

ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e Scienze Informatiche

PROGETTAZIONE E SVILUPPO DI UN SISTEMA
DOMOTICO INTEGRATO CON ASSISTENTE
VOCALE A SUPPORTO DI SERVIZI
SOCIO-ASSISTENZIALI BASATI SU
GAMIFICATION

Elaborato in
SISTEMI EMBEDDED E INTERNET OF THINGS

Relatore
Prof. ALESSANDRO RICCI

Presentato da
JAHRIM GABRIELE CESARIO

Correlatori
Ing. MASSIMILIANO MALAVASI
Dott.ssa LAURA BUGO

Anno Accademico 2020 – 2021

Indice

Introduzione	v
1 Definizione del problema	1
1.1 Scopo del progetto	1
1.2 Contesto	2
1.2.1 La domotica	2
1.2.2 Gli assistenti vocali	3
1.2.3 Gamification e giochi cognitivi	5
1.2.4 Le funzioni cognitive del cervello	5
2 Studio del problema	9
2.1 Analisi dei requisiti	9
2.1.1 Must Have	9
2.1.2 Should Have	11
2.1.3 Could Have	11
2.2 Vincoli del Sistema	12
3 Progettazione del sistema	13
3.1 Architettura del Sistema	13
3.1.1 Architettura di massima	13
3.1.2 Progetto del sistema	14
3.2 Tecnologie utilizzate	15
3.2.1 OpenHAB come sistema di controllo domotico	16
3.2.2 Alexa come assistente vocale	18
3.2.3 Comunicazione tra Alexa e OpenHAB	21
3.3 Limitazioni del progetto	22
4 Progettazione delle skill	23
4.1 Number List Game	23
4.1.1 Analisi del modello di dialogo	24
4.1.2 Progettazione del modello di dialogo	25
4.2 Category Game	26

4.2.1	Analisi del modello di dialogo	27
4.2.2	Progettazione del modello di dialogo	30
4.3	Labyrinth Game	31
4.3.1	Analisi del modello di dialogo	32
4.3.2	Progettazione del modello di dialogo	34
5	Implementazione del prototipo	37
5.1	Implementazione del sistema domotico	37
5.1.1	Struttura dei file di configurazione	37
5.1.2	Accesso al sistema domotico	38
5.1.3	Esempio d'integrazione con OpenHAB	39
5.2	Implementazione delle skill	41
5.2.1	Struttura dell'applicazione	41
5.2.2	Funzionamento dell'applicazione	42
5.2.3	Esempio d'implementazione di una skill	44
6	Validazione del sistema	47
6.1	Metodi di validazione	47
6.2	Risultati ottenuti	48
6.2.1	Il sistema domotico	48
6.2.2	Number List Game	49
6.2.3	Category Game	49
6.2.4	Labyrinth Game	49
Conclusioni		51
Ringraziamenti		53

Introduzione

L'uomo ha da sempre cercato di sopperire ai propri limiti, inventando e costruendo strumenti che lo aiutassero nel soddisfare i propri bisogni di libertà. Recentemente, uno di questi strumenti è stato la domotica, che, tra le altre cose, si propone di facilitare la vita in ambito domestico, permettendo di velocizzare o addirittura automatizzare, quelle piccole azioni quotidiane che possono rubare via del tempo, oggi più che mai prezioso. Non è tutto però. Infatti è proprio attraverso la domotica che molte persone ora vedono davanti a sé una prospettiva d'indipendenza, nella quale quelle azioni quotidiane, prima per loro impraticabili, diventano finalmente accessibili.

Con il passare del tempo, l'accessibilità ha acquisito sempre maggiore importanza. Più precisamente, per accessibilità s'intende la capacità di un servizio di essere fruibile dal maggior numero di persone possibile. L'accessibilità è quindi cruciale per includere le persone, ma anche per assicurare il nostro futuro. Infatti, man mano che si invecchia, sempre più strumenti diventano sempre meno accessibili.

Ormai la domotica è un concetto ben consolidato e sta cominciando a diffondersi su larga scala, permeando nelle case delle persone, a cui sta diventando sempre più familiare. Per questo motivo, si presta come ottimo punto di appoggio per lo sviluppo di servizi integrativi, che possano estendere le funzionalità del proprio sistema domotico.

In questo contesto, in collaborazione con la AUSL [1] e il CRA [2] di Bologna, sotto la guida del coordinatore del CRA *Ing. Massimiliano Malavasi*, la neuropsicologa *Dott.ssa Arianna Gherardini* e la sviluppatrice software *Dott.ssa Laura Bugo*, si è pensato di progettare un servizio integrativo a un impianto domotico, che consenta di mantenere in allenamento le proprie capacità cognitive, oltre a fornire assistenza nello svolgimento delle attività di vita quotidiana. Questa suite è stata pensata soprattutto come supporto per gli anziani o per le persone con difficoltà cognitive, ma potrà anche svolgere una funzione di agevolazione e intrattenimento per i giovani. Più in particolare, si vorrebbe applicare la suite a un contesto socio-sanitario per assistere persone con disabilità e/o con difficoltà cognitive e monitorarne i progressi durante il loro percorso psico-educativo, sotto la supervisione di personale qualificato. La suite prevederà anche un'assistente vocale che avrà sia la funzione di aumentare l'accessibilità dell'impianto domotico, sia di permettere l'accesso al servizio integrativo.

Questa tesi illustrerà come è stato sviluppato questo progetto dalla fase di analisi del problema fino ai primi risultati prodotti. Sarà quindi descritta una panoramica generale

dei concetti necessari a comprendere le tecnologie coinvolte e l'obiettivo preposto. Successivamente, sarà analizzato il problema, delineando quelli che saranno i requisiti del sistema. Sarà poi proposta una possibile soluzione concettuale, non vincolata a dei servizi specifici, che descriverà la natura del problema. A questa seguirà una soluzione concreta, basata su servizi oggi esistenti, di cui se ne motiverà l'adeguatezza, illustrandone vantaggi e svantaggi. Infine, sarà presentata l'implementazione di un prototipo e commentati i risultati dei test eseguiti durante la fase di validazione.

Capitolo 1

Definizione del problema

In questo capitolo, sarà introdotto lo scopo del progetto e saranno illustrati i concetti fondamentali che consentiranno al lettore di comprendere a pieno l'obiettivo del progetto e le tecnologie utilizzate per realizzarlo.

1.1 Scopo del progetto

Questo progetto nasce con lo scopo di assistere persone anziane, fragili o con disabilità, permettendo loro di ottenere una certa indipendenza nello svolgere le azioni quotidiane in ambito domestico. Inoltre, si propone di fornire un servizio integrativo per poter mantenere in allenamento le proprie capacità cognitive. Questo servizio potrà essere utilizzato anche in un contesto socio-sanitario, dagli operatori competenti, per realizzare un percorso educativo per pazienti con difficoltà cognitive e monitorarne i progressi, così da tenere traccia dei miglioramenti ottenuti.

L'idea è quella d'integrare un'assistente vocale a un sistema domotico, creando una suite che comprenderà i servizi di entrambi. Nello specifico, il *sistema domotico* si occuperà di agevolare la vita domestica degli utenti, rendendo più accessibili gli strumenti di uso quotidiano, come porte e finestre. L'*assistente vocale* fornirà invece un supporto al sistema domotico, permettendo di controllarlo tramite comandi vocali, ma consentirà anche di accedere agli esercizi cognitivi di cui è composto il servizio integrativo, che si baserà perciò su interazioni prettamente vocali. Per gli esercizi cognitivi, si è pensato di progettare dei giochi che richiedano l'impiego di alcune funzioni cognitive del cervello, seguendo i principi di *gamification*. In questo modo, si rende il servizio più accattivante, stimolando l'utente all'utilizzo dello stesso.

1.2 Contesto

Di seguito sarà contestualizzato il progetto, approfondendo i concetti che sono stati utilizzati per descriverlo.

In particolare, si introdurrà la domotica, distinguendo il concetto d'impianto domotico da quello di *smart home* e descrivendo alcuni degli approcci usati per controllare un sistema domotico. Seguirà poi, una breve storia sullo sviluppo degli assistenti vocali e delle funzioni che ricoprono oggigiorno. Infine, saranno illustrate le capacità cognitive del cervello, in modo da poter comprendere come il servizio che si vuole progettare possa esercitarle, dunque mantenendole in allenamento.

1.2.1 La domotica

Il termine *domotica* [3] deriva dal latino *domus*, ovvero *casa*, e dal suffisso greco -*ticos*, che indica le discipline che si applicano a un determinato concetto. Nello specifico, il termine indica la scienza che studia e applica delle tecnologie adatte a migliorare la qualità della vita in ambiente domestico.

L'applicazione della domotica si manifesta spesso sotto forma di un impianto domotico. Un impianto domotico è l'insieme composto da alcuni dispositivi elettronici e dell'infrastruttura che questi utilizzano per comunicare tra di loro (un esempio, è l'impianto *KNX* [4]). Un dispositivo domotico è infatti uno strumento non più solo passivo, ovvero che subisce solamente le sollecitazioni dall'ambiente esterno, ma ha anche una parte attiva, che processa tali sollecitazioni e le trasmette come eventi identificabili e recepibili da altri strumenti. In questo modo, si crea un organismo di strumenti che producono eventi e che possono reagire ad alcuni di essi, modificando lo stato in cui si trovano.

I costi di un impianto domotico sono piuttosto ingenti e dipendono in parte dai dispositivi da acquistare, ma soprattutto dalla progettazione e installazione dell'infrastruttura su cui comunicheranno. Per questo, oggi la domotica si manifesta molto più comunemente sotto forma di *smart home* (Figura 1.1). Una *smart home* è simile a un impianto domotico, ma non necessita dell'installazione di un'infrastruttura ad hoc, infatti ne utilizza una preesistente: Internet. I dispositivi che ricevono e trasmettono eventi attraverso una connessione a Internet prendono il nome di dispositivi smart, e sono alla base dell'***Internet Of Things***.

Normalmente, il controllo di questi dispositivi è affidato ad applicazioni particolari, magari legate a uno specifico produttore. Queste prediligono un setup facile e veloce, consentendo a chiunque di domotizzare la propria casa senza troppe difficoltà e di controllarla generalmente tramite cellulare (ad esempio, *Philips Hue* [5]). Tuttavia, questi sistemi sono spesso vincolati ai dispositivi di uno specifico produttore.

Quando alla *smart home* si integrano dispositivi anche molto diversi tra loro o ad dirittura incompatibili, diventa invece necessario un sistema di controllo centralizzato. Solitamente, questi sistemi richiedono un setup complesso e delle conoscenze abbastanza

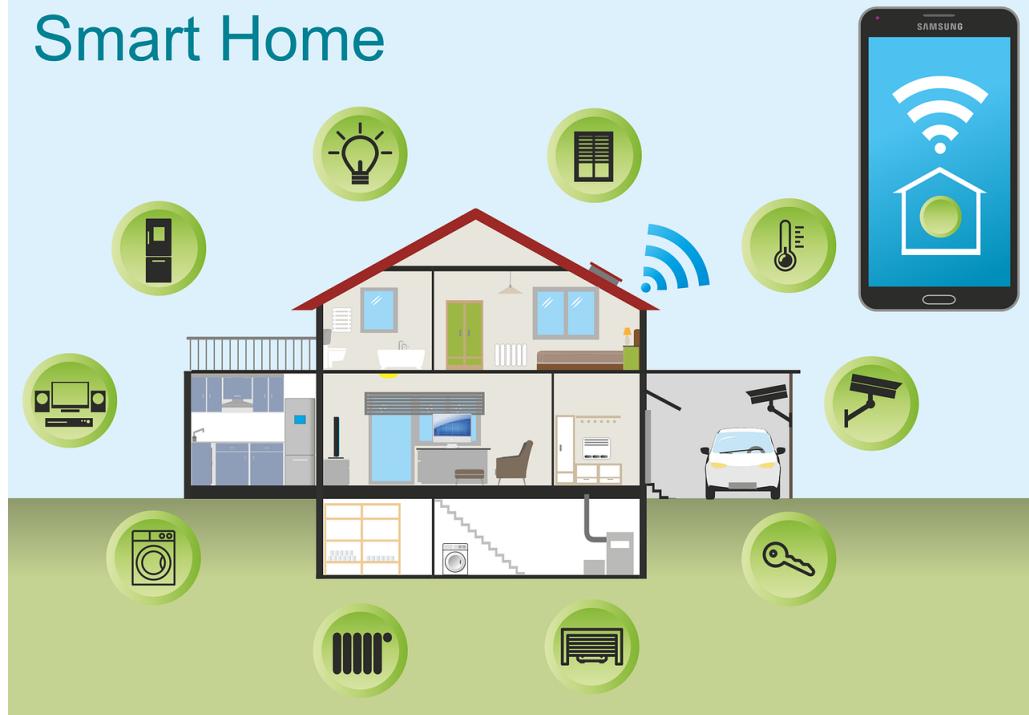


Figura 1.1: L’idea di *smart home*: tutti i dispositivi domestici sono connessi tra loro tramite Internet e possono essere controllati da remoto.

approfondite in materia, ma permettono d’intermediare le interazioni tra i dispositivi, di controllarli molto più liberamente e di creare delle regole di automazione più complesse (ad esempio, *OpenHAB* [6]).

Un altro dei vantaggi di una *smart home* è la possibile integrazione con un’assistente vocale: una tecnologia che consente d’interagire con il proprio sistema domotico attraverso comandi vocali e di dare una voce ai propri strumenti.

1.2.2 Gli assistenti vocali

Gli *assistenti vocali* sono una tecnologia che si è diffusa recentemente, il cui sviluppo risale tuttavia già al 1952, con il primo dispositivo in grado di riconoscere delle cifre decimali in linguaggio parlato: *Audrey* di *Bell Labs*. Dieci anni più tardi, fu *IBM* a proporre *Shoebox*, il primo calcolatore in grado di riconoscere operazioni matematiche in linguaggio parlato. Così, dagli anni ’70 lo studio degli assistenti vocali iniziò a diffondersi, fino a portare i risultati odierni. [7]

Oggi un’assistente vocale è uno strumento che permette d’interagire con dei servizi attraverso dei comandi vocali. Nello specifico, è esso stesso un servizio, basato sull’intelligenza artificiale e il *machine learning*, capace di tradurre il linguaggio parlato in dati

comprensibili a una macchina (*STT: Speech To Text*) e viceversa (*TTS: Text To Speech*), così proponendosi come ottimo intermediario nelle interazioni tra le persone e le macchine.

Normalmente, si interagisce con un'assistente vocale tramite cellulare, computer o tramite un apposito dispositivo del produttore specifico. Queste però sono solo interfacce, infatti si occupano solo di registrare le richieste dell'utente, per poi inviare le registrazioni al servizio vero e proprio per essere interpretate. Tale servizio è solitamente installato su un *cloud*, ovvero su un insieme di cluster di macchine in remoto.

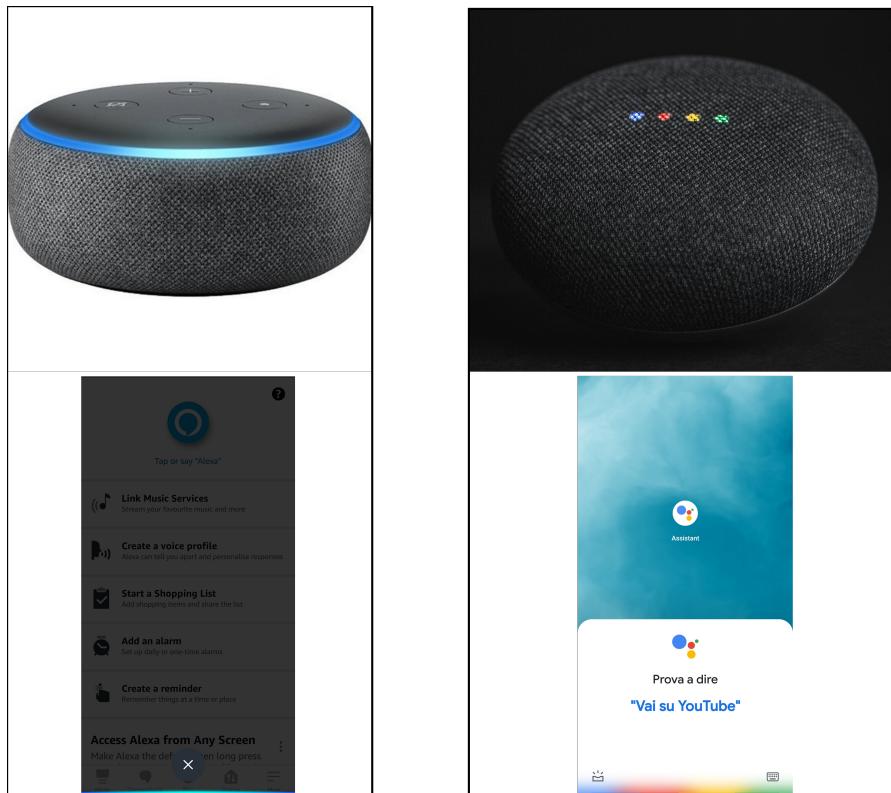


Figura 1.2: Alcuni degli assistenti vocali più conosciuti insieme alla loro applicazione per cellulare: a sinistra, *Alexa* di Amazon; a destra, *Google Assistant* di Google.

Gli assistenti vocali possono essere utilizzati per agevolare l'accesso a diversi servizi, ma spesso svolgono anche una funzione d'intrattenimento. Tra le diverse funzionalità che offrono, esistono infatti dei giochi, basati interamente su interazioni vocali. Prendendo ispirazione da ciò e seguendo i principi di *gamification*, si è deciso di sviluppare alcuni giochi per un'assistente vocale, che avessero tuttavia anche un fine educativo, specificamente il fine di esercitare alcune funzioni cognitive del cervello.

1.2.3 Gamification e giochi cognitivi

Con il termine *gamification* s'intende "l'utilizzo di meccanismi tipici del gioco e, in particolare, del videogioco (punti, livelli, premi, beni virtuali, classifiche), per rendere gli utenti o i potenziali clienti partecipi delle attività di un sito e interessarli ai servizi offerti" [8]. Nel dettaglio, i principi di gamification sono applicati a un servizio quando si vuole:

- *Aumentare la soddisfazione dell'utente*: la continua documentazione dei risultati del giocatore consente di visualizzare i propri progressi, facilitando la derivazione di obiettivi personali realizzabili e offrendo un feedback immediato sul comportamento dell'utente;
- *Trasmettere ottimismo all'utilizzatore*: la *gamification* promuove l'autodeterminazione, oltre all'esperienza di un senso di realizzazione, o più specificamente la speranza di raggiungere gli obiettivi preposti;
- *Facilitare l'interazione sociale tra gli utilizzatori*: giocare significa spesso essere coinvolti in una comunità di giocatori, dando quindi spazio a scambi sociali e/o competizioni;
- *Motivare l'utente a utilizzare il servizio*: ciò assume particolare importanza se l'utente può realmente trarre vantaggio dal servizio (ad esempio, se il servizio ha uno scopo formativo) [9].

In questo elaborato, la *gamification* viene applicata a un servizio che permetterà l'accesso ad alcuni esercizi cognitivi. Ciò implica la progettazione di *giochi cognitivi*, ovvero di giochi che stimolano le funzioni cognitive del cervello.

1.2.4 Le funzioni cognitive del cervello

Il cervello è uno degli organi più importanti e complessi del nostro corpo. La sua complessità deriva dal fatto che ha la funzione di coordinare gran parte dei meccanismi del nostro organismo, insieme alla percezione, il movimento, il linguaggio, le emozioni e il pensiero.

Tra le funzioni del cervello vi sono quelle che prendono il nome di funzioni cognitive. Le funzioni cognitive sono processi mentali che consentono d'interagire con il mondo esterno e processare gli stimoli ricevuti producendo pensieri complessi. Esse si distinguono in alcune macro-categorie:

- **Attenzione**: è un insieme di processi complessi che permettono di selezionare alcuni stimoli specifici tra tutti quelli provenienti dall'ambiente circostante. Esistono diversi livelli di attenzione, organizzati su una struttura piramidale, per cui l'attivazione di un livello richiede quella di tutti i livelli sottostanti. I livelli di attenzione sono, dalla base alla cima della piramide:

- o **Attenzione sostenuta:** è la capacità di mantenere l'attenzione nel tempo, reagendo agli stimoli provenienti dall'ambiente circostante, anche detta concentrazione o vigilanza;
 - o **Attenzione selettiva:** è la capacità di rimanere concentrati su uno specifico compito, inibendo gli altri stimoli esterni (come il rumore di sottofondo);
 - o **Attenzione alternata:** è la capacità di eseguire più compiti, concentrandosi su un compito per volta in modo alternato;
 - o **Attenzione divisa:** è la capacità di rimanere concentrati su più compiti contemporaneamente.
- **Memoria:** è il processo che permette la codifica degli stimoli esterni, in modo da essere memorizzati e quindi recuperabili in un secondo momento. La memoria è applicata ai soli eventi a cui si è prestata attenzione. In particolare, si distinguono:
 - o **Memoria a lungo termine:** è la memoria riservata a enormi quantità d'informazioni per un periodo di tempo molto lungo;
 - o **Memoria a breve termine:** è la memoria riservata a piccole quantità d'informazioni per una durata di al massimo 30 secondi circa; queste informazioni possono essere rielaborate per passare alla memoria a lungo termine, altrimenti sono perse.
 - **Orientamento:** è la capacità di riconoscere e comprendere ciò che dovrebbe esserci familiare. Comprende sia l'orientamento spaziale, ma anche il riconoscimento dei volti e degli oggetti.
 - **Linguaggio:** è la capacità di esprimere o comprendere dei pensieri attraverso dei simboli, che spesso si manifestano come lingue. Alcune funzioni del linguaggio sono:
 - o **Semantica:** la capacità di distinguere parole che hanno significati, tali da essere accomunabili sotto a una stessa categoria;
 - o **Fonemica:** la capacità di distinguere parole che hanno un suono simile.
 - **Funzioni esecutive:** sono l'insieme di tutte le funzioni cognitive superiori, che includono il controllo della cognizione e la regolazione dei pensieri, come ad esempio l'astrazione, la deduzione e la pianificazione.

Con l'invecchiamento, si assiste a un decremento delle prestazioni cognitive, soprattutto per le abilità cognitive soggette a disuso o poco stimolate. Questo può prevedere:

- Un *rallentamento dei tempi di reazione*, prolungando il tempo impiegato a svolgere determinati compiti;

- Un *deficit dell'attenzione*, che è in realtà la causa di molti dei problemi di memoria accusati dagli anziani;
- Un *deficit delle funzioni esecutive*, che può ad esempio rendere più difficile monitorare processi complessi o inibire certi comportamenti inadeguati;
- Un *deficit della memoria*, soprattutto *a breve termine*, mentre quella a lungo termine è meno affetta.

Per contrastare queste conseguenze, in generale, è sufficiente cercare di mantenere uno stile di vita sano, in termini di socialità, dieta, esercizio fisico e attività mentale. Un'altra ricetta anti-invecchiamento è invece l'impostazione di abitudini costruttive, perché per imparare a fare una cosa, come per ricordarsela, bisogna continuare a farla nel tempo. [10]

Capitolo 2

Studio del problema

In questo capitolo si analizzerà la natura del problema che si vuole risolvere, identificando i requisiti che il sistema dovrà soddisfare per adempiere al suo scopo.

Nello specifico, si formalizzeranno le funzioni che il sistema deve poter assolvere, classificandole in base alla loro priorità. Inoltre, si discuteranno quei vincoli del sistema che dipendono dalla sua stessa natura.

2.1 Analisi dei requisiti

Ora si descriveranno in un modo più ordinato e formale i requisiti del sistema da progettare.

Più in dettaglio, i requisiti saranno categorizzati attraverso la prioritizzazione MoSCoW [11]. Si distingueranno quindi i requisiti *must have*, ovvero quelli vitali, che devono essere soddisfatti dal sistema a breve termine, i requisiti *should have*, ovvero quelli importanti, che dovranno essere soddisfatti dal sistema a lungo termine, e i requisiti *could have*, ovvero quelli opzionali, che potranno essere integrati una volta ultimato il sistema. In questa tesi progettuale, lo scopo preposto è quello di soddisfare almeno i requisiti *must have*, quindi tutti gli schemi trattati, faranno riferimento esclusivamente a quei requisiti.

2.1.1 Must Have

Il sistema dovrà assolutamente soddisfare i seguenti requisiti funzionali:

- A. ***Controllo dei dispositivi domotici da sistema di controllo.*** Sarà necessario fornire all'utente un sistema di controllo, attraverso il quale si potrà interagire per monitorare e controllare i propri dispositivi domotici.
- B. ***Controllo dei dispositivi domotici attraverso comandi vocali.*** Sarà necessario fornire all'utente un'assistente vocale, integrato con il sistema di control-

lo domotico, attraverso il quale si potrà controllare i propri dispositivi domotici attraverso comandi vocali.

- C. **Ricezione delle notifiche dai dispositivi domotici.** Sarà necessario fornire all'utente la possibilità di ricevere delle notifiche sugli eventi rilevati dai dispositivi domotici, sia tramite il sistema di controllo domotico, sia tramite l'assistente vocale (quando più opportuno).
- D. **Accesso allo storico degli eventi rilevati dai dispositivi domotici.** Sarà necessario registrare tutti gli eventi rilevati dai dispositivi domotici fino a un certo periodo di tempo, così da fornire all'utente la possibilità di consultare lo storico degli eventi tramite il sistema di controllo domotico.
- E. **Accesso ad almeno tre giochi cognitivi distinti tramite l'assistente vocale.** Sarà necessario fornire all'utente un servizio comprendente almeno tre giochi cognitivi tra cui poter scegliere e a cui poter accedere attraverso l'assistente vocale. I giochi cognitivi dovranno distinguersi in base alle capacità cognitive di cui permetteranno l'allenamento.

Di seguito, il diagramma dei casi d'uso che schematizza questi specifici requisiti, che saranno quelli trattati successivamente nella tesi (Figura 2.1).

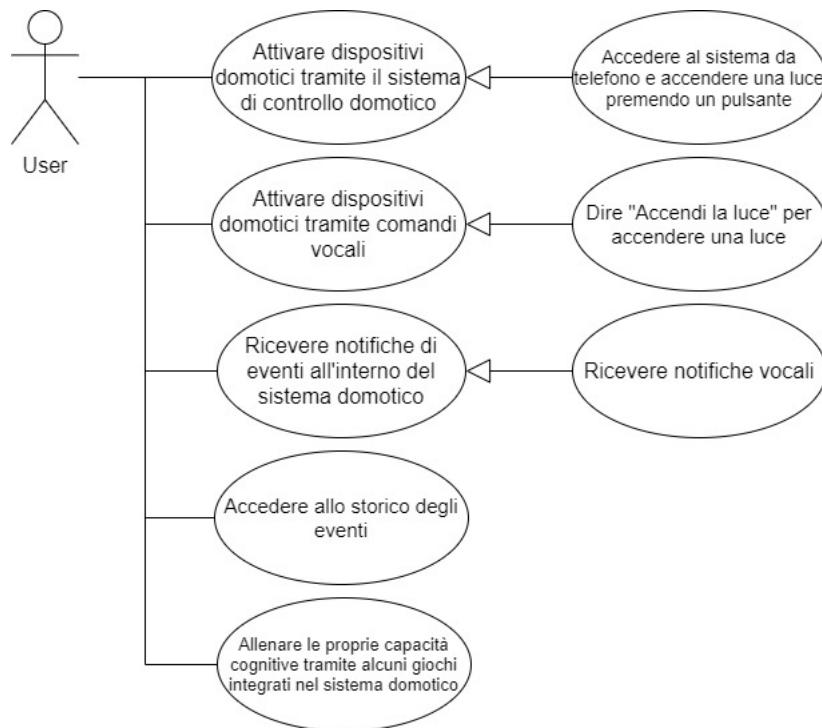


Figura 2.1: Il diagramma dei casi d'uso che riassume i requisiti funzionali del sistema.

2.1.2 Should Have

In un secondo momento, il sistema dovrà evolversi soddisfacendo i seguenti requisiti funzionali:

- F. **Gestione dell'autenticazione come paziente o personale socio-sanitario.** Sarà necessario gestire l'autenticazione, introducendo delle credenziali per essere riconosciuti come pazienti o come personale socio-sanitario.
- G. **Accesso allo storico dei risultati ottenuti sui diversi giochi cognitivi di un determinato paziente.** Sarà necessario registrare il risultato ottenuto dal paziente alla fine di ciascuna partita, per ogni gioco cognitivo, così da fornire un campione di dati analizzabile per monitorare e comprendere i miglioramenti conseguiti dal paziente. Questo storico sarà accessibile solo a personale socio-sanitario autorizzato.
- H. **Analisi dei risultati ottenuti sui diversi giochi cognitivi.** Sarà necessario implementare un servizio, possibilmente scalabile, che potrà accedere ai risultati ottenuti da uno specifico paziente, così da poter eseguire un'analisi più dettagliata dei risultati del paziente e produrre un report dei progressi, visualizzabile dal personale socio-sanitario autorizzato.
- I. **Gestione del percorso educativo di un paziente.** Sarà necessario permettere al personale socio-sanitario autorizzato d'impostare un percorso educativo per un dato paziente, che preveda una routine di giochi cognitivi, da proporre a tale paziente attraverso l'assistente vocale.
- J. **Aggiunta di nuovi giochi cognitivi.** Sarà necessario espandere il servizio che fornisce i giochi cognitivi, in modo da coprire più capacità cognitive, ma anche fornire più varietà per allenare la stessa capacità cognitiva.

2.1.3 Could Have

Di seguito, i requisiti considerati opzionali, ovvero che potrebbero essere implementati nei tempi successivi al completamento del sistema:

- K. **Accesso ai giochi cognitivi al di fuori del contesto socio-sanitario.** Sarà opzionale consentire l'accesso ai giochi cognitivi al di fuori del contesto socio-sanitario, così da poterli installare su altri sistemi domotici pubblici o privati.
- L. **Suggerimento casuale di un gioco cognitivo.** Sarà opzionale permettere all'assistente vocale di proporre all'utente uno dei giochi cognitivi installati, a discrezione dell'utente.

M. **Ulteriore aggiunta di nuovi giochi cognitivi.** Sarà opzionale espandere nuovamente il servizio che fornisce i giochi cognitivi, in modo da coprire più capacità cognitive, ma anche fornire più varietà per allenare la stessa capacità cognitiva.

2.2 Vincoli del Sistema

Ora si illustreranno alcuni dei vincoli del sistema, dovuti alla natura del sistema stesso.

Il primo vincolo è dovuto ai **costi**. Infatti configurare un sistema domotico, che sia un impianto o una *smart home* (o entrambi), ha dei costi non irrisoni per l'acquisto dei dispositivi, soprattutto se si tratta di dispositivi progettati per l'accessibilità.

Il secondo vincolo è dovuto all'uso di un assistente vocale. Gli assistenti vocali si basano su *cloud computing*, che ha enormi vantaggi in termini di fruibilità di risorse e scalabilità, tuttavia ha anche qualche svantaggio.

In primo luogo, gli assistenti vocali necessitano di una connessione a Internet, anzi più specificamente necessitano di una **connessione verso il cloud del produttore**. In altre parole, se, per un qualche motivo, un giorno non è possibile raggiungere il cloud del produttore, quel giorno l'assistente vocale non potrà funzionare (perciò non si potrà accedere ai giochi cognitivi, né utilizzarlo per controllare il proprio sistema domotico).

In secondo luogo, vi è la questione della **privacy** e della **sicurezza**. Infatti, gli assistenti vocali inviano le registrazioni vocali al cloud del produttore per essere processate e dunque interpretate come linguaggio parlato. Questo chiaramente richiede un alto livello di fiducia da parte dell'utilizzatore. Risulta quindi necessaria un'infrastruttura di rete sicura e un rispetto adeguato delle norme di privacy e sicurezza da parte dei produttori.

Fortunatamente, molti produttori hanno recentemente adottato un'infrastruttura che promuova l'*edge computing*, ovvero una forma di cloud computing, che sposta i centri di calcolo verso gli utilizzatori, riducendo il tempo per cui i loro dati si trovano in rete e migliorando anche la responsività e fruibilità dei servizi forniti, tra cui gli assistenti vocali.

Un terzo e ultimo vincolo è infine dovuto alla natura dei giochi, che si basano strettamente su interazioni vocali. Ciò rende **difficile realizzare giochi troppo complessi**, in cui la quantità d'informazioni da scambiare è molto elevata. In questi giochi, infatti, risulterebbe più efficace una comunicazione parallela, basata su più sensi (come vista e udito), piuttosto che una seriale, come un'interazione vocale, che si basa solo sull'udito. Per fortuna però, molti esercizi cognitivi si basano su interazioni semplici e si prestano piuttosto bene a essere implementati attraverso un'assistente vocale.

Capitolo 3

Progettazione del sistema

In questo capitolo si proporrà una possibile soluzione che soddisfa i requisiti *must have* del sistema descritto precedentemente.

Più in dettaglio, si analizzeranno le tecnologie scelte per implementare il sistema, focalizzandosi prima sul loro funzionamento individuale e successivamente su come possono interagire tra di loro. Inoltre, sarà giustificata la scelta di tali tecnologie nel contesto in cui è stato sviluppato il progetto. Infine, si illustreranno alcune possibili limitazioni del sistema, non più dipendenti dalla natura del problema stesso, ma derivanti dalle tecnologie specifiche utilizzate per implementarlo.

3.1 Architettura del Sistema

Ora si descriverà una possibile soluzione al problema introdotto nei capitoli precedenti, facendo riferimento ai requisiti *must have* del sistema.

In particolare, si progetterà prima un'architettura di massima che descriva quali sono i componenti e le interazioni che appartengono al sistema, dopodiché s'illustrerà l'architettura effettivamente implementata, vincolata alle specifiche tecnologie scelte.

3.1.1 Architettura di massima

Di seguito, viene proposta un'architettura generica di massima che possa soddisfare i requisiti *must have* del problema (Figura 3.1).

L'utente potrà interfacciarsi con il sistema attraverso il **sistema di controllo domotico** e l'**assistente vocale**. Entrambi gli permetteranno di controllare il proprio **ambiente domotico**, composto da tutti i dispositivi domotici del sistema. Il sistema di controllo domotico registrerà anche gli eventi ricevuti dall'ambiente domotico in un **database** e consentirà all'utente di accedere allo stato corrente e allo storico degli stati dei suoi dispositivi domotici. Inoltre, potrà notificare l'utente di certi eventi accaduti,

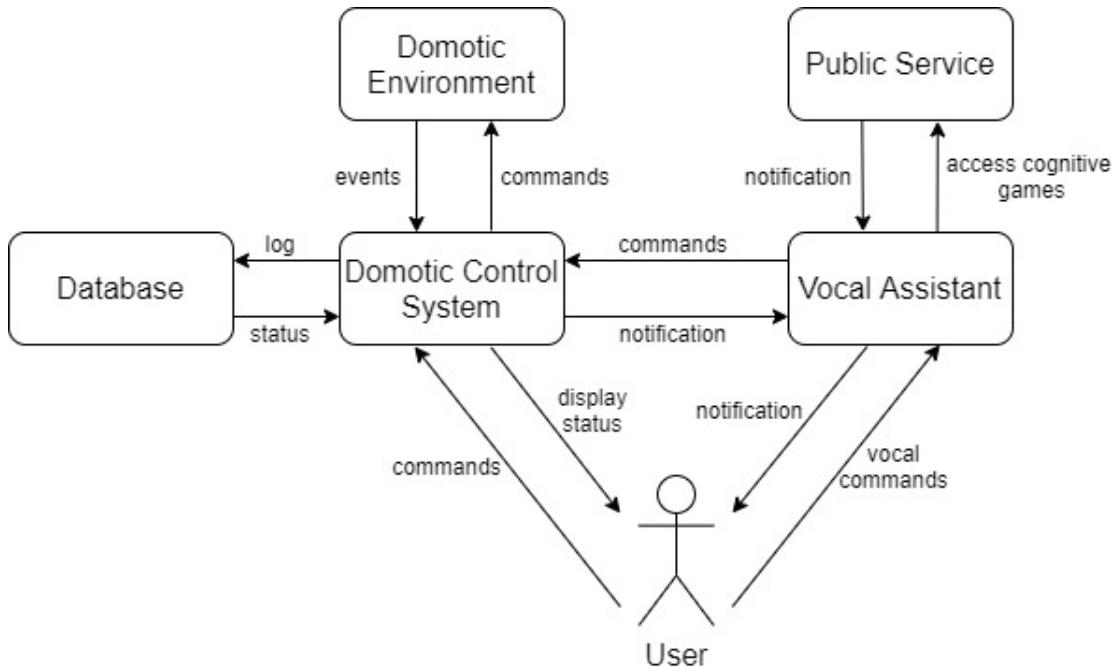


Figura 3.1: Lo schema delle interazioni in una generalizzazione del sistema.

attraverso messaggi vocali mediati dall'assistente vocale. L'assistente vocale, oltre a delegare i comandi vocali rivolti verso l'ambiente domotico al sistema di controllo domotico, permetterà l'accesso agli esercizi cognitivi. Questi saranno esposti attraverso un **servizio pubblico**, quindi accessibile dall'assistente vocale.

3.1.2 Progetto del sistema

Dopo aver descritto l'architettura di massima, ora si riporterà un'architettura vincolata alle tecnologie scelte per implementare il sistema (Figura 3.2).

Più in dettaglio, le tecnologie scelte sono:

- Per l'*ambiente domotico*, si è scelta una smart home composta da un impianto domotico **KNX** [4] e alcuni dispositivi smart della **Philips** [5]. Questo sistema era già predisposto su di un appartamento usato come caso di studio;
- Per il *sistema di controllo domotico*, è stato scelto **OpenHAB** [6], perché consente di gestire una grande varietà di dispositivi domotici e quindi di non vincolarsi a uno specifico produttore. In aggiunta, è anche comprensivo di un database. Questo sistema era in gran parte già predisposto per controllare l'appartamento domotico;
- Per l'*assistente vocale*, è stata scelta **Amazon Alexa** [12], nello specifico un dispositivo **Amazon Echo Dot**, poiché già disponibile da ricerche e studi precedenti;

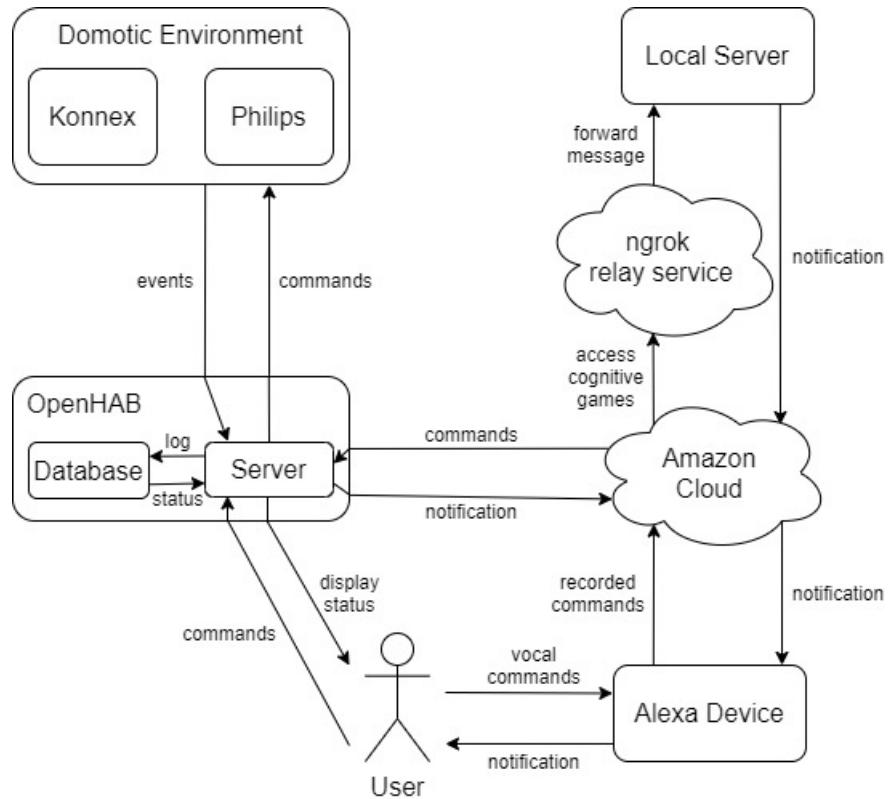


Figura 3.2: Lo schema delle interazioni nel sistema che sarà implementato in questo elaborato.

- Per gli *esercizi cognitivi*, allo scopo di eliminare i costi di manutenzione di un server pubblico, è stato scelto d'implementare un servizio su un server locale, esposto come server pubblico tramite **ngrok** [13], ovvero un servizio gratuito in cloud che inoltra le richieste ricevute a una specifica porta di una macchina locale.

3.2 Tecnologie utilizzate

In questo capitolo, si entrerà più nel dettaglio rispetto alle tecnologie utilizzate nel progetto, spiegando come vengono realizzate le interazioni tra i vari componenti del sistema.

Dunque, saranno prima descritte le caratteristiche di OpenHAB, come sistema di controllo domotico, poi di Alexa, come assistente vocale, e infine saranno descritti i servizi che offrono per interfacciarsi l'uno con l'altro.

3.2.1 OpenHAB come sistema di controllo domotico

OpenHAB (Open Home Automation Bus) è una piattaforma che gestisce la comunicazione e l'automazione in un ambiente domotico, svincolandosi dalle specifiche tecnologie utilizzate dai dispositivi che controlla. Le sue caratteristiche principali sono:

- La possibilità d'*integrare dispositivi domotici anche molto diversi*, permettendo loro di comunicare indirettamente, persino se usano tecnologie e/o protocolli direttamente incompatibili;
- La possibilità di controllare il proprio ambiente domotico attraverso un'*unica interfaccia utente* e di creare *regole di automazione per la casa uniche e valide su tutto il sistema*, evitando di utilizzare un'applicazione diversa per ogni specifico produttore;
- È un software *open source*, quindi, oltre a essere gratuito, possiede una comunità di sviluppatori che estendono continuamente le sue funzionalità. [14]

3.2.1.1. La comunicazione con i dispositivi domotici

La struttura con cui OpenHAB consente l'interazione con i dispositivi connessi, si basa su diversi livelli d'interfacciamento (come descritto in Figura 3.3).

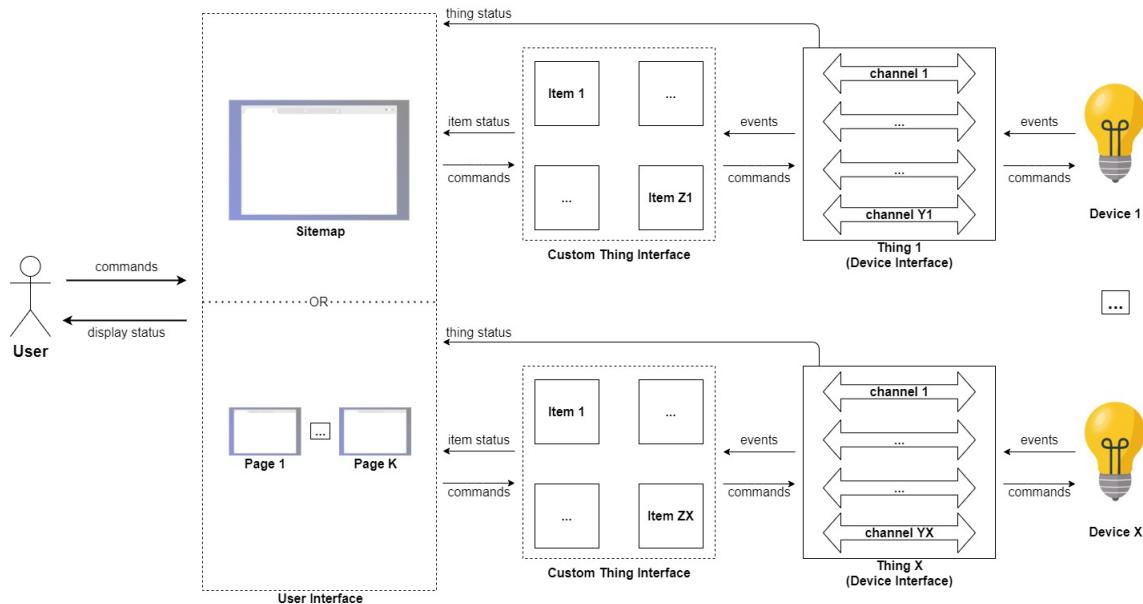


Figura 3.3: I diversi livelli d'interfacce poste tra l'utente e i suoi dispositivi domotici in OpenHAB.

Il primo livello d'interfacciamento è costituito dalle **things** [15], che espongono le funzionalità di un dispositivo fisico all'interno di OpenHAB, attraverso i **channel** di cui

sono composte. Ciascun *channel* espone una specifica funzione del dispositivo domotico. Ogni *thing* mantiene lo stato della connessione tra il dispositivo fisico che controlla e OpenHAB, ad esempio se il dispositivo è raggiungibile (*ONLINE*) o meno (*OFFLINE*).

Il secondo livello d’interfacciamento è costituito dagli *items* [16]. Esistono diversi tipi di *item*, ognuno dei quali si distingue per gli stati che può assumere e i comandi che può ricevere. Un *item* può essere associato a più *channel*, che controllerà inoltrando loro i comandi che riceve. Viceversa, un *channel* può essere associato a più *item*, da cui sarà controllato. Gli *item* offrono un certo grado di libertà e permettono di scegliere con quali comandi si vuole controllare i propri dispositivi, al contrario delle *thing*, che sono spesso vincolate dal dispositivo fisico che rappresentano.

L’ultimo livello d’interfacciamento è costituito dall’**interfaccia utente**, esposta attraverso il *Web Server di OpenHAB* e quindi accessibile tramite browser. All’interno di OpenHAB ne esistono di diversi tipi, ma le più recenti sono:

- **Sitemap** [17]: un’interfaccia dalla grafica minimale e funzionale;
- **Pages** [18]: un’interfaccia dalla grafica più accattivante e user-friendly.

Le componenti grafiche con cui si controlla un *item* dall’interfaccia utente sono determinate dal tipo di quell’*item*.

Di seguito, si riporta un esempio che integra quanto detto finora (Figura 3.4).

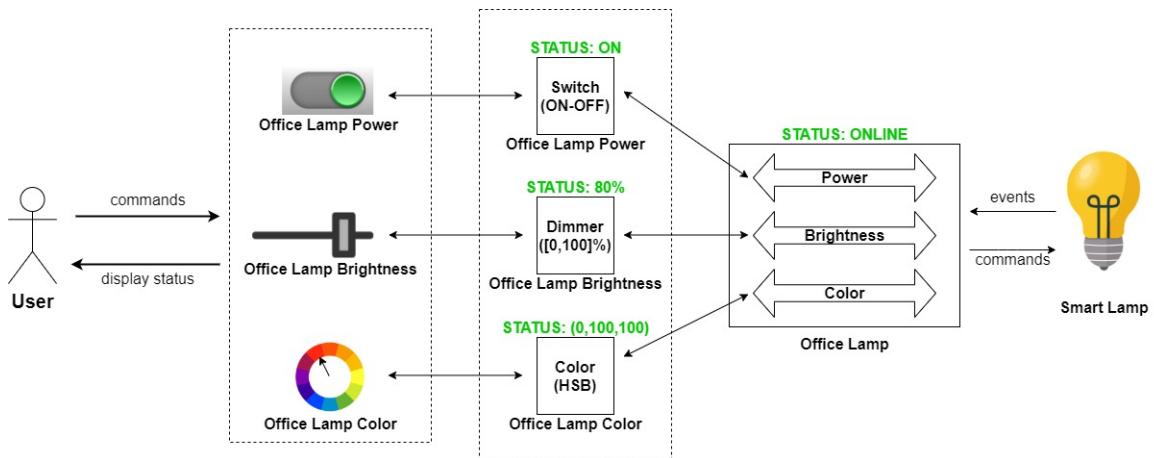


Figura 3.4: Esempio d’interfacciamento tra l’utente e una lampadina smart tramite OpenHAB.

3.2.1.2. Add-On, Automazione e Persistenza

Alle componenti principali, che permettono la comunicazione tra OpenHAB e i dispositivi domotici, si aggiungono delle componenti più avanzate.

In primo luogo, i ***binding*** [19] sono degli *add-on* che aggiungono ulteriori funzionalità a OpenHAB. Normalmente, i *binding* forniscono nuove tipologie di *thing*, che possiedono *channel* preconfigurati per connettere a OpenHAB dispositivi di una certa categoria. Ad esempio, esistono *binding* per dispositivi domotici di diversi produttori (come *Philips Hue Binding*).

In secondo luogo, le ***rule*** [20] gestiscono la parte di automazione di OpenHAB. In particolare, sono degli *handler* che possono accedere e inviare comandi agli *item* in risposta a particolari eventi.

Infine, la ***persistence*** [21] consente di costruire uno storico degli stati e degli eventi ricevuti, memorizzandoli sul database integrato di OpenHAB. È infatti possibile configurare OpenHAB in modo che campioni automaticamente lo stato di alcuni *item* indicati. Il metodo di campionamento può essere descritto attraverso una ***strategy***, che indica ogni quanto e sotto quali condizioni campionare gli *item*.

3.2.2 Alexa come assistente vocale

Amazon Alexa [12], o semplicemente Alexa, è un’assistente vocale sviluppato da Amazon. Più precisamente, è un servizio del cloud di Amazon capace d’interpretare la voce umana producendo del testo, e viceversa, dunque permettendo un’interazione vocale con altri servizi, sotto la sua mediazione.

Le applicazioni di Alexa sono numerose, ma principalmente includono:

- Lo *sviluppo di Alexa Skills*, ovvero di servizi pubblici accessibili tramite comandi vocali;
- Lo *sviluppo di dispositivi Alexa-Integrated*, ovvero di strumenti controllabili attraverso comandi vocali;
- L’agevolazione del monitoraggio e della gestione degli ambienti di lavoro tramite *Alexa for Business*.

In questo progetto, Alexa sarà utilizzata per interagire, oltre che con il sistema domotico, anche con gli esercizi cognitivi, che saranno quindi implementati come ***skills*** di Alexa.

3.2.2.1. Alexa Skill

Una *skill* di Alexa è composta principalmente da:

- Un **nome d’invocazione**, che la identifica ed è utilizzato per accedere alla *skill*;
- Un **modello di dialogo**, situato all’interno del cloud di Alexa, che indica le intenzioni che l’utente può esprimere all’assistente vocale all’interno della *skill*;

- Un **modello logico**, situato nell'endpoint specificato, che indica quali saranno le reazioni della *skill* in relazione alle intenzioni espresse dall'utente;
- Un **endpoint**, ovvero un indirizzo URL verso il servizio che implementa il modello logico della *skill*.

Per interagire con una certa *skill* è necessario richiamare il suo nome d'invocazione all'interno dei comandi vocali espressi. In alternativa, è possibile aprirla come un'applicazione qualsiasi (ad esempio, dicendo "Alexa, apri Corriere della Sera."). In tal caso, Alexa si aspetterà solo comandi compresi nel modello di dialogo di quella *skill*.

L'interazione con una *skill* di Alexa avviene come mostrato in Figura 3.5.

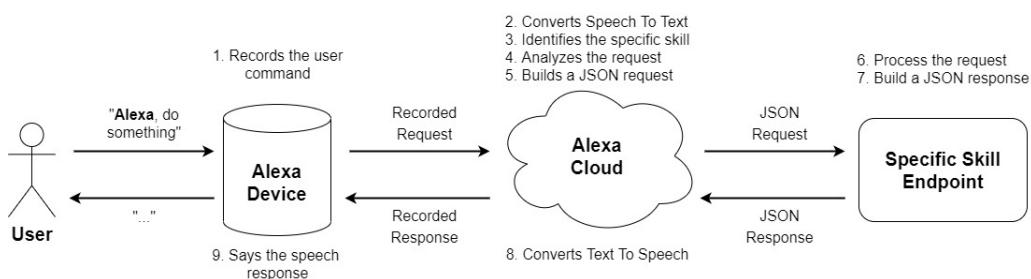


Figura 3.5: Il flusso d'interazione tra l'utente e una *skill* di Alexa che permette d'innescare la reazione di un sistema tramite comandi vocali.

Inizialmente, l'utente esprime la propria intenzione a un **dispositivo Alexa**, ad esempio un *Alexa Echo Dot*. Per farlo, dovrà precedere il comando vocale da una **wake-word** (di default "Alexa"), ovvero una parola che indicherà al dispositivo di cominciare a registrare. Una volta terminata la registrazione del comando vocale, il dispositivo la inoltrerà al **cloud di Alexa**, per essere interpretata. Giunta al cloud di Alexa, la registrazione sarà convertita in un comando testuale, che sarà analizzato per identificare la *skill* a cui si riferisce e l'intenzione espressa dall'utente all'interno di quella *skill*. Il cloud inoltrerà quindi i risultati dell'analisi all'**endpoint** della *skill* specifica. Ricevuta l'intenzione dell'utente, l'*endpoint* potrà reagire di conseguenza e produrre una risposta testuale, che sarà successivamente tradotta dal cloud di Alexa in una risposta vocale e infine comunicata dal dispositivo Alexa.

Come *endpoint* dove collocare la logica della *skill*, è possibile scegliere tra diverse soluzioni, come mostrato in Figura 3.6, ognuna con i propri vantaggi e svantaggi:

- **Amazon Web Services (AWS) Lambda** [22]: è un servizio in cloud di Amazon che permette di eseguire il proprio codice senza dover configurare e mantenere un proprio server; è scalabile e facile da usare, tuttavia ha un costo mensile basato sul numero di richieste ricevute e sulle risorse utilizzate;

- Un ***proprio cloud***, ovvero un gruppo di cluster di macchine gestito in proprio; è scalabile e completamente personalizzabile, tuttavia ha una configurazione relativamente più lunga e un costo di gestione elevato;
- Un ***proprio server pubblico***, ovvero una macchina accessibile da Internet; è la soluzione meno costosa ed è completamente personalizzabile, tuttavia ha una configurazione relativamente più lunga e soprattutto non è scalabile, che per un servizio accessibile pubblicamente è un grande svantaggio.

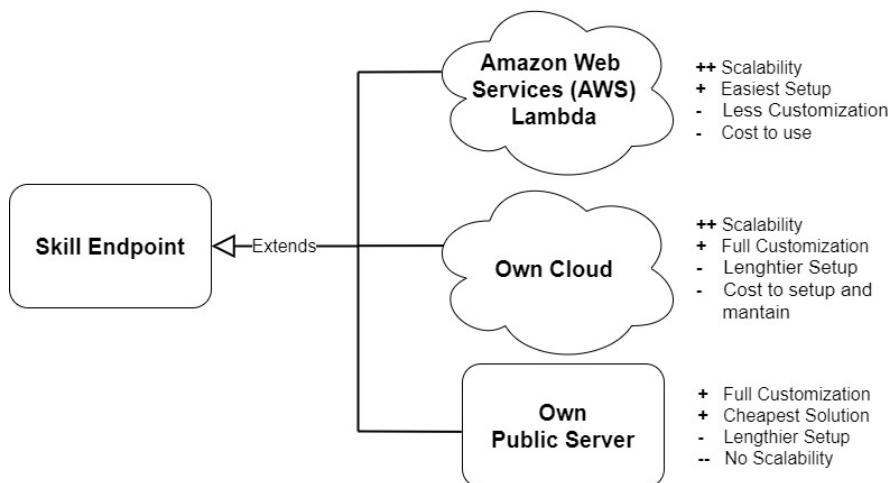


Figura 3.6: Le possibili soluzioni adottabili per implementare una *skill* di Alexa.

Per minimizzare i costi di progettazione, si è scelta una soluzione a costo zero, basata su un server locale esposto come server pubblico attraverso il servizio gratuito *ngrok*.

È possibile sviluppare una nuova *skill* per Alexa interamente attraverso l'**Amazon Developer Console** [23]: una piattaforma online, gestita da Amazon, che permette l'accesso a diversi strumenti di sviluppo software. In particolare, per lo sviluppo di una *skill* esiste uno strumento chiamato **ASK** (Alexa Skills Kit) [24].

3.2.2.2. Il modello di dialogo di una Alexa Skill

Il modello di dialogo di una *skill* di Alexa definisce le intenzioni (***intents***) che l'utente può esprimere all'interno della *skill*. Ogni *intent* è associato a una serie di proposizioni (***utterances***), che l'utente può pronunciare per esprimere quell'intenzione.

Un *utterance* può contenere degli ***slot***, ovvero degli spazi all'interno della proposizione, identificati da un nome, che possono assumere un certo dominio di valori, definito dal **tipo di slot**.

Esistono due categorie a cui può appartenere un tipo di slot:

- *Custom slot type*: è un tipo di slot il cui dominio di valori è un insieme finito definito dallo sviluppatore della *skill*;
- *Pre-built slot type*: è un tipo di slot il cui dominio di valori è predisposto da Amazon.

Di seguito, viene riportato uno schema che riassume i componenti appena descritti (Figura 3.7).

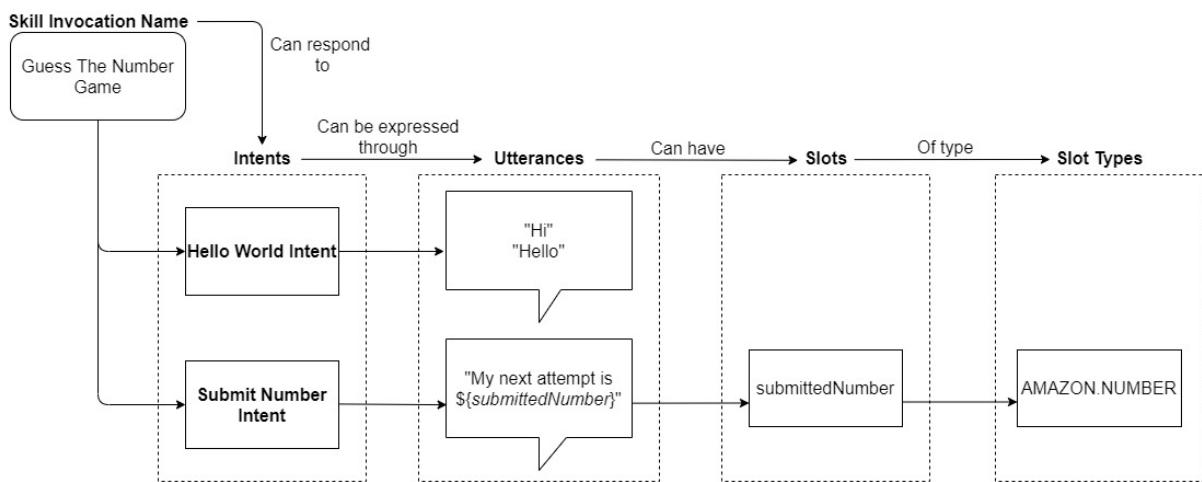


Figura 3.7: Le componenti del modello di dialogo di una *skill* di Alexa.

3.2.2.3. Il modello logico di una Alexa Skill

Il modello logico di una *skill* è costituito dal codice situato nell’*endpoint* della *skill*. In generale, le richieste che il cloud Alexa esegue sull’*endpoint* sono semplici richieste *http post*, che possono essere gestite liberamente. Tuttavia, Amazon predispone e suggerisce l’uso del suo framework, chiamato **ASK SDK** [25], che predilige una logica ad *handler di eventi*, o, nello specifico, ad *handler d’intenzioni (intent handlers)*. In aggiunta, il framework offre diverse funzionalità, tra cui la *validazione delle richieste*, ovvero una verifica sul mittente delle richieste ricevute dall’*endpoint*, che garantisca la loro provenienza dal cloud di Alexa (requisito necessario al deployment della *skill*).

3.2.3 Comunicazione tra Alexa e OpenHAB

In questo capitolo, si descriverà come è stata realizzata la comunicazione bidirezionale tra Alexa e OpenHAB, attraverso i servizi che offrono per interfacciarsi l’uno con l’altro.

3.2.3.1. Amazon Echo Control Binding per OpenHAB

Amazon Echo Control Binding [26] è un *add-on* di OpenHAB che permette d'integrare un dispositivo Alexa al sistema OpenHAB. Il *binding* sarà quindi utilizzato per gestire la comunicazione **da OpenHAB verso Alexa**.

Più in dettaglio, consente di collegare degli account Amazon a OpenHAB e pertanto di controllare i dispositivi Alexa associati a quegli account, come se fossero delle *things*. Risulta dunque possibile utilizzare un *Alexa Echo Dot* per dare voce al proprio sistema domotico, inviando notifiche vocali di eventi rilevati da OpenHAB.

3.2.3.2. OpenHAB Skill per Alexa

OpenHAB [27] è anche il nome di una *skill* per Alexa, che permette di mappare ogni dispositivo connesso a OpenHAB alle interfacce predisposte da Alexa, che descrivono le capacità di un certo dispositivo, consentendo così un'interazione vocale verso di esso. La *skill* sarà perciò utilizzata per gestire la comunicazione **da Alexa verso OpenHAB**.

3.3 Limitazioni del progetto

Ora si illustreranno alcune delle limitazioni del sistema, dovute stavolta alla particolare progettazione scelta.

La prima limitazione è dovuta ai protocolli di comunicazione utilizzati. Infatti, gran parte delle comunicazioni nel sistema avvengono tramite il protocollo *http*, che è un protocollo particolarmente lento e pesante su dispositivi con poche risorse per la computazione. Questo provocherà una certa **latenza nelle interazioni con il proprio sistema domotico**, in particolare le interazioni vocali, che sono le più complesse.

La seconda limitazione è dovuta alla **non scalabilità del servizio pubblico per gli esercizi cognitivi**, che è installato su un singolo server. Trattandosi di un servizio pubblico, la cui domanda potrebbe variare molto nel tempo, la sua scalabilità risulta un fattore necessario. Nonostante ciò, il singolo server si presta bene almeno allo studio e alla realizzazione di un prototipo.

La terza limitazione è dovuta al metodo con cui è stato esposto il server locale, per offrire un servizio pubblico. Infatti, utilizzare il servizio *ngrok* aggiunge un ulteriore intermediario nella comunicazione, aumentando dunque la latenza nelle interazioni con il proprio servizio. Inoltre la licenza gratuita di *ngrok* non concede di dare un nome pubblico al proprio server, ma restituisce un identificatore casuale.

Capitolo 4

Progettazione delle skill

In questo capitolo, si descriverà il processo di progettazione delle *skill* per i giochi cognitivi.

Più in dettaglio, per ciascuna *skill*, sarà prima delineato l'esercizio cognitivo implementato e il suo scopo, facendo riferimento alle capacità cognitive di cui permette l'allenamento. Successivamente, se ne illustrerà un caso d'uso specifico, che ne riassumerà i requisiti, ovvero la logica del gioco implementato. Infine, sarà presentato il progetto di una sua possibile soluzione, descrivendo i contesti in cui si può trovare l'utente nella *skill* e le intenzioni che può esprimere all'interno di tali contesti.

I giochi implementati avranno alcune caratteristiche in comune. Innanzitutto, saranno giochi che, per come sono gestite le interazioni con Alexa, si svolgeranno **necessariamente a turni**. Ogni turno, l'utente potrà esprimere una propria volontà attraverso un'intenzione, che potrà essere soddisfatta dall'endpoint, producendo una risposta adeguata da parte di Alexa. Inoltre, si è deciso di seguire un **modello d'interazione simile** per proseguire all'interno dei giochi della suite. In questo modo, una volta raggiunto un certo livello di familiarità con uno specifico gioco, all'utente risulterà più facile interagire con gli altri giochi. In particolare, esisteranno dei comandi comuni nei giochi per conoscere le funzionalità della *skill*, le regole del gioco implementato e per cominciare una partita.

Altri comandi condivisi, sono invece comandi standard richiesti da Amazon per poter eseguire il deployment di una *skill*, come quelli per aprire e chiudere la *skill*.

4.1 Number List Game

Number List Game è il primo gioco sviluppato per questa suite. È stato utilizzato per sondare le capacità di un'assistente vocale nel contesto dell'allenamento delle funzioni cognitive del cervello.

Il gioco consiste semplicemente nel dover ripetere una lista di numeri espressa da Alexa. La lista di numeri cambia e diventa più lunga a ogni turno. Al primo errore, il

gioco termina, annunciando al giocatore il suo punteggio finale, ovvero il numero di liste che è riuscito a ricordarsi correttamente.

Lo scopo del gioco è quello di allenare la **memoria a breve termine**, infatti sarà richiesto all'utente di ricordarsi una cosa sempre diversa e sempre più difficile.

4.1.1 Analisi del modello di dialogo

Di seguito, viene riportata la progettazione di un esempio d'interazione tra l'utente e la *skill* Number List Game, che riassume i requisiti fondamentali della *skill*, ovvero la logica di base del gioco (Figura 4.1).

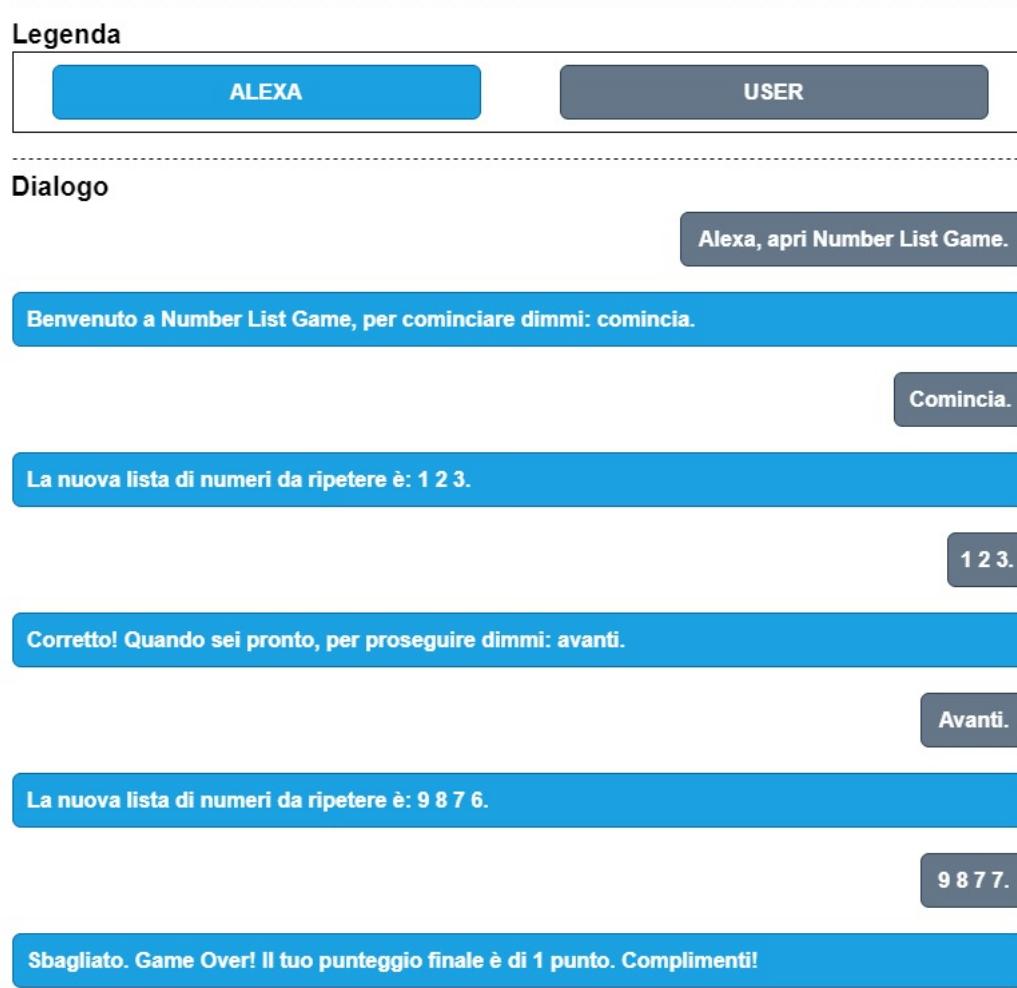


Figura 4.1: Un esempio d'interazione con la *skill* Number List Game, progettato per riassumere la logica del gioco.

4.1.2 Progettazione del modello di dialogo

Nel capitolo seguente, viene riportato il progetto di una possibile soluzione che soddisfi i requisiti della *skill* Number List Game. In particolare, il progetto consiste in un diagramma di stati della *skill*, in cui gli stati sono da intendersi come **contesti** e **sotto-contesti**, in cui ha senso esprimere una certa **intenzione** o meno. Il contesto in cui si trova una *skill* può cambiare in base alle intenzioni espresse dall'utente. A ogni intenzione, corrisponde un'azione che sarà eseguita dalla *skill*.

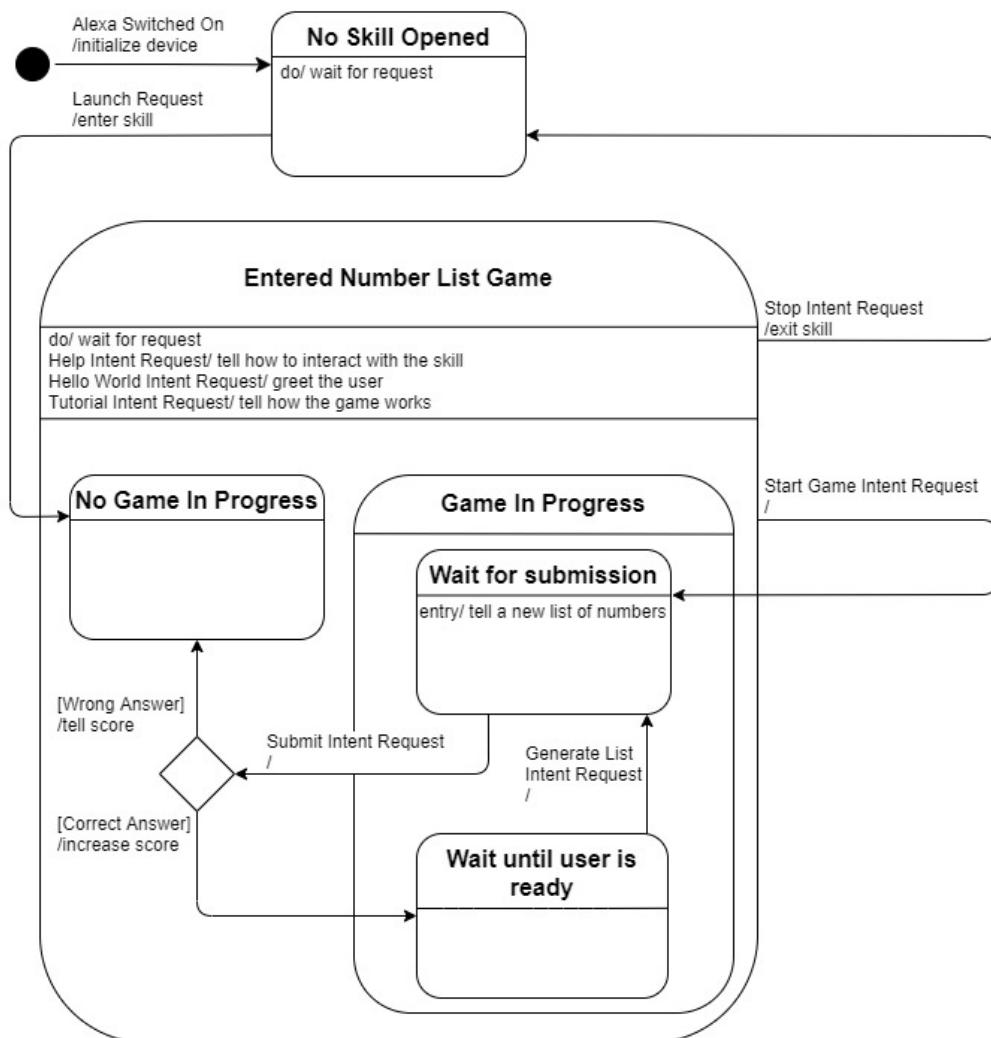


Figura 4.2: Diagramma degli stati della *skill* Number List Game.

Come si vede dalla Figura 4.2, sarà possibile interagire con la *skill* attraverso le seguenti *intenzioni*, all'interno degli opportuni *contesti* e *sotto-contesti*:

- **NO SKILL OPENED:** contesto in cui la skill non è ancora stata aperta; è anche il contesto in cui si trova Alexa subito dopo l'accensione:
 - ***Launch Request***: intenzione che esprime la volontà dell'utente di *aprire la skill Number List Game*;
- **ENTERED NUMBER LIST GAME:** contesto in cui la skill è stata aperta:
 - ***Stop Intent Request***: intenzione che esprime la volontà dell'utente di *uscire dalla skill Number List Game*;
 - ***Hello World Intent Request***: intenzione che esprime la volontà dell'utente di *essere salutato dalla skill Number List Game* (utilizzata per testare la comunicazione con la skill);
 - ***Help Intent Request***: intenzione che esprime la volontà dell'utente di *conoscere com'è possibile interagire con la skill Number List Game*;
 - ***Tutorial Intent Request***: intenzione che esprime la volontà dell'utente di *conoscere come funziona il gioco*;
 - ***Start Game Intent Request***: intenzione che esprime la volontà dell'utente di *cominciare una nuova partita*;
 - **NO GAME IN PROGRESS:** contesto in cui non è ancora stata cominciata nessuna partita;
 - **GAME IN PROGRESS:** contesto in cui è già in corso una partita:
 - **WAIT FOR SUBMISSION:** contesto in cui la skill ha comunicato una lista di numeri e aspetta che l'utente la ripeti:
 - ***Submit Intent Request***: intenzione che esprime la volontà d'*inviare una lista di numeri alla skill*;
 - **WAIT UNTIL USER IS READY:** contesto in cui la skill ha processato il tentativo corretto dell'utente di ripetere la lista di numeri precedentemente comunicata e aspetta una conferma da parte del giocatore:
 - ***Generate List Intent Request***: intenzione che esprime la volontà di *ricevere una nuova lista di numeri da ripetere*.

4.2 Category Game

Un altro gioco sviluppato per questa suite è **Category Game**. Il gioco consiste nel categorizzare le parole comunicate da Alexa. In particolare, possiede due diverse modalità:

- Nella *modalità facile*, Alexa comunicherà al giocatore una categoria. Successivamente, a ogni turno, riferirà una parola diversa che il giocatore dovrà indicare come appartenente alla categoria o meno. Lo scopo della modalità è quello di allenare **l'attenzione** e la **categorizzazione**, infatti sarà richiesto all'utente di riconoscere solo alcune parole, tra tutte, come appartenenti a un certo gruppo;
- Nella *modalità difficile*, Alexa comunicherà al giocatore due categorie. Dopodiché, a ogni turno, riferirà una parola diversa appartenente a una delle due categorie menzionate. In questo caso però, il giocatore dovrà indicare la parola come appartenente alla categoria contraria a quella corretta. Lo scopo della modalità è quello di allenare **l'attenzione** e **l'inibizione degli automatismi**, infatti sarà richiesto all'utente di andare contro le sue abitudini e di rispondere in modo errato ai quesiti posti.

Il gioco termina al primo errore commesso dal giocatore, annunciando il suo punteggio finale, ovvero il numero di parole che è riuscito a categorizzare correttamente.

4.2.1 Analisi del modello di dialogo

Di seguito, viene riportata la progettazione di due esempi d'interazione tra l'utente e la *skill* Category Game, uno per modalità di gioco (Figura 4.3 e Figura 4.4).

Legenda

ALEXA

USER

Dialogo

Alexa, apri Category Game.

Benvenuto a Category Game, per cominciare dimmi: comincia.

Comincia.

A che difficoltà vuoi giocare: facile o difficile?

Facile.

Ho scelto la categoria Pesce. La prossima parola è: trota.

Pesce.

Corretto! La prossima parola è: barca.

Altro.

Corretto! La prossima parola è: pescatore.

Pesce.

Sbagliato. Game Over! Il tuo punteggio finale è di 2 punti. Complimenti!

Figura 4.3: Esempio d'interazione con la *skill* Category Game, progettato per riassumere la logica del gioco nella modalità facile.

Legenda

ALEXA

USER

Dialogo

Alexa, apri Category Game.

Benvenuto a Category Game, per cominciare dimmi: comincia.

Comincia.

A che difficoltà vuoi giocare: facile o difficile?

Difficile.

Ho scelto le due categorie: Frutta e Verdura. La prossima parola è: mela.

N.B.: bisogna dire la categoria contraria a quella corretta.

Verdura.

Corretto! La prossima parola è: carota.

Frutta.

Corretto! La prossima parola è: pomodoro.

N.B.: il pomodoro è un frutto, quindi la risposta è sbagliata.

Frutta.

Sbagliato. Game Over! Il tuo punteggio finale è di 2 punti. Complimenti!

Figura 4.4: Esempio d'interazione con la *skill* Category Game, progettato per riassumere la logica del gioco nella modalità difficile.

4.2.2 Progettazione del modello di dialogo

Ora viene riportato il progetto di una possibile soluzione che soddisfi i requisiti della skill Category Game.

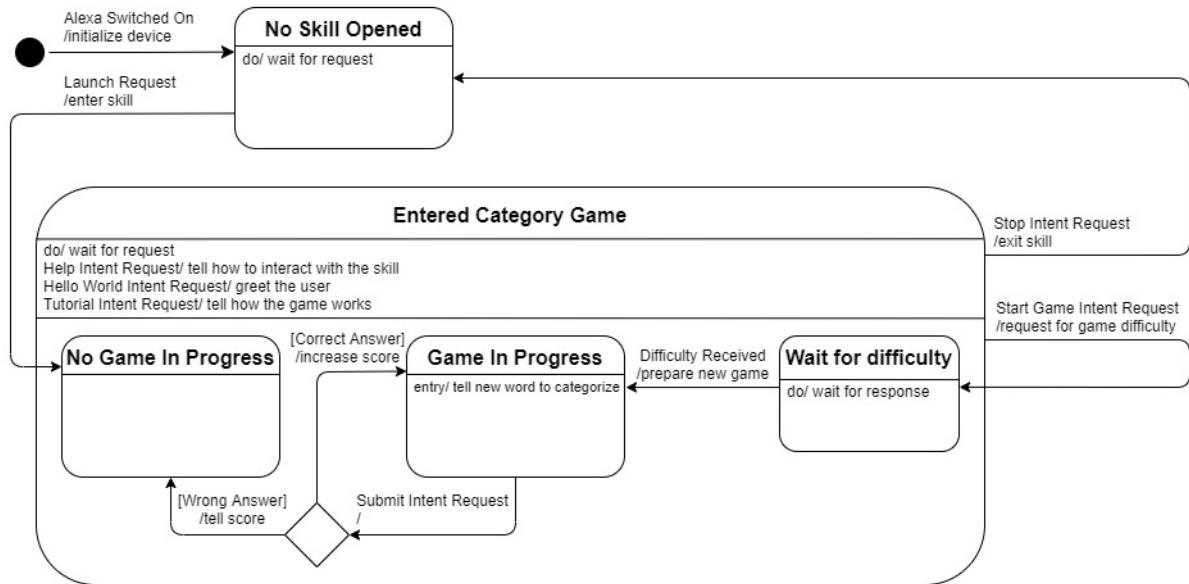


Figura 4.5: Diagramma degli stati della skill Category Game.

Come si vede dalla Figura 4.5, sarà possibile interagire con la skill attraverso le seguenti *intenzioni*, all'interno degli opportuni *contesti* e *sotto-contesti*:

- **NO SKILL OPENED**: contesto in cui la skill non è ancora stata aperta; è anche il contesto in cui si trova Alexa subito dopo l'accensione:
 - **Launch Request**: intenzione che esprime la volontà dell'utente di *aprire la skill Category Game*;
- **ENTERED NUMBER LIST GAME**: contesto in cui la skill è stata aperta:
 - **Stop Intent Request**: intenzione che esprime la volontà dell'utente di *uscire dalla skill Category Game*;
 - **Hello World Intent Request**: intenzione che esprime la volontà dell'utente di *essere salutato dalla skill Category Game* (utilizzata per testare la comunicazione con la skill);

- o **Help Intent Request**: intenzione che esprime la volontà dell’utente di *conoscere com’è possibile interagire con la skill*;
- o **Tutorial Intent Request**: intenzione che esprime la volontà dell’utente di *conoscere come funziona il gioco*;
- o **Start Game Intent Request**: intenzione che esprime la volontà dell’utente di *cominciare una nuova partita*. In questo caso, prima di cominciare la partita, sarà richiesto all’utente di specificare anche la difficoltà a cui vuole giocare, che sarà utilizzata per determinare la modalità di gioco;
- o **NO GAME IN PROGRESS**: contesto in cui non è ancora stata cominciata nessuna partita;
- o **GAME IN PROGRESS**: contesto in cui è già in corso una partita ed è stata comunicata una parola da categorizzare al giocatore:
 - **Submit Intent Request**: intenzione che esprime la volontà del giocatore *d’inviare una categoria alla skill*.

Si noti come in questo caso si sia deciso di non richiedere ogni turno la conferma per proseguire nel gioco. Questa decisione è stata presa per alleggerire il flusso di dialogo e diminuire il numero d’interazioni necessarie per progredire in una partita.

4.3 Labyrinth Game

Labyrinth Game è il gioco più complesso sviluppato per questa suite.

Il gioco consiste nel doversi muovere e orientare all’interno di un labirinto bidimensionale, raggiungendone eventualmente l’uscita. All’inizio della partita, Alexa genererà il labirinto. Successivamente, a ogni turno, Alexa descriverà i dintorni del giocatore, il quale dovrà scegliere una direzione verso cui spostarsi.

Come mostrato nella Figura 4.6, il labirinto sarà composto di:

- *Mura*: componenti non attraversabili;
- *Mura di perimetro*: componenti non attraversabili, che costituiscono il perimetro del labirinto;
- *Percorsi*: componenti attraversabili;
- *Torri*: componenti attraversabili, che forniscono dei suggerimenti al giocatore per aiutarlo a raggiungere l’uscita del labirinto;
- *Un’entrata*: il percorso su cui si trova il giocatore all’inizio della partita;
- *Un’uscita*: il percorso che dovrà raggiungere il giocatore per terminare la partita.

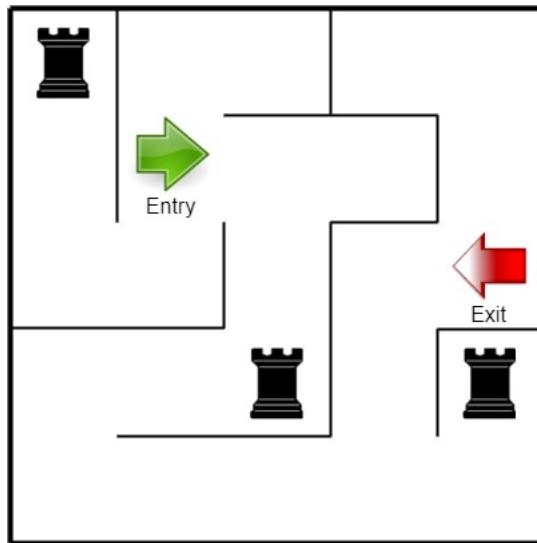


Figura 4.6: Esempio di layout di un labirinto nella *skill Labyrinth Game*.

Il gioco termina quando il giocatore raggiunge l'uscita del labirinto, annunciando il suo punteggio finale, ovvero il numero di passi che ha compiuto prima di raggiungere l'uscita.

Lo scopo del gioco è quello di allenare la **memoria a lungo termine**, l'**orientamento** e l'**immaginazione**, infatti sarà richiesto all'utente di ricordare e costruirsi mentalmente il layout del labirinto man mano che lo esplora, riconoscendo la posizione in cui si trova.

4.3.1 Analisi del modello di dialogo

Di seguito, viene riportata la progettazione di un esempio d'interazione dell'utente con la *skill Labyrinth Game* (Figura 4.7).

Legenda

ALEXA

USER

Dialogo

Alexa, apri Labyrinth Game.

Benvenuto a Labyrinth Game, per cominciare dimmi: comincia.

Comincia.

Ho generato il labirinto.

- A Nord, vedi un muro di perimetro;
- A Est, vedi un percorso;
- A Sud, vedi un percorso;
- A Ovest, vedi un muro di perimetro.

Dove vuoi andare?

Est.

- A Nord, vedi un muro di perimetro;
- A Est, vedi un muro;
- A Sud, vedi una torre;
- A Ovest, vedi l'entrata del labirinto.

Dove vuoi andare?

Sud.

Dall'alto della torre, vedi che l'uscita è distante 1 passo e che il prossimo passo è verso Sud.

- A Nord, vedi un percorso;
- A Est, vedi un percorso;
- A Sud, vedi l'uscita del labirinto;
- A Ovest, vedi un muro.

Dove vuoi andare?

Sud.

Game Over! Hai raggiunto l'uscita in 3 passi. Complimenti!

Figura 4.7: Esempio d'interazione con la skill Labyrinth Game, progettato per riassumere la logica del gioco.

4.3.2 Progettazione del modello di dialogo

Ora viene riportato il progetto di una possibile soluzione che soddisfi i requisiti della skill Labyrinth Game.

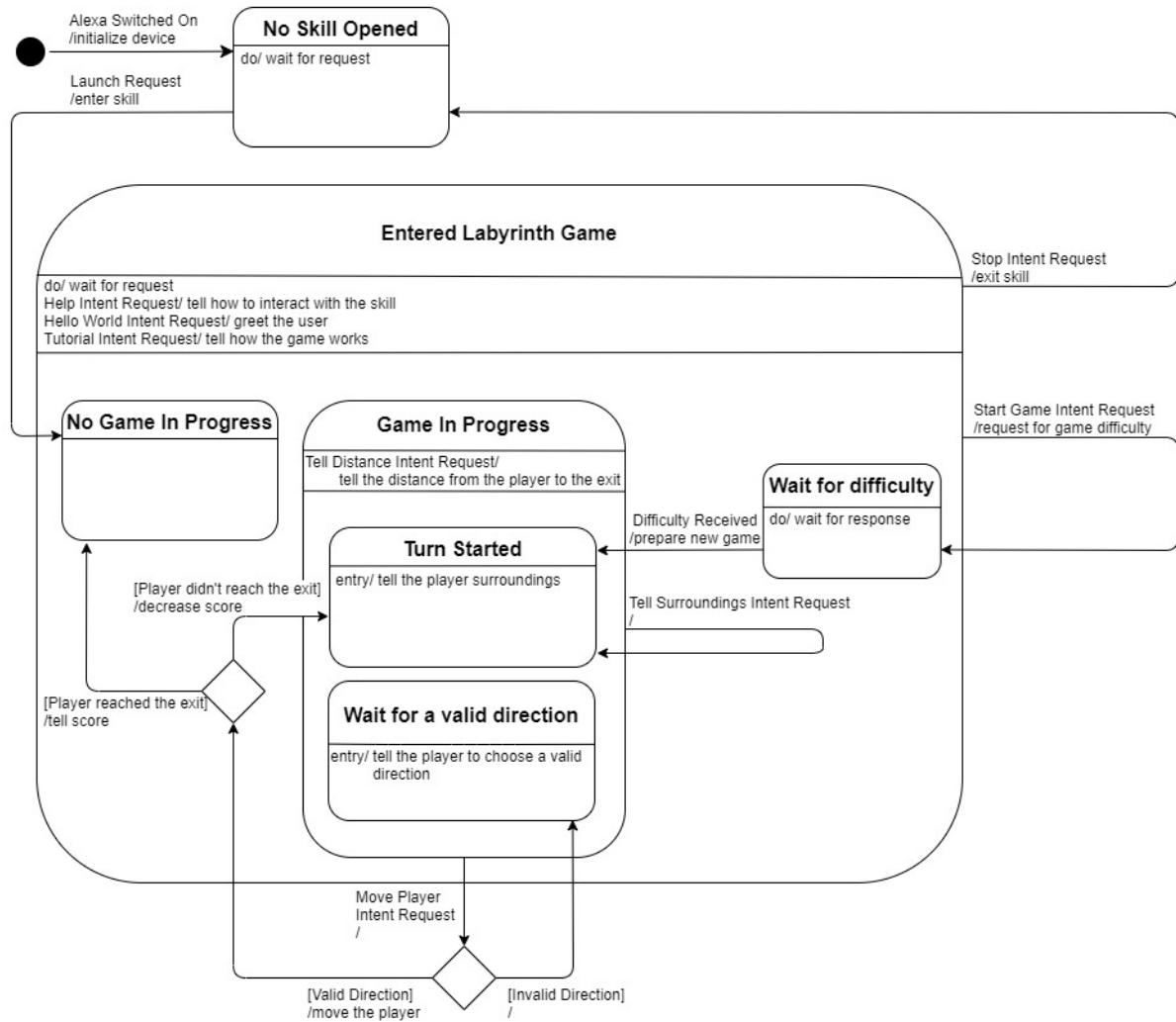


Figura 4.8: Diagramma degli stati della skill Labyrinth Game.

Come si vede dalla Figura 4.8, sarà possibile interagire con la skill attraverso le seguenti *intenzioni*, all'interno degli opportuni *contesti* e *sotto-contesti*:

- **NO SKILL OPENED**: contesto in cui la skill non è ancora stata aperta; è anche il contesto in cui si trova Alexa subito dopo l'accensione:

- o **Launch Request**: intenzione che esprime la volontà dell’utente di *aprire la skill Labyrinth Game*;
- **ENTERED NUMBER LIST GAME**: contesto in cui la skill è stata aperta:
 - o **Stop Intent Request**: intenzione che esprime la volontà dell’utente di *uscire dalla skill Labyrinth Game*;
 - o **Hello World Intent Request**: intenzione che esprime la volontà dell’utente di *essere salutato dalla skill Labyrinth Game* (utilizzata per testare la comunicazione con la skill);
 - o **Help Intent Request**: intenzione che esprime la volontà dell’utente di *conoscere com’è possibile interagire con la skill Labyrinth Game*;
 - o **Tutorial Intent Request**: intenzione che esprime la volontà dell’utente di *conoscere come funziona il gioco*;
 - o **Start Game Intent Request**: intenzione che esprime la volontà dell’utente di *cominciare una nuova partita*. Prima di cominciare la partita, sarà richiesto all’utente di specificare anche la difficoltà a cui vuole giocare, che sarà utilizzata per determinare le dimensioni del labirinto;
 - o **NO GAME IN PROGRESS**: contesto in cui non è ancora stata cominciata nessuna partita;
 - o **GAME IN PROGRESS**: contesto in cui è già in corso una partita ed è stato generato il labirinto su cui giocherà l’utente:
 - **Tell Distance Intent**: intenzione che esprime la volontà del giocatore di *conoscere la sua distanza dall’uscita*;
 - **TURN STARTED**: contesto in cui sono già stati descritti i dintorni del giocatore:
 - o **Tell Surroundings Intent**: intenzione che esprime la volontà del giocatore di *conoscere i propri dintorni*;
 - o **Move Player Intent**: intenzione che esprime la volontà del giocatore di *muoversi verso una certa direzione*.

A questo punto, una volta presentato il progetto, è possibile passare alla descrizione della sua effettiva implementazione.

Capitolo 5

Implementazione del prototipo

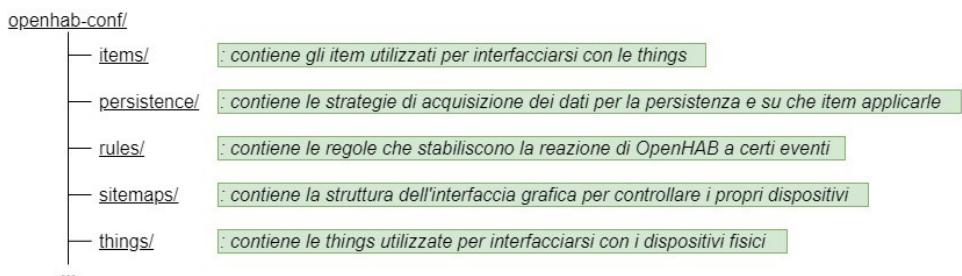
In questo capitolo, si discuterà l'implementazione prima del sistema domotico e successivamente delle *skill* di Alexa per gli esercizi cognitivi. In entrambi i casi, saranno anche forniti degli esempi di codice che descriveranno le interazioni tra le componenti di ciascun sistema.

5.1 Implementazione del sistema domotico

In questa sezione, si illustrerà l'organizzazione del codice per il sistema domotico e l'implementazione del servizio che permetterà all'utente di controllare e monitorare i propri dispositivi domotici. Successivamente, sarà fornito un esempio d'integrazione di un dispositivo domotico a OpenHAB, per il quale è stata prevista la possibilità di una comunicazione bidirezionale tra Alexa e OpenHAB.

5.1.1 Struttura dei file di configurazione

Di seguito, viene riportata la struttura standard dei file di configurazione di OpenHAB (Figura 5.1).



I file di configurazione definiscono:

- *Things* e *Channels*: i dispositivi fisici connessi al sistema e le loro funzionalità;
- *Items*: i metodi e i componenti grafici utilizzabili per controllare i dispositivi, all'interno dei file di configurazione di OpenHAB e dell'interfaccia grafica rispettivamente;
- *Sitemaps*: l'organizzazione dei componenti nell'interfaccia grafica;
- *Rules*: le regole di automazione del sistema domotico, ovvero le sue reazioni agli eventi che riceve dai dispositivi che lo compongono;
- *Persistence*: le regole utilizzate per costruire uno storico degli stati degli item specificati.

Questi file sono caricati all'avvio della macchina in cui è stato installato OpenHAB e a ogni loro aggiornamento durante l'esecuzione. Nel caso specifico, il sistema domotico è stato installato su un **Raspberry PI**, con sistema operativo a base **Linux**.

5.1.2 Accesso al sistema domotico

Per controllare il proprio sistema domotico è possibile accedere alla **Dashboard di OpenHAB**, disponibile all'indirizzo IP della macchina su cui è installato, su una specifica porta.

La dashboard permette di configurare il proprio sistema tramite interfaccia grafica e d'interagire con le interfacce grafiche realizzate per controllare i propri dispositivi domotici. In particolare, queste definiscono l'organizzazione dei componenti grafici che potranno essere usati per controllare i dispositivi domotici, ma non i componenti grafici stessi (definiti invece dalle *tipologie di item* usate per rappresentare le funzionalità di quei dispositivi). I componenti grafici, oltre a controllare un determinato dispositivo domotico, permettono di accedere allo storico dei suoi stati in un certo periodo di tempo d'interesse.

Per controllare il sistema domotico, sono state rese disponibili due interfacce grafiche:

- Una **sitemap**: privilegia un'esposizione agevole e compatta delle funzionalità del sistema; quest'interfaccia era già stata predisposta in un tempo precedente rispetto a questo progetto;
- Delle **pages**: privilegiano l'estetica dell'interfaccia grafica.

Per le *pages*, si è deciso di sfruttare una nuova funzionalità introdotta nella versione 3.x di OpenHAB: il **modello semantico**.

Il modello semantico di un certo locale consente di organizzare logicamente i propri dispositivi domotici, prima in base alla loro posizione nel locale e successivamente in base alla loro funzionalità. OpenHAB è quindi in grado di:

- costruire il modello semantico automaticamente, se il codice rispecchia una certa struttura logica;
- dal modello semantico, ricavare automaticamente un’interfaccia grafica ben strutturata, dalla quale è possibile controllare i propri dispositivi domotici.

Pertanto, è stato sufficiente riorganizzare la struttura di alcuni file di configurazione, per ottenere un’interfaccia grafica già impostata, che consentisse all’utente di controllare e monitorare il proprio sistema domotico.

5.1.3 Esempio d’integrazione con OpenHAB

Ora sarà illustrato un esempio su come è possibile integrare una lampadina Philips Hue al sistema domotico.

Innanzitutto, è necessario installare sul proprio sistema domotico un *add-on*, chiamato **Philips Hue Binding** [28]. Questo definirà alcune *tipologie di thing*, che hanno *channels* preconfigurati per comunicare con dispositivi fisici della Philips Hue.

A questo punto è possibile iniziare a integrare i propri dispositivi domotici, dichiarando le **things** del sistema in un file *.things*.

```
Bridge hue:bridge:1 [ ipAddress="...", port="...", userName="..." ]{
    0210 lightbulb1 "Lampadina Test 1" @ "Office" [ lightId="1" ]
    0820 button1 "Pulsante Test 1" @ "Office" [ sensorId="8" ]
}
```

Figura 5.2: Un esempio di file di configurazione per le *thing*: in questo caso è stato dichiarato un bridge con due dispositivi connessi, una lampadina e un pulsante.

Nell’esempio (Figura 5.2), viene integrato un **bridge Hue** a cui sono collegati una **lampadina** e un **pulsante** della Philips Hue. Queste sono le *things* vere e proprie e sono contraddistinte da due proprietà principali:

- Il **tipo di thing**, che dichiara i *channel* disponibili a quella *thing*, ovvero le funzionalità del dispositivo. In questo caso, sono forniti dal *binding* del produttore specifico (nell’esempio: *0210* e *0820*);
- Il **nome della thing**, che serve per riconoscerla all’interno dei file di configurazione di OpenHAB (nell’esempio: *lightbulb1* e *button1*).

Una volta integrati i dispositivi domotici al sistema OpenHAB, bisogna definire come sarà possibile controllarli nei file di configurazione e nell’interfaccia grafica, dichiarando gli **items** del sistema in un file *.items*.

```

Switch light_switch "Lampadina Hue" <lightbulb> (EQP_hue_lamp) ["Control", "Light"] {
    alexa="PowerController.powerState"
    [category="LIGHT", friendlyNames="Lampadina hue", language="it"],
    channel="hue:0210:1:lightbulb1:color"
}
Color light_color "Colore luce" <colorpicker> (EQP_hue_lamp) ["Control", "Light"] {
    alexa="PowerController.powerState,BrightnessController.brightness,ColorController.color"
    [category="LIGHT", friendlyNames="Colore luce hue", language="it"],
    channel="hue:0210:1:lightbulb1:color"
}
Dimmer light_temperature "Temperatura della luce" <colorpicker> (EQP_hue_lamp) ["Control", "ColorTemperature"] {
    alexa="PercentageController.percentage"
    [category="LIGHT", friendlyNames="Temperatura luce hue", language="it"],
    channel="hue:0210:1:lightbulb1:color_temperature"
}

```

Figura 5.3: Un esempio di file di configurazione per gli *items*: in questo caso sono state esposte le funzionalità di una lampadina Hue, che comprendono l'accensione, il colore e la temperatura del colore.

Nell'esempio (Figura 5.3) sono stati creati tre *items* per la lampadina integrata precedentemente: uno **Switch** per accendere e spegnere la lampadina, un **Color** per impostarne il colore e un **Dimmer** per regolarne la temperatura del colore. I punti focali della sintassi di un *item* sono:

- Il ***tipo di item***, che definisce come sarà possibile interagire con le funzionalità che controlla. I *tipi di item* sono standard e ognuno fornisce gli stati che può assumere l'*item* e i comandi che può ricevere. Ad esempio, uno Switch può essere acceso (*ON*) o spento (*OFF*);
- Il ***nome dell'item***, che serve per riconoscerlo all'interno dei file di configurazione di OpenHAB (nell'esempio *light_switch*, *light_color* e *light_temperature*);
- Il ***channel associato***, che definisce le funzionalità che l'*item* controlla e il ponte tra *item* e *thing*. In questo caso, la sintassi usata per identificare un *channel* è definita dal *binding* a cui appartiene, ovvero Philips Hue Binding (nell'esempio: *channel="hue:0210:1:lightbulb1:color"* permette di controllare il colore e lo stato di accensione della *thing* *lightbulb1*).
- L'***interfaccia di Alexa implementata***, che serve per configurare la **skill di OpenHAB** per Alexa. In particolare, indica con quali comandi vocali si potrà controllare quell'*item* attraverso un dispositivo Alexa (nell'esempio: *alexa="PowerController.PowerState" [...]*).

A questo punto è già possibile controllare i propri dispositivi domotici tramite Alexa e, assumendo che la sintassi rispetti le regole del modello semantico di OpenHAB, anche attraverso un'interfaccia grafica, generata automaticamente, su OpenHAB.

L'ultima cosa che rimane da fare, è quella di notificare l'utente degli eventi rilevati da OpenHAB. A questo proposito, sarà necessario installare un altro *add-on* sul proprio sistema domotico: **Amazon Echo Control Binding**. Questo *binding* permetterà d'integrare un dispositivo Alexa a OpenHAB, con un metodo analogo a quello utilizzato per la lampadina Hue. Dunque, sarà possibile controllarlo attraverso una regola di automazione, definendo una **rule** in un file *.rules*.

```
rule "Light on"
when
    Item light_switch changed
then
    if (light_switch.state == ON) {
        echo_device_1_voice.sendCommand("<speak>La lampadina hue è stata accesa.</speak>");
    } else {
        echo_device_1_voice.sendCommand("<speak>La lampadina hue è stata spenta.</speak>");
    }
end
```

Figura 5.4: Un esempio di file di configurazione per le *rules*: in questo caso è stato definito un *handler* che reagisce ai cambiamenti di stato della lampadina Hue, comunicandoli all'utente tramite un dispositivo Alexa.

Nell'esempio (Figura 5.4), viene definita una regola che risponde ai cambiamenti dello stato di accensione della lampadina Hue (ovvero se è stata accesa o spenta), controllando un dispositivo Alexa Echo e quindi comunicando all'utente il nuovo stato della lampadina.

5.2 Implementazione delle skill

In questa sezione, si illustrerà l'organizzazione del codice per il progetto del servizio che espone gli esercizi cognitivi e l'implementazione dell'interazione tra l'utente e tale servizio. Infine, saranno presentati alcuni estratti di codice che aiutino a visualizzare nel concreto le componenti del sistema e le loro interazioni.

Per l'implementazione è stato utilizzato il linguaggio di programmazione **TypeScript** [29], che è un'estensione tipizzata di **JavaScript**. Il servizio è stato costruito sul framework **node.js** [30], il cui scopo è gestire applicazioni di rete scalabili, e in particolare sul framework **Express** [31], utilizzato per costruire applicazioni web.

5.2.1 Struttura dell'applicazione

Di seguito, viene riportata la struttura organizzativa utilizzata nella directory del progetto sugli esercizi cognitivi (Figura 5.5). Sono quindi evidenziate, le componenti principali del progetto e la loro funzione.

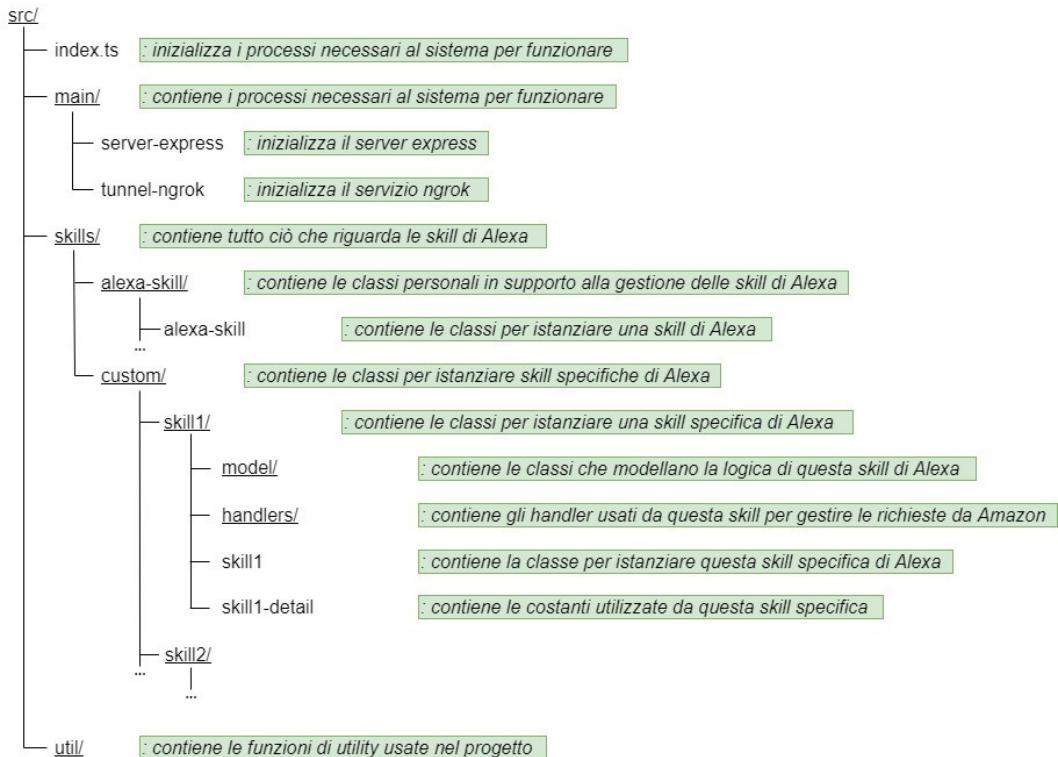


Figura 5.5: L'organizzazione della directory che contiene l'applicazione che gestisce il servizio per le *skill* cognitive.

5.2.2 Funzionamento dell'applicazione

All'avvio, l'applicazione crea il **server Express** utilizzato per accedere al servizio. Questo potrà ricevere le *richieste http* dell'utente, delegandone la gestione ad alcuni *handler*, selezionati in base al *percorso* specificato dall'utente. Successivamente, viene verificata l'esistenza di un **tunnel ngrok** verso la porta del *server Express*: se non è presente, viene avviato un nuovo processo, che eseguirà *ngrok* in modo indipendente dall'applicazione. Il processo *ngrok* dovrà rimanere attivo anche dopo la chiusura dell'applicazione, così non sarà riavviato insieme all'applicazione. Infatti, per inizializzare l'applicazione, è stato utilizzato il processo *Nodemon* per Node.js, che, una volta avviato, rileva ogni aggiornamento sulla directory indicatagli, reagendo con un riavvio automatico dell'applicazione, quindi applicando le modifiche eseguite sul codice.

All'inizializzazione del *server Express*, vengono anche creati i gestori delle richieste http. Questi consistono nelle vere e proprie **skills di Alexa**. Ogni *skill* viene associata al *server Express* su uno specifico percorso, tramite un **Express Adapter**, che si occuperà di validare le richieste http in entrata, controllando che effettivamente provengano da Amazon Alexa.

Una *skill* avrà il compito di gestire tutte le richieste http sul percorso assegnato.

Più in dettaglio, riceverà tutte le *intenzioni* dell’utente che le competono e le smisterà verso gli opportuni **intent handlers**. Ogni *intent handler* gestirà una specifica intenzione dell’utente, interagendo con il **modello della skill**, che contiene le classi che descrivono la logica del gioco cognitivo implementato. Al termine dell’esecuzione, un *intent handler* produrrà un’opportuna risposta, che sarà comunicata mediante il dispositivo Alexa.

I progressi di un relativo giocatore durante il gioco, vengono memorizzati sulle variabili di sessione della *skill*, chiamate **session attributes**. La sessione di una certa *skill* è relativa all’interazione con uno specifico utente e viene creata all’apertura della *skill*, permanendo fino alla sua chiusura. Specificamente, è mantenuta all’interno dei messaggi scambiati tra l’utente e il servizio.

Di seguito, viene riportato uno schema che riassume i componenti descritti precedentemente (Figura 5.6).

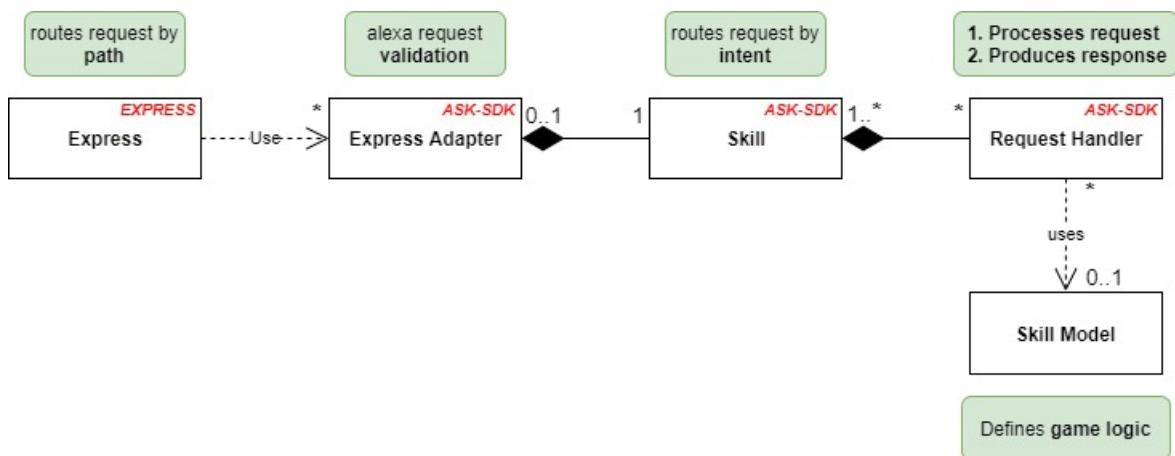


Figura 5.6: Schema che descrive i componenti principali dell’applicazione in base alla loro funzione.

Al termine dell’avvio, il *server Express* mostrerà le caratteristiche del servizio, come mostrato in Figura 5.7 (da notare che il caricamento del server impiega molto tempo quando il servizio *ngrok* non è già in esecuzione).

Siccome l’indirizzo pubblico del proprio servizio cambia a ogni esecuzione di *ngrok* (nella licenza gratuita), è necessario ricordarsi di mantenere aggiornati gli **endpoint** utilizzati dalle *skill*. Gli **endpoint** sono modificabili dal modello di dialogo delle *skill*, accessibile dall’*Alexa Skills Kit* dell’*Amazon Developer Console*. Nella figura, questi endpoint hanno gli indirizzi URL mostrati nella sezione *LOADED SKILLS*.

Una volta finita la configurazione, il servizio è pronto per essere acceduto da un qualsiasi dispositivo Alexa, che sia associato a un account Amazon in cui sono installate le *skill*. Siccome non è ancora stato eseguito il deployment delle *skill*, queste saranno accessibili solo dall’account che le sta sviluppando.

```

PS C:\[REDACTED]\Tesi\alexa-project> cd .\server_node\
PS C:\[REDACTED]\Tesi\alexa-project\server_node> npx nodemon
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): src\**\*
[nodemon] watching extensions: ts
[nodemon] starting `npx ts-node ./src/index.ts`

-----
SERVER DETAILS
Server listening on http://localhost:3001
[server-status]: NOW_RUNNING (pid:5384)

-----
NGROK DETAILS
Ngrok listening on https://84dcfa2c8745.ngrok.io
[ngrok-status]: NOW_RUNNING (pid:8364)

-----
LOADED SKILLS
[Number List Game]: https://84dcfa2c8745.ngrok.io/skill/number-list-game
[Category Game]: https://84dcfa2c8745.ngrok.io/skill/category-game
[Labyrinth Game]: https://84dcfa2c8745.ngrok.io/skill/labyrinth-game

-----
OUTPUT
Loading took 11.059s...
[REDACTED]

```

Figura 5.7: Output del *server Express* dopo la sua inizializzazione. Di particolare importanza è la sezione *LOADED SKILLS*.

5.2.3 Esempio d’implementazione di una skill

Ora sarà illustrato un esempio su come è possibile implementare una *skill*, raggiungibile attraverso un server realizzato con *Express*.

Inizialmente, è necessario configurare il **server Express** (Figura 5.8). Ciò significa creare le **skills** che dovrà gestire e mapparle a un determinato percorso nel server. Siccome, su quei percorsi si vuole accettare solo richieste provenienti da Alexa, viene utilizzato un **Express Adapter**, il cui scopo è modificare gli *handler* di una *skill*, in modo che eseguano una verifica sulla sorgente della richiesta, preliminarmente alla sua gestione. Il *server Express* si occuperà di smistare le richieste in entrata in base al percorso da loro indicato, delegandole alla *skill* opportuna.

La creazione di una *skill* (Figura 5.9) avviene attraverso uno **Skill Builder**, che ne determina gli *handler*. Lo *Skill Builder* utilizzato è un’estensione di quello fornito dalla libreria **ASK-SDK** e distingue gli *handler* in due categorie:

- Gli *handler standard*, che sono molto simili tra le varie *skill*, quindi si prestano bene a essere raggruppati in un **template**, applicabile su *skill* diverse. Ad esempio, sono considerati standard gli *handler* indicati da Amazon come necessari al deployment di una *skill*;
- Gli *handler custom* che caratterizzano una specifica *skill*, in base alle funzionalità che offre.

```

/* SERVER CREATION: create the application server, which will handle alexa skills. */
const app = express();

/* SKILL CREATION: create the skills that will be handled by this server, assigning each a path. */
const skillRouters : AlexaRouter[] = [
    new AlexaRouter(NumberListGameDetails.registeredPath, new NumberListGameSkill()),
    new AlexaRouter(CategoryGameDetails.registeredPath, new CategoryGameSkill()),
    new AlexaRouter(LabyrinthGameDetails.registeredPath, new LabyrinthGameSkill())
];

/* SKILL LOADING: prepare the server to resolve alexa requests. */
skillRouters.forEach( (skillRouter) => {
    app.post(
        '/' + env.skill_path_base + '/' + skillRouter.getPath(),
        new ExpressAdapter(skillRouter.getSkillWrapper().getSkill(), true, false).getRequestHandlers()
    );
});

/* SERVER START: start the server. */
app.listen(env.port);

```

Figura 5.8: Inizializzazione del *server Express*. Include la creazione delle *skill* e la loro mappatura sul server.

```

export class LabyrinthGameSkill extends AlexaSkill {
    constructor(){ super(LabyrinthGameDetails.skillName); }

    public getSkill() : Alexa.Skill {
        return new AlexaSkillBuilder()
            .fromTemplates(new NamedDefaultTemplate(this.getName(), {
                launchRequestHandler: LGLaunchRequestHandler,
                stopIntentHandler: LGStopIntentHandler,
                helpIntentHandler: LGHelpIntentHandler,
                helloWorldIntentHandler: LGHelloWorldIntentHandler,
                fallbackIntentHandler: LGFallbackIntentHandler
            }))
            .addRequestHandlers(
                StartGameHandler,
                MovePlayerHandler,
                TellSurroundingsHandler,
                TellDistanceHandler,
                TutorialHandler,
                DefaultHandlerFactory.createIntentReflectorHandler()
            ).create();
    }
}

```

Figura 5.9: Inizializzazione di una *skill*. Qui sono inizializzati tutti gli *intent handler* utilizzati dalla *skill*.

Ogni *skill* si occuperà di gestire le richieste su di uno specifico percorso del server in base all'intenzione espressa dall'utente, delegandole all'*handler* opportuno.

Gli *handler* di una *skill* sono caratterizzati da due metodi principali (Figura 5.10). Una funzione, chiamata ***canHandle***, permette alla *skill* di verificare se l'*handler* può gestire una certa richiesta. Una volta verificato ciò, può essere applicata la seconda funzione, chiamata ***handle***, che gestisce la richiesta. Gli *handler* interagiranno con il modello della *skill*, controllando il progresso nel gioco, la cui logica è definita nel modello stesso. Il progresso nel gioco sarà memorizzato nei parametri della sessione attiva tra l'utente e la *skill* specifica.

```
export const StartGameHandler : Alexa.RequestHandler = {
    canHandle(handlerInput : Alexa.HandlerInput) {
        return handlerInput.requestEnvelope.request.type === 'IntentRequest' &&
            handlerInput.requestEnvelope.request.intent.name === Skill.registeredIntents.startGame;
    },
    handle(handlerInput : Alexa.HandlerInput) {
        const input : AlexaInputHelper = new AlexaInputHelper(handlerInput);
        input.clearSessionAttributes();

        let difficulty : number = Number.parseInt(input.resolveCustomSlotValue(Skill.slotValueNames.difficulty).getOrDefault("1"));
        input.addSessionAttribute(
            'labyrinth',
            new LabyrinthGame(Skill.env.difficultyToSize.get(difficulty).getOrDefault(4))
                .printMazeLayout()
                .printSolvedMaze()
        )
        .addSessionAttribute('currentTurn', 0)
        .addSessionAttribute('from', Skill.registeredIntents.startGame);
    }
}
return TellSurroundingsHandler.handle(handlerInput);
```

Figura 5.10: Esempio d’implementazione di un *intent handler*. Si notano i caratteristici metodi di un intent handler e l’interazione con il modello della skill (ovvero la classe *LabyrinthGame*).

Normalmente, un *handler* produce anche una risposta, che sarà comunicata dal dispositivo Alexa. Nell’esempio però, la generazione della risposta è delegata a un altro *handler* della *skill*.

Capitolo 6

Validazione del sistema

In questo capitolo, si descriveranno i metodi utilizzati per validare l'applicazione, ovvero i test che sono stati eseguiti per verificarne la correttezza e l'adeguatezza rispetto i requisiti preposti.

6.1 Metodi di validazione

Al tempo della scrittura di questo elaborato, viste anche le limitazioni legate all'emergenza *COVID-19*, non era ancora stato possibile eseguire delle prove che coinvolgessero il target dell'applicazione. Questi test sugli utenti finali sono necessari ancor di più nel contesto della *gamification*, quindi saranno sicuramente eseguiti in futuro.

Per il momento, nondimeno, sono stati eseguiti numerosi *friendly-trial* in ambiente chiuso all'interno del CRA di Bologna. Nello specifico, sono state eseguite delle prove che verificassero:

- La **possibilità di controllare i dispositivi domotici attraverso OpenHAB**; per questo è stata utilizzata l'interfaccia grafica costruita, accertandosi che a un comando dell'utente corrispondesse una reazione adeguata del sistema domotico (ad esempio, in riferimento alle figure 5.2 e 5.3, alla pressione del pulsante *light-switch*, la lampadina *lightbulb1* si deve accendere o spegnere);
- La **possibilità di controllare i dispositivi domotici con dei comandi vocali attraverso Alexa**; per questo è stata utilizzata la skill di OpenHAB attraverso Alexa, accertandosi che a un comando vocale dell'utente corrispondesse una reazione adeguata del sistema domotico (ad esempio, in riferimento alle figure 5.2 e 5.3, dicendo ad Alexa *"Accendi / Spegni la Lampadina Hue"*, la lampadina *lightbulb1* si deve accendere o spegnere);
- La **capacità di OpenHAB di notificare l'utente di certi eventi relativi ai dispositivi domotici connessi**; per questo sono stati generati degli eventi

nel sistema domotico, accertandosi che questi fossero comunicati correttamente attraverso Alexa (ad esempio, in riferimento alla figura 5.4, Alexa deve comunicare quando la lampadina *lightbulb1* viene accesa o spenta);

- L'***adeguatezza delle skill nell'esercitare le capacità cognitive dell'utente***; per questo si è lasciato giocare alcuni utenti che non fossero direttamente coinvolti nell'implementazione delle *skill*, prendendo atto delle recensioni e del feedback ricevuto, così da effettuare cambiamenti che ne migliorassero l'usabilità.

Il funzionamento del sistema è stato testato da alcuni membri del personale del CRA di Bologna. Nello specifico, i risultati ottenuti sul sistema domotico e sulle *skill* di Alexa sono stati approvati da alcuni componenti del team multidisciplinare del CRA, tra cui la neuropsicologa *Dott.ssa Arianna Gherardini*, e i correlatori *Ing. Massimiliano Malavasi*, coordinatore del CRA, e *Dott.ssa Laura Bugo*, sviluppatrice software.

6.2 Risultati ottenuti

Di seguito, saranno riportati e commentati i risultati ottenuti dai test eseguiti sul sistema domotico e sulle *skill* implementate.

6.2.1 Il sistema domotico

Il funzionamento del sistema domotico si è rilevato adeguato e soddisfacente. Tuttavia, è possibile fare alcuni commenti sulle responsività constatate nelle diverse interazioni rese disponibili all'utente.

L'*interazione attraverso l'interfaccia grafica* è quella con minor latenza. Ciò ha senso se si tiene conto del fatto che i comandi inviati tramite interfaccia grafica sono inviati a OpenHAB senza intermediari. Maggiore è invece la latenza delle *interazioni vocali attraverso Alexa*. In questo caso, i comandi devono essere interpretati dal cloud di Alexa, per questo trascorrono un paio di secondi prima che l'utente possa osservare un riscontro sul sistema domotico. Considerando che lo scopo dell'integrazione con Alexa era in primo luogo di fornire ulteriori strumenti accessibili per controllare il proprio sistema domotico, i tempi di risposta ottenuti sono stati valutati come accettabili. Altrettanta latenza si è stata constatata nella *ricezione delle notifiche* sugli eventi accaduti nel sistema domotico. Questa funzione era stata pensata per ricordare all'utente alcuni compiti da svolgere, a orari prestabiliti, in base allo stato del sistema domotico (come ricordare all'utente di chiudere le finestre d'inverno prima di dormire, se ce n'è qualcuna aperta). Qualche secondo di differenza rispetto a tali orari, non influenza il risultato finale, quindi anche in questo caso si è ritenuto il sistema soddisfacente.

6.2.2 Number List Game

Riguardo alla *skill* Number List Game, sono stati ricevuti alcuni feedback costruttivi, soprattutto riguardo ad alcuni problemi di progettazione. Nel complesso, è stato un buon primo tentativo d'implementazione di una *skill* per Alexa.

I feedback ricevuti riguardano soprattutto il flusso di dialogo. Infatti, la *skill* richiede una conferma ogni volta che l'utente invia la risposta al quesito precedentemente comunicato da Alexa. Considerando che, nelle interazioni con una *skill*, ovvero con il cloud, esiste anche una certa latenza, ciò può diventare abbastanza tedioso per il giocatore. Nelle *skill* progettate successivamente, si è dunque deciso di snellire il modello di dialogo, raggiungendo gli stessi risultati attraverso meno interazioni.

6.2.3 Category Game

La *skill* Category Game ha ricevuto feedback molto positivi. Non significa però, che non possano ancora essere migliorati alcuni suoi aspetti.

Il problema principale è la mancanza di contenuti. In effetti, la *skill* necessita di una maggiore varietà di categorie e di parole conosciute: capita spesso che sia richiesto all'utente di categorizzare la stessa parola in un breve periodo di tempo, oppure di fare partite consecutive in cui siano coinvolte le stesse categorie. Siccome le categorie e le parole sono definite in modo statico, una soluzione potrebbe essere l'aggiornamento dei file che le contengono, o ancora meglio, se possibile, richiederle a servizi terzi che già gestiscono dati simili.

6.2.4 Labyrinth Game

Labyrinth Game è stato un esperimento che ha avuto abbastanza successo. In fase di progettazione, si pensava infatti che sarebbe stato troppo difficile per l'utente seguire il flusso di dialogo del gioco e pertanto interagire con la *skill*. Tuttavia, si è scoperto invece che ci si abitua velocemente alla struttura con cui si riferiscono le informazioni nel gioco. Dunque, già all'interno della prima partita, le difficoltà iniziali vengono riscontrate sempre meno.

Questi risultati sono stati ottenuti dopo alcune fasi di raffinamento, in cui lo scopo principale era di bilanciare il dialogo tra l'utente e Alexa: in questo tipo di gioco, l'utente riceve molte più informazioni rispetto a quelle che fornisce. Per agevolare la concentrazione dell'utente, si è quindi cercato di essere il più possibile concisi e schematici nel riferirgli le informazioni necessarie per giocare, così da rendere il dialogo più interattivo.

Conclusioni

In questo elaborato si è cercato d'illustrare al meglio le caratteristiche di un sistema che potesse soddisfare i requisiti funzionali preposti. Il risultato è stato nel complesso adeguato e soddisfacente, ma possono essere fatte alcune considerazioni finali.

In primo luogo, il sistema è ancora in fase prototipale e saranno necessari alcuni raffinamenti. Nello specifico, per quanto riguarda il sistema domotico, sarà necessario cercare di migliorarne la reattività. Ciò dipende in gran parte dalle tecnologie utilizzate, quindi si dovrà mantenere il sistema aggiornato, in modo da adottare le nuove funzionalità e i miglioramenti che queste offriranno. Per quanto riguarda le *skill* implementate, sarà invece necessario continuare a perfezionarne il modello di dialogo, rendendo sempre più agevoli e interessanti le interazioni con Alexa.

In secondo luogo, il sistema dovrà essere verificato non solo in ambiente chiuso, ma anche attraverso il coinvolgimento di un campione del target indicato. Questo risulterà necessario per dimostrare l'effettiva adeguatezza del sistema, soprattutto per gli esercizi cognitivi implementati.

In futuro, il progetto potrà essere esteso ed essere integrato in un contesto socio-sanitario, nel quale potrà sperabilmente assistere persone con disabilità o difficoltà cognitive a raggiungere una maggiore indipendenza.

In conclusione, si spera che questo progetto possa essere ultimato, producendo tutti i risultati desiderati, e che possa concretamente assistere altre persone, offrendo loro l'opportunità di ottenere una maggiore libertà.

Ringraziamenti

Ringrazio,

la mia famiglia e i miei genitori, per avermi supportato in questo percorso, permettendomi di proseguire con gli studi e offrendomi questa opportunità per crescere, ma anche per avermi insegnato a essere indipendente e sostenuto nella mia ricerca d'indipendenza;

i miei amici, per le esperienze e le conoscenze condivise, i loro diversi punti di vista e le discussioni anche scherzosamente animate, ma sicuramente costruttive;

i membri del team del CRA di Bologna, per avermi ospitato calorosamente, aiutato a integrarmi nell'ambiente di lavoro e formato negli argomenti trattati dalla tesi; in particolare, il tutor aziendale e correlatore della tesi, Ing. Massimiliano Malavasi, e la correlatrice, Dott.ssa Laura Bugo, che mi hanno seguito nello sviluppo del progetto; inoltre, la neuropsicologa Dott.ssa Arianna Gherardini, per avermi formato e assistito nella progettazione dei giochi cognitivi;

i professori universitari, per offrire l'opportunità d'intraprendere questo tipo di percorso didattico e formativo; in particolare, il professore Alessandro Ricci, che mi ha introdotto alla gestione della comunicazione tra sistemi diversi e poi seguito nella stesura di questa stessa tesi.

Grazie a tutti voi.

Bibliografia

- [1] Azienda Unità Sanitaria Locale di Bologna.
URL: <https://www.ausl.bologna.it/>.
[Ultimo accesso: 03-07-2021].
- [2] Centro Regionale Ausili di Bologna.
URL: <https://www.ausilioteca.org/cra>.
[Ultimo accesso: 03-07-2021].
- [3] VIMAR. Che cos'è la domotica?
URL: <https://www.vimar.com/it/it/che-cos-e-la-domotica-13772122.html>.
[Ultimo accesso: 03-07-2021].
- [4] Impianto domotico KNX (Konnex).
URL: <https://knx.it/chi-siamo/>.
[Ultimo accesso: 03-07-2021].
- [5] Dispositivi smart della Philips.
URL: <https://www.philips-hue.com/en-us/explore-hue/works-with>.
[Ultimo accesso: 03-07-2021].
- [6] OpenHAB - Homepage.
URL: <https://www.openhab.org/>.
[Ultimo accesso: 06-07-2021].
- [7] Wikipedia Contributors. Storia degli assistenti virtuali.
URL: https://en.wikipedia.org/wiki/Virtual_assistant#Experimental_decades:_1910s%E2%80%941980s.
[Ultimo accesso: 03-07-2021].
- [8] Treccani. Definizione di Gamification.
URL: https://www.treccani.it/vocabolario/gamification_%28Neologismi%29/.
[Ultimo accesso: 08-07-2021].

- [9] Blohm I. e Leimeister J.M. Gamification. Business & information systems engineering, 2013 - Springer
URL: <https://link.springer.com/content/pdf/10.1007/s12599-013-0273-5.pdf>.
[Ultimo accesso: 08-07-2021].
- [10] L'importanza del potenziamento cognitivo nell'invecchiamento sano.
URL: <https://www.stateofmind.it/2017/06/potenziamento\protect\discretionary{\char\defaulthyphenchar}{}{}cognitivo\protect\discretionary{\char\defaulthyphenchar}{}{}invecchiamento/>.
[Ultimo accesso: 05-07-2021].
- [11] Prioritizzazione MoSCoW.
URL: https://en.wikipedia.org/wiki/MoSCoW_method.
[Ultimo accesso: 03-07-2021].
- [12] Amazon Alexa.
URL: <https://developer.amazon.com/en-US/alexa>.
[Ultimo accesso: 03-07-2021].
- [13] Servizio di tunnelling ngrok.
URL: <https://ngrok.com/>.
[Ultimo accesso: 03-07-2021].
- [14] Documentazione di OpenHAB.
URL: <https://www.openhab.org/docs/>.
[Ultimo accesso: 03-07-2021].
- [15] OpenHAB - Things.
URL: <https://www.openhab.org/docs/configuration/things.html>.
[Ultimo accesso: 05-07-2021].
- [16] OpenHAB - Items.
URL: <https://www.openhab.org/docs/configuration/items.html>.
[Ultimo accesso: 05-07-2021].
- [17] OpenHAB - Sitemaps.
URL: <https://www.openhab.org/docs/ui/sitemaps.html>.
[Ultimo accesso: 05-07-2021].
- [18] OpenHAB - Pages.
URL: <https://www.openhab.org/docs/ui/layout-pages.html>.
[Ultimo accesso: 05-07-2021].

- [19] OpenHAB - Bindings.
URL: <https://www.openhab.org/addons/>.
[Ultimo accesso: 05-07-2021].
- [20] OpenHAB - Rules.
URL: <https://www.openhab.org/docs/configuration/rules-dsl.html>.
[Ultimo accesso: 05-07-2021].
- [21] OpenHAB - Persistence.
URL: <https://www.openhab.org/docs/configuration/persistence.html>.
[Ultimo accesso: 05-07-2021].
- [22] AWS Lambda di Amazon.
URL: <https://aws.amazon.com/lambda/>.
[Ultimo accesso: 03-07-2021].
- [23] Amazon Developer Console.
URL: <https://developer.amazon.com/settings/console/home>.
[Ultimo accesso: 03-07-2021].
- [24] Alexa Skills Kit di Amazon.
URL: <https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html>.
[Ultimo accesso: 03-07-2021].
- [25] ASK Software Development Kit per Node.js.
URL: <https://developer.amazon.com/en-US/docs/alexa/alexa-skills-kit-sdk-for-nodejs/overview.html>.
[Ultimo accesso: 03-07-2021].
- [26] Amazon Echo Control Binding per OpenHAB.
URL: <https://www.openhab.org/addons/bindings/amazonechocontrol/>.
[Ultimo accesso: 03-07-2021].
- [27] Amazon Alexa Smart Home Skill - OpenHAB.
URL: <https://www.openhab.org/docs/ecosystem/alexa/>.
[Ultimo accesso: 03-07-2021].
- [28] Philips Hue Binding per OpenHAB.
URL: <https://www.openhab.org/addons/bindings/hue/>.
[Ultimo accesso: 03-07-2021].
- [29] Typescript Documentation.
URL: <https://www.typescriptlang.org/docs/handbook/>

- typescript-from-scratch.html.
[Ultimo accesso: 03-07-2021].
- [30] Node.js.
URL: <https://nodejs.org/en/>.
[Ultimo accesso: 03-07-2021].
- [31] Express for Node.js.
URL: <https://expressjs.com/>.
[Ultimo accesso: 03-07-2021].