un mod intuitiv, prin intermediul comenzilor vocale. Numărul dispozitivelor care integrează *Alexa* a crescut, iar tipul acestora a început să se diversifice.

O parte din serviciile pe care *Alexa* le furnizează se dovedesc foarte utile în bucătărie, fapt evidențiat din statistici 85% dintre utilizatorii de *Alexa* au setat o alarmă, iar 45% au adăugat un

obiect în coșul lor de cumpărături de pe Amazon. Probabil acest motiv a condus la colaborarea dintre *Amazon* și *LG*, care cu propus frigiderul inteligent *Instaview*.

Ford a integrat *Alexa* cu aplicația *Ford Sync*, ceea ce permite utilizatorilor o mulțime de operații prin intermediul comenzilor vocale în timp ce aceștia conduc. Aceștia pot folosi *Alexa* pentru a seta GPS-ul sau pentru a afla informații legate de mașină, drum, vreme, dar o pot utiliza și pentru a nu se plictisi în timpul drumului. *Alexa* este utilizată de multe ori pentru divertisment. Statisticile spun că 82% dintre utilizatori au ascultat muzică cu ajutorul *Alexa*, iar 66% au ascultat știrile din diverse ziare.

Alexa Skill Kit (ASK) este o colecție de API-uri care ajută dezvoltatorii să creeze rapid și cu ușurință abilități (eng. skills) pentru *Alexa*. Abilitățile sunt aplicațiile vocale pe care un utilizator le poate accesa prin intermediul lui *Alexa*. O abilitate este formată din două părți: **interfața abilității** și **skill service**. Pentru a avea o abilitate funcțională trebuie ca cele două părți să fie scrise astfel încât comunicarea dintre ele să fie perfectă.

4.4.1 Skill service

Skill service poate fi văzut ca nivelul de business al aplicației, deoarece acesta conține logica aplicației. Acesta determină ce acțiuni trebuie făcute în funcție de ceea ce utilizatorul a rostit și gestionează comunicarea cu serviciile externe, accesul la baza de date și informații legate de sesiune, dar se ocupă și de generarea răspunsului pe care *Alexa* îl furnizează utilizatorilor. De exemplu, în aplicația care sunt expuse informații despre starea vremii, componenta *skill service* face apel la un API extern, apoi procesează datele primite, punându-le sub o formă ușor de înțeles pentru utilizatorii umani.

Acestă componentă se poate dezvolta în orice limbaj de programare în care este posibil să fie dezvoltate servere HTTPS. Pentru a putea utiliza orice server configurații suplimentare sunt necesare, deoarece o abilitate *Alexa* are nevoie de suport SSL și de un certificat semnat.

De cele mai multe ori, dezvoltatorii aleg să utilizeze *AWS Lambda* care este o opțiune bună deoarece poate manipula evenimente în siguranță, astfel permițându-i-se serviciului *Alexa* să comunice automat și sigur cu AWS. Acesta nu necesită configurări suplimentare și totodată asigură toate beneficiile pe care nivelul PaaS îl furnizează.

Majoritatea dezvoltatorilor de aplicații pentru *Alexa* aleg ca în partea de backend să folosească **NodeJS**, deși *AWS Lambda* suportă și **Java**, **Python** sau **C#**.

4.4.2 Voice Service

Voice Service este partea abilității configurată de către dezvoltator pentru a procesa ceea ce un utilizator comunică dispozitivului integrat cu *Alexa*. Ea manipulează transformarea de limbajului natural în evenimente pe care *skill service* le poate utiliza. În cadrul *Voice Service* trebuie definit modelul de interacțiune cu abilitatea.

Voice Service este antrenată astfel încat aceasta să știe cum trebuie să asculte utilizatorii, pentru ca apoi să transforme cuvintele acestora în acțiuni ce vor fi efectuate de către *skill service*. Pentru a antrena modalitatea de ascultare, dezvoltatorilor le revine sarcina să definească în cadrul modelului de interacțiune legătura dintre anumite propoziții și acțiuni (eng. *intent*), prin furnizarea de exemple (eng. *utterance*). Cele două concepte reprezintă inima unui model de interacțiune deoarece cu ajutorul lor este generat modelul care ajută la înțelegerea limbajului natural utilizat.

Un **intent** reprezintă o acțiune ce va îndeplini o cerere venită de la un utilizator. Opțional un intent poate avea parametrii, denumite **slot-uri**.

Utterances sunt șiruri de caractere ce reprezintă o modalitate prin care un utilizator poate comunica cu o abilitate împreuna cu intent-urile care vor genera răspunsul.

De exemplu, în figura 5.2 ceea ce comunică utilizatorul aplicației *"Alexa, ask Greeter to say Hello"* este cautat în lista de perechi utterence, intent (detaliată în figura `5.2`, în partea stângă a cercului portocaliu), apoi *Voice Service* decide că intent-ul corespunzător este `helloWorldIntent`.

Figura 4.2: Exemplu de procesare a perechilor utterance, intent (*The Big Nerd Ranch*, 2016)

Atunci când un dezvoltator definește utterance-urile acesta trebuie să ia în considerare diferitele modalități în care un utilizator poate comunica cu abilitatea. Este important să fie adăugate cât mai multe utterances deoarece crește probabilitatea ca la interacțiunea unui utilizator cu aplicația

acesta să poată traduce limbajul natural într-un intent. Dacă aplicația nu este capabilă să facă transformarea limbajului natural într-un intent, utilizatorul nu va primi răspunsul dorit.
În cazul în care un dezvoltator dorește să creeze abilități mai complexe, acesta are nevoie de slot-uri. Slot-urile ajută la definirea utterances-urilor astfel încât același model de interacțiune poate procesa mai multe valori ale informațiilor pe care un utilizator le furnizează abilității. Putem să le privim ca pe niște variabile ce sunt furnizate unui intent. Modelul de interacțiune are sarcina de a face legătura dintre inten-uri și slot-uri. Acestea pot avea valori diverse precum nume, date, numere, localități, nume de aeroporturi etc..
De exemplu, dacă analizăm modul de interacțiune cu abilitatea care ne comunică starea vremii într-o locație specifică. Atunci când cerem să ne ne fie comunicată vremea în Londra, în modelul de interacțiune, mai precis în utterance nu se află cuvântul Londra ci un slot al cărei valori este o localitate, iar printre acele localități se află și Londra.
La fel ca și variabilele dintr-un limbaj de programare slot-urile au tipuri care indică dezvoltatorilor ce valori poate lua un anumit slot. Pentru a putea utiliza întru-un intent un slot trebuie definite în cadrul modelului de interacțiune slot-urile utilizate pentru un intent și tipul acestora. Există trei tipuri de **tipuri ale sloturilor** dintre care dezvoltatorii pot alege:

- *Slot-urile standarde* sunt utilizate pentru cuvinte comune precum date, numere sau nume. Dacă un dezvoltator va defini un slot de acest tip atunci cuvintele utilizatorului vor fi transformate în cea mai similară valoare a slot-ului.

- Slot-uri ce au *tipuri definite de dezvoltator* sunt utilizate atunci când cele standarde nu oferă resursele necesare dezvoltării abilității dorite. Ele sunt definite în modelul de interacțiune și se comportă similar cu cele standard.

- Slot-uri de *tipul literal* acest tip nu este recomandat pentru utilizare și este inclus doar din motive de compatibilitate cu abilități dezvoltate anterior.

4.5 Interactiunea dintre Voice Service și Skill Service

După cum am precizat mai sus, un skill este suma a două componente: *Voice Service* și *Skill Service*. Informațiile procesate de către Voice Service, adică tipul intentului cât și valorile slot-urilor sunt transmise către Skill service, care va procesa datele și va forma răspunsul pentru utilizator. Cele două componente trebuie să interacționeze perfect și să se completeze reciproc pentru a crea o experiență cât mai bună pentru utilizator atunci când acesta folosește *Alexa*.
Dacă în partea de Service Skill am ales să lucrăm cu *AWS Lambda*, atunci Lambda funcția

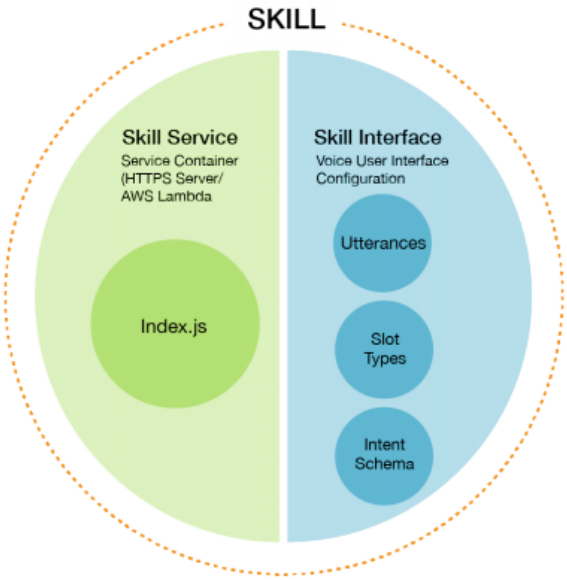

Figura 4.3: Elementele componente ale unei abilități (*The Big Nerd Ranch*, 2016)

va fi rulată de fiecare dată când un utilizator va interacționa cu aplicația. Acesta va primi la intrare de la *Alexa* un `JSON`, reprezentând o procesare a textului rostit de către utilizator în timpul interacțiunii cu dispozitivul care este integrat cu *Alexa*. Procesarea va fi făcută pe baza modelului de interacțiune. Rezultatul procesării informațiilor primite va fi un obiect `JSON` care va conține răspunsul cererii precum și alte informații. Drumul unei cereri făcută de către un utilizator către un dispozitiv integrat cu *Alexa* este prezentat în figura `5.4`.

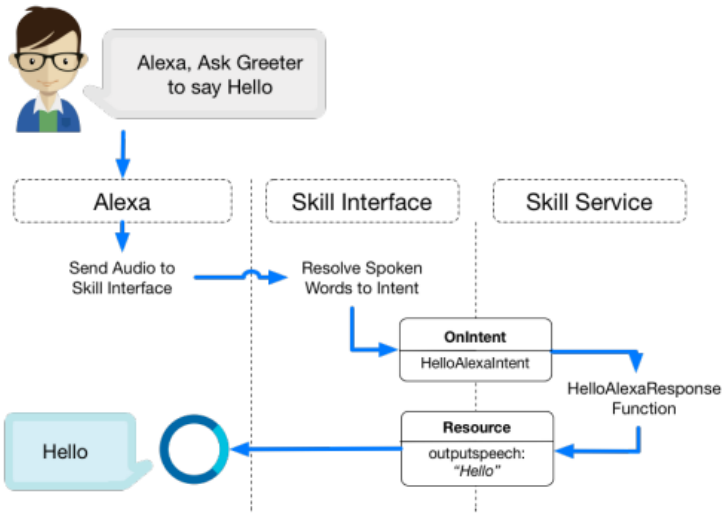

Figura 4.4: Drumul unei cereri de la utilizator (*The Big Nerd Ranch*, 2016)

4.6 Ontologie

Termenul de *ontologie* este folosit multe domenii cu sensuri diferite. Acesta a fost împrumutat în informatică din filosofie, unde acesta definește un punct de vedere sistematic asupra existenței. O ontologie din punct de vedere computațional este o modalitate de a structura un univers, prin modelarea unui sistem care organizează entități și relațiile dintre acestea. Prin entitate se înțelege ”lucru” identificabil în mod distinct pe baza unei semnificații (en. *denotation*) și a unui conținut (en. *connotation*) (Kingsley Idehen, 2014). Fiecare entitate este identificată printr-un identificator unic, eventual specificat de un standard, de exemplu URI, cod QR, UUID. Relațiile dintre entități oferă organizarea structurată a ontologiilor și mijlocesc descrierea acestora, dând valoare informațiilor.

Clasele sunt utilizate pentru a descrie concepte. De exemplu, în figura 5.5 clasa pusă în evidență clasa vinurilor. Tipurile de vinuri reprezintă instanțe ale clasei (ex. *Bordeaux*). Proprietățile descriu clase sau instanțe ale claselor. În exemplul amintit *Chateau Lafite Rothschild Pauillac* este un tip de vin, adică o instanță, care este descrisă cu ajutorul relațiilor ca fiind produs de *Chateau Lafite Rothschild*, producătorul reprezintă o proprietate în acest context.

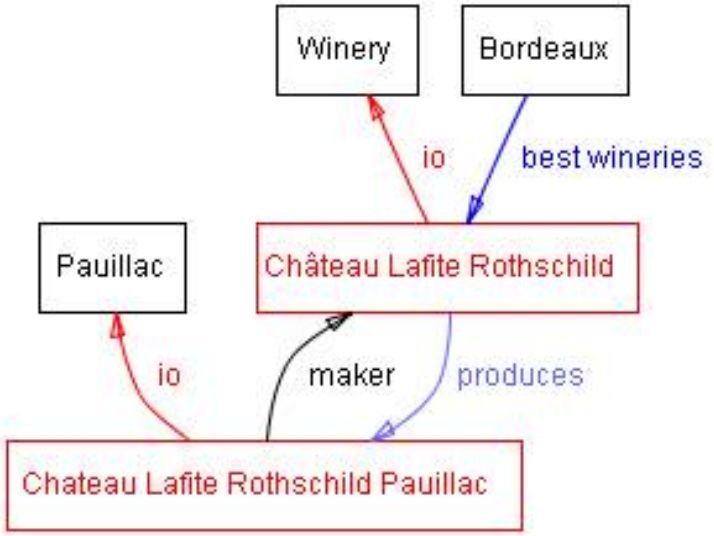

Figura 4.5: O ontologie ce modelează o parte din domeniul vinului, clasele sunt marcate cu negru, iar instanțele cu roșu.(*Noy*, 2001)

Inima unei ontologii o reprezintă conceptele de generalizare și particularizare. O ierahie de concepte se numește taxonomie. De exemplu, suntem interesați de aspecte legate de animale, atunci mamifer, câine și pisica ar putea fi concepte de interes, unde primul este un superconcept pentru cele două din urmă.

4.7 Knowledge Graph

În momentul de față web-ul poate fi privit ca o mulțime de documente care conțin legături între ele. *Sir Tim Berners-Lee* a adus o nouă viziune asupra web-ului în anul 2004. Acesta susține că în momentul de față web-ul în sensul clasic ar trebui să fie de domeniul trecutului, omenirea având nevoie de un web al datelor, unde orice relațiile dintre două entități, de exemplu persoane, localități, companii etc., să poată fi reprezentată pe web. Acest nou concept este cunoscut sub denumirea de **Semantic Web**.

În prezent **Knowledge Graph**-urile sunt văzute drept unul dintre elementele de bază pentru Semantic Web. Acestea sunt definite ca o combinație dintre o ontologie și instanțe ale claselor dintr-o ontologie, conținând un număr mare de informații despre entități. (*Färber*, 2015)

Cele mai cunoscute exemple de reprezentare ale Knowledge Graphs sunt *Resource Description Framework* (RDF) și *Web Ontology Language* (OWL). Fiecare obiect, nod în graf, este reprezentat prin intermediul unui identificator unic, de cele mai multe ori URI, și este de obicei asignat unei clase, de exemplu locuri, lacuri, orașe, țări, persoane, actori etc. Pentru fiecare nod sunt definite o serie de proprietăți, pe care le putem privi drept relații sau muchii în graf. De exemplu pentru un actor o proprietate asignată ar putea fi data nașterii sau premiile câștigate. Acest mod de reprezentare deschide noi orizonturi de interconectare a datelor.

4.8 Resource Description Framework

Resource Description Framework (RDF) a avut ca scop inițial modelarea metadatelor resurselor prezente pe Web. Această formă de reprezentare și-a dovedit însă utilitatea și în cadrul aplicațiilor de gestionare a cunoștiintelor.

RDF oferă un model de specificare a triplelor (subiect, predicat, obiect) prin intermediul unor identificatori uniformi de resurse (URI). Această formă de reprezentare generează un Knowledge Graph, fiecare nod al grafului RDF fiind referit în mod unic prin intermediul unui URI.

Subiectul reprezintă un obiect, predicatul stabilește natura relației dintre un subiect și un obiect. Poate fi o caracteristică, un atribut sau o relație, iar obiectul este o resursă care poate deveni de cele mai multe ori la rândul său ca un subiect. De exemplu, pentru tripleta (Sir Timothy John, was born, London) Londra este în cazul de față un obiect, însă aceasta poate deveni oricând subiect. În tripleta (Londra, este străbătută, Tamisa) Londra este subiect. Diferența dintre un subiect și un obiect îl reprezintă faptul ca un obiect poate avea a valoare specifică, precum un număr întreg, iar un subiect trebuie să fie specificat printr-un URI.

Fiecare element al unei triplete RDF trebuie să fie inclus într-un **namespace**. Acesta este utilizat

pentru a asigura unicitatea subiectelor, predicatelor și al obiectelor ce compun o tripletă RDF. Namespace-urile sunt de obicei reprezentate prin URI-uri.

În figura 5.6 este un exemplu de graf RDF. Observăm că nodurile *Messi* și *FC Barcelorna* fac parte dintr-o tripletă RDF, unde *Messi* este subiect, iar *FC Barcelona* este obiect. Acestea se află în relația joacă pentru (en. *plays for*). *FC Barcelona* este sunbiect în tripleta (*FC Barcelona*, population, 5.5M), iar obiectul 5.5M este o valoare specifică, care nu poate fi la rândul ei subiect.

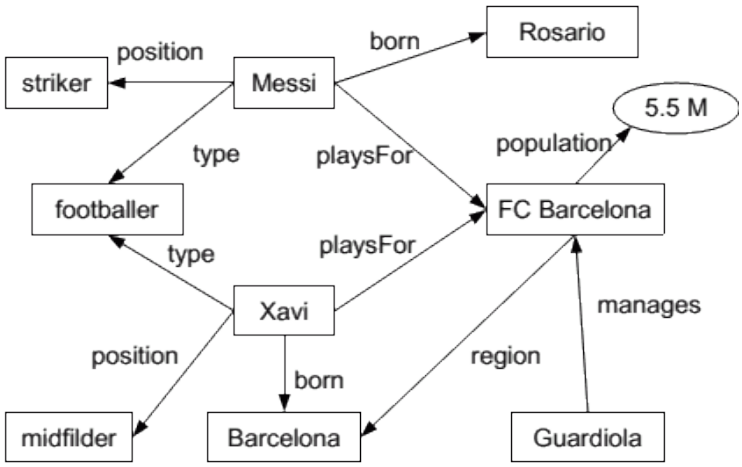

Figura 4.6: Exemplu de graf RDF (http://0agr.ru/wiki/index.php/SPARQL)

Cele mai cunoscute doua metode de serializare a unui graf RDF sunt XML/RDF și Turtle.

Exemplul de serializare turtle a unui graf *RDF* din figura 5.7 începe cu o linie de comentariu în care este specificat numele fișierului. Următoarele două linii conțin triplete reprezintă namespace-urile pe care le vom utiliza în cadrul fișierului. Cele două linii de fapt definesc un *syntactic sugar* pentru mai departe astfel încât dezvoltatorii să nu fie nevoiți pentru fiecare element din namespace-ul *http://learningsparql.com/ns/addressbook#* să fie scris întreg URI-ul, acesta este înlocuit de caracterele *ab*. Aceste abrevieri se numesc **prefixe**. Următoarele linii conțin o listă de triplete RDF, unde subiectele și predicatele sunt date prin ID-urile lor unice, adică URI-uri sau prefixe urmate de numele obiectelor. Obiectele pot fi la rândul lor identificate prin ID-uri sau pot fi valori specifice. În exemplul din figura 5.7 obiectele au valori specifice.

```
# filename: ex012.ttl

@prefix ab: <http://learningsparql.com/ns/addressbook#> .
@prefix d: <http://learningsparql.com/ns/data#> .

d:i0432 ab:firstName "Richard" .
d:i0432 ab:lastName "Mutt" .
d:i0432 ab:homeTel "(229) 276-5135" .
d:i0432 ab:email "richard49@hotmail.com" .

d:i9771 ab:firstName "Cindy" .
d:i9771 ab:lastName "Marshall" .
d:i9771 ab:homeTel "(245) 646-5488" .
d:i9771 ab:email "cindym@gmail.com" .

d:i8301 ab:firstName "Craig" .
d:i8301 ab:lastName "Ellis" .
d:i8301 ab:email "craigellis@yahoo.com" .
d:i8301 ab:email "c.ellis@usairwaysgroup.com" .
```

Figura 4.7: Exemplu de serializare *Turtle* a unui graf RDF (*DuCharme*, 2011)

4.8.1 Dbpedia

Wikipedia este o bază de cunoștințe extrem de importantă. Conform site-ului *alexa.com* în momentul de față (mai 2017) *Wikipedia* este al cincilea cel mai vizitat site din lume. Resursele pe care aceasta le conține sunt extrem de numeroase, de exemplu doar în limba engleză sunt scrise peste cinci milioane de articole, conform datelor publicate de aceștia. Cu toate acestea din punct de vedere computațional utilizarea acesteia este aproape imposibilă din cauza faptului că informația nu este într-o formă structurată.

Proiectul **DBpedia** are ca scop structurarea conținutului *Wikipedia*. În cadrul acestuia a fost dezvoltat un framework care este capabil să transforme conținutul Wikipedia în conținut RDF. Această structurare oferă dezvoltatorilor de aplicații software să construiască o varietate de aplicații pentru Semantic Web, dar poate fi utilizată și pentru aplicații în inteligența artificială.

Versiunea în limba engleză a *DBpedia* baza de cunoștiințe conține 4,58 de milioane de lucruri dintre care 4,22 milioane sunt clasificate într-o ontologie consistentă care include 1,4 milioane de persoane, 735 de mii de locuri (incluzând 478 de mii de locuri populate), 411 de mii de lucrări de artă (incluzând 123 de mii de albume, 87 de mii de filme și 19 de mii de jocuri video), 241 de mii de organizații (incluzând 58 de mii de de companii și 49 de mii de de instituții educaționale), 251 de mii de specii și 6 mii de de boli. (*http://wiki.dbpedia.org/about*, 2017)

Pentru a accesa conținutul *DBpedia* există mai multe metode pot fi accesate online prin intermediul interfeței grafice, prin reprezentări RDF ale resurselor sau prin intermediul unui endpoint SPARQL, utilizat de obicei de către aplicațiile software (vezi figura 5.8).

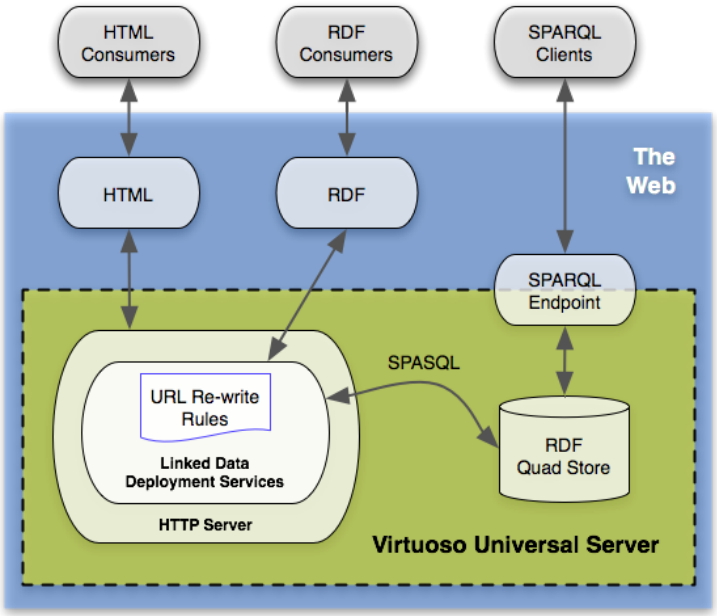

Figura 4.8: Exploatarea conținutului *DBpedia* (http://wiki.dbpedia.org/about/architecture)

4.8.2 Wikidata

Wikidata este o altă bază de cunoștințe extrem de importantă și este operată de către *Wikimedia Foundation*. Aceasta a fost lansată la finalul anului 2012 și este o ontologie colaborativă, adică la fel ca și în cazul *Wikipedia* oricine poate contribui la informațiile pe care aceasta le conține. Comunitatea a ajutat la dezvoltarea acesteia, astfel într-un timp scurt numărul de entități reprezentate a crescut considerabil. Probabil această se datorează legăturii dintre *Wikipedia* și *Wikidata*. Astăzi majoritatea paginilor de pe *Wikipedia* conțin elemente preluate de pe *Wikidata*, folosite mai ales pentru a înterconecta articolele despre același subiect însă scrise în limbi diferite.
Conținutul este structurat în triplete RDF, iar conceptele de bază se numesc elemente (en. *items*). Elementele, adică clasele și instanțele, sunt identificate printr-un identificator unic, care este un număr precedat de litera *Q*, indiferent de limba în care sunt prezentate. De exemplu, toate informațiile despre România, indiferent dacă o numești Romania, *Romania* (en. România) sau *Rumunia* (pl. România) sunt unificate sub ID-ul *Q218*. Proprietățile, adică relațiile, la fel ca și elementele sunt identificate printr-un ID. Acesta este un număr precedat de litera *P*. De exemplu, relația limbă oficială are ID-ul *P37*.
Wikidata se afla într-o continuă dezvoltare. Pentru a avea acces la conținutul *Wikidata* se poate

folosi pagina web, sau endpoint-ul SPARQL fie programatic, fie prin interfața pusă la dispoziție pe pagina lor web.

4.9 SPARQL

SPARQL definește două concepte. Pe de o parte acesta este un acronim recursiv pentru protocolul SPARQL, iar pe de altă parte este un acronim pentru *RDF Query Language*, care este descris printun set de reguli de către *World Wide Web Consortium* (W3C). În esență SPARQL este un mod de a interoga RDF la fel cum SQL este un mod de a interoga baze de date relaționale.
SPARQL funcționează la fel ca și SQL adică sunt selectate o parte din obiectele din baza de date, apoi o parte din proprietățile acestora sunt selectate și furnizate celui care a facut interogarea (vezi figura 5.9).

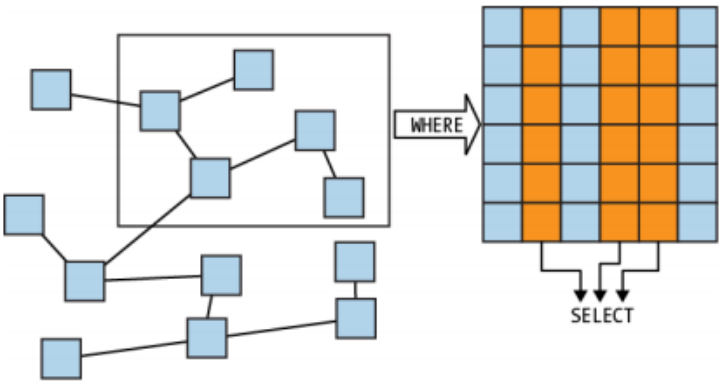

Figura 4.9: Modul de funcționare al unei interogări SPARQL (*DuCharme*, 2011)

Inițial interogările SPARQL par asemănătoare cu cele SQL, deoarece conțin o serie de cuvinte cheie comune (*SELECT*, *WHERE*), însă SPARQL permite și o serie de operații pe care SQL nu le suportă prin intermediul unor cuvinte cheie. Acestea sunt *FILTER* și *OPTIONAL*. Condițiile pentru selectarea subiectelor în SPARQL sunt descrise prin triplete asemănătoare cu cele RDF, dar pot conține și variabile care adaugă flexibilitate interogărilor. Un exemplu de interogare este în figura 5.10, iar acesta se face asupra datelor din graful RDF al cărui serializare o este în figura 5.7.
La fel ca și în serializarea *turtle* pentru SPARQL se definesc prefixe care sunt de fapt *syntactic sugar* pentru namespace-uri. După cuvântul cheie *SELECT* vor fi numele de variabile ce reprezintă proprietățile subiectelor pe care interogarea le va furniza. Apoi urmează o serie de triplete RDF, unde predicatele și obiectele au valori cunoscute, iar subiectele sunt variabile. Întotdeauna variabilele sunt precedate de *?* (vezi figura 5.10).
Rezultatul interogării din figura 5.10 este în figura 5.11.

```
# filename: ex013.rq
PREFIX ab: <http://learningsparql.com/ns/address1book#>
SELECT ?craigEmail
WHERE
{
 ?person ab:firstName "Craig" .
 ?person ab:email ?craigEmail .
}
```

Figura 4.10: Exemplu de serializare *Turtle* a unui graf RDF (*DuCharme*, 2011)

Figura 4.11: Rezultatul interogării SPARQL din figura 5.10 (*DuCharme*, 2011)

SPARQL este utilizat de cele mai multe ori pentru a interoga resurse publice precum *DBpedia* sau *Wikidata* prin intermediul unui endpoint SPARQL, care este un serviciu web care acceptă interogări SPARQL.

4.10 Concluzii

În această secțiune au fost prezentate tehnologiile și conceptele ce au fost folosite în cadrul dezvoltării aplicației. Au fost expuse noțiuni elementare pe care un dezvoltator trebuie să le aibă atunci când dezvoltă o abilitate pentru *Alexa*, dar și elemente care țin de elemente utilizate pentru dezvoltarea abilității *Bob*, a cărei modalitate de dezvoltare este descrisă în secțiunea 6.

5 Dezvoltarea aplicației

5.1 Introducere

În secțiunile ce urmează va fi prezentată arhitectura aplicației, dar și elemente care țin crearea mediului de lucru pentru ca abilitatea să funcționeze. Prezentarea se concentrează asupra serviciilor utilizate, modului de utilizare, dar totodată va avea în vedere și motivarea alegerilor arhitecturale făcute. În figura 6.1 este prezentată o imagine de ansamblu despre interacțiunea dintre toate serviciile utilizate pentru dezvoltarea abilității.
În cele ce urmează se vor dezvolta următoarele idei:

- configurarea modelului de interacțiune care furnizează *Voice Sevice-ului* informații despre modul în care trebuie procesat limbajul natural
- cum funcționează componenta *Skill Sevice* pentru abilitatea dezvoltată, adică cum se configurează *AWS Lambda* astfel încât să primească cereri de la *Voice service*, cum este procesată o cerere de către serviciu, cum este furnizat răspunsul, dar și elemente arhitecturale.
- cum poate un utilizator să se autentifice cu un cont pe care acesta îl are la o terță parte? Cum a fost implemenetată pentru abilitatea dezvoltată.
- necesitatea utilizării unei baze de date în cadrul abilitații dezvoltate
- selectarea datelor neceresare pentru popularea bazei de date, dar și modalitatea de a modifica datele prezente în aceasta

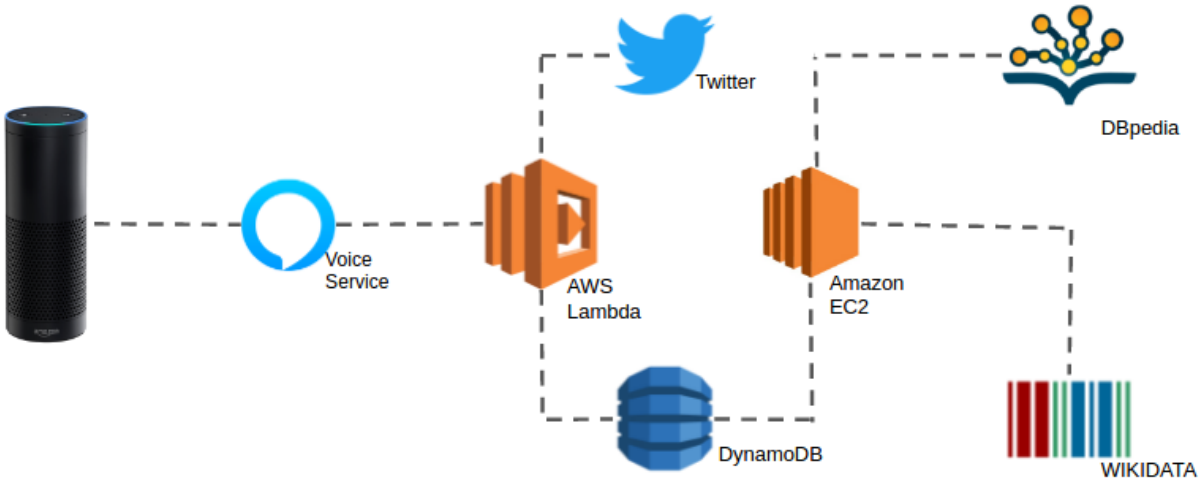

Figura 5.1: Interacțiunea dintre serviciile folosite pentru dezvoltarea aplicației

5.2 Configurarea *Voice service-ului*

Pentru a configura partea de `Voice service` trebuie mai întâi să ne avem un cont pe site-ul *https://developer.amazon.com*, apoi să creăm o noua abilitate pentru *Alexa*. După ce este creată o noua abilitatea, ea va primi un identificator unic, iar dezvoltatorul va trebui să completeze mai multe informații: despre modelul de interacțiune, despre integrarea cu *Skill Service* sau despre publicarea abilității (descrierea abilității, instrucțiuni de testare etc.). Cea din urmă categorie va fi completată la finalul dezvoltării abilității.

Pentru configurarea *Voice service-ului* este necesară crearea modelului de interacțiune. Acesta conține *scheama de intent-uri*, *tipurile slot-urilor*, *utterance-urile* (vezi secțiunea `5.4.2`).

5.2.1 Schema de *intent-uri*

Schema de *intent-uri* este un obiect `JSON` care conține o listă de obiecte `JSON` ce înglobează informații despre fiecare *intent* pe care abilitatea dezvoltată este capabilă să îl manipuleze.

Obiectele ce conțin informații despre un *intent* vor stoca numele acestuia și opțional o listă de obiecte ce indică *slot-urile* care pot apărea în cadrul *intent-ului* împreuna cu tipul acestora. Numele pentru *intent-uri* pot fi atât oferite de către `ASK`, precum și definite de către dezvoltator, la fel ca și tipurile slot-urilor.

```
{ "intents": [
    {
      "intent": "AMAZON.CancelIntent"
    }, {
      "intent": "BobBeginTest"
    }, {
      "intent": "BobAnswer",
      "slots": [
        {
          "name": "AnswerL",
          "type": "LAKE_LIST"
        },
        {
          "name": "AnswerC",
          "type": "AMAZON.Country"
        } ],
    }
  ]
}
```

Figura 5.2: O parte dintre intenturile definite în cadrul aplicației *Bob*

Aplicația *Bob* are mai multe intent-uri, o parte din schema acestora este reprezentată de în figura 6.2. Numele de *intent*:

- `AMAZON.CancelIntent` este definit în `ASK`, iar sub acest nume sunt adunate o serie de replici pe care utilizatorul le poate comunica atunci când acesta dorește să oprească abilitatea;
- `BobBeginTest` este definit de către dezvoltator, iar acesta ascunde replici pe care utilizatorul le va putea spune atunci când dorește să înceapă un nou test de cultura generală;
- `BobAnswer` este definit de către dezvoltator, iar acesta înglobează replici pe care utilizatorul le poate spune atunci când dorește să răspundă la o întrebare de cultură generală. Deoarece există foarte multe posibile răspunsuri o serie de *slot-uri* sunt utilizate. Două dintre acestea se pot vedea în figura 6.2:
  - `AnswerL`, care are tipul `LAKE_LIST`, definit de către utilizator
  - `AnswerC`, care are tipul `AMAZON.Country`, acest tip este definit în `ASK`

5.2.2 Tipuri pentru *slot-uri*

Pentru a defini tipul pentru un *slot* trebuie enumerate toate valorile pe care acesta le poate lua. Pentru a adăuga un nou tip în cadrul modelului de interacțiune la secțiunea *Custom Slot Types* vom apasă butonul *Add Slot Type*, apoi este necesar să primească un nume și valorile pe care acesta le înglobează. În figura *6.3* sunt prezentate *tipurile slot-urilor* create de către dezvoltator care sunt folosite în cadrul aplicației *Bob*.

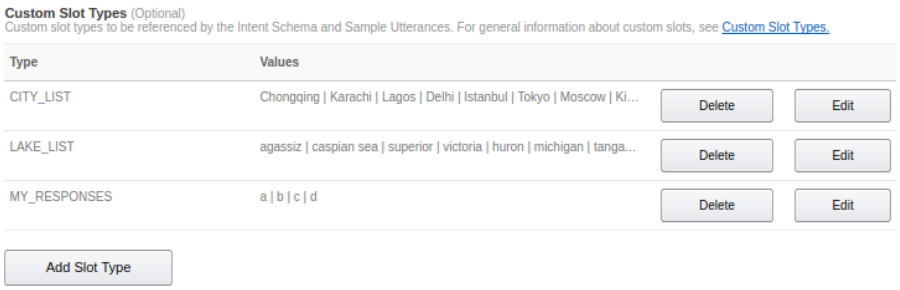

Figura 5.3: Interfața ce prin intermediul căreia se face managementul pentru tipul *slot-urilor*

Acestea sunt:

- `CITY_LIST` care conține numele tuturor orașelor ce constituie un răspuns corect la întrebările despre orașe. Nu a fost folosit unul dintre tipurile din `AMAZON.GB_CITY`, `AMAZON.DE_CITY` sau altele, deoarece acestea conțin cele mai utilizate orașe în vorbirea curentă pentru vorbitorii dintr-o anumita zonă (`AMAZON.GB_CITY` pentru Marea Britanie).

- `LAKE_LIST` care conține numele lacurilor orașelor ce constituie un răspuns corect la întrebările despre orașe. Nu a fost folosit unul dintre tipurile din `AMAZON.Landform`, deoarece nu conține numele tuturor lacurilor ce pot fi un raspuns.
- `MY_RESPONSES` care conține literele ce pot fi variante de răspuns pentru întrebări. Această formă a răspunsului poate fi folosită indiferent de domeniul din care este întrebarea.

5.2.3 *Utterance-uri*

Utterance-urile trebuie adaugate în cadrul secțiunii *Sample Utterances* a modelului de interacțiune. Fiecare *utterance* va fi despărțit prin *enter*. După cum este specificat și în secțiunea `5.4.2`, fiecare posibilă replică a utilizatorului este precedată de *intent-ul* pe care acestă îl generează. O parte din utterance-urile folosite de către aplicația *Bob* sunt în figura `6.4`.

Figura 5.4: Interfața ce prin intermediul căreia se face managementul *utterance-urilor*

5.3 *Skill Service*

Pentru partea de *skill service-ului* a dezvoltării aplicației *Bob* a fost ales serviciul *`AWS` Lambda*, deoarece aceasta oferă suport pentru lucrul cu abilități pentru *Alexa* și totodată pentru ușurința cu care poți apela alte servicii `AWS` prin intermediul unei funcții lambda. Limbajul de programare utilizat este `NodeJS`. A fost ales acest limbaj de programare, deoarece se recomandă a fi utilizat în aplicații de mici dimensiuni, pe partea de server, unde operațiile non blocante sunt foarte importante.

5.3.1 Modalitatea de creare a legăturii între `AWS` Lambda și *Voice Sevice*

Pentru a crea o lambda funcție este necesar ca dezvoltatorul să aibă un cont `AWS` și să acceseze secțiunea ce se ocupă de management-ul acestui serviciu. În secțiunea de creare a unei lambda funcții va fi întâi necesar ca dezvoltatorul să aleagă un blueprint, adică mediul de rulare, care în

cazul aplicației *Bob* este Node.js `6.10`, și un template pentru funcția ce urmează a fi dezvoltată. Pasul al doilea constă în configurarea declanșatorului funcției, în cazul nostru va fi `Alexa Skill Kit` (vezi figura `6.5`). În continuare în cadrul secțiunii *Configure function* vor fi furnizate informații despre permisiuni, limite de memorie, timeout, etc. După urmarea acestor pași lambda funcția a fost creată.

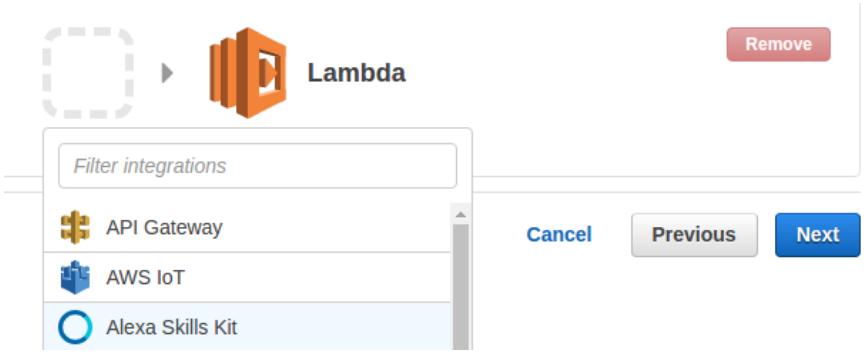

Figura 5.5: Interfața de configurare a declanșatorului pentru o lambda funcție

După crearea funcției la secțiunea configurare (en. *configuration*) a *voice service-ului* va trebui furnizat `ID-ul` funcției. Acest pas asigură trimiterea informațiilor provenite de la utilizator, după procesarea acestora, către lambda funcția ce va decide răspunsul pe care acesta îl va primi. Pentru a avea o abilitate funcțională trebuie să facem *deploy* codului dezvoltat. Acesta se poate face în patru moduri:

- scriind codul direct în IDE-ul pus la dispoziție în cadrul secțiunii pentru funcția lambda;
- încărcând codul în Amazon S3, iar apoi să fie furnizat un link către acesta;
- creând o arhiva cu extensia `.zip` care apoi va fi încărcată în cadrul secțiunii pentru funcția lambda;
- folosind *AWS Command Line Interface* (`CLI`). Pentru a putea face deployment-ul cu ușurință am creat un alias, denumit `lambda`, care avea ca scop crearea arhivei cu extensia `.zip`, apoi *deployment-ul* propriu zip. Pentru a crea acest alias am adăugat în fisierul `/.bash_aliases` ceea ce se află în figura `6.6`

```
alias lambda="rm Archive.zip ; zip -r Archive.zip my_modules/* node_modules/ ind
ex.js ; aws lambda update-function-code --function-name bobTrigger --zip-file fi
leb://Archive.zip"
```

Figura 5.6: Interfața de configurare a declanșatorului pentru o lambda funcție

5.4 Descrierea manipulării intent-urior în aplicația Bob

Aplicația *Bob* oferă utilizatorilor posibilitatea de efectua următoarele acțiuni principale:

- primi întrebări de cultura generală din domeniul geografiei
- iniția teste de cultură generală în urma cărora vor obține un punctaj care le va posibilitatea să avanseze în clasamentul general al jucătorilor
- pot afla locul în clasamentul general și numărul de puncte pe care aceștia l-au acumulat
- pot publica pe *Twitter* punctajul acumulat

În continuare va fi descris cum au fost obținute funcționalitățile de mai sus, presupunând că în baza de date deja avem informații care pot ajuta la formarea unei întrebări.

5.4.1 Construirea întrebărilor

Aplicația *Bob* poate primi de la utilizatori cereri să formeze întrebări despre orașe, țări, lacuri sau o categorie aleatoare. Aceste se traduce de către componenta *Voice Service* în *intent-urile* `BobAskCity`, `BobAskCountry`, `BobAskLake`, respectiv `BobRandomQuestion`. În continuare este prezentat un studiu de caz despre cum este manipulat *intent-ul* `BobAskLake`, la fel se va proceda și pentru celelalte *intent-uri* amintite mai sus.
Atunci când un *intentul* `BobAskLake` este primit de către componenta *Service Skill* atunci va fi apelată funcția *handleLakeIntent* ce face parte din modulul *lake*. În urma apelării aceasta va furniza utilizatorului o întrebare despre un lac pentru care există informații în baza de date. Funcția *handleLakeIntent* este prezentată în figura `6.7`. Ea formează un obiect de configurare ce conține: ID-ul cu valoarea maximă din baza de date și pe cel cu valoarea minimă, numele tabelei în care se află informații despre categoria din care face parte întrebarea ce urmează a fi formulată, în cazul de față acesta este *lake*, și obiectul ce corespunde funcției care construiește întrebarea. La final acesta apelează funcția `ask_answer.handleAsk` care indiferent de categoria din care face parte întrebarea ce urmează a fi construită va alege în mod aleatoriu o înregistrare din baza de date, apoi funcția `ask_config.question` va forma o întrebare. Totodată înainte de a furniza întrebarea utilizatorului răspunsul corect va fi adăugat la sesiune, împreună cu alte informații, pentru ca acestea să poată fi utilizate ulterior.
Cererea primită de la *Voice Service*, reprezentat printr-un obiect `JSON`, conține cheia *request*, care are drept valoare un obiect JSON ce conține informații despre ultima interacțiune a clientului cu interfața vocală *Alexa*. În cadrul acestui obiect se găsește un rezumat al cererii, fiind precizat

```
function handleLakeIntentLocal(session, callback, pre_ask = "") {
    var ask_answer = require('./ask_answer.js');
    var ask_config = {
        table: "lake",
        min: minimum,
        max: maximum,
        question: construct_question
    }
    ask_answer.handleAsk(ask_config, session, callback, pre_ask);
}
exports.handleLakeIntent = handleLakeIntentLocal;
```

Figura 5.7: Manipularea unui *intent-ului BobAskLake*

tipul *intent-ului* și eventual valoarea *slot-urilor* dacă acesta există. Fiecare *slot* specific pentru un *intent-ul* curent este prezent în obiect, însă dacă unul dintre ele nu este utilizat în cererea clientului câmpul *value* nu va exista (vezi figura 6.8).

```
"request": {
    "type": "IntentRequest",
    "requestId": "EdwRequestId.cf29c698-7d71-4498-98c9-9313860ffe6c",
    "locale": "en-US",
    "timestamp": "2017-05-28T06:39:43Z",
    "intent": {
      "name": "BobAnswer",
      "slots": {
        "AnswerC": {
          "name": "AnswerC"
        },
        "AnswerLetter": {
          "name": "AnswerLetter",
          "value": "d"
        },
        "Answer": {
          "name": "AnswerCi"
        },
        "AnswerL": {
          "name": "AnswerL"
        }
      }
    }
  }
```

Figura 5.8: Exemplu pentru cheia *request* a unui obiect JSON ce reprezintă o cerere de la utilizator

Pentru a analiza răspunsul acesta este extras din obiectul JSON de răspuns apoi este comparat cu datele din sesiune ce reprezintă răspunsul corect. Dacă acestea sunt identice, atunci răspunsul este considerat corect, utilizatorul este notificat și este întrebat dacă dorește să afle mai multe informații despre răspuns, dacă în baza de date acestea există. Utilizatorul va răspunde la acesta întrebare cu *Yes* sau *No*, ceea ce va genera un *intent-ul* AMAZON.YesIntent, respectiv AMAZON.NoIntent. Aceste două *intent-uri* pot fi generate oricând. De exemplu, un utilizator poate spune *yes* drept răspuns la o întrebare sau în locul unei operații pe care acesta o cere lui *Alexa*. Pentru a face distincția între momentele în care utilizatorul este cu adevărat îndreptățit să dea aceste răspunsuri și momentele când nu, se va adăuga la sesiune câmpul MoreInfo ia valoarea true atunci când este legal ca următorul *intent* să fie de tipul AMAZON.YesIntent sau AMAZON.NoIntent.

5.4.2 Formarea testelor

Aplicația *Bob* poate primi de la utilizatori cereri să înceapă un test de cultură generală nou. Aceste se traduce de către componenta *Voice Service* în *intent-urile* BobBeginTest.
Prima întrebare din cadrul unui test este aleasă aleatoriu din baza de date, în mod similar cu modalitatea în care este manipulat și *itent-ul* BobRandomQuestion. Pentru a marca faptul că utilizatorul este în timpul unui test se adaugă la sesiune cheia test ce conține un obiect. Acesta la rândul lui are două elemente de tipul cheie valoare reprezentând numărul de întrebări rămase, respectiv numărul de întrebări la care a răspuns corect utilizatorul.
Singura diferență dintre *intent-ul* BobRandomQuestion și BobBeginTest este componența sesiunii.
Răspunsul este transmis către dispozitivul care este integrat cu *Alexa* prin intermediul unui *callback*, care primește ca parametrii un obiect ce reprezintă sesiunea și un obiect ce conține răspunsul pe care *Alexa* îl va furniza utilizatorului împreuna cu alte informații. Pentru a nu duplica codul am ales să trimit funcției care face tratează *intent-ul* BobRandomQuestion, un callback fals care adaugă la sesiune elementele specifice unui test, iar apoi apelează *callback-ul* real, care transmite utilizatorului răspunsul(vezi figura 6.9)
Pentru răspunsul la întrebările din cadrul testului acestea se tratează asemănător cu modalitatea prezentată în secținea 6.4.1. Singurele defierențe sunt modificările care trebuie aduse elementelor din sesiune care sunt specifice unui test și faptul că la o întrebare se poate încerca să se răspundă doar o singură dată. Cea din urmă diferență se rezolvă astfel: atunci când un utilizator este în cadrul unui test după ce răspunde la o întrebare este apelată funcția ce tratează *intent-ul* BobRandomQuestion.

```
exports.handleTestBegin = function(session, callback) {
  var my_callback = function (sessionAttributes, speachOutput) {
    sessionAttributes["sessionAttributes"]["test"] = {}
    sessionAttributes["sessionAttributes"]["test"]["left"] = 6;
    sessionAttributes["sessionAttributes"]["test"]["correct"] = 0;
    callback(sessionAttributes, speachOutput);
  }
  var ask_answer = require('./ask_answer');
  ask_answer.handleRandomQuestion(session, my_callback);
}
```

Figura 5.9: Manipuleare *intent-ului* `BobBeginTest`

5.4.3 Alegerea întrebărilor pentru un test

După finalizarea unui test fiecare jucător va primi 4 sau 10 puncte în funcție de nivelul cunoștințelor de geografie pe care acesta în are. Vom considera că există două nivele: începător și maestru, începătorii primind 4 puncte la finalul jocului, iar maeștrii 10. Pentru a determina în mod imparțial din ce categorie face parte un jucător am utilizat un algoritm care adaptează întrebările din timpul unui test în mod automat (procedeu numit *Computer Adaptive Testing* sau CAT). Obiectivul de a clasifica cel mai bine nivelul de cunoștințe al unui jucător se va face pe baza răspunsurilor la întrebări, probabilitățile apriori ale întrebărilor, probabilitățile apriori ale clasificării populației în nivele de măiestrie. Pentru a determina întrebarea care clasifică cel mai bine jucătorul vom folosi urmatorele relații:

- $P(m_k)$ probabilitatea ca un jucător să aibă măiestria m_k, în cazul nostru m_k poate fi începător sau maestru;
- $P(z_i|m_k)$ probabilitatea ca răspunsul să fie z_i, care poate fi corect sau greșit, dată fiind măiestria m_k;
- fie Z vectorul de răspunsuri pe care un jucător le dă într-un test
- $P(m_k|Z) = \frac{P(Z|m_k)P(m_k)}{P(Z)} = \frac{P(Z|m_k)P(m_k)}{\sum_{n=1}^{K} P(Z|m_k)P(m_k)}$ reprezintă probabilitatea ca un jucător să aibă nivelul m_k dat fiind că a răspuns la întrebări cu Z, iar K este numărul de măiestrii care în cazul nostru este cu 2
- Presupunem că răspunsurile întrebărilor sunt independente ceea ce implică faptul că $P(Z|m_k) = \prod_{i=1}^{N} P(z_i|m_k)$, unde N reprezintă numărul de întrebări, iar m_k reprezintă un nivel de măiestrie.

- $P(z_i = v) = \sum_{n=1}^{K} P(z_i = v|m_j)P(m_j)$ probabilitatea ca răspunsul la întrebarea i să fie v care in cazul de față poate fi corect sau greșit.

- Probabilitatea ca răspunzând v la o întrebare aceasta i jucătorul să aibă măiestria m_i $P(m_i) = \frac{P(z_i=v|m_i)P(m_i)}{\sum_{j=1}^{K} P(z_i=v|m_j)P(m_j)}$

- Entropia așteptată după ce se răspunde la întrebarea i, $H(S_i) = P(z_i = 1)H(S_i|z_i = 1) + P(z_i = 0)H(S_i|z_i = 0)$.

Pentru a alege întrebarea cea mai potrivită se va selecta cea care are câștigul de informații maximal dintre cele 3 intrebări extrase aleator din fiecare categorie.

5.4.4 Publicarea scorului pe Twitter

Aplicația *Bob* poate primi de la utilizatori cererea de a posta scorul acestora pe *Twitter*. Aceste se va traduce de către componenta *Voice Service* în *intent-urile* `BobPost`.
Pentru a putea implementa acestă funcționalitate va trebui mai întâi configurată abilitatea astfel încât aceasta să suporte legătura cu contul de *Twitter* al utilizatorului. modalitatea prin care se realizează acesta va fi prezentată ulterior în acest capitol. Astfel dezvoltatorii vor putea accesa *accessToken-ul* utilizatorului prin intermediul căruia pot să acceseze, să modifice sau să adauge informații în contul utilizatorului în funcție de scopul aplicației lor. Atunci când un *intent* de tipul `BobPost` este primit de către aplicație va fi extras din baza de date punctajul pe care un utilizator îl are împreună cu locul acestuia. `ID-ul` unui utilizator se poate extrage din obiectul `JSON` pe care componenta *Voice Service* îl formează pentru tratarea cererilor clienților (vezi figura `6.12`). După ce am obținut datele despre utilizator prin intermediul pachetul de `Node.js`, *twitter* se va posta sorul și locul acestuia.
OAuth este un protocol standard utilizat în industrie pentru autentificare și partajarea de permisiuni dintre diferiți utilizatori ce au conturi pe severe diferite. Există două versiuni principale ale acestui protocol `1.0` și `2.0`. Pentru interconectarea de conturi *Amazon* suportă în mod oficial protocolul `2.0`. Unele servicii de exemplu *Facebook* sau *Google* suportă această versiune a protocolului, spre deosebire de *Twitter* care încă utilizează protocolul *OAuth `1.0`*.
Pentru a integra un *endpoint* care implementează protocolul *OAuth `2.0`* trebuie furnizat un `URL` de autorizare în *Alexa Skill Developer Portal* la secțiunea *Configuration*, împreuna cu alte informații dacă acestea sunt necesare. Pentru a putea vedea informațiile ce necesită a fi furnizate se va selecta *Yes* pentru intebarea *Do you allow users to create an account or link to an existing account with you?*. În figura `6.10` este prezentată o versiune simplificată a modului în care se

face autentficarea folosind versiunea 2.0 a protocolului *OAuth*, atunci când acesta este utilizat prin intermediul unui *endpoint*. Este de remarcat faptul ca pentru ca dezvoltatorii să utilizeze acest protocol nu este necesară utilizarea unui client de *OAuth*.

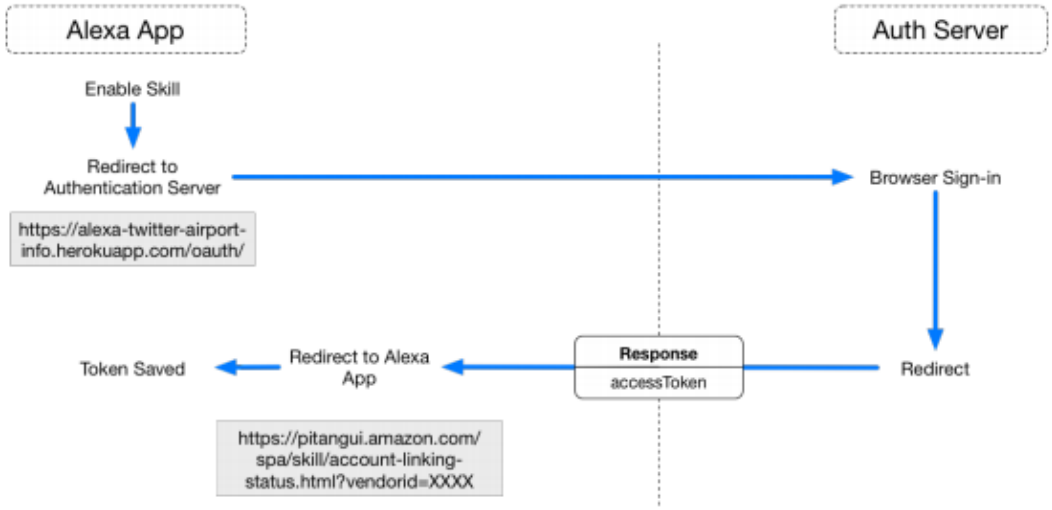

Figura 5.10: Integrarea unui abilități cu protocolul *OAuth 2.0*

Pentru integrarea *Twitter-ului*, care folosește protocolul *OAuth 1.0* este necesară folosirea unui client de *OAuth* care să facă integrarea să meargă corect. Clientul este un serviciu web care va trebui să primească URL-ul de redirectare, *consumer key* și *consumer secret*. URL-ul de redirectare va fi furnizat de către *Alexa* atunci când se face configurarea pentru legarea conturilor, iar *consumer key* și *consumer secret* vor fi obținute atunci când se va face configurarea aplicației în consola de developer furnizată de *Twitter*. Figura 6.11 oferă o imagine de ansamblu asupra drumului unei cereri de legare a contului de *Twitter* de abilitatea dezvoltată.

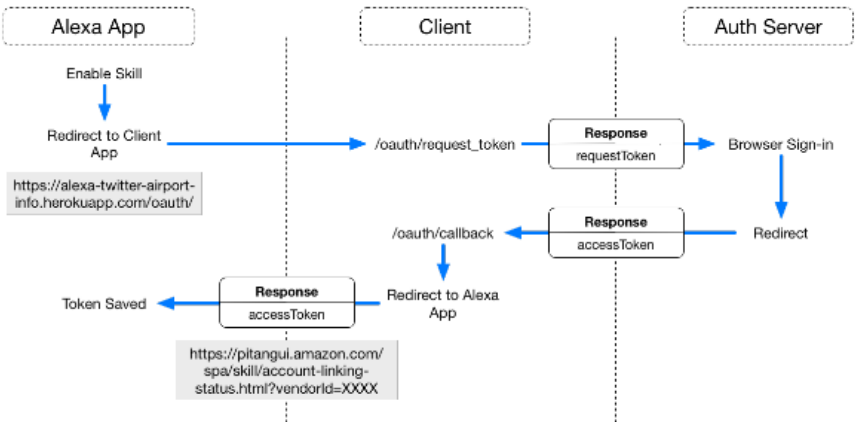

Figura 5.11: Integrarea unui abilități cu protocolul *OAuth 1.0*

Clientul de *OAuth* folosit în aplicația *Bob* este cel pus la dispoziție de către aplicația *Airport Info*. Această aplicație intermediază redirectarea către pagina de *login* de la *Twitter* și extrage *accessToken-ul*, pe care apoi îl transmite mai departe către URL-ul de redirectare pus la dispoziție

de către abilitatea dezvoltată. Cu ajutorul *accessToken-ului* se va face autentificarea atunci când aplicația dezvoltată de noi va face operații cu utilizând `SDK-ul` pentru *Twitter*. El va putea fi găsit în obiectul `JSON` pe care componenta *Voice Service* îl transmite către *Skill Service* (vezi figura `6.12`).

Figura 5.12: *AccessToken* în obiectul ce este trasmis către *Skill Service*

5.5 Extragerea datelor pentru întrebări

Pentru a extrage datele necesare formulării întrebărilor au fost folosite cele mai cunoscute baze de cunoștințe, *DBpedia* și *Wikidata*. Accesul la cele două baze de cunoștințe a fost făcut prin intermediul unei aplicații separate care are scopul de a extrage datele și de a le filtra, astfel încât în baza de date să existe doar informațiile pe care abilitatea dezvoltată se va folosi.

5.5.1 De ce aceasta este o componentă separată?

DBpedia sau *Wikidata* pot fi accesate prin intermediul unui *endpoint* `SPRQL`. Inițial componenta *Skill Service* manipula cererile făcute către cele două baze de cunoștiințe. Deși la prima vedere comunicarea acestea se face rapid, o dată cu creșterea numărului de instanțe extrase și cu crearea unor operații asemănătoare cu cele de *join* din bazele de date relaționale latența a crescut considerabil, ajungând la mai mult de o secundă. Efectul creșterii latenței se putea observa cu ușurință de către utilizatorul abilității *Bob*. Interacțiunea dintre *Alexa* și utilizator nu mai provoca aceeași senzație a unei conversații naturale.

Pentru a micșora latența a fost decizia ca informațiile ce sunt necesare construcției unei întrebări vor fi adăugate în baza de date. Am ales *DynamoDB*, deoarece este o bază de date `NoSQL`. Acest tip de bază de date pare a fi alegerea majorității dezvoltatorilor atunci când se confruntă cu

aplicații de tipul IoT. Pentru a micșora și mai mult latența indusă de rețea am ales baza de date în aceeași regiune în care am hostat și componenta *Skill Service*, adică funcția *Lambda*.

5.5.2 Extragerea datelor

Datele pentru întrebări se extrag în două moduri. Pentru extragerea de date necesare întrebărilor se va utiliza o combinație dintre *Wikidata* și *Dbpedia*, procedându-se astfel:

1. se face o interogare folosind *endpoint-ul* SPARQL pus la dispoziție de Wikidata. Din această interogare se vor extrage date despre instanțele dorite

2. dacă datele extrase în pasul 1 nu sunt suficiente, atunci se va face o interogare la *endpoint-ul* SPARQL pus la dispoziție de DBpedia.

3. extragerea posibilelor variante de răspuns astfel încât acestea să fie relevante, de exemplu răspunsul la întrebare să fie lacul *Marea Carspică*, situat în *Asia*, iar toate celelalte variante de răspuns să fie lacuri din *America de Nord*.

S-a impus utilizarea ambelor baze de cunoștințe deoarece după cum precizează și un studiul comparativ realizat între *DBpedia*, *Freebase*,*OpenCyc*, *Wikidata* și YAGO (*Färber*, 2015), *DBpedia* tinde să nu aibă datele structurate inconsistent.

```
SELECT DISTINCT ?item WHERE {
    {
        SELECT (MAX(?population) AS ?population) ?country WHERE {
            ?item wdt:P31/wdt:P279* wd:Q515 .
            ?item wdt:P1082 ?population .
            ?item wdt:P17 ?country .
        }
        GROUP BY ?country
        ORDER BY DESC(?population)
    }
    ?item wdt:P31/wdt:P279* wd:Q515 .
    ?item wdt:P1082 ?population .
    ?item wdt:P17 ?country .
    ?item wdt:P625 ?loc
} order by desc(?population) LIMIT 100
```

Figura 5.13: Exemplu de interogare SPARQL

În continuare vor fi detaliați cei trei pași precizați mai sus pentru extragerea datelor în cazul orașelor. În **pasul 1** al extragerii informațiilor, o interogare relevantă o constituie cea din figura

6.13, care furnizează din fiecare țară orașul cu populația maximă ordonate descrescător după numărul de locuitori.

În **pasul 2** va fi apelat *endpoint-ul* SPARQL al bazei de cunoștințe *DBpedia* pentru a afla informații despre fiecare oraș în parte. Acest pas poate produce uneori probleme, deoarece *Wikidata* și *Dbpedia* au metode diferite prin care identifică instanțele (vezi secțiunile 5.8.1 și 5.8.2). În cazul *DBpedia* este utilizat numele, însă uneori pot apărea diferențe între numele pe care îl folosește drept ID și ceea ce furnizează *Wikidata* ca fiind numele instanței. De exemplu, la extragerea datelor despre țări, *Wikidata* furnizează numele *People's Republic of China*, care este denumirea oficială a țării, dar *DBpedia* folosește numele China. Pentru a diminua efectele produse de această neconcordanță, se pot identifica alte predicate care specifică denumirea sub alte forme, precum dbo:longName pentru exemplul prezentat anterior. Apoi se va construi o interogare care va furniza instanta care are numele dat de către *Wikidata* egal cu unul dintre obiectele furnizate de către *DBpedia* ca fiind numele instanței.

În **pasul 3** se vor extrage câteva posibile variante de răspuns astfel să fie îngreunat jucătorului să identifice răspunsul corect. În cazul variantelor de răspuns pentru întrebările despre orașe au fost selectate cele din aceeași țară cu cel ce constituie răspunsul corect, dar și câteva orașe fără nicio regulă, pentru ne asigura că avem cel puțin patru variante de răspuns la dispoziție. Orașele din aceeași țara cu răspunsul corect vor fi selectate după următorul criteriu: ca ele să fie printre cele ce au populație mare, adică sa fie în top 12 al numărului de locuitori. În acest pas una dintre greutățile întâmpinate a fost faptul că nu toate orașele au ca obiect pentru predicatul P31 (*instance of* tradus drept *instanta a*), Q515, ID pentru oraș (en. *city*). După analiza obiectelor pentru predicatul P31 am observat că instanțele ce sunt orașe au legătură cu instanța Q15284 (ID-ul pentru municipalitate). Ceea ce semnifică faptul că:

1. există un obiect pentru care valoarea predicatului P31 are obiectul predicatului P279 egală cu Q15284 (vezi figura 6.14), ceea ce ar putea fi tradus în limbaj natural astfel: subiectul selectat este o instanță a unei subclase pentru conceptul de municipalitate;

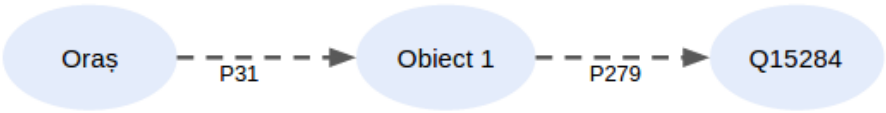

Figura 5.14: Interfața de configurare a declanșatorului pentru o lambda funcție

2. subiectul selectat este o instanță a unei subclase care este la rândul ei subclasă pentru conceptul de municipalitate. Varianta formală a acesteia este prezentată în figura 6.15;

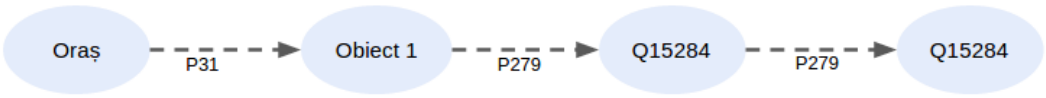

Figura 5.15: Interfața de configurare a declanșatorului pentru o lambda funcție

În figura 6.16 este un exemplu de interogare care va furniza toate variantele de răspuns pentru *Paris*.

```
SELECT DISTINCT ?item {
{
  SELECT DISTINCT ?item WHERE {
    ?item wdt:P17 wd:Q90 .
    ?item wdt:P31 ?itemInstance.
    ?itemInstance wdt:P279 wd:Q15284 .
    ?item wdt:P1082 ?population
  } ORDER BY desc(?population) LIMIT 6
}
UNION {
  SELECT DISTINCT ?item WHERE {
    ?item wdt:P17 wd:Q90 .
    ?item wdt:P31 ?itemInstance.
    ?itemInstance wdt:P279 ?subclass .
    ?subclass wdt:P279 wd:Q15284 .
    ?item wdt:P1082 ?population
  }   ORDER BY desc(?population) LIMIT 6
}
UNION {
  SELECT DISTINCT ?item WHERE {
    ?item wdt:P17 wd:Q90 .
    ?item wdt:P31 ?itemInstance.
    ?itemInstance wdt:P279 ?subclass .
    ?subclass wdt:P279 ?subsubclass .
    ?subsubclass wdt:P279 wd:Q15284 .
    ?item wdt:P1082 ?population
  }   ORDER BY desc(?population) LIMIT 6
}
UNION {
  SELECT DISTINCT ?item WHERE {?item wdt:P31 wd:Q515} LIMIT 4
}
}LIMIT 6
```

Figura 5.16: Exemplu de interogare SPARQL pentru aflarea variantelor de răspuns

5.6 Concluzii

În această secțiune a fost prezentată arhitectura aplicației, dar și elemente care țin crearea mediului de lucru pentru ca abilitatea să funcționeze. Prezentarea se concentrează asupra serviciilor utilizate, modului de utilizare, dar totodată va avea în vedere și motivarea alegerilor arhitecturale făcute .

6 Concluzii

Scopul acestui proiect este de a utiliza interfața vocală *Alexa* pentru a interacționa cu un joc de cultură generală. Acest tip de interacțiune este benefic deoarece permite utilizatorilor cu anumite dizabilități să poată interacționa cu aplicația. Expunerea pe care aplicația o are va crește o data cu integrarea interfeței vocale *Alexa* în cât mai multe dispozitive. Adoptarea acesteia pe o scară din ce în ce mai largă se poate observa din din faptul că unele companii ce produc autovehicule sau electrocasnice au integrat-o în aplicațiile lor. Pentru activitățile pe care le fac utilizatorii în jurul produselor companiilor de tipul menționat anterior *Bob* poate fi o modalitate de a produce divertisment atunci când așteptăm în trafic sau atunci când gătim.

Pentru a îmbunătăți aplicația se pot adăuga noi categorii de întrebări, se pot face statistici reale legate de probabilitățile ca răspunsurile utilizatorilor la anumite întrebări să fie corecte sau greșite în funcție de nivelul acestora, se pot adăuga mai multe nivele de dificultate sau se pot adăuga mai multe rețele sociale pe care să se poată posta scorul utilizatorului.

Bibliografie

[1] When You See These 10 Awesome Amazon Echo Alternatives You'll Be Sold. https://beebom.com/cool-amazon-echo-alternatives/. Accessed: 2017-05-1.

[2] 10 Cool Amazon Echo Alternatives You Can Use. http://smartwatches.org/learn/amazon-echo-alternatives/. Accessed: 2017-05-1.

[3] Amazon's Echo is building a coffin that's custom-made for Google. http://www.businessinsider.com/amazon-echo-success-could-spell-big-trouble-for-google-2017-1/. Accessed: 2017-05-1.

[4] THE BIG NERD RANCH GUIDE. Developing alexa skills. 2016.

[5] Amazon Alexa and the Alexa Skill Kit. https://www.slideshare.net/AmazonWebServices/please-meet-amazon-alexa-and-the-alexa-skills-kit/. Accessed: 2017-05-4.

[6] Amazon Alexa and the Alexa Skill Kit. https://developer.amazon.com/blogs/post/Tx27NAUCY0KQ34D/Rapidly-Create-Your-Alexa-Skill-Backend-with-AWS-CloudFormation/. Accessed: 2017-03-1.

[7] NoSQL (Not Only SQL database). http://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL. Accessed: 2017-05-10.

[8] Amazon dynamodb product details. https://aws.amazon.com/dynamodb/details/. Accessed: 2017-05-10.

[9] Ontology. http://tomgruber.org/writing/ontology-definition-2007.htm. Accessed: 2017-05-14.

[10] Guarino Nicola, Oberle Daniel, and Staab Steffen. What Is an Ontology? 2009.

[11] Färber Michael, Ell Basil, Menne Carsten, and Rettinge. Achim. A Comparative Survey of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. In *Semantic Web 1*, 2015.

[12] Heim Philipp, Hellmann Sebastian, Lehmann Jens, Lohmann Steffen, and Stegemann Timo. RelFinder: Revealing Relationships in RDF Knowledge Bases. 2009.

[13] Auer S., Bizer C., Kobilarov G., Lehmann J., Cyganiak R., and Ives Z. DBpedia: A Nucleus for a Web of Open Data. In *Aberer K. et al. (eds) The Semantic Web. Lecture Notes in Computer Science, vol 4825. Springer, Berlin, Heidelberg*, 207.

[14] The dbpedia data provision architecture. http://wiki.dbpedia.org/about/architecture. Accessed: 2017-05-25.

[15] Product update: In voice, every millisecond matters. http://voicelabs.co/2016/11/01/product-update-in-voice-every-millisecond-matters/. Accessed: 2017-05-30.

[16] The average mobile game day. http://www.vertoanalytics.com/average-mobile-game-day/. Accessed: 2017-06-20.