

UNIVERSITÀ DEGLI STUDI DI PERUGIA
Dipartimento di Matematica e Informatica

CORSO DI LAUREA MAGISTRALE IN INTELLIGENT AND MOBILE COMPUTING



Tesi di Laurea

Car diagnostics con Amazon Alexa

Laureando:

Dominici Alex

Relatore:

Prof. Osvaldo Gervasi

Dr. Damiano Perri

Dr. David Berti

Anno Accademico 2020–2021

Ai miei genitori

Indice

Introduzione	ix
1 Human Machine Interface	1
1.1 Voice User Interface	13
1.2 Standard guidelines	15
1.3 Migliorare la User Experience di bordo	18
2 Assistanti vocali intelligenti	22
2.1 Cosa sono gli assistenti vocali?	22
2.1.1 L'evoluzione dell'assistenza vocale nei veicoli	25
2.2 L'assistenza vocale oggi	31
2.3 La tecnologia dell'assistenza vocale	34
2.4 Tipologie di assistenti vocali	41
3 Alexa	49
3.1 Alexa Auto SDK	52
3.1.1 Architettura di Alexa Auto SDK	53
3.2 Alexa Skills Kit	55
3.2.1 Software e strumenti di sviluppo	60
3.2.2 Processo di sviluppo	64

4	Funzionamento dell'applicazione	78
4.1	Struttura	79
4.2	Fase di Autenticazione	82
4.3	Panoramica delle pagine	83
4.4	DynamoDB	92
4.4.1	Connessione tra applicativo e DynamoDB	94
4.4.2	Connessione tra custom Skill e DynamoDB	97
4.4.3	Alternativa: MongoDB	98
4.5	Casi d'uso	100
	Conclusioni	110
	Bibliografia	112

Elenco delle figure

1.1	Car HMI	2
1.2	Lamborghini Countach 25th Anniversary, modello del 1989; foto scattata il 3 Febbraio 2015 © Thesupermat. Immagine sotto licenza Creative Commons Attribution-Share Alike 4.0 International.	4
1.3	Lamborghini Countach LPI 800-4, modello del 2021; © Au- tomobili Lamborghini S.p.A.	5
1.4	Working Memory Model	8
1.5	esempi di Chunking delle parole e dei numeri di telefono . .	9
1.6	Tasso di mortalità per 100 mln di miglia percorse dal veicolo	10
1.7	Modello dell'esperienza utente, J.D.Power	13
1.8	Adobe Voice Technology Study, July 2019	14
1.9	Statistica del desiderio per lo stesso servizio in macchina . .	19
1.10	Statistica della soddisfazione con gli assistenti vocali	20
2.1	Storia degli assistenti vocali (prima parte).	24
2.2	Storia degli assistenti vocali (seconda parte).	25
2.3	Storia degli assistenti vocali in auto (prima parte).	27
2.4	Storia degli assistenti vocali in auto (seconda parte).	29

2.5	Il sistema LIS.	30
2.6	Storia degli assistenti vocali in auto (terza parte).	31
2.7	Come vengono utilizzati gli assistenti vocali dentro l'auto oggi.	33
2.8	Familiarità con le funzionalità dell'assistente vocale.	34
2.9	Funzionamento Assistenti Vocali	35
2.10	Funzionamento Speech Recognition.	37
2.11	Funzionamento Voice Recognition.	38
2.12	NLP e NLU a confronto.	39
2.13	Assistente vocale ibrido	42
2.14	Assistenti vocali usati in auto	44
2.15	Assistenti vocali offerti dalle Case Automobilistiche	45
3.1	Logo Amazon Alexa	49
3.2	Schema di funzionamento di Alexa	51
3.3	Architettura di Alexa Auto SDK	53
3.4	Sample App	55
3.5	Flusso di invocazione di una Skill	57
3.6	Alexa Developer Console: Build page	60
3.7	Alexa Developer Console: Test page	61
3.8	Alexa Developer Console: Endpoint page	63
3.9	Intents di Car Maintenance	66
3.10	Sample utterances di getCarStatus	67
3.11	Sample utterances di GetComponentStatus	68
3.12	Riempimento del custom slot	69
3.13	Flusso conversazione di diagnostica	72
3.14	Esempio richiesta inviata da Alexa	74

3.15	Handler di GetComponent	75
3.16	Risposta in formato JSON	77
3.17	SkillBuilder	77
4.1	QT Creator	79
4.2	Struttura interfaccia grafica, HomePage	81
4.3	Flusso di avvio dell'autorizzazione CBL	83
4.4	Data Flow tra prodotto e AWS	84
4.5	Weather Page	85
4.6	Media Page	86
4.7	Clima Page	88
4.8	Diagnostic Page	90
4.9	Information Card in Diagnostic Page	91
4.10	Solution Card in Diagnostic Page	92
4.11	Tabella DynamoDB	94
4.12	Configurazione client DynamoDB	95
4.13	Update tabella DynamoDB	96
4.14	Configurazione permessi Lambda Function	97
4.15	Policy applicata ruolo	98
4.16	Metodi per leggere la tabella del DynamoDB	99
4.17	Interfaccia autenticazione utente	100
4.18	Login utente	101
4.19	Caso d'uso di Autenticazione	102
4.20	Interfaccia controllo generale auto	103
4.21	Caso d'uso Skill Car Maintenance	103
4.22	Interfaccia ricerca locale	104

4.23 Caso d'uso ricerca locale	105
--	-----

Acronimi

AI Artificial Intelligence.

AWS Amazon Web Services.

GUI Graphical User Interface.

H2M Human-To-Machine.

HCI Human-Computer Interaction.

HDI Human-Device Interaction.

HMI Human Machine Interface.

ISO International Organization for Standardization.

LVC Local Voice Control.

NHTSA National Highway Traffic Safety Administration.

NLP Natural Language Processing.

SDK Software Development Kit.

STT Speech To Text.

TRL Transport Research Laboratory.

TTS Text To Speech.

UX User Experience.

VCS Voice Control System.

VR Voice Recognition.

VUI Voice User Interface.

Introduzione

La presente tesi punta a delineare come le Voice User Interface (VUI) stiano ricoprendo un ruolo fondamentale nel graduale processo di miglioramento dell'esperienza utente (UX) a bordo veicolo. In particolare, ne introduce il concetto di Human Machine Interface (HMI), approfondendo le VUI, e ne ripercorre in primo luogo le fasi che, storicamente, da primi esperimenti embrionali hanno portato allo stato dell'arte degli assistenti vocali oggi disponibili a bordo. Vengono poi analizzate le relative problematiche, individuate le linee guida esistenti per lo sviluppo e marcate le differenze tra assistenti vocali integrati e cloud-based. In secondo luogo, viene proposto un esempio degli attuali punti di forza degli assistenti vocali intelligenti, prendendo in esame un applicativo basato sul servizio Amazon Alexa, presentandone alcuni casi d'uso che permettano di eseguire un servizio di diagnostica dell'auto e di assistenza al conducente. Tali servizi non sono presenti di base in Alexa ma possono essere integrati attraverso lo sviluppo delle Skills. Successivamente viene descritto DynamoDB, database di cui è proprietaria Amazon (tramite il suo AWS) che ha permesso di rendere accessibili le informazioni relative ai componenti del veicolo. In terzo luogo, sulla base degli assunti e dei casi d'uso proposti, vengono tracciate alcune potenziali prospettive e tendenze future che le VUI apriranno in ambito di UX a bordo vettura.

Capitolo 1

Human Machine Interface

Con Human Machine Interface (HMI) ci si riferisce all'interfaccia utente di un determinato dispositivo che permette alle persone di interagire con i device; tale termine è comunemente utilizzato nel contesto di un processo industriale, ma in questo caso ci si focalizzerà più nel settore automobilistico. Ad oggi il "disegnare una HMI" ha come scopo il mettere insieme caselle e tasselli di un grande puzzle, dentro cui rientrano non solo le più classiche interfacce fisiche, quali potrebbero essere componentistiche fisiche e hardware di interazione con l'abitacolo, si pensi a tasti fisici sul volante adibiti alla gestione degli impianti di climatizzazione o di fruizione dei media e radio disponibili nell'impianto di bordo; si parla anche di interfacce che tendono ad "ibridare" l'esperienza utente, basate su touchscreen, componentistica a feedback aptico, nonché interfacce di assistenza vocale all'utente (VUI). Una indagine di Cisco [1] rivela che l'industria automobilistica non sta soddisfacendo la domanda dei consumatori per quanto riguarda la tecnologia. Nel rapporto si evince anche che i consumatori sono propensi a scambiare le proprie informazioni personali come altezza, peso e abitudini di guida



Figura 1.1: Car HMI

se ciò consentisse un'esperienza di guida più personalizzata. Con la graduale elettrificazione del parco auto globale, la nuova mobilità stimola le crescenti aspettative dell'utenza in termini di personalizzazione, ottimizzazione funzionale degli spazi, sicurezza, riduzione dei costi di accesso ai servizi di trasporto. Questo avviene poiché le persone sono diventate sempre più dipendenti dai propri dispositivi e si aspettano che la tecnologia venga incorporata in altre parti della loro vita garantendo continuità esperienziale. In un mercato della componentistica di base sempre più globalizzato e normato, dove le differenze tecnologiche si assottigliano gradualmente anno dopo anno, ecco che creatività, design e interattività diventano i caratteri distintivi su cui le Case Automobilistiche giocano la vera sfida della differenziazione dai competitor e del posizionamento nel mercato.

È in tal senso che l'HMI diventa gradualmente portatrice del carattere distintivo di ciascun Marchio. L'HMI salvaguarda l'identità comunicati-

va, il "DNA" del Costruttore, ed accoglie - fidelizzando - gli utenti grazie all'ecosistema di applicazioni e funzionalità a bordo vettura.

Dal secolo scorso ad oggi, l'automobile ha mantenuto invariata la sua struttura fondamentale: ciò l'ha resa un bene universalmente accessibile, alla portata di tutti; a catalizzare un mutamento profondo nell'ultimo quarantennio, in senso squisitamente tecnologico, è stato l'avvento dell'autronica, di cui i Sistemi di Infotainment e i navigatori GPS sono risultati emblematici. Tale fenomeno ha motivato una crescente attenzione sull'interazione fra l'individuo e il "sistema di bordo", aprendo un dibattito - tuttora in corso nello scenario globale dell'Industria Automobilistica - su complessità, modelli cognitivi, sovraccarico cognitivo, e definendo se e come sia possibile individuare un punto d'incontro "ideale" nell'equazione della mobilità che tenga conto anche delle esigenze e delle aspettative degli utenti.

Negli interni delle auto di una volta, come è possibile riscontrare in Figura 1.2, la disposizione spaziale dei quadranti, selettori, potenziometri e vari altre tipologie di organi di regolazione e di comando fisici permetteva di sviluppare nel tempo una mappa mentale dell'HMI e di incorporarla nella memoria muscolare. Ciò era reso possibile poiché c'erano meno elementi meccanici ed elettronici, come manopole e interruttori, i quali fornivano sia funzione di controllo che feedback diretto; di conseguenza, l'interfaccia utente - grazie alla sua tangibilità - possedeva una semplicità intrinseca.¹

Con la moderna HMI (vedi Figura 1.3) invece, ci sono molti altri elementi all'interno del veicolo ed esiste un mix di interfacce grafiche che si comporta come la principale fonte di feedback per diverse funzioni. Tutto

¹Thesupermat, Lamborghini Countach 25th Anniversary, [Immagine sotto licenza Creative Commons Attribution-Share Alike 4.0 International.], <https://bit.ly/3sgCK3N>



Figura 1.2: Lamborghini Countach 25th Anniversary, modello del 1989; foto scattata il 3 Febbraio 2015 © Thesupermat. Immagine sotto licenza Creative Commons Attribution-Share Alike 4.0 International.



Figura 1.3: Lamborghini Countach LPI 800-4, modello del 2021; © Automobili Lamborghini S.p.A.

ciò ha aumentato esponenzialmente la complessità cognitiva, non consentendo la costruzione di un'unica mappa mentale da seguire. Adesso passare da una modalità all'altra, capire dove sono i controlli e quali sono le relative funzioni risulta essere meno intuitivo. Oggi se una persona cambia il modello dell'auto, a differenza di quello che succedeva negli anni passati, deve imparare quasi da zero i controlli base del sistema. Con l'avvento delle nuove funzionalità all'interno del HMI, le persone si trovano a relazionarsi con un livello di complessità discretamente alto e questo svolge un ruolo faticosamente impattante nel processo decisionale a causa della natura limitata nella memorizzazione e nell'accesso ai dati dentro le nostre menti. ²

Per capire meglio i limiti cognitivi delle persone vengono analizzate come queste interagiscono con il mondo, attraverso la formazione di ricordi di breve o lunga durata e l'accesso a tali ricordi con il passare del tempo. A tal proposito viene citato il "Working Memory Model" di Baddeley e Hitch [2], il quale descrive appunto l'interazione umana attraverso i sistemi di:

1. Memoria a lungo termine detta "memoria cristallizzata";
2. Memoria a breve termine detta "memoria fluida";
3. Un contenitore episodico, detto "episodic buffer", che ha capacità limitata.

Nella memoria cristallizzata (o memoria a lungo termine) troviamo il *linguaggio*, la *semantica visuale* - i.e. il riconoscimento delle forme - e la *memoria muscolare* - i.e. una forma di memoria che permette il consolidamento

²Automobili Lamborghini S.p.A., Lamborghini Countach LPI 800-4, <https://cdn.carbuzz.com/gallery-images/1600/883000/600/883639.jpg>

di un'attività specifica attraverso la sua ripetizione. Sono esempi di questa tipologia di memoria molte attività quotidiane, come l'andare in bicicletta, praticare uno sport o ricordare una password.

La memoria fluida (o memoria di breve termine) comprende la gestione dell'*Episodic Buffer* (attento a captare informazioni visuo-spaziali, consente di tenere traccia del mondo fisico in cui ci troviamo sulla base di altri oggetti nell'ambiente che ci circonda), delle informazioni visuo-spaziali temporaneamente immagazzinate (*Visuo-spatial Sketchpad*) e del *Phonological loop* (i.e. le parole che pronunciamo); il phonological loop può essere considerato una sorta di "nastro", continuamente scritto e sovrascritto, su cui le tracce dei suoni permangono per pochi secondi. Infine abbiamo L'esecutivo centrale (*Central-executive*) che è la componente più importante del modello. Esso è responsabile del coordinamento dei sistemi e li mette in relazione con la memoria a lungo termine, decidendo a quali informazioni prestare attenzione e a quali parti della memoria recapitarle.

Possiamo considerare l'attività della guida come appartenente al sistema cristallizzato, visto che utilizza la memoria muscolare, mentre le informazioni di attività secondarie possono essere considerate come parte dell'*episodic buffer*. Avendo questo ultimo una capacità limitata, se l'HMI ha un livello di complessità alto può rischiare di sovraccaricarlo facilmente. Questo porta a prendere decisioni sbagliate da parte del conducente, nonché ad una frustrazione dovuta all'eccessivo carico cognitivo.

Gli studi sull'HMI hanno ipotizzato diversi metodi chiave per affrontare tale problema del sovraccarico cognitivo. Uno dei primi metodi fu proposto da George Miller nel 1956 tramite il famoso articolo: "The Magical Number

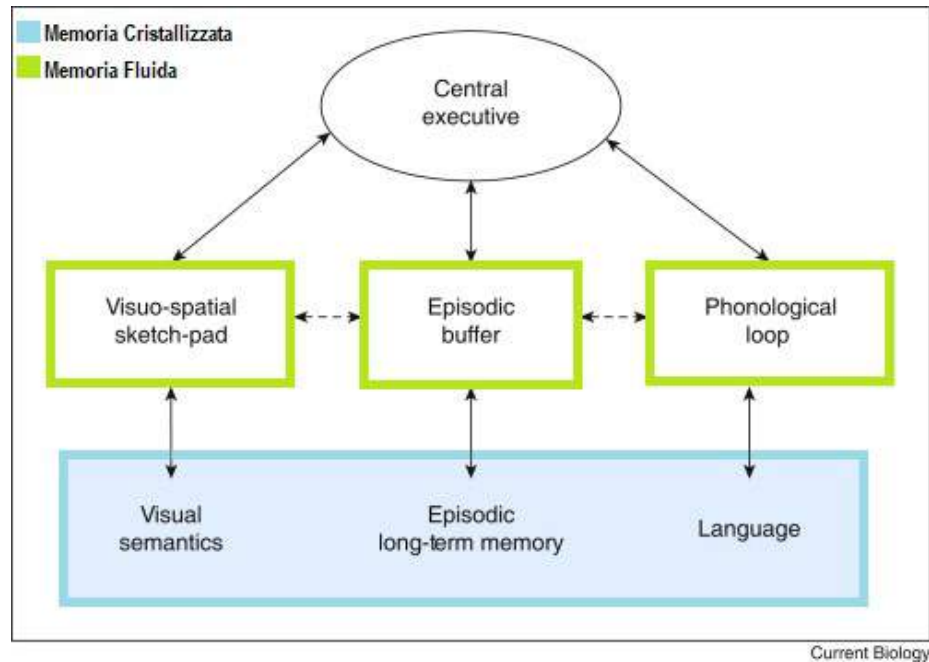


Figura 1.4: Working Memory Model

Seven, Plus or Minus Two" [3]. Questo venne utilizzato successivamente proprio da Baddeley per ipotizzare il *Phonological loop* presente nel Working Memory Model citato precedentemente e si domandò quale fosse l'utilità di questa memoria a breve termine, arrivando a definirla come un'area dedicata ad attività come l'aritmetica mentale, il ragionamento e la risoluzione dei problemi. Nel presente articolo il metodo utilizzato sfrutta il concetto del "*chunking*", cioè raggruppare in blocchi, e viene spiegato che un soggetto è in grado di ricordare ed elaborare solo sette più o meno due blocchi di informazioni nel corretto ordine proposto e quindi essenzialmente raggruppando tali informazioni risulta più facile la loro memorizzazione visto che il raggruppamento aiuta la memoria a breve termine. Sono stati forniti diversi esempi di questa metodologia come i numeri di cellulare che suddividendoli

in gruppi la loro memorizzazione risulta semplificata; oppure raggruppando le parole in categorie diverse, vedi Figura 1.5. Tuttavia, il valore sette più o

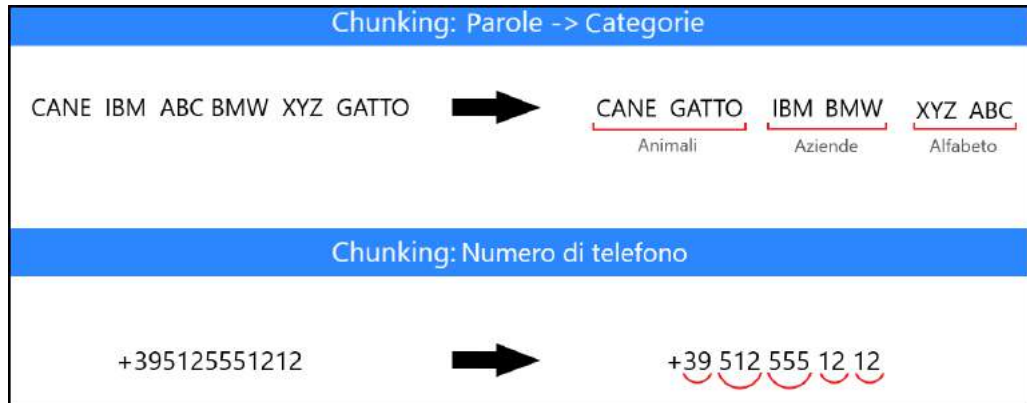


Figura 1.5: esempi di Chunking delle parole e dei numeri di telefono

meno due, come misura della memoria a breve termine, è una leggenda metropolitana [4], applicabile solo a chi parla inglese cercando di ricordare una sequenza di cifre. Le prestazioni effettive della memoria umana dipendono da molti fattori e non possono essere approssimate da un valore numerico.

Dunque, la complessità dell'HMI nelle auto moderne, con i limiti della cognizione umana, portano ad un sovraccarico cognitivo per i conducenti e causano potenziali distrazioni. Sulle strade degli Stati Uniti [5] durante il 2019 ci sono state 36.096 persone uccise in incidenti stradali automobilistici, come mostrato in Figura 1.6. Ciò rappresenta una diminuzione del 2% rispetto ai 36.835 decessi del 2018. Il numero stimato di feriti invece è aumentato nel 2019 a 2,74 milioni con un aumento dell'1,1%. Lo stesso vale per il numero di incidenti segnalati dalla polizia in Italia che è aumentato a 6,76 milioni nel 2019, con un aumento dello 0,3%. In Italia, come riportato dall'Istat [6], nel 2019 sono stati registrati 172.183 incidenti stradali

con lesioni a persone. Numero in lieve calo rispetto al 2018, con una diminuzione del 4,8% di morti. Ci sono stati dunque meno decessi nel 2019 rispetto al 2018 in diverse categorie, tuttavia un dato risulta essere aumentato particolarmente, ovvero la mortalità in incidenti causati da distrazioni del conducente i quali sono aumentati del 10%, passando da 2.585 nel 2018 a 3.142 nel 2019. Molti studi hanno dimostrato che l'eccessiva interazio-

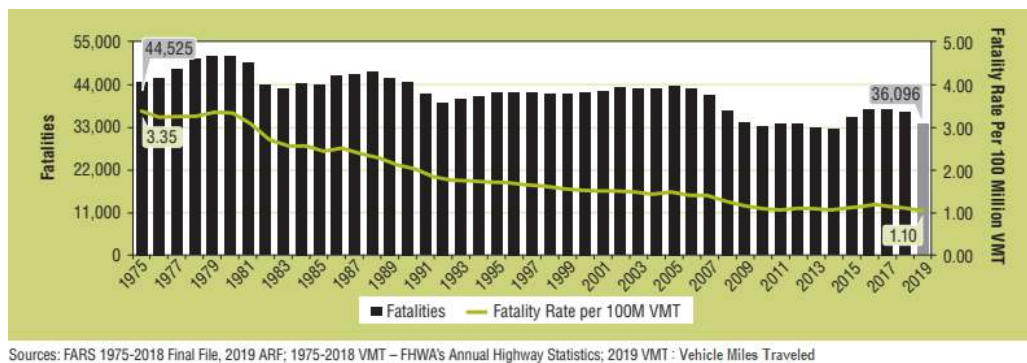


Figura 1.6: Tasso di mortalità per 100 mln di miglia percorse dal veicolo

ne con il sistema di infotainment compromette l'attenzione del conducente nell'attività di guida. Nel 2012, la National Highway Traffic Safety Administration (NHTSA) ha pubblicato uno studio [7] per valutare la distrazione visiva e manuale misurando il comportamento dello sguardo. L'agenzia statunitense classifica tre tipi di distrazione del conducente: *distrazione visiva*, *distrazione manuale* e *distrazione cognitiva*. In caso di *distrazione visiva*, il conducente distoglie lo sguardo dalla carreggiata per ottenere informazioni o per controllare qualche dispositivo, ad esempio controllando e regolando il GPS o cambiando stazione radio. La *distrazione manuale* si verifica quando un guidatore toglie una mano dal volante, ad esempio per mangiare, bere o fumare. Infine, la *distrazione cognitiva* è definita come il carico di lavoro

mentale associato a un compito che implica pensare a qualcosa di diverso dalla guida. Tuttavia, lo studio non ha considerato alcun metodo per rilevare la distrazione cognitiva. In realtà, questa è un tipo di distrazione più difficile da affrontare, perché si verifica all'interno del cervello del conducente. Tale fenomeno, a volte indicato come "guardare ma non vedere", non è lo stesso della distrazione visiva. I risultati di questo studio [7] hanno supportato l'ipotesi che durante la guida con un alto livello di carico cognitivo il movimento involontario degli occhi diventa anomalo. L'alto livello di carico di lavoro mentale, soprattutto mentre si svolge un'attività secondaria come rispondere al telefono cellulare o interagire con il display di bordo, può portare ad una perdita del controllo della vettura, con annessi tutti i rischi che ne conseguono. Infatti, l'80% degli incidenti, sono causati dalla distrazione del conducente da un compito secondario [8]. Su questi dati l'Industria dell'Auto può individuare importanti sfide per i propri attori di progetto - ingegneri e designer, in primis - nel ridurre la driver distraction. Ogni progetto, dopotutto, è opportunità di dialogo e scambio fra i Team progettuali; e oggi, soprattutto chi studia, ricerca e progetta in ambito HMI è chiamato a realizzare soluzioni efficaci che agevolino l'interazione fra features del veicolo e conducente mitigando più possibile le difficoltà di apprendimento. Con User Experience [9], in questo caso, si intende la relazione tra il conducente e il veicolo, comprendendo tutto ciò che ruota attorno a questa interazione. In generale la User Experience racchiude tutte le forme affettive, esperienziali e attribuisce il senso e il valore che ha il prodotto o il servizio per l'utente, ma include anche le sensazioni personali su aspetti come l'utilità, la semplicità nell'utilizzo e l'efficienza del sistema. Il design e lo sviluppo

partono dal comportamento dell'utente, dalle sue necessità e da quali capacità e limiti possiede, permettendo così di valorizzare non solo il prodotto ma l'esperienza che vive l'utente, poiché soggettiva. Per aiutare a definire gli elementi di ciò che crea una buona esperienza utente, J.D. Power [10] - un'azienda americana di analisi dei dati - ha creato un modello composto da quattro attributi chiave: *comprensibilità*, *usabilità*, *fiducia* e *utilità*, vedi Figura 1.7. Ciascun elemento del modello è interdipendente dagli altri. I prodotti che hanno un'esperienza utente forte saranno considerati forti in ogni quadrante; viceversa, esperienze più deboli porteranno a una causa principale di debolezza in una o più aree del quadrante. La definizione di esperienza utente è cambiata nel tempo e continuerà a cambiare poiché legata alle molteplici e complesse dinamiche del quotidiano. L'attributo della *fiducia* è la percezione che ha il consumatore dell'accuratezza del prodotto e quindi la sua capacità di iniziare a costruire una relazione con esso. L'*utilità* invece misura se un prodotto vale o meno il tempo o il denaro speso. Questa misura culmina nella disponibilità del consumatore a volere di nuovo il prodotto in futuro e a raccomandarlo.

L'ISO 9241 è lo standard che descrive lo sviluppo di questi sistemi e mira a renderli usabili dagli utenti. Questa ISO punta a normare l'interazione Uomo-Macchina in generale, coprendo quindi tutte le interfacce possibili, mentre per quanto riguarda l'esperienza utente all'interno del veicolo non sono stati trovati standard specifici.

Un report, pubblicato dal *Transport Research Laboratory* (TRL) [11], mostra che i livelli di distrazione del conducente sono molto più alti quando si utilizzano le tecnologie touchscreen rispetto ai sistemi ad attivazione

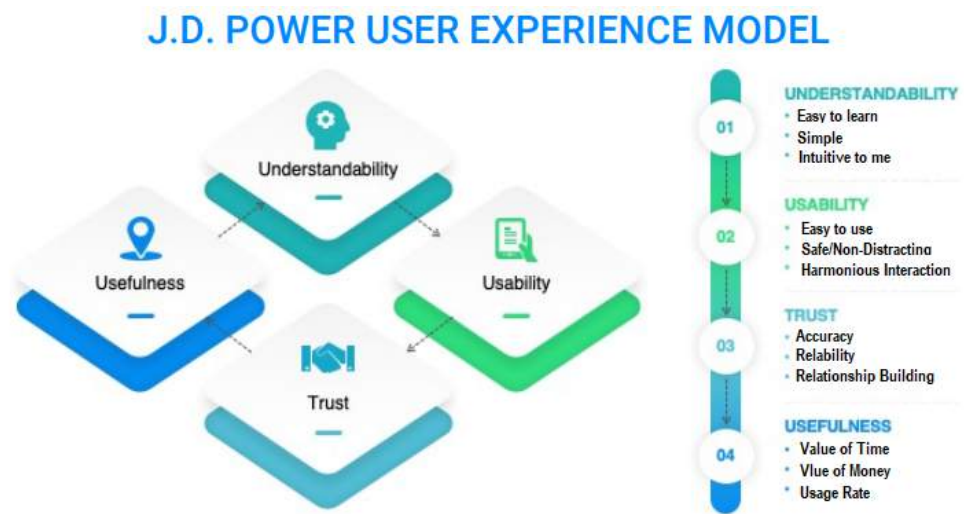


Figura 1.7: Modello dell'esperienza utente, J.D.Power

vocale. Nei prossimi capitoli verrà mostrato come l'introduzione delle VUI ha migliorato in parte l'esperienza utente a bordo vettura.

1.1 Voice User Interface

La Voice User Interface (VUI) consente l'interazione da parte di un utente con un sistema attraverso il riconoscimento vocale per comprendere i relativi comandi e in genere riprodurre una risposta tramite la sintesi vocale. Esempi di prime VUI possono essere le segreterie telefoniche che chiedono di dire nome e cognome per il riconoscimento dell'utente, oppure navigatori di qualsiasi genere che forniscono un output vocale ma non permettono una vera interazione. Queste interfacce oramai sono presenti nelle nostre case, automobili e elettrodomestici, a dimostrazione del fatto che stanno acquisendo popolarità già da diversi anni e sono considerate il principale metodo

per interagire con gli assistenti vocali come Siri, Google Assistant e Alexa. Il loro vantaggio risiede nel fatto che permettono di avere occhi e mani libere mentre si sta eseguendo un'attività secondaria, mantenendo quindi la concentrazione per l'attività principale.

Nel 2019 è stato stimato che il 48% dei consumatori utilizza la voce per eseguire delle ricerche online e che il 44% utilizza le interfacce vocali in qualche modo ogni giorno [12]. In Figura 1.8 viene mostrato che la maggior parte degli utenti utilizza questa tecnologia tramite il proprio smartphone, anche se l'utilizzo tramite gli altoparlanti e il sistema multimediale della vettura sia in crescita. Dunque, è in questo scenario globale che interfacce

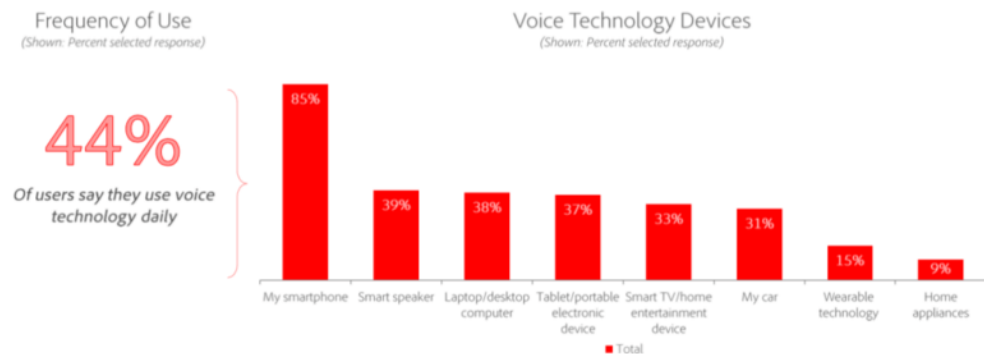


Figura 1.8: Adobe Voice Technology Study, July 2019

vocali (VUI) e interfacce grafiche (GUI) si contendono il miglior posto per un'esperienza utente sempre più fluida e appagante. Per semplici attività, come ottenere indicazioni stradali o eseguire dei controlli, le persone preferiscono parlare invece di scrivere. Tuttavia, l'esperienza deve funzionare perfettamente; infatti quasi sempre alle tecnologie vocali vengono associate tecnologie tattili per migliorare l'esperienza. Qualsiasi dispositivo potrebbe dotarsi di un'interfaccia vocale, ma ciò non significa che dovrebbero averla

tutti. I dispositivi che vengono utilizzati in pubblico riscontrano problemi nella variazione della voce e nel rumore ambientale. Inoltre, quando si ha a che fare con dati sensibili, l'ultima cosa che vuole l'utente è parlare ad alta voce. Quindi le VUI trovano spazio per quelle attività che richiedono le mani libere, come cucinare o guidare, dispositivi intelligenti per la casa e applicativi che hanno una componente emotiva.

Per molte persone, gli assistenti vocali sono ancora sinonimo di smart speaker, come l'echo di Alexa. Questo è dato dalla forte influenza di Amazon e Google, nonché dalla diffusa adozione dei loro prodotti. Tuttavia, la voce non è un'esclusiva degli altoparlanti ma attraversa una vasta gamma di dispositivi. Un esempio sono proprio le auto. Secondo uno studio condotto da VoiceBot.AI [13], ci sono più utenti attivi che utilizzano un assistente vocale in macchina che sugli smart speaker. L'utilizzo dell'assistente vocale a bordo vettura ha consentito ai conducenti di vivere un'esperienza a mani libere sicura, permettendo di effettuare chiamate telefoniche, controllare la musica e navigare. Dunque, gli assistenti vocali stanno fornendo risposte ad una serie di aspettative completamente nuove da parte dei consumatori riguardo all'esperienza di guida e i produttori di automobili sono in forte competizione proprio per soddisfare queste aspettative e reinventare l'esperienza di guida.

1.2 Standard guidelines

Di conseguenza, uno dei principali obiettivi delle politiche di tutto il mondo è stato proprio quello di ridurre al minimo tali distrazioni, limitando l'utilizzo di dispositivi come display o interfacce mobili, i quali sono la colonna

portante della progettazione dell'HMI. Infatti la Commissione Europea [14] ha emesso dei principi sull'HMI di bordo, affermando che il sistema deve essere progettato per supportare il conducente e non deve dar luogo a comportamenti potenzialmente pericolosi. Inoltre l'attenzione che il conducente riserva ai display o ai controlli del sistema deve essere compatibile con l'attenzione richiesta dalla guida e quindi non distrarlo visivamente. Più recentemente la National Highway Traffic Safety Administration ha pubblicato delle proprie linee guida per la progettazione e lo sviluppo di HMI sulla base delle attuali conoscenze delle capacità e dei limiti del conducente. Tali direttive sostengono che il conducente dovrebbe essere in grado di tenere almeno una mano sul volante durante l'esecuzione di un'attività secondaria e che comunque questa ultima può essere interrotta in qualsiasi momento. Non deve essere lasciato al sistema il controllo del ritmo dell'interazione tra le diverse attività ma deve essere stabilito dal conducente.

Entrando un po' più nello specifico, viene descritto come la posizione di un display visivo sia un fattore chiave che influenza la facilità con cui i conducenti possono ottenere informazioni [15]. I design tradizionali dei cruscotti stanno cambiando man mano che diventano disponibili nuovi sistemi di avviso e informazione; un posizionamento appropriato della componente visiva di questi sistemi faciliterà l'accesso alle informazioni riducendo l'impatto sul compito di guida.

Nonostante i vantaggi che possono fornire gli avvisi multimodali, cioè quelli costituiti da più modalità come quella visiva, tattile e uditiva insieme, se questi vengono riprodotti sequenzialmente potrebbero presentare alcuni inconvenienti. Un esempio di avviso multimodale può essere la con-

clusione di una chiamata, premendo il tasto del volante che restituisce un feedback aptico e contemporaneamente l'impianto multimediale riproduce il suono di avvenuta "call ended". Se un conducente sente un avviso sonoro, mentre sta rispondendo ne vede uno visivo, tale avviso ridondante deve ancora essere elaborato e quindi in teoria può aumentare il tempo necessario per rispondere. Lee et al [16] hanno dimostrato che il tempo di reazione della frenata da parte del conducente aumenta con la presenza di un display multimodale. Suggestiscono che le prestazioni possono peggiorare quando un avviso multimodale viene percepito come più segnali, ma è possibile ottenere miglioramenti delle prestazioni abbinando le caratteristiche dei segnali multimodali in modo che l'avviso venga percepito come un singolo segnale.

Applicare le stesse linee guida di progettazione delle interfacce grafiche per l'utente (GUI) alle VUI non è possibile. In queste gli utenti non hanno indicazioni chiare su cosa può effettivamente fare l'interfaccia e quali sono le loro opzioni, data la mancanza di suggerimenti visivi. Risulta essere importante durante la progettazione che il sistema indichi in modo chiaro quali sono le opzioni per interagire con esso, fornendo all'utente un numero adeguato di informazioni da poter ricordare. A volte gli utenti non sono consapevoli della complessità che può comprendere l'interfaccia, poichè associano la voce ad una normale comunicazione, o meglio naturale, piuttosto che ad un'interazione uomo-macchina. Dunque, non è richiesta solo una buona capacità da parte dell'interfaccia di comprendere il linguaggio, ma anche una sorta di addestramento degli utenti per poter capire quali tipi di comandi vocali utilizzare e poter interagire al meglio con la VUI.

Gli utenti abbandonano spesso le interfacce vocali a causa dei problemi

di usabilità che incontrano e dallo scostamento fra le loro aspettative e ciò che forniscono realmente. Ad oggi uno dei principali problemi che la progettazione di VUI deve affrontare è la mancanza di principi condivisi e universali

1.3 Migliorare la User Experience di bordo

Sempre più automobili adottano sistemi di infotainment di bordo, che può essere ad esempio un grande schermo o più schermi. Questi sistemi consentono da una parte di migliorare la mobilità e il comfort del conducente, fornendo le tradizionali funzionalità dell'auto in modo smart come la riproduzione di musica o la navigazione, ma dall'altra parte aumentano il rischio di distrazione durante la guida della vettura. Dunque, abbiamo diverse motivazioni per migliorare l'esperienza utente in auto:

1. Cercare di ridurre il più possibile le distrazioni causate dai display di bordo, sfruttando a pieno le VUI, poiché le statistiche ci dicono che il gran numero degli incidenti sono provocati proprio da queste distrazioni;
2. Spesso il vocal assistant di bordo è limitato e non sufficientemente allenato, rispetto a quelli cloud-based come Alexa. Secondo alcuni dati abbiamo sempre più persone che vorrebbero un'esperienza utente che non finisse una volta usciti di casa ma che per esempio continuasse in macchina;
3. Ancora più intriganti sono i casi d'uso per preordinare il cibo prima di arrivare in un ristorante con servizio rapido o cercare un prodotto

e poi chiedere di navigare verso il luogo più vicino in cui è disponibile. Oppure, considerare come l'intrattenimento e i giochi possono essere adattati all'esperienza dell'assistente vocale in auto.

Come già specificato, i servizi vocali stanno diventando una parte sempre più naturale della vita quotidiana delle persone e le loro aspettative non cambiano quando salgono sul veicolo [17]. Vogliono vivere la stessa esperienza che hanno a casa quando sono in viaggio. Dalla Figura 1.9 è possibile vedere che oltre i due terzi dei consumatori hanno affermato di volere lo stesso assistente vocale che hanno a casa nel loro prossimo veicolo. Quello

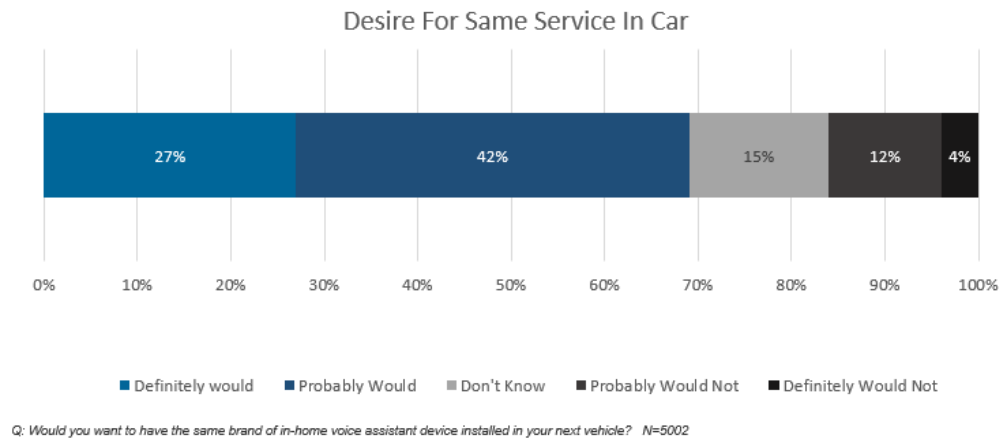


Figura 1.9: Statistica del desiderio per lo stesso servizio in macchina

che permette di motivare questa tendenza, oltre al desiderio di avere un'esperienza coerente fra il vissuto casalingo e quello della guida e quindi di mantenere la personalizzazione insegnata precedentemente all'assistente, è che i consumatori avrebbero già familiarità sul come utilizzare il sistema e le relative funzioni.

Stiamo parlando di miglioramento dell'esperienza utente nei veicoli proprio perché il riconoscimento vocale tradizionale, cioè quello integrato dalle Case Automobilistiche, è stato valutato in modo inferiore rispetto a quelli cloud-based, come mostrato in Figura 1.10. Gli assistenti vocali cloud-

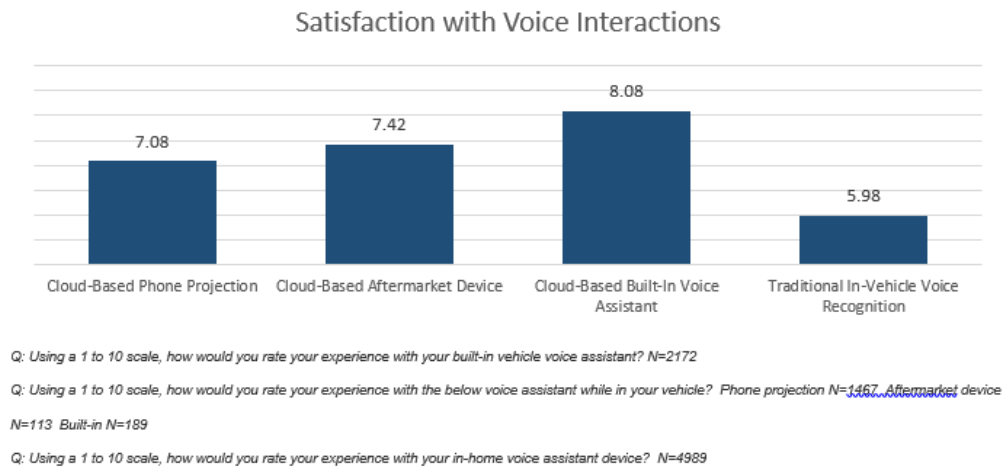


Figura 1.10: Statistica della soddisfazione con gli assistenti vocali

based forniscono un'esperienza che va oltre le funzionalità dei servizi vocali di bordo, rappresentando un'estensione di tale esperienza la quale permette di ottenere una soddisfazione da parte dei consumatori che l'utilizzano. Proprio per questo i servizi vocali cloud-based, come Alexa, sono i preferiti nei veicoli dei clienti.

Dunque, l'integrazione dell'esperienza vocale in auto con la funzionalità domestica e la creazione di un ecosistema a lungo termine creerà un vantaggio duraturo per i consumatori di assistenti vocali per auto.

Infine è possibile dire che la user experience del futuro avrà molto a che vedere con questa tipologia di assistenza, di fatto sono forme di intelligenza che possono essere applicate in qualsiasi supporto abbia la possibilità di

farle funzionare. Questo dimostra che un ecosistema di questo tipo può acquisire ubiquità se progettato bene. La buona progettazione vede come sfida più grande lo sviluppo - a cui sottostà una ampia e profonda comprensione delle dinamiche interrelazionali umane - e la modularizzazione di sistemi tecnicamente più completi possibile sotto il profilo dell'accettabilità e della ricchezza empatica, emotiva e relazionale, da poter essere considerati al pari di un essere umano che assiste l'utente. Si punta ad un rapporto doverosamente paritetico. La sfida è quella di progettare interfacce che siano sì più semplici possibile, più user centered possibile, più accessibili possibile, ma anche complete e affidabili [18].

Capitolo 2

Assistenti vocali intelligenti

2.1 Cosa sono gli assistenti vocali?

Meglio conosciuti come Assistenti Personali Intelligenti, sono agenti software che tramite l'input di comandi o domande sono in grado di eseguire attività e servizi. Nel caso preso in considerazione si parla proprio di assistenti vocali, ovvero quelle applicazioni che sono in grado di interpretare un linguaggio umano così da rispondere tramite voci sintetizzate di sempre crescente verosimiglianza.

Vengono definiti intelligenti poiché sono in grado di simulare l'interazione umana attraverso un sistema di intelligenza artificiale che utilizza le *deep neural networks* e il *machine learning*. Le deep neural networks vengono utilizzate ad esempio per convertire il modello acustico della voce in una distribuzione di probabilità sui suoni del parlato. Questo consente di attivare l'assistente vocale se la probabilità di aver pronunciato una frase di attivazione (wake up word) è abbastanza alta. Gli assistenti vocali utilizzano il machine learning per allenarsi, diventando sempre più utili. Un esempio

del suo utilizzo è la possibilità di insegnare a riconoscere meglio la voce in modo da poter ricevere risultati personalizzati. L'intelligenza artificiale [19] ha la capacità di eseguire compiti comunemente associati agli esseri intelligenti e alla base di questa tecnologia sono presenti l'apprendimento, la comprensione e il ragionamento, i quali sono elementi chiave per emulare una conversazione umana.

Il primo assistente ad attivazione vocale venne rilasciato nel 1922 e fu un giocattolo di legno a forma di cane di nome Rex [20], che quando veniva chiamato con il proprio nome usciva dalla propria cuccia. Seguì poi nei primi anni '60 un altro strumento per il riconoscimento vocale chiamato Shoebox, il quale venne fornito dall'IBM [21] ed era in grado di riconoscere fino a 16 parole pronunciate e le cifre da 0 a 9. Tutto ciò ha posto le basi per permettere che gli assistenti vocali diventassero disponibili per i consumatori, difatti nei primi anni '90 viene prodotto il software di riconoscimento vocale sviluppato da Dragon Systems [22], il quale tra ogni parola pronunciata richiedeva una pausa discreta. Nonostante tale pausa fosse fastidiosa per la maggior parte degli utenti, il software ha guadagnato popolarità, così nel '97 dopo essere stata acquisita da Nuance viene rilasciato un nuovo software chiamato Dragon Naturally Speaking. Esso era in grado di riconoscere e trascrivere in un documento il linguaggio umano evitando le pause tra una parola e l'altra, raggiungendo una velocità di 100 parole al minuto. Questo ultimo ha aperto le porte all'assistente virtuale di Microsoft Office, cioè Clippy [23] il quale ha mostrato come il linguaggio naturale nel testo può essere interpretato e usato come feedback interattivo. Tuttavia, invece di supportare l'utente con una guida chiara e precisa, gli studi han-

no mostrato che Clippy era considerato fastidioso e scortese, comparando nell'interfaccia grafica mentre si lavorava. Passando all'era contemporanea, entra in scena Siri di Apple che è stato il primo degli assistenti vocali a raggiungere un ampio pubblico nel 2011. Ben presto seguirono altri assistenti come Google Now nel 2012 e Cortana di Microsoft nel 2013. Amazon, nel 2014, ha introdotto l'assistente vocale Alexa dando inizio ad una vera e propria rivoluzione degli Smart Speaker, vedi Figura 2.1.

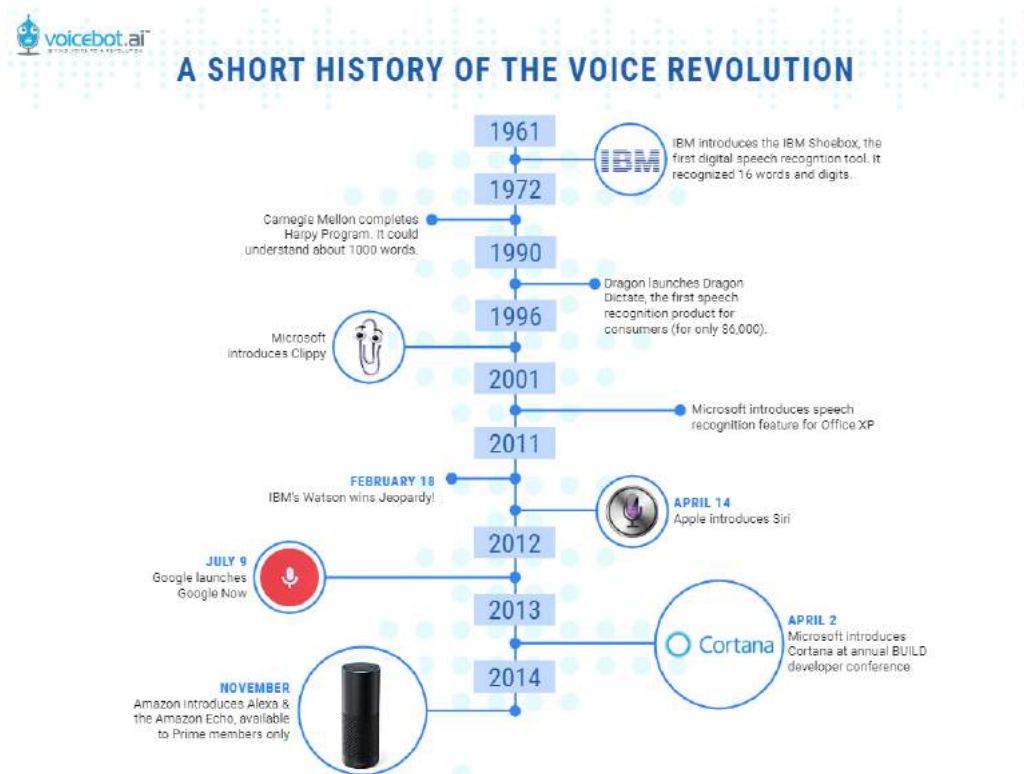


Figura 2.1: Storia degli assistenti vocali (prima parte).

Infine, come è possibile vedere in Figura 2.2, i principali avvenimenti di questi ultimi anni sono l'introduzione nel 2015 da parte di Amazon delle

Skills per Alexa, nel 2016 l'azienda SoundHound lancia l'assistente virtuale *Hound* e nel 2018 Amazon rilascia **Alexa Auto SDK**.

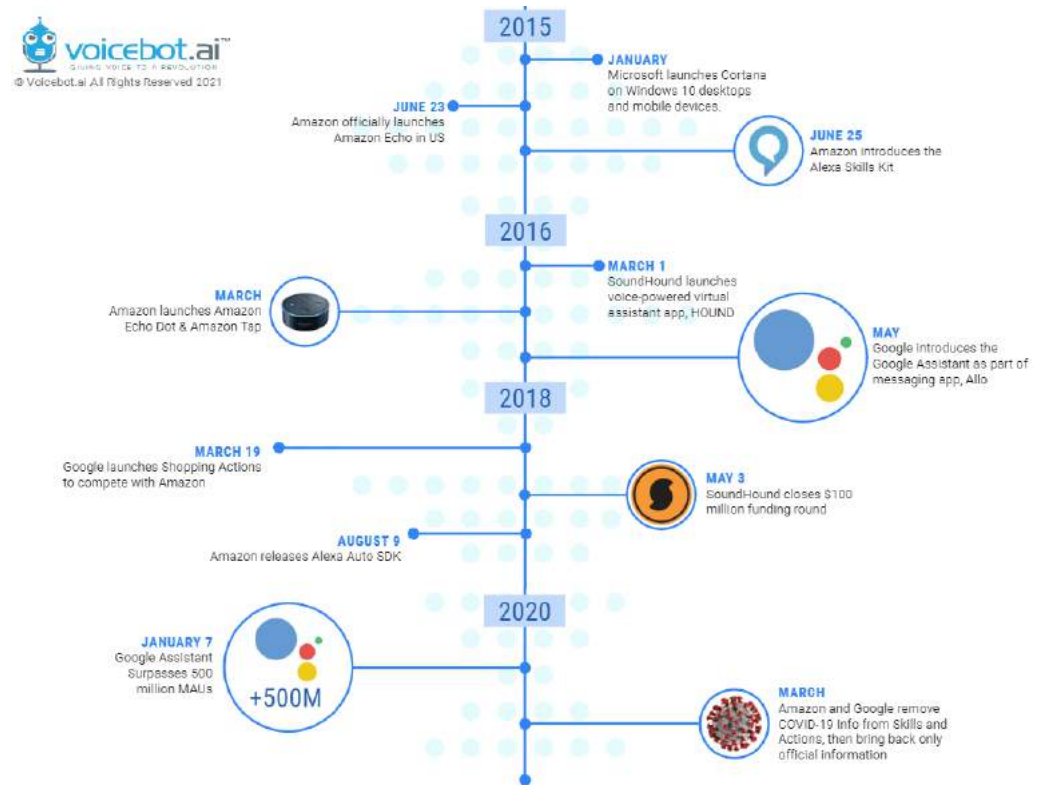


Figura 2.2: Storia degli assistenti vocali (seconda parte).

2.1.1 L'evoluzione dell'assistenza vocale nei veicoli

L'evoluzione del controllo vocale nelle auto è iniziata nel 2004 con Honda, che in collaborazione con IBM, è stata una delle prime Case Automobilistiche ad abilitare un sistema di navigazione vocale che comprendesse funzionalità di comando e controllo vocale per climatizzazione, navigazione e audio. La prima auto con controllo vocale che è arrivata sul mercato è stata la Honda *Acura* del 2005 [24] e per essere attivato era necessario premere un

pulsante situato sul volante. I limiti principali di questo assistente riguardavano la capacità di riconoscere quello che l'utente stava dicendo, quindi era necessario che le parole fossero pronunciate chiaramente, ridurre il rumore di fondo regolando la ventilazione lontana dal microfono e chiudendo i finestrini.

Nel 2007, Ford e Microsoft stabilirono una partnership che portò allo sviluppo della tecnologia *Ford Sync*. Questa ha dato la possibilità ai conducenti utilizzare telefoni dotati di Bluetooth e lettori multimediali attraverso i comandi vocali o tramite i pulsanti del volante. In seguito è stato aggiunto un assistente vocale di nome *Samantha* in grado di inviare e ricevere messaggi di testo. Microsoft [25] ha anche scoperto che per molti consumatori l'importanza della navigazione GPS stava diminuendo, mentre aumentava l'importanza attribuita alla connettività con telefoni e lettori di musica digitale. Ciò aveva senso perché i telefoni avevano più rilevanza quotidiana nella vita della maggior parte delle persone rispetto alle funzioni di navigazione in un'auto. Il telefono era un dispositivo più personalizzato perché aiutava le persone a tenersi in contatto con il lavoro, la famiglia e gli amici. Infine come per Acura, Sync richiede un'ambiente silenzioso per poter funzionare correttamente, vedi Figura 2.3.

Apple, nel 2013, ha introdotto *Siri Eyes-Free* per CarPlay. I conducenti possono utilizzare, sia tramite Bluetooth che collegato tramite USB, le funzionalità di comando vocale di Siri per effettuare e ricevere chiamate, rispondere ai messaggi e altro senza dover togliere le mani dal volante. Ancora una volta, è responsabilità del conducente assicurarsi che l'ambiente sia silenzioso affinché Siri possa funzionare. Nello stesso anno anche la casa

A BRIEF HISTORY OF THE VOICE IN CAR EVOLUTION



2007

Ford partnered with Microsoft to present Sync to the market, which lets drivers interact with mobile devices to make calls, send texts, and control music from a USB-connected MP3 player or Smartphone..

2004

Honda, in collaboration with IBM, was one of the first automobile manufacturers to enable a voice-navigation system that comprised voice command-and-control capabilities for audio, DVD, navigation, and climate control.



Figura 2.3: Storia degli assistenti vocali in auto (prima parte).

automobilistica Škoda ha rilasciato il suo primo sistema di controllo vocale. Tuttavia, non è stato un successo, poiché il vocabolario limitato del sistema e la complessa struttura dei comandi hanno reso gli utenti frustrati. Inoltre, il suo menu richiedeva ai conducenti di leggere le istruzioni, distogliendo lo sguardo dalla strada e sminuendo l'aspetto della sicurezza.

Nel 2017 Lexus introduce la terza generazione di funzioni vocali in auto, ma la necessità di leggere i menu sullo schermo ne ha ostacolato il successo e gli utenti hanno abbandonato tale sistema di controllo vocale. Oltreoceano, la National Highway Traffic Safety Administration, nel suo "In-Vehicle Voice Control Interface Performance Evaluation" del 2016 [26], ha dichiarato che "la complessità del compito, l'accuratezza del sistema di riconoscimento vocale e le interazioni naturali con un VCS (Sistema di Controllo Vocale)" come tre dei principali fattori che inibiscono l'adozione della voce nelle vetture. Sempre nel 2017 Ford ha portato l'integrazione vocale in macchina a un livello superiore, annunciando che avrebbe collaborato con Amazon per poter usare i loro dispositivi per la casa intelligente direttamente dalle proprie automobili. I conducenti potevano chiedere di avviare la vettura quando erano ancora nelle loro case, bloccare e sbloccare le porte da remoto o continuare ad ascoltare gli audiolibri da casa all'auto. Ben presto, più produttori di macchine hanno iniziato a creare i propri assistenti vocali incorporati lavorando con aziende di tecnologia vocale. Ad esempio, Mercedes-Benz ha creato la *Mercedes-Benz User Experience* o MBUX, lavorando con la tecnologia NLP (Natural Language Processing) di *Nuance*. Amazon ha contrastato questa tendenza del controllo vocale incorporato già nelle vetture introducendo *Amazon Echo Auto* nello 2018. Il dispositi-

vo, che ha le dimensioni di un piccolo portafoglio sottile, riconosce la voce dell conducente filtrando i rumori di fondo [27] e funziona con l'app Alexa di uno smartphone tramite l'ingresso ausiliario o la connessione Bluetooth dello smartphone. Ha reso disponibile il controllo vocale a mani libere per i veicoli più vecchi, con o senza Bluetooth, vedi Figura 2.4.

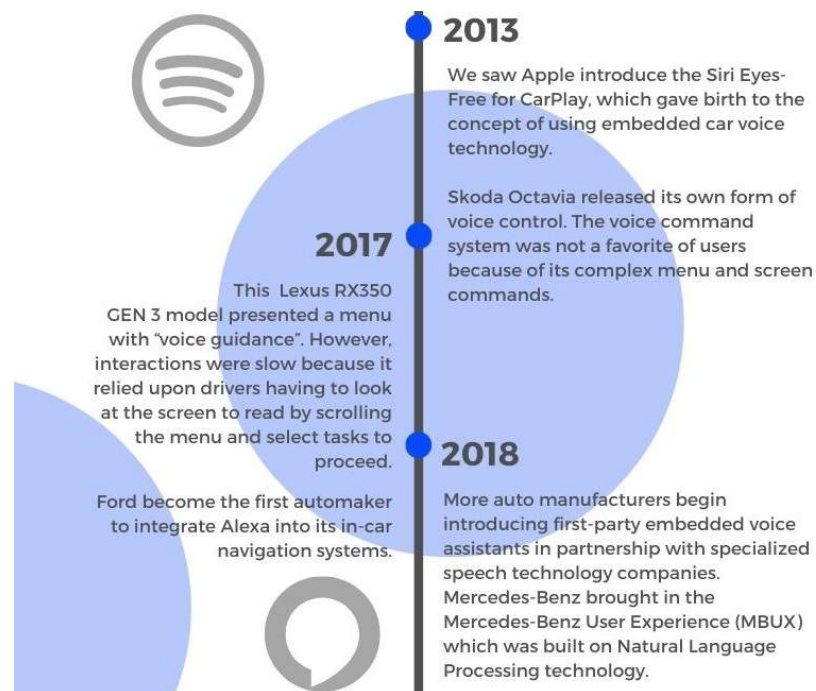


Figura 2.4: Storia degli assistenti vocali in auto (seconda parte).

Nel 2020 Lamborghini, con il suo modello Huracán Evo, è stata la prima casa automobilistica a integrare Alexa per il controllo in auto. Per raggiungere questo obiettivo è stata stipulata una partnership con l'azienda umbra *ART S.p.A.*, che si è occupata della progettazione del sistema di infotainment di Lamborghini (LIS), vedi Figura 2.5. L'integrazione di Alexa garantisce un'esperienza immersiva all'interno del veicolo, un aspetto chiave

nel prossimo futuro. Con il sistema LIS i proprietari di auto migliorano la propria esperienza di guida, potendo accedere direttamente a tutti i servizi Amazon utilizzando il proprio profilo vocale. I conducenti che utilizzano il comando vocale possono regolare la climatizzazione interna, ottenere indicazioni stradali, effettuare telefonate e molto altro che comprende la gestione dell'auto. Inoltre, se connesso ad Alexa, possono interagire con il proprio impianto domotico casalingo.



Figura 2.5: Il sistema LIS.

General Motors, stabilendo una collaborazione con Google, ha annunciato che entro il 2021 mira a integrare Google Assistant e Google Maps per i sistemi di navigazione, i comandi vocali e le altre funzioni di infotainment dei suoi veicoli. Tuttavia, la partnership non sembra limitarsi ai pacchetti Assistant e Maps, visto un ulteriore annuncio di Ford che prevede l'integrazione di un pacchetto Cloud per il 2023, vedi Figura 2.6.

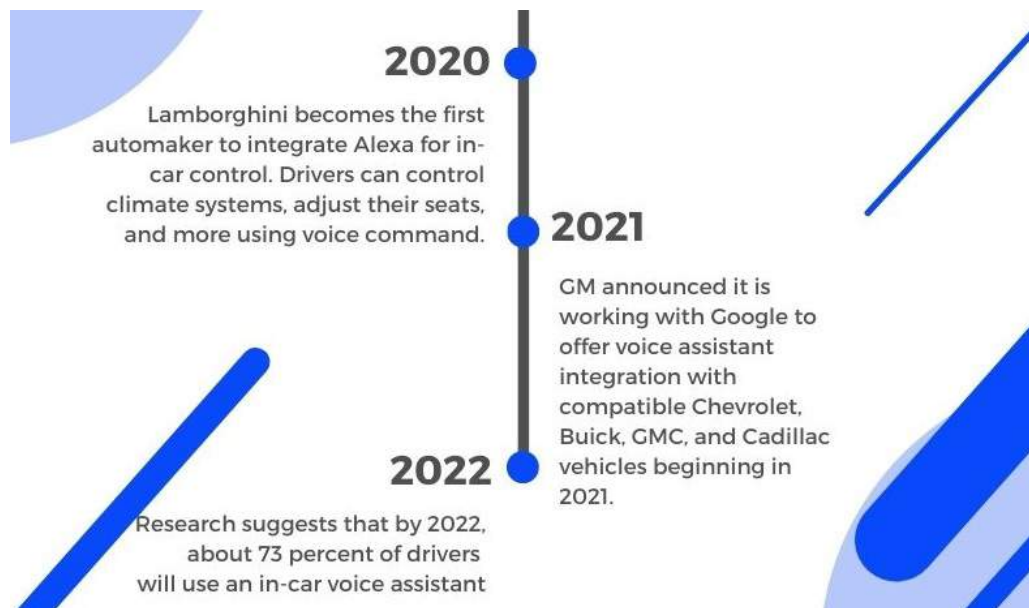


Figura 2.6: Storia degli assistenti vocali in auto (terza parte).

2.2 L'assistenza vocale oggi

La comunicazione con dispositivi che utilizzano la voce è oggi un compito comune a molte persone. Gli Assistenti Vocali Intelligenti, vedi Amazon Alexa, Microsoft Cortana, Google Assistant, o Apple Siri, come già detto sono solo alcuni dei più famosi, integrati in molti dispositivi e non solo. Essi consentono alle persone di cercare vari argomenti, pianificare una riunione o effettuare una chiamata dalla propria auto o casa in vivavoce, quindi non è più necessario tenere in mano alcun dispositivo mobile. Questi assistenti intelligenti utilizzano Voice User Interface per interagire con gli utenti e fornire informazioni su meteo, mappe, orari, chiamate, eventi, ecc. Discipline e ambiti di studio piuttosto recenti come lo *Human-to-Machine* (H2M), *Human-Device Interaction* (HDI) e *Human-Computer Interaction*

(HCI) consentono di integrare VUI, che implica la traduzione delle intenzioni umane nei comandi di controllo dei dispositivi attraverso il riconoscimento vocale. Questi sono il risultato dell'intelligenza artificiale (AI), delle applicazioni ed implementazioni tecniche del natural language processing (NLP), consolidando il concetto di assistenti vocali **intelligenti**. Poiché le persone interagiscono a voce con un numero crescente di dispositivi, la conversazione sta diventando una modalità essenziale di interazione uomo-macchina. I vantaggi non sono solo il controllo vocale, ma anche la natura dialogica delle interazioni, ovvero di come il linguaggio naturale configura discorsivamente ciò che per senso comune viene definito come “realtà”.

Di seguito viene spiegato come sono utilizzati oggi gli assistenti vocali. In un'intervista a *Voicebot* [28], vengono mostrati quali sono i principali casi d'uso degli assistenti vocali in macchina, come si può evincere dalla Figura 2.7. In cima all'elenco c'è l'esecuzione di una *telefonata* seguita dalla richiesta di *indicazioni stradali* o di *navigazione*, *messaggi*, riproduzione di *musica* e *radio*. Questi sono seguiti da domande di *cultura generale* o sul *tempo*. Un caso d'uso a frequenza piuttosto bassa è il *controllo delle funzioni dell'auto* come il riscaldamento e l'aria condizionata. È possibile che i controlli manuali esistenti per queste funzioni siano così comunemente usati che i comportamenti dei consumatori non sono ancora passati alla voce. Tuttavia, è anche probabilmente influenzato dal fatto che molti consumatori utilizzano assistenti vocali che non possono controllare le funzionalità del veicolo. Se ne conclude che gli utenti che possiedono assistenti vocali incorporati hanno quasi il doppio delle probabilità di utilizzarli per i controlli dell'abitacolo, mentre hanno circa la metà delle probabilità di porre una

domanda di carattere generale. Nonostante ci sia stato un crescente uso di

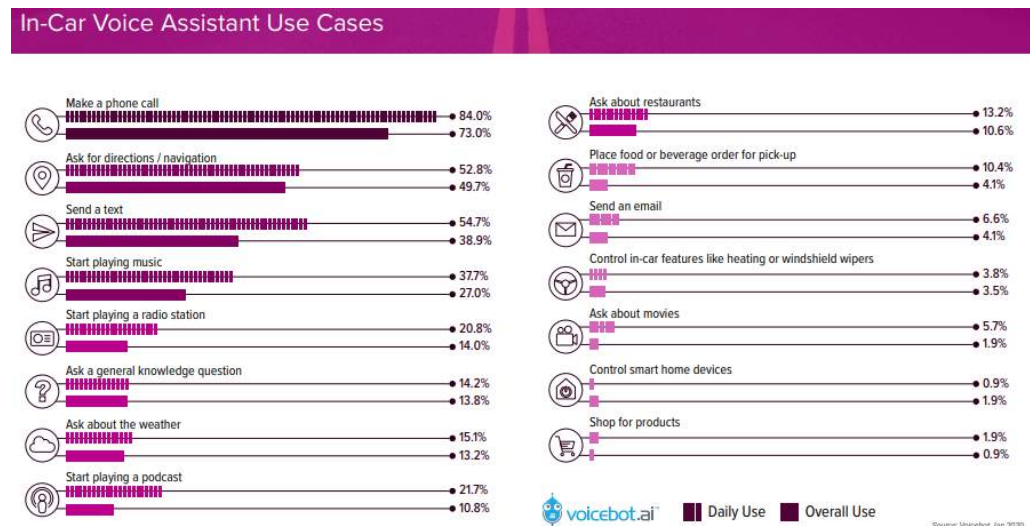


Figura 2.7: Come vengono utilizzati gli assistenti vocali dentro l'auto oggi.

assistenti vocali in macchina, molti utenti non hanno molta familiarità con ciò che le soluzioni integrate possono fare per loro. Come riporta la Figura 2.8 circa il 32% è sicuro di sapere cosa può fare il proprio assistente vocale in macchina e un altro 23,6% crede di conoscere molte funzionalità. Oltre il 35% ritiene di conoscere solo le basi e poco meno del 10% esprime poca conoscenza in materia. Questi risultati suggeriscono che l'utilizzo potrebbe aumentare se i consumatori fossero più istruiti su come gli assistenti vocali possono avere un impatto positivo sull'esperienza di guida. È probabile che vi sia una maggiore consapevolezza in merito a casi d'uso come chiamate, messaggi e navigazione che mostrano un'elevata frequenza di utilizzo.

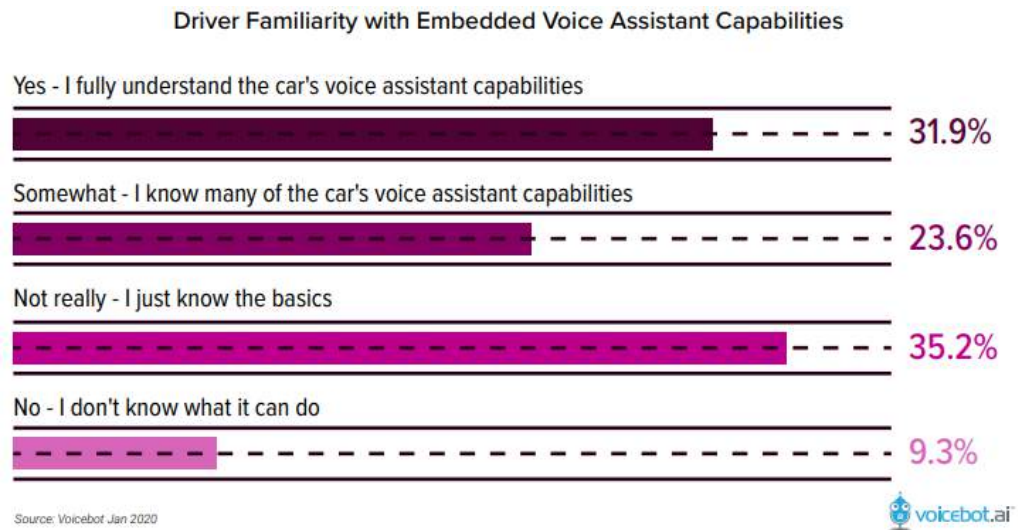


Figura 2.8: Familiarità con le funzionalità dell'assistente vocale.

2.3 La tecnologia dell'assistenza vocale

Con il progredire dell'intelligenza artificiale e dell'apprendimento automatico, gli assistenti vocali [29] adesso sono sistemi in grado di comprendere gli interessi e il comportamento dell'utente, rispondendo di conseguenza. Utilizzano funzionalità basate sull'*intelligenza artificiale* per svolgere azioni in modo proattivo o, come avviene il più delle volte, suggerire contenuti che potrebbero essere interessanti per l'utente basandosi sull'analisi delle scelte effettuate precedentemente. Dunque, usano molti concetti di intelligenza artificiale per tradurre l'interazione umana a mani libere in una serie di comandi comprensibili dal computer. Il software utilizza un microfono per ascoltare la richiesta dell'utente mentre per restituire una risposta verbale viene utilizzato un altoparlante. In questo processo vengono svolte diverse attività, vedi Figura 2.9, come:

1. Convertire le espressioni dell'utente in testo (*Speech Recognition*);
2. Convertire il segnale analogico vocale, ovvero viene registrato e convertito in un file binario (*Voice Recognition*);
3. Classificare le parole e le frasi in strutture semantiche (*Natural Language Understanding*);
4. Interagire e rispondere alle domande dell'utente (*Dialog Management*);
5. Riconvertire il testo generato in voce (*Speech Synthesis*).

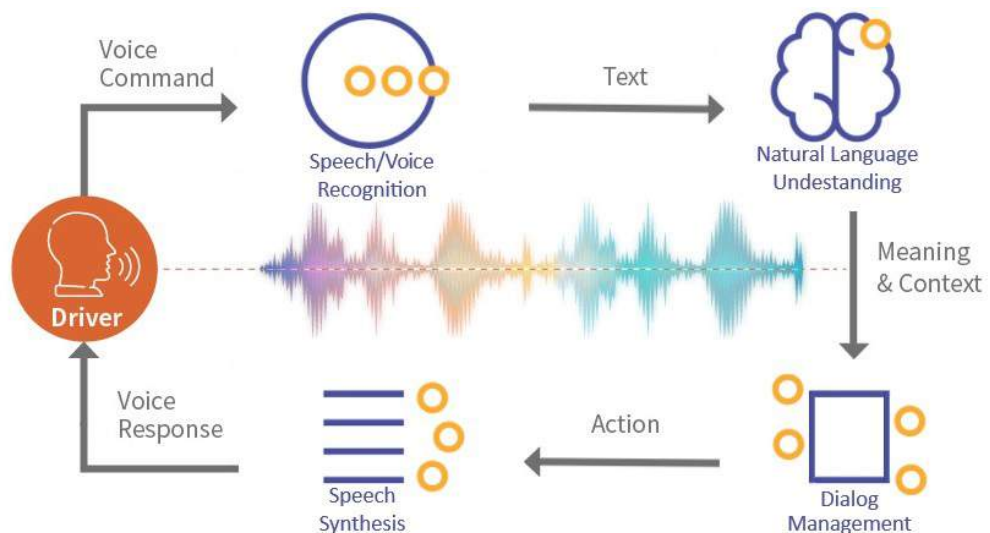


Figura 2.9: Funzionamento Assistenti Vocali

A primo impatto *Speech Recognition* e *Voice Recognition* potrebbero sembrare la stessa cosa, ma in realtà non lo sono. La spiegazione più semplice delle differenze tra i due sono che il primo traduce la voce di chiunque in testo (*SpeechToText*), mentre il secondo comprende la voce di una persona specifica.

Ci sono voluti anni di approfondite ricerche per sviluppare tecnologie di speech recognition utilizzabili nelle odierne interfacce utente vocali (VUI). La Speech Recognition si basa sulla “*feature analysis*”, vedi Figura 2.10. Questo metodo elabora l’input vocale utilizzando un’unità chiamata *phonetic recognition* e trova somiglianze tra gli input previsti e l’effettivo input vocale. In poche parole, abbina il discorso di un utente a generici modelli vocali. Il suo funzionamento effettivo consiste in un microfono che traduce le vibrazioni della voce in un segnale elettrico. Successivamente viene convertito in un segnale digitale, al quale vengono applicati dei filtri per attenuare il rumore. Infine il software di speech recognition analizza il segnale utilizzando un *acoustic model*, il quale permette di registrare unità distinte del suono vocale così da differenziare una parola dall’altra. Un esempio di questa tecnologia sono le piattaforme come Speechmatics o il motore di sintesi vocale di Google. Anche quando vengono impartiti comandi vocali come la riproduzione di musica o l’ottenimento di indicazioni stradali si sta utilizzando speech recognition. Tale tecnologia è risultata determinante nell’aiutare persone diversamente abili come non udenti o con problemi di apprendimento, poiché ha permesso loro di interagire con i media o di utilizzare altre componenti hardware tramite sottotitoli generati automaticamente e ripetitori di testo.

Mentre con Speech Recognition si distinguono quasi tutti i discorsi, a seconda della lingua, degli accenti e così via, con Voice Recognition si considera la capacità di una macchina di identificare la voce di un utente specifico, vedi Figura 2.11. Questo si basa su un processo chiamato *template matching* il quale identifica il passaggio fondamentale nella processing pipeline in cui

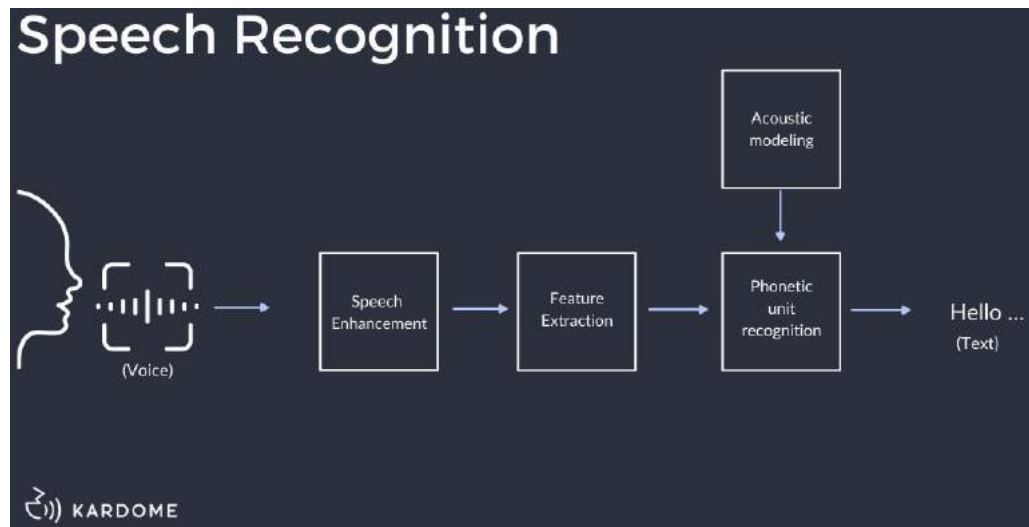


Figura 2.10: Funzionamento Speech Recognition.

avviene un confronto a livello di sistema fra un "pattern" input recepito dai microfoni di sistema e il template che il sistema ha in precedenza salvato nella propria memoria. In questo caso il template rappresenta proprio il modello della voce dell'utente. Successivamente un programma deve essere allenato a riconoscere la voce dell'utente. Quindi verrà mostrata una parola o una frase che l'utente dovrà ripetere più volte in modo da poter allenare il software di voice recognition. A questo punto viene calcolata una media statistica di più campioni della stessa parola o frase. Infine viene memorizzato il campione medio come modello nella struttura dati. L'utilizzo più comune che si ha per voice recognition è proprio tramite gli assistenti vocali, i quali permettono di fornire risposte personalizzate solo all'utente che ha allenato l'assistente a riconoscere la propria voce. Altri settori in cui si sta impiegando voice recognition è il finanziario e bancario, i quali stanno implementando sistemi di riconoscimento biometrico per motivi di sicurezza.

La voce così sta gradualmente acquisendo un peso sempre più significativo, anche grazie agli sviluppi delle tecnologie della cattura e della riproduzione sonora. Una persona adesso può usare la propria voce per accedere ai dati in sicurezza.

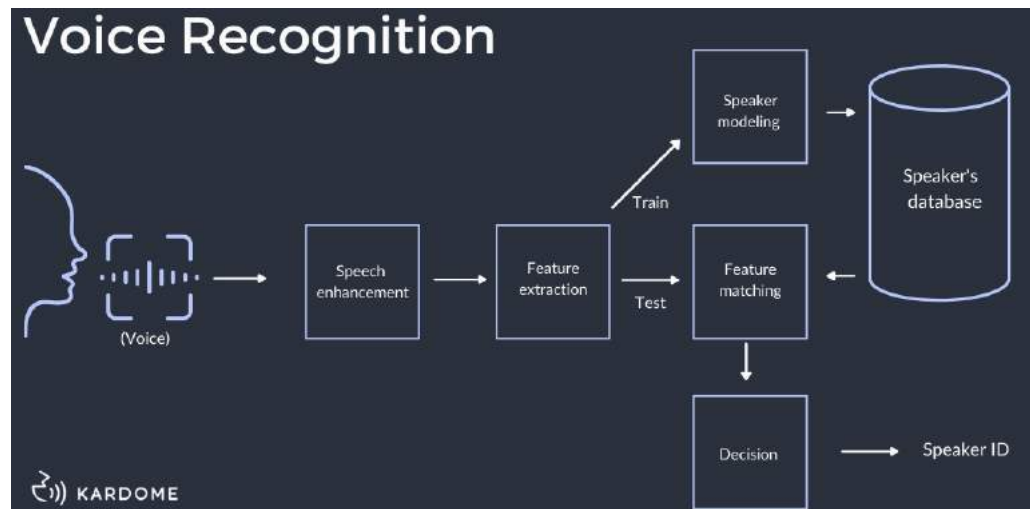


Figura 2.11: Funzionamento Voice Recognition.

Dopo le prime due attività di Speech Recognition e Voice Recognition quello che si ottiene è un testo scritto. La terza attività che viene eseguita è quindi l'interpretazione di tale testo. Grazie al *Natural Language Understanding* (NLU), che è un sotto argomento del Natural Language Processing (NLP), l'assistente può identificare l'intento dietro all'input vocale fornito dall'utente e quindi comprendere il significato delle parole e del contesto. A questo punto, dopo aver colto l'intenzione dell'utente, viene ricercata una risposta valida e restituita di conseguenza. Molto spesso NLU e NLP vengono confuse ma sono parti diverse dello stesso processo di elaborazione del linguaggio naturale, come mostrato in Figura 2.12. La NLU è una componente

della NLP e interpreta il significato che l'utente comunica, classificandolo in intenti propri. Ad esempio, è relativamente semplice capirsi tra persone che parlano la stessa lingua, anche se le pronunce errate, la scelta del vocabolario o delle frasi possono complicare la situazione. NLU è responsabile di questo compito, ovvero distinguere cosa si intende applicando una serie di processi come la categorizzazione del testo, l'analisi del contenuto e l'analisi del sentimento, che consente all'assistente vocale di gestire input diversi. Quindi NLU simula la capacità inferenziale propria della mente umana attraverso il machine learning e sfruttando la velocità della rete, così facendo il sistema identifica sfumature semantiche che in passato non era possibile cogliere. Sin dal suo inizio negli anni '50, la ricerca sulla NLP [30] si è concentrata su attività come traduzione automatica, recupero di informazioni, riepilogo di testo, risposta alle domande, estrazione di informazioni, modellazione di argomenti e, più recentemente, estrazione di opinioni.

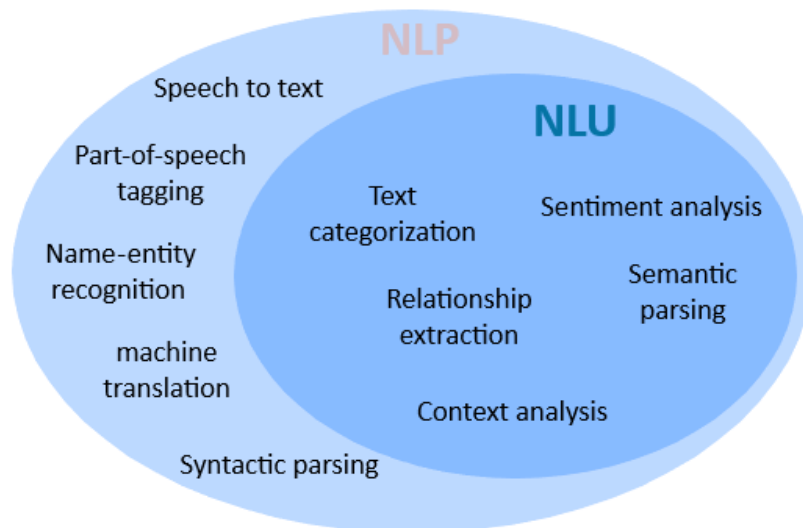


Figura 2.12: NLP e NLU a confronto.

Dunque, si è stabilito che con Speech Recognition possiamo convertire l'input audio in una sequenza di parole. Questo viene inoltrato al Natural Language Understanding (NLU) [31] per estrarre la semantica dell'enunciato. A sua volta per decidere l'azione da intraprendere in base alla strategia di dialogo adottata viene utilizzato un *Dialog Management* (DM). Un DM è un componente software responsabile della gestione dello stato dei dialoghi, il quale mappa gli input vocali dell'utente con le relative risposte di output dell'assistente. La strategia rappresenta lo stato dei dialoghi e delle rispettive operazioni, generando eventi rilevanti per l'interazione. Il DM può utilizzare informazioni contestuali memorizzate dal precedente uso dell'applicazione e una delle azioni che può intraprendere è la generazione dell'output vocale. Pertanto, la generazione della risposta, genera il testo come output che viene passato al motore di sintesi vocale (Speech Synthesis) per essere sintetizzato in un'espressione vocale. La ricerca è stata incentrata sul DM per molti anni, concentrando i propri sforzi sullo sviluppo di strategie di dialogo adeguate per un'esperienza utente più naturale possibile. Con *Speech Synthesis*, anche abbreviato come TTS (Text-to-Speech) [32], si intende quella tecnologia che permette la produzione artificiale di voci umane. Il principale utilizzo è quello che ne ha indotto la creazione, è la capacità di tradurre automaticamente un testo scritto in parlato. I sistemi di Speech Recognition utilizzano delle unità chiamate *fonemi* per ritagliare gli input vocali, essi rappresentano la più piccola unità di suono e possono essere messi insieme per formare parole. Dall'altra parte i sistemi di sintesi vocale si basano sui cosiddetti *grafemi*, lettere o gruppi di lettere che trascrivono un fonema, quindi la risorsa principale qui non è il suono ma il

testo. Questo processo di solito viene fatto in due passaggi:

1. Tagliare il testo in frasi e parole (rappresentano i grafemi), assegnando le trascrizioni fonetiche, cioè la pronuncia, a tutti questi gruppi. Questo processo viene identificato come *G2P* [33], cioè *Grapheme to phoneme*, e si riferisce al compito di trovare la pronuncia di una parola data la sua forma scritta;
2. Convertire in suono queste rappresentazioni linguistiche, ovvero leggere queste indicazioni per produrre una voce il più naturale possibile.

2.4 Tipologie di assistenti vocali

Viene posta adesso l'attenzione sulle varie tipologie di assistenti vocali [34] che si possono trovare oggi dentro un'automobile. In base al grado di funzionalità richieste dall'assistente vocale e della quantità di memoria e potenza di elaborazione disponibile a bordo vettura, i produttori di automobili possono scegliere di abilitare la voce con opzioni di connettività diverse:

- Completamente *Embedded*, nessuna connessione al cloud per riconoscere determinate frasi e inoltre richiede molta meno potenza di calcolo rispetto a una soluzione ibrida;
- *Ibride*, connessione al cloud parziale in modo tale che un assistente vocale ibrido risulta essere sempre attivo e in grado di fornire informazioni aggiornate e accurate;
- Solo *Cloud-Based*, richiede una connessione al cloud al 100%.

Quando l'assistente è connesso al cloud, gli utenti hanno la possibilità di porre domande su una vasta gamma di argomenti come ad esempio meteo, ricerca locale, notizie e molto altro. Molti produttori di automobili, per garantire agli utenti un'esperienza di guida a mani libere, hanno implementato assistenti vocali embedded così da fornire tutte le funzionalità di cui un utente avrebbe bisogno. Solitamente le Case Automobilistiche desiderano soluzioni ibride, ovvero assistenti vocali in grado di elaborare la richiesta del conducente in modalità offline, inviando contemporaneamente la stessa richiesta al cloud e fornendo la miglior risposta disponibile, vedi Figura 2.13. Se il cloud risponde abbastanza velocemente con un risultato accurato, allora viene immediatamente restituito all'utente. Tuttavia, se il cloud non risponde entro un tempo minimo o se la risposta non è sufficientemente buona, viene restituita la risposta dal software embedded.

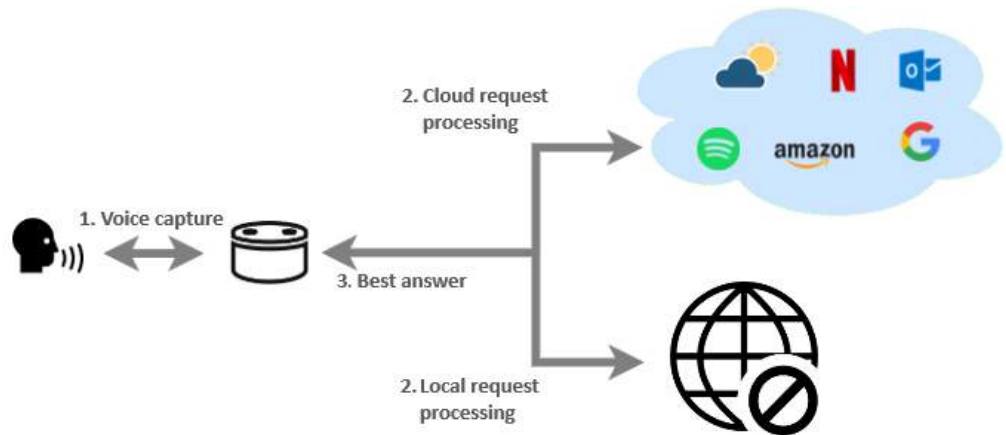


Figura 2.13: Assistente vocale ibrido

Parlare con la propria auto è stata una delle prime soluzioni interattive vocali offerte ai consumatori ed è arrivata più di un decennio prima di Siri o Alexa [28]. Da allora le Case Automobilistiche hanno sperimentato i propri

sistemi specifici per il marchio. Ad esempio, Ford e General Motors dispongono di un sistema digitale di bordo proprietario - rispettivamente Sync e OnStar - che incorpora il telefono e la voce del conducente nella navigazione, nell'intrattenimento e in altre funzioni limitate. Sfortunatamente, come hanno appurato molte Case Automobilistiche, la suite completa di funzionalità non è sempre facile da incorporare. I primi sistemi di riconoscimento vocale in auto sono meglio caratterizzati come abilitanti al controllo vocale. Tali sistemi riconoscevano solo un vocabolario limitato e ciò significava che i consumatori dovevano imparare una sintassi rigida per utilizzare con successo la propria voce per controllare le caratteristiche dell'auto. Solo un numero molto limitato di comandi veniva riconosciuto e la possibilità di essere fraintesi era frustrantemente alta. I problemi si sono aggravati quando alcune Case Automobilistiche hanno sostituito i microfoni a basso costo che erano meno efficaci nel catturare ciò che i conducenti dicevano in una cabina rumorosa. Ciò ha portato a molti consumatori frustrati che hanno trovato i sistemi *difficili da usare e inaffidabili*. Anno dopo anno, i proprietari di veicoli identificano i tradizionali sistemi di riconoscimento vocale come uno dei principali problemi in auto, questo deriva in gran parte dalla difficoltà a capire il funzionamento (*comprensibilità*), dalla frustrazione per la sua capacità limitata e da problemi del sistema di comprendere con precisione i comandi impartiti [10]. Da notare come gli elementi del modello di UX citato nel capitolo precedente vengono ripresi. Sia i produttori che i conducenti hanno presto scoperto che una "macchina parlante" non è molto utile se non comprende la loro lingua o il loro accento, o funziona solo quando non c'è assolutamente alcun rumore di fondo.

Gli assistenti vocali come Siri, Google Assistant e Alexa generalmente sono stati considerati come un'offerta di un nuovo livello di esperienza utente, permettendo l'accesso a una serie più ampia di servizi. Di conseguenza, era naturale che questi popolari assistenti migrassero nell'auto. Questo cambiamento ha creato una folla di assistenti vocali nelle automobili, vedi Figura 2.14. Il *Bluetooth* è stata la prima tecnologia che ha condotto gli assistenti vocali di uso generale all'interno del veicolo, seguito da sistemi come *Apple CarPlay* e *Android Auto*. Questi sono stati poi seguiti da dispositivi aftermarket e SDK (ad esempio *Alexa Auto*), nel tentativo di interfacciarsi con gli assistenti embedded, o poterli sostituire. [28]. Questi sviluppi hanno stimolato una rapida innovazione per diventare l'assistente vocale preferito dai conducenti.

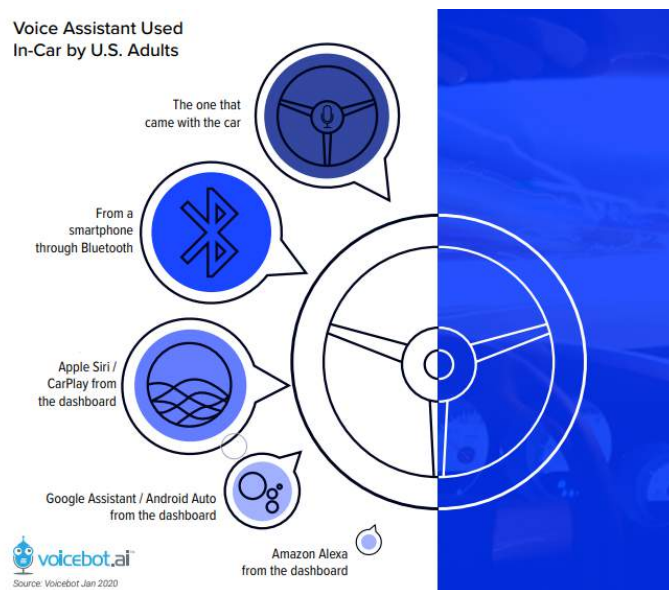


Figura 2.14: Assistenti vocali usati in auto

La disponibilità dell'assistente vocale nelle auto è in rapida espansione.

Solo pochi anni fa, le funzionalità vocali per i conducenti erano disponibili quasi esclusivamente per i proprietari di macchine di lusso. Gli assistenti vocali proprietari delle Case Automobilistiche sono ora offerti su centinaia di modelli di vetture e molti includono anche il supporto per Apple CarPlay, Android Auto e Alexa auto. Il risultato è una scelta sempre più ampia per il consumatore, Figura 2.15.



Brand	Alexa Auto	Apple CarPlay	Android Auto	Proprietary
Acura		✓	✓	✓
BMW	✓	✓	✓	✓
Chevrolet	✓	✓	✓	✓
Chrysler	✓ (Fiat)	✓	✓	✓
Dodge		✓	✓	✓
Ford	✓	✓	✓	✓
GMC	✓	✓	✓	✓
Honda		✓	✓	✓
Hyundai		✓	✓	✓
Infiniti		✓		✓
Jeep		✓	✓	✓
Kia		✓	✓	✓
Lexus	✓	✓	✓	✓
Mercedes		✓	✓	✓
Nissan		✓	✓	✓
Tesla				✓
Toyota	✓	✓	✓	✓
Volkswagen	✓ (Announced)	✓	✓	
Volvo		✓	✓	✓

voicebot.ai
Source: Voicebot Analysis 2020

Figura 2.15: Assistenti vocali offerti dalle Case Automobilistiche

Le soluzioni integrate in genere si concentrano sul controllo delle funzionalità di bordo e su alcuni altri servizi come la navigazione, mentre gli

assistenti che si trovano più comunemente su smartphone e altoparlanti intelligenti tendono a soddisfare le richieste che non comprendono la guida. Tuttavia, anche questo sta cambiando, poiché gli assistenti come *Alexa Auto* e *Google Android Auto OS* stanno estendendo il controllo vocale alle funzionalità della macchina. La domanda oggi è meno su quali assistenti vocali saranno disponibili nell'auto e più su quali saranno scelti per soddisfare la varietà delle preferenze dei consumatori.

Ricollegandosi al discorso precedente sulla distrazione del conducente, anche gli attuali sistemi di controllo vocale aumentano i tempi di reazione di questi ultimi e rimangono una preoccupazione per la sicurezza stradale [11]. Nello specifico questo peggiora con la presenza di due assistenti vocali all'interno della macchina che lavorano in parallelo ma non contemporaneamente. Solitamente gli assistenti presenti sono quello di base - fornito dalla Casa Automobilistica - e quello integrato che può essere ad esempio *Alexa* o *Siri*. Dunque, in questa situazione il conducente potrebbe richiedere dei servizi ad uno dei due senza sapere quale di questi possa soddisfare la richiesta. Tutto ciò quindi richiede concentrazione da parte del conducente poiché in base al servizio richiesto deve scegliere accuratamente uno dei due assistenti vocali, portando ad aumentare così il livello di distrazione. Inoltre in caso di scelta sbagliata dell'assistente, viene provocato un senso di malessere e frustrazione. Perciò anche da qui nasce l'esigenza di avere o un unico assistente vocale o assistenti vocali che possano collaborare tra loro, poiché gli utenti vogliono un servizio che semplifichi le loro vite invece di aggiungere complicazioni. Quando ciò accade, i consumatori hanno una grande esperienza, che alla fine avvantaggia anche le Case Automobilistiche.

Per questo è importante combinare tale funzionalità con un unico servizio vocale, producendo un'esperienza positiva del consumatore in auto. User Experience e fidelizzazione del cliente sono direttamente proporzionali; e un cliente soddisfatto, con un passaparola entusiasta, è la più importante forma di pubblicità. [10].

Questo è un ambiente altamente specializzato in cui l'assistente digitale deve lavorare con un sistema avanzato di assistenza alla guida detto *ADAS* (Advanced Driver-Assistance Systems) per supportare il conducente. Un assistente vocale come Alexa potrebbe non necessariamente comprendere il contesto della situazione che deve affrontare il conducente del veicolo. Ciò di cui gli utenti avrebbero bisogno è un assistente automobilistico che abbia:

1. Una profonda conoscenza del veicolo, così da rispondere a qualsiasi domanda relativa ad esso;
2. La capacità di soddisfare le esigenze di guida e le personalizzazioni necessarie per svolgere il lavoro;
3. La possibilità di fornire assistenza durante le emergenze o risolvere i problemi;
4. Il modo di indirizzare l'utente alla stazione di servizio più vicina;

Il tutto garantendo la sicurezza dei passeggeri all'interno della vettura. Un assistente che in tutta sicurezza agevola l'interazione con le features della vettura risulta utile, migliorando la percezione che l'utente ha del servizio fruito.

Quello che verrà proposto nei prossimi capitoli è una possibile soluzione che unisca tutti questi concetti per lo sviluppo di un applicativo tramite

l'Auto SDK di Alexa. Per soddisfare i requisiti degli utenti e diminuire il livello di distrazione del conducente viene aggiunto all'assistente vocale la capacità di reperire informazioni sempre aggiornate sui diversi componenti del veicolo, così da poter fornire un servizio di diagnostica e assistenza.

Capitolo 3

Alexa

Prima di proseguire con la presentazione dell'applicativo, viene data una panoramica generale dell'assistente vocale scelto, ovvero Alexa.



Figura 3.1: Logo Amazon Alexa

Alexa [35] è il servizio vocale cloud-based di Amazon disponibile su milioni di dispositivi. Essendo commercializzato come un assistente personale

intelligente, viene utilizzato per riprodurre musica, rispondere a domande generiche, impostare sveglie e timer o controllare dispositivi in rete.

Studi di settore [36] indicano una rapida adozione di questa tecnologia manifestata nei dati di vendita che sono passati da 2,4 milioni di unità nel 2015 a 5,2 milioni nel 2016. Ad oggi sono stati venduti più di 100 milioni di dispositivi basati su Alexa, diventando così il leader nel settore. Amazon [37], partendo dalla casa fino ad arrivare nell'auto, si concentra sui luoghi in cui può avere senso parlare ad alta voce; cercando di rendere l'utilizzo di Alexa il più semplice possibile nei relativi ambienti.

La Figura 3.2 illustra lo schema di funzionamento di Alexa. L'assistente viene attivato quando il suo software di riconoscimento vocale riceve una parola o una frase di attivazione da un utente, questa viene identificata come "Wake Up Word"; ad esempio, la parola "Alexa" può essere utilizzata per attivare il dispositivo, ma questa parola di attivazione può essere personalizzata dall'utente. Gli utenti inviano una richiesta, che viene filtrata ed elaborata da Alexa sfruttando *speech recognition*, *machine learning* e *natural language understanding*. Alexa accede ai servizi ospitati sul Web e fornisce una risposta all'utente utilizzando *text to speech*. All'interno della risposta è presente la "scheda" di informazioni che fornisce un record e i relativi risultati all'utente.

Dunque, per diventare leader nel settore degli assistenti vocali, ciò che ha fatto realmente la differenza sono stati i progressi compiuti in ambito di intelligenza artificiale. Questo ha permesso così di ridurre gli errori di riconoscimento e apprendimento, ma non solo, aumentando anche la naturalezza del discorso tra Alexa e l'utente stesso.



Figura 3.2: Schema di funzionamento di Alexa

La strategia generale per l'intelligenza artificiale di Alexa [38] è stata quella di combinare il machine learning - nello specifico un suo sottocampo cioè il *deep learning* - con i dati su larga scala disponibili tramite Amazon Web Services. AWS è una piattaforma di cloud computing completa e in evoluzione che include una combinazione di pacchetti software e strumenti come potenza di calcolo, archiviazione di database e servizi di distribuzione dei contenuti. Questi miglioramenti delle prestazioni sono il risultato di ricerche su una varietà di argomenti specifici tra cui:

1. *Apprendimento semi-supervisionato*, cioè l'utilizzo di una combinazione di dati non etichettati ed etichettati per migliorare il sistema di machine learning;
2. *Apprendimento attivo*, cioè la strategia di apprendimento in cui il sistema di machine learning seleziona i campioni più informativi per ricevere le etichette manualmente;

3. *Addestramento distribuito su larga scala*, ovvero parallelizzare modelli di machine learning per il training;
4. *Modellazione sensibile al contesto*, consiste nell'utilizzo di un'ampia varietà di informazioni per migliorare la precisione.

3.1 Alexa Auto SDK

Il team di Amazon ha introdotto nell'agosto 2018 *Alexa Auto SDK* [39], dove SDK sta per Software Development Kit. Tale *SDK* permette di semplificare l'integrazione di Alexa nei sistemi di infotainment di bordo, portando nel veicolo l'esperienza dell'assistente vocale che ha deliziato i clienti a casa. Aggiunge funzionalità specifiche per l'automotive e contestualizza l'esperienza dell'utente per il veicolo. Include codice sorgente in C++ e Java e librerie che consentono al veicolo di elaborare input e trigger audio, stabilire una connessione con il servizio e gestire le interazioni. L'*SDK* include le funzionalità principali di Alexa, come *speech recognition* e *speech synthesis*, e altre funzionalità come lo streaming multimediale, il controllo dei dispositivi domestici intelligenti, le notifiche, i bollettini meteorologici e anche le *custom Skills*, le quali verranno approfondite in seguito. Inoltre, l'*SDK* fornisce gli hook necessari per connettersi a una parola di attivazione detta "Wake Up Word", un lettore multimediale locale, un telefono locale e un sistema di navigazione locale. Si ha quindi la possibilità di accedere a determinate funzionalità e capacità di Alexa anche quando la connettività Internet è instabile o assente.

3.1.1 Architettura di Alexa Auto SDK

Auto SDK ha un'architettura modulare [40], come si può evincere dalla Figura 3.3, con un sistema organizzato in gruppi di funzionalità logicamente correlati denominati "moduli" che ne forniscono l'implementazione a runtime, rappresentato dall'*Engine*. Questo ultimo è quindi un sistema di componenti che fornisce l'implementazione principale di tutte le funzionalità di Auto SDK e ne registra i moduli utilizzati. Ogni modulo espone delle interfacce per gestire funzionalità specifiche e consente di selezionare solo quelle che si desidera includere nella propria integrazione.

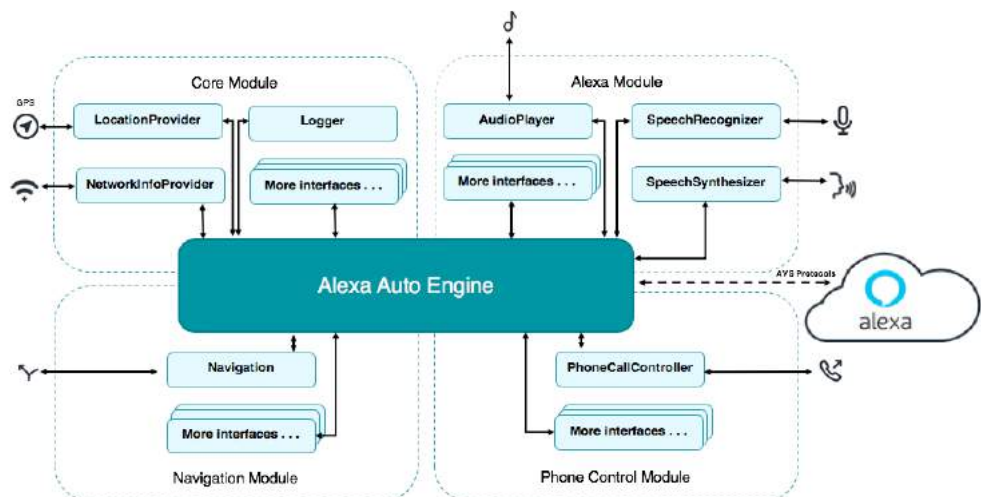


Figura 3.3: Architettura di Alexa Auto SDK

I principali moduli presenti sono:

- *Core Module*, è il cuore dell'SDK e funge da base per tutte le altre funzionalità. Fornisce l'infrastruttura di base dell'engine ad altri moduli, che la estendono per aggiungere funzionalità specifiche del

modulo. L'infrastruttura è necessaria per qualsiasi modulo che utilizza le interfacce di messaggistica, così da poter segnalare informazioni essenziali come la posizione del dispositivo e lo stato della connessione di rete.

- *Alexa Module*, supporta le funzionalità di Alexa come Speech Recognition, Speech Synthesis o Audio Player.
- *Navigation Module*, consente all'applicazione che ospita l'SDK di utilizzare le funzionalità di navigazione di Alexa e fornisce supporto per interfacciarsi con il sistema di navigazione di bordo.
- *Phone Control Module*, permette all'applicazione di utilizzare le funzionalità di controllo delle telefonate di Alexa, indipendentemente dal meccanismo di connessione al dispositivo chiamante. Utilizzando questo modulo viene consentito all'utente di interagire con chiamate nuove o in corso, fornendo ad Alexa lo stato del dispositivo.
- *Code-Based Linking (CBL) Module*, implementa il *meccanismo CBL* per l'acquisizione di token di accesso. Ogni richiesta al Servizio Vocale di Alexa (AVS) richiede appunto un token di accesso che tramite l'interfaccia *Authorization* permette di avviare, annullare e disconnettersi al servizio.
- *Car Control Module*, consente all'applicazione di creare un'esperienza di controllo del veicolo personalizzata permettendo all'utente di controllare le funzioni del veicolo tramite Alexa.

Per creare un'applicazione basata su l'Auto SDK di Alexa, è necessario eseguire la build di questo ultimo e installare i relativi pacchetti nel di-

spositivo. Creare ed eseguire un'istanza dell'Engine tramite il modulo Core. Successivamente, è possibile estendere le interfacce Auto SDK creando degli handler personalizzati per ogni interfaccia da implementare e registrandoli nell'Engine.

L'SDK fornisce anche una "Sample app" per vedere il suo funzionamento e per testare i diversi moduli, come riportato Figura 3.4. Si è partiti proprio da essa per sviluppare l'applicativo finale, tuttavia quello che manca ancora nel kit è un modulo che permetta di fornire un servizio diagnostica e assistenza al conducente. Per ovviare a questa mancanza nel nostro caso, sono state impiegate le cosiddette "Skills" di Alexa (ASK, Alexa Skills Kit), che verranno approfondite di seguito.



Figura 3.4: Sample App

3.2 Alexa Skills Kit

Alexa Skills Kit (ASK) [41] è un framework di sviluppo software che permette di creare contenuti, chiamate *Skills*. Queste sono un insieme di azioni o attività svolte da Alexa e sono come delle app che aiutano i clienti a svolgere attività quotidiane come controllare le notizie o ascoltare la musica. ASK può essere utilizzato anche per creare Skill personalizzate dette *custom Skill*,

queste includono sia il codice sotto forma di servizio cloud-based, sia la configurazione fornita sulla console dello sviluppatore. Dunque, si espanderanno le capacità di Alexa tramite la creazione di una custom Skill per ovviare alla mancanza di un modulo che curi la diagnostica del veicolo e l'assistenza del conducente. Questo permetterà di personalizzare al meglio l'esperienza in auto poiché l'assistente vocale al suo interno dovrebbe essere il più versatile possibile. Nell'Alexa Skills Store, è possibile scoprire nuove Skills e tramite l'app di Alexa l'utente può effettivamente invocarle e utilizzarle, solo se prima le abilita. Una volta abilitata è collegata all'account Amazon e può essere invocata da qualsiasi dispositivo associato a questo ultimo. Le Skills possono essere installate anche a voce o addirittura Alexa può proporle per soddisfare le richieste del cliente, quando pensa che una particolare Skill possa soddisfare le richieste del cliente. Pertanto, le capacità di Alexa possono essere continuamente ampliate.

Alexa attraverso un'interfaccia vocale interattiva offre agli utenti un modo a mani libere per interagire con le proprie Skills. Un utente è in grado di accedere a queste ultime chiedendo semplicemente di invocarle. Quando viene pronunciata la Wake Up Word, "Alexa", il dispositivo trasmette il discorso ad Alexa Voice Services (AVS) nel cloud. Alexa riconosce il discorso, determina ciò che l'utente desidera e quindi invia una richiesta per invocare la Skill che può soddisfarla. Il servizio gestisce Speech Recognition e Natural Language Processing, comunicando con la Skill attraverso l'interfaccia HTTPS la quale usa un meccanismo *request-response*. Una volta invocata, la Skill riceve una richiesta di tipo POST contenente un JSON. Il contenuto della richiesta fornisce i parametri necessari affinché la Skill la comprenda,

ne esegua la logica e ne generi una risposta. In questo processo, vedi Figura 3.5, è possibile vedere come la Voice User Interface (VUI) e il gestore logico siano due entità separate. Sono infatti le due componenti principali di una Skill, dove la VUI è definita dal *modello di interazione* mentre la logica della Skill dal codice di back-end. Il compito di Alexa è raccogliere i dati necessari per comporre una richiesta corretta tramite il modello di interazione e poi inviarli al codice di back-end.

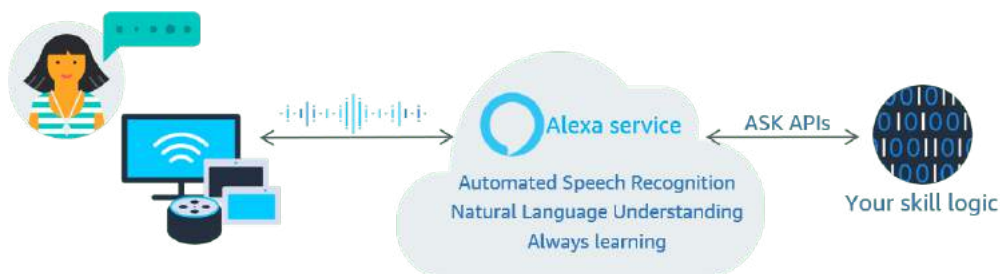


Figura 3.5: Flusso di invocazione di una Skill

Ogni Skill di Alexa ha un *voice interaction model*, che definisce le parole e le frasi che gli utenti possono pronunciare per fare in modo che questa ultima faccia ciò che vogliono gli utenti. Questo modello determina il modo in cui gli utenti comunicano e controllano la Skill. Quando vengono effettuate domande e richieste, Alexa utilizza il modello di interazione per interpretare e tradurre le parole in una richiesta specifica per la Skill identificata.

Alexa [42] supporta due tipi di modelli di interazione vocale:

1. *Pre-built voice interaction model*, in questo modello, ASK definisce l'insieme di parole pronunciate dagli utenti per invocare la Skill. Quelle che utilizzano questo tipo di modello avranno meno controllo sull'esperienza dell'utente, ma lo sviluppo risulterà essere semplificato

poiché non sarà necessario creare una VUI, basterà sfruttare il modello predefinito. Un vantaggio delle Skill che adottano questo modello è che non richiedono di essere attivate tramite l'invocazione di un nome specifico, ma basta semplicemente richiamare Alexa e fornire la richiesta. Tuttavia, hanno funzionalità abbastanza limitate in uno specifico campo, come ad esempio il controllo di dispositivi connessi al cloud o la selezione e l'ascolto di contenuti audio trasmessi in streaming tramite un dispositivo abilitato.

2. *Custom voice interaction model*, offre il massimo controllo sull'esperienza dell'utente. Permette di progettare l'insieme di parole e frasi per ogni azione che la Skill può eseguire in modo da offrire un'esperienza vocale completamente personalizzata. Le *custom Skill*, essendo più versatili, permettono una maggiore flessibilità e controllo della progettazione e del codice. Tuttavia, risultano essere più complesse dato che il modello di interazione deve essere costruito da zero.

Oltre all'interaction model quando si sta progettando una custom Skill vengono definiti:

- Gli *Intents* che la Skill può gestire. Questi fanno parte del modello di interazione e rappresentano le azioni che gli utenti possono eseguire. Ciascun intent richiama una specifica funzionalità e può avere degli argomenti, detti "slots", che raccolgono i valori necessari alla Skill per soddisfare una richiesta dell'utente.
- Le *Utterances*, cioè le espressioni, che gli utenti dicono per invocare gli intents. Queste vengono quindi mappate a intents specifici andando a

costituire il modello di interazione della Skill. Può esistere una mappatura "many-to-one" delle espressioni agli intents perchè il modello deve definire ogni modo in cui un utente può comunicare la stessa richiesta.

- L'*Invocation Name* che un utente pronuncia quando avvia una conversazione con la Skill. Questo è il nome che Alexa utilizza per identificarla.
- La *Skill logic* per soddisfare gli intents personalizzati.
- Un *Endpoint* accessibile da internet per permettere l'host del servizio cloud-based che accetta questi intents come richieste strutturate. Alexa Skills Kit fornisce un'opzione chiamata *Alexa-hosted* per creare, archiviare e ospitare le Skill su AWS. Altre opzioni per l'host del servizio sono, sfruttare una *AWS Lambda Function* utilizzando le risorse personali di AWS o scrivere un *Web Service* e ospitarlo con qualsiasi hosting-provider.
- Una *AWS Lambda Function*, ovvero un servizio di calcolo che consente di eseguire codice senza configurare o gestire i server. Lambda esegue il codice su un'infrastruttura di elaborazione ad alta disponibilità ed amministra le risorse di elaborazione, inclusa la manutenzione del server e del sistema operativo. Con Lambda può essere eseguito il codice, che è organizzato nella funzione, praticamente per qualsiasi tipo di applicazione o servizio di back-end.

- Un *ambiente di sviluppo* appropriato per il linguaggio di programmazione che si intende utilizzare. Ad esempio, una Lambda Function può essere scritta in Node.js, Java, Python o C#.

3.2.1 Software e strumenti di sviluppo

ASK offre più opzioni per lo sviluppo delle Skills, tra cui *Alexa developer console*, *Alexa-Hosted Skills*, *ASK Toolkit* per Visual Studio Code, *ASK SDKs*, *ASK Command Line Interface* (CLI).

Alexa Developer Console [43] è il software utilizzato per lo sviluppo della Skill del progetto e offre un'esperienza semplificata per la creazione, la gestione e la pubblicazione delle Skills, vedi Figura 3.6.

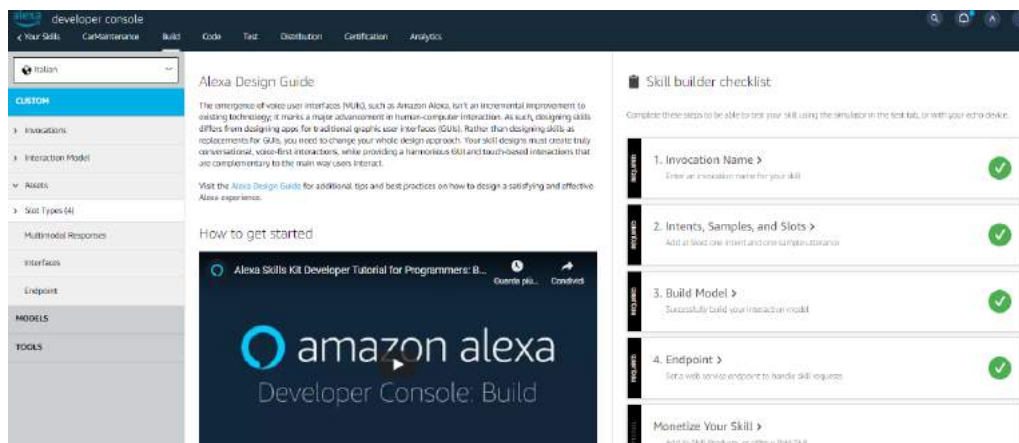


Figura 3.6: Alexa Developer Console: Build page

Questa console permette di impostare tutte le configurazioni del modello di interazione e di specificare gli endpoint del servizio, ovvero dove si trova il codice della Skill. Inoltre, come è possibile vedere in Figura 3.7, permette di testare tutte le Skills con testo o voce, inviarle per la certificazione e

utilizzare l'analisi per rivedere le metriche, come il numero di clienti e gli intents invocati.

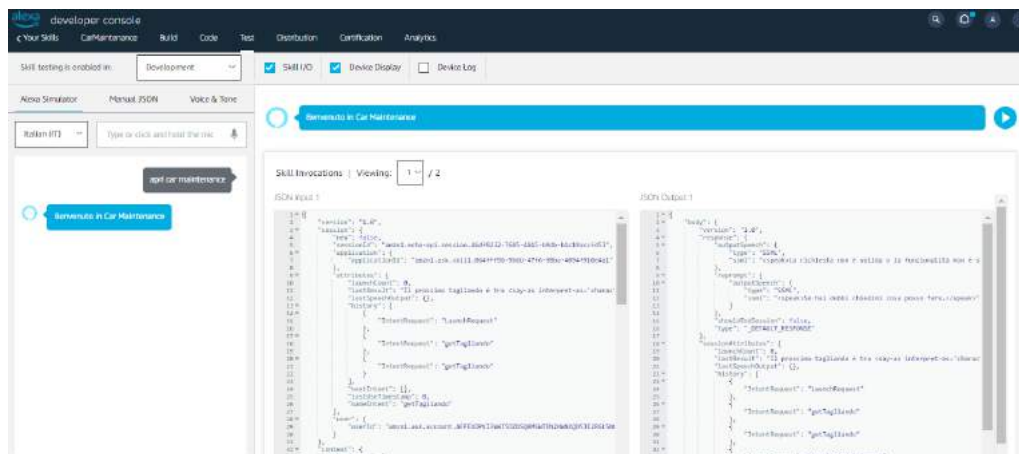


Figura 3.7: Alexa Developer Console: Test page

Una limitazione riscontrata è che solo tramite la console si può accedere a questo tipo di Skill-code, per qualsiasi altra opzione, questa funzionalità non è accessibile e lo sviluppo deve dipendere da altri strumenti. Nonostante questo Alexa Developer Console rimane lo strumento migliore per il design della VUI data la sua facilità d'uso e capacità di lavorare con altri software.

Alexa-Hosted Skills permette di memorizzare il codice e le risorse su AWS, così da iniziare rapidamente a lavorare alla Skill utilizzando i templates disponibili nella console per sviluppatori. Sostanzialmente è il modo più veloce per creare una Skill. Alexa fornisce tutte le risorse AWS necessarie, inclusa una funzione Lambda, e in questo caso la connessione della Skill all'endpoint avviene automaticamente. Tuttavia, come già annunciato esistono altri due metodi che permettono l'host del servizio cloud-based della Skill:

- Tramite un *Web Service* [44] che accetta richieste e invia risposte al servizio Alexa nel cloud. Per gestire le richieste inviate da Alexa, il servizio web deve essere accessibile tramite internet e accettare richieste HTTP.
- Attraverso una *AWS Lambda Function* [45], che esegue il codice nel cloud solo quando è necessario e si ridimensiona autonomamente, quindi senza gestire i server. Dopo aver caricato il codice per la Skill su una funzione Lambda, questa fa il resto, eseguendolo in risposta alle interazioni vocali di Alexa e gestendo le risorse di calcolo. Tutto ciò elimina la complessità della configurazione e della gestione di un proprio endpoint. In questo modo, lo sviluppatore non deve occuparsi di amministrare o gestire nessuna delle risorse di calcolo per il servizio. Non ha bisogno di un *certificato SSL*, esso è un protocollo sviluppato per inviare informazioni in modo sicuro su Internet e consiste nel crittografare tutti i dati che vanno tra un utente e un server web, identificando questo ultimo e cambiando il suo indirizzo HTTP in HTTPS, rendendolo più sicuro. Spetta dunque ad Alexa crittografare la comunicazione con la Lambda Function utilizzando TLS. Inoltre non è necessario verificare le richieste che provengono dal servizio Alexa, poiché gli accessi sono controllati automaticamente. Tuttavia, lo sviluppatore deve connettere la Lambda Function alla Skill, così da sapere dove vengono inviate le richieste. Questa connessione avviene aggiornando l'endpoint della Skill con l'*Amazon Resource Names* (ARN) della funzione, come mostrato in Figura 3.8.

Per ridurre la latenza percepita quando si accede a una Skill, Alexa fornisce

Service Endpoint Type

Select how you will host your skill's service endpoint. Best practices in choosing lambda regions. [Learn more here](#)

<input checked="" type="radio"/> AWS Lambda ARN (Recommended)	Your Skill ID	amzn1.ask.skill.064fff96-9b6b-47f6-98ba-4094f910e4a Copy to Clipboard
	Default Region (Required)	arn:aws:lambda:us-east-2:316246626242:function:serverlessrepo-a
	North America (Optional)	arn:aws:lambda:us-east-2:316246626242:function:serverlessrepo-a
	Europe and India (Optional)	arn:aws:lambda:us-east-2:316246626242:function:serverlessrepo-a
	Far East (Optional)	arn:aws:lambda:us-east-2:316246626242:function:serverlessrepo-a

Figura 3.8: Alexa Developer Console: Endpoint page

tre endpoint AWS Lambda, in tre diverse regioni. Ciò rende molto semplice la creazione di una Skill, perché lo sviluppatore non deve occuparsi di tutto ciò che sta dietro la VUI. Questi servizi sono forniti gratuitamente, ma ci sono alcuni limiti di utilizzo, come il numero di richieste al server che, nel caso di un account gratuito, è impostato di default a 1 milione. Tra questi il più semplice e veloce da utilizzare è proprio Alexa-hosted Skills ma risulta essere anche il più limitato e meno flessibile. Per questo è stato adottato il metodo che sfrutta la Lambda Function per lo sviluppo della Skill. A differenza del metodo precedente il codice non è legato alla Skill e alla console per sviluppatori, risultando così più versatile e consentendo di gestire le risorse utilizzate dalla funzione.

ASK Toolkit per Visual Studio Code (VS Code) [46] è un'estensione che semplifica la creazione, il testing e la distribuzione delle Skills di Ale-

xa, fornendo uno spazio di lavoro dedicato per queste ultime in VS Code. In questo ambiente è possibile creare e modificare le Skill, visualizzare le anteprime ed eseguire il debug tramite un simulatore di Alexa.

ASK SDKs [47], sono strumenti di sviluppo software e librerie che danno accesso programmatico alle funzionalità di Alexa. Sono disponibili per Node.js, Java e Python, e permettono di semplificare lo sviluppo delle Skills di Alexa consentendo di dedicare più tempo all'implementazione delle funzionalità e meno tempo alla scrittura del codice standard.

ASK Command Line Interface (CLI) [48], è uno strumento per la gestione delle Skills e delle relative risorse dalla riga di comando. La ASK CLI consente di creare nuove Skills e distribuirle nella console dello sviluppatore, caricando il codice direttamente tramite Lambda function se opportunatamente configurata. Inoltre l'interfaccia permette di costruire e aggiornare il Model Interaction della Skill, di simulare questa ultima e inviarla per ottenere la certificazione. Questa soluzione risulta non essere semplice da utilizzare quindi è necessario che lo sviluppatore possieda una certa esperienza nella gestione delle Skills.

3.2.2 Processo di sviluppo

Come già enunciato, per la creazione della custom Skill viene utilizzata la console per sviluppatori di Alexa. La quale necessita di un *nome* per identificare e invocare la Skill, di una *lingua* iniziale e dell'*Interaction model*. Dunque, il passo più importante nel processo di sviluppo della Skill è la scelta dell'**Invocation Name**. Gli utenti inizieranno ad interagire con la custom Skill solo dopo averla pronunciata. Inoltre tale nome può essere

combinato con un comando o un'azione così da invocare un intent specifico della Skill. In questo caso il nome utilizzato è *Car Maintenance* e la lingua iniziale è impostata in *Italiano*. Sono presenti quindi diversi modi in cui l'Invocation Name può essere pronunciato per iniziare ad usare la Skill, come ad esempio:

- "*Alexa, chiedi a Car Maintenance qual'è lo stato della macchina*", qui viene invocata la Skill con una richiesta particolare;
- "*Alexa, apri Car Maintenance*", qui viene invocata semplicemente la Skill. Solitamente viene restituito un intent di benvenuto tramite una *LaunchRequest* la quale è attivata quando viene fatta una richiesta non intenzionale;
- "*Alexa, Car Maintenance*", anche in questo caso viene invocata semplicemente la Skill.

L'Invocation Name [49] deve rispettare molti requisiti affinché sia valido. Ad esempio non deve violare i diritti di proprietà intellettuale di un'entità o di una persona. Non sono consentiti nomi composti da una sola parola, salvo casi specifici. Inoltre l'inclusione di nomi di persone o luoghi non sono consentiti.

Interaction model [50], con Alexa Skills Kit è possibile creare un modello di interazione personalizzato, definendo la logica della custom Skill e l'interfaccia vocale con cui gli utenti interagiscono. Per la definizione del modello è necessario fornire:

1. **Intents**, un intent rappresenta un'azione che soddisfa la richiesta vocale di un utente, può facoltativamente avere argomenti chiamati

"slot" ed almeno una "sample utterance". Come è possibile vedere nella Figura 3.9, questi sono gli intents personalizzati creati.

NAME	UTTERANCES	SLOTS	TYPE
<code>getCarStatus</code>	9	1	Custom
<code>getComponentStatus</code>	18	1	Custom
<code>getInfo</code>	-	1	Custom
<code>getSolution</code>	1	1	Custom
<code>getComponent</code>	5	-	Custom
<code>getTagliando</code>	3	-	Custom

Figura 3.9: Intents di Car Maintenance

Ciascuno intent viene rappresentato nell'interaction model ed ognuno di essi ha una funzionalità diversa. Per esempio, *getCarStatus* consente al conducente di ottenere una diagnosi completa del veicolo, riportando i componenti che hanno riscontrato un problema. Altrimenti *getComponentStatus* permette di analizzare un componente specifico del veicolo, fornendo poi assistenza in caso di guasti. Tuttavia, questi intent devono essere attivati in qualche modo, così da consentire ad Alexa di capire cosa vuole l'utente. Proprio per questo viene inserita una lista di Sample utterances.

2. **Sample utterances**, sono delle probabili frasi pronunciate dagli utenti, le quali vengono mappate agli intents per attivarli. Maggiore è il

numero di varianti possibili di ciò che l'utente può dire e più è alta la probabilità che Alexa riconosca l'intent richiesto. Questo viene fatto proprio perché non dovrebbe esserci nessuna differenza se viene detto "Qual'è lo stato della macchina?" piuttosto che "In che condizioni è la vettura?". Al fine di evitare conflitti tra le sample utterances, devono essere univoche e non possono essere mappate a intent differenti.



Figura 3.10: Sample utterances di getCarStatus

3. *Custom slot types* [51], si tratta di parole o frasi all'interno delle sample utterances che rappresentano informazioni variabili. Dunque, viene sostituita la parola originale con il nome dello slot tra parentesi graffe ({}). In questo modo Alexa è in grado di capire cosa vuole l'utente quando va a colmare il vuoto dello slot, permettendo al sistema di eseguire attività diverse a seconda del valore. Un esempio può essere lo slot *Componente* utilizzato in *getComponentStatus*, il quale permette di eseguire la diagnosi di un componente specifico del veicolo, vedi Figura 3.11.

Quando i possibili valori per uno slot non sono coperti da quelli integrati da Amazon allora vengono utilizzati i *custom slot types*, cioè

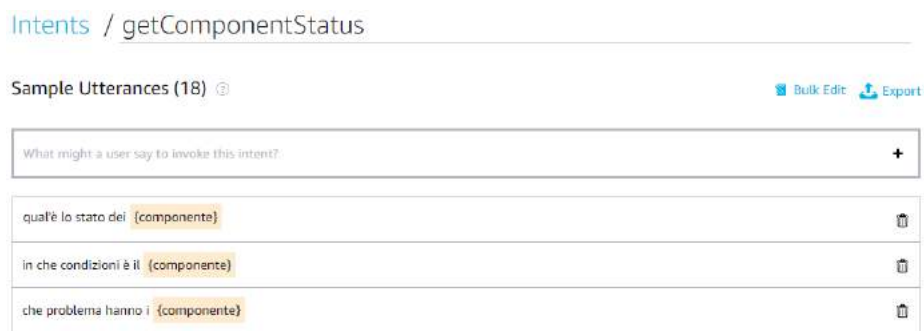


Figura 3.11: Sample utterances di `getComponentStatus`

un tipo di slot personalizzato. Questo è proprio quello utilizzato per lo slot *Componente*, il quale contiene come valori tutti i componenti diagnosticabili nella macchina. Dunque, oltre a scegliere il tipo di slot possono essere abilitate altre impostazioni come il *riempimento*, la *conferma* e la *convalida*. In breve, il *riempimento* rende lo slot necessario affinché l'intent possa essere soddisfatto. Può essere specificata anche una frase che richieda all'utente il valore mancante, un esempio di come viene gestita questa parte della conversazione è mostrato in Figura 3.12. La *conferma* chiede all'utente se il valore dello slot è corretto prima di continuare il dialogo. Infine la *convalida* permette di accettare solo valori che fanno parte della definizione dello slot e di rifiutare quelli che appartengono ad un determinato insieme di parole.

4. **Dialog model** [52], è una conversazione con più turni in cui Alexa pone domande all'utente al fine di raccogliere, convalidare e confermare i valori degli slot. La conversazione è legata a un intent specifico che rappresenta la richiesta complessiva dell'utente e fino a quando



Figura 3.12: Riempimento del custom slot

tutti gli slot necessari non sono stati riempiti e confermati secondo le regole definite nel modello la conversazione continua. Dunque, in sostanza contiene la struttura della conversazione quando non viene effettuata una richiesta completa o se l'intent deve essere confermato prima dell'invio. Così facendo è Alexa che gestisce il proseguimento della conversazione in base a quello che l'utente dice, rispondendo tramite i prompt definiti per richiedere le informazioni mancanti. Di conseguenza quando si devono gestire semplici dialoghi non spetta allo sviluppatore di controllare se la richiesta dell'utente è completa, scrivendo del codice per richiedere i valori degli slot, ma la gestione viene delegata ad Alexa.

Esistono essenzialmente due modi per delegare il dialogo:

- *Auto delegation*, in questo caso Alexa invia alla Skill un intent di richiesta chiamato "IntentRequest" solo quando il dialogo è completato;
- *Manual delegation*, tramite la direttiva "Dialog.Delegate", Alexa

invia alla Skill un `IntentRequest` per ogni turno di conversazione. Nella richiesta è presente un `"dialogState"` che indica lo stato corrente del dialogo ed può essere impostato su `STARTED`, `IN_PROGRESS` o `COMPLETED`. La Skill restituisce questa direttiva per i dialoghi incompleti, dicendo ad Alexa di controllare il modello per il passaggio successivo e di utilizzare un prompt per chiedere all'utente ulteriori informazioni, se necessario. Una volta completati tutti i passaggi, la Skill riceve l'`IntentRequest` finale con `dialogState` impostato su `COMPLETED`.

La delegazione del dialogo [53] può essere configurato per l'intera Skill oppure per ogni intent. Ciò significa che può essere utilizzata l'`Auto delegation` per alcuni intents e la `Manual delegation` per altri. La giusta strategia di delegazione dipende dalla complessità del dialogo. `Auto delegation` è più semplice poiché non è necessario gestire alcun dialogo nel codice. `Manual delegation` è più flessibile, dato che la Skill può prendere decisioni in fase di esecuzione e dirigere il flusso della conversazione. È inoltre possibile utilizzare `"Dialog.Delegate"` in combinazione con altre direttive `Dialog` per assumere il controllo completo del dialogo. Tra queste abbiamo `"Dialog.ElicitSlot"` che permette di richiedere manualmente il valore di uno slot e `"Dialog.ConfirmSlot"` che richiede invece la conferma di uno slot. Alexa così facendo è in grado di cambiare l'intent e avere un ruolo attivo nella conversazione, rendendo l'esperienza dell'utente più naturale e verosimile.

Tutta via questo metodo di dialogo permette esclusivamente la realizzazione di un solo intent alla volta, rendendo la conversazione poco

articolata. Dunque, per esempio, l'utente dovrebbe chiedere ogni volta se c'è un problema con un componente, quali sono le relative informazioni e se può essere fornita una soluzione adeguata. Tutto questo porterebbe appesantire l'esperienza dell'utente nell'utilizzare un assistente vocale che in realtà dovrebbe guidarlo e assisterlo.

Per ovviare a questa mancanza viene settato un attributo che rappresenta l'intent corrente, permettendo ad Alexa di tenere il punto della conversazione e di proporre nuovi intents in base a quanto detto dall'utente. La logica utilizzata per avanzare nella conversazione è quella di sfruttare gli intent pre-costruiti da Amazon: "YesIntent" e "NoIntent". Tuttavia, al fine di evitare di avere una serie di domande da parte di Alexa a cui l'utente potesse rispondere solo con "si" o "no", vengono utilizzati *Custom Slot Type* così da proporre nuovi intents agli utenti in base al valore catturato dallo slot.

Adottando questo approccio viene resa possibile un'adeguata conversazione per la diagnostica del veicolo, così che quando il conducente richiede lo stato di un componente la Skill prende il controllo del flusso. Alexa sarà ora in grado di chiedere all'utente se desidera maggiori informazioni sul problema o se necessita direttamente di trovare una soluzione. In Figura 3.13 viene mostrato cosa accade quando viene eseguita una richiesta di questo genere.

Dunque, tutti gli elementi illustrati definiscono l'interaction model su cui si basa Alexa. Questo permette all'utente di interagire con il veicolo sfruttando l'invocazione della Skill. L'intent desiderato dal conducente viene

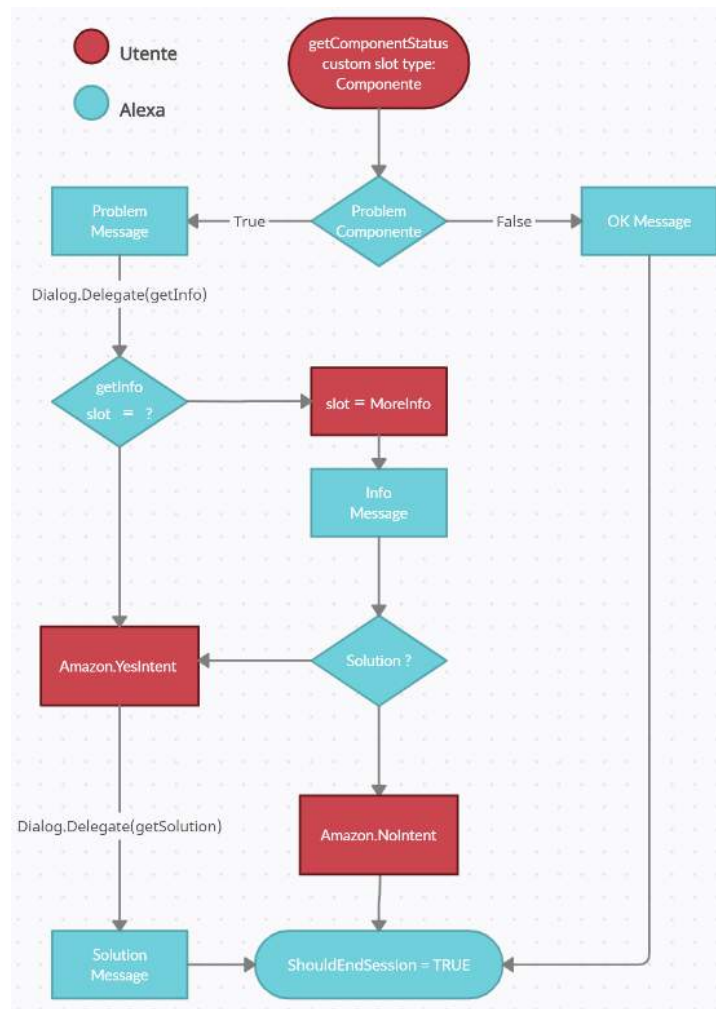


Figura 3.13: Flusso conversazione di diagnostica

inserito all'interno di una richiesta in formato JSON, la quale viene costruita da Alexa e inviata alla Skill, avviandone una nuova sessione.

Tutte le richieste [54] includono oggetti informativi a cui la Skill può accedere. Nel corpo di una richiesta possiamo trovare l' *oggetto sessione*, il quale fornisce un contesto aggiuntivo sulla richiesta. Degli esempi possono essere i parametri come "new", "sessionId" e "attributes", il quale consiste in una mappatura di coppie chiave-valore che se incluse nella proprietà "sessionAttributes" vengono ritrasmessi alla Skill nella richiesta successiva. Viene utilizzato questo metodo proprio per fare sì che Alexa mantenga il flusso della conversazione. Sempre nel corpo della richiesta è possibile trovare anche il *contesto*, che contiene informazioni sullo stato di Alexa al momento dell'invio, e gli *oggetti di richiesta* i quali contengono ogni dettaglio, come il tipo di richiesta e il nome dell'intent.

Alexa può inoltre creare diversi tipi di richiesta da inviare alla Skill, gli standard principali sono:

- *LaunchRequest*, inviata quando l'utente invoca la Skill senza fornire un intent specifico;
- *IntentRequest*, inviata quando l'utente effettua una richiesta che corrisponde a uno degli intents definiti nello schema;
- *SessionEndedRequest*, inviata quando la sessione della Skill corrente termina per qualsiasi motivo diverso dal codice che chiude la sessione.

L'intero codice scritto nella ***Lambda Function*** può essere visto come una grande lista di gestori per tutti gli intent definiti nel modello, la quale identifica le richieste, le elabora e ne restituisce una risposta, vedi Figura

```
{
  "version": "1.0",
  "session": {
    "new": false,
    "sessionId": "amzn1.echo-api.session.97f28ccd-144f-4231-b66e-459424ad3009",
    "application": {},
    "attributes": {},
    "user": {}
  },
  "context": {},
  "request": {
    "type": "IntentRequest",
    "requestId": "amzn1.echo-api.request.849d53d0-2a22-4011-a031-07fd53cc19c7",
    "locale": "it-IT",
    "timestamp": "2021-12-31T11:33:54Z",
    "intent": {
      "name": "getComponentStatus",
      "confirmationStatus": "NONE",
      "slots": {}
    },
    "dialogState": "COMPLETED"
  }
}
```

Figura 3.14: Esempio richiesta inviata da Alexa

3.15. Ogni *Handler Intent* solitamente accetta le richieste in base al tipo dell'intent desiderato solo dopo aver identificato quale intent soddisfa tale richiesta. Quindi la Skill ha il compito di indirizzare la richiesta dell'utente al relativo Handler Intent, successivamente il codice può fare tutto ciò di cui ha bisogno per soddisfarla. Tramite ASK SDK, è possibile implementare un Handler Intent definendo i metodi *canHandle()*, il quale permette di stabilire se la richiesta identificata può essere processata, e *handle()* che si occupa dell'elaborazione dei dati per fornire una risposta. Tutte e due i metodi accettano l'oggetto *handlerInput*, il quale contiene la richiesta completa in formato JSON inviata alla Skill. Tramite questo oggetto è possibile recuperare tutti i dati specificati nella richiesta, fornendo anche vari strumenti per elaborarla. Tra questi c'è la *requestEnvelope*, la quale rappresenta l'intero corpo della richiesta, permettendo all'handler di accedere a ogni dettaglio. Un altro strumento è il *responseBuilder*, che permette di costruire le rispo-

ste alle richieste. Una volta completato il lavoro, viene inviata una risposta, cioè una struttura JSON che dice ad Alexa cosa fare dopo.

```
const getComponent_Handler = {
  canHandle(handlerInput) {
    const request = handlerInput.requestEnvelope.request;
    return request.type === 'IntentRequest' && request.intent.name === 'getComponent' ;
  },
  async handle(handlerInput) {
    const request = handlerInput.requestEnvelope.request;
    const responseBuilder = handlerInput.responseBuilder;
    let sessionAttributes = handlerInput.attributesManager.getSessionAttributes();

    let say = 'I componenti a disposizione sono: ';
    lista_componenti=[];

    await getFromDb();

    for(let i=0; i<data.length;i++){
      say+= data[i].name+', ';
    }

    sessionAttributes.lastResult = say;
    sessionAttributes.nameIntent = request.intent.name;
    handlerInput.attributesManager.setSessionAttributes(sessionAttributes);
    return responseBuilder
      .speak(say)
      .reprompt(say)
      .withSimpleCard('componenti:', say)
      .getResponse();
  },
};
```

Figura 3.15: Handler di getComponent

Inoltre, come mostrato in Figura 3.15, viene reso possibile l'accesso all'*attributeManager* che è utilizzato per gestire gli attributi descritti precedentemente. Gli Handler Intents possono recuperare ed elaborare gli attributi memorizzati solo se la richiesta fa parte della stessa sessione, poiché al termine questi vengono persi. Come accennato poco fa, per sessione si intende un oggetto che fornisce un contesto aggiuntivo sulla richiesta. Nello specifico i *sessionAttributes* sono stati utilizzati per ricordare l'*ultima frase* detta da Alexa, così da poterla ripetere all'occorrenza e per memo-

rizzare l'*ultimo intent invocato*, permettendo di mantenere il flusso della conversazione.

Il *responseBuilder* fornisce dei metodi tramite i quali Alexa può essere istruito a parlare, tramite il comando *speak()* ed a mostrare delle cards tramite *withSimpleCard()*. Mediante il comando *getResponse()* viene fornita la risposta finale in formato JSON, come mostrato in Figura 3.16. Questa è composta dagli oggetti *sessionAttributes* e *response*, il quale contiene al suo interno:

- Un oggetto *outputSpeech*, per impostare ciò che Alexa deve dire dopo;
- Un oggetto *card*, ovvero schede grafiche che descrivono o migliorano l'interazione vocale;
- Un oggetto *reprompt*, contenente l'*outputSpeech* per impostare il messaggio di reprompt e indicare ad Alexa di ascoltare una risposta dall'altoparlante. L'utente ha pochi secondi, altrimenti Alexa chiude la sessione;
- Un oggetto *directives*, un array di direttive che specificano l'azione che il dispositivo deve intraprendere.

Infine attraverso lo *SkillBuilder* è possibile configurare e costruire un'istanza completa della Skill, il quale permette di instradare le richieste ricevute e le risposte fornite verso i corrispettivi Handler Intents. Esso deve includere tutti gli handlers definiti precedentemente e vanno inseriti nel giusto ordine poiché vengono elaborati dall'alto verso il basso.

```
"body": {
  "version": "1.0",
  "response": {
    "outputSpeech": {},
    "card": {},
    "directives": [],
    "type": "_DEFAULT_RESPONSE"
  },
  "sessionAttributes": {},
  "userAgent": "ask-node/2.7.0 Node/v14.18.1"
}
```

Figura 3.16: Risposta in formato JSON

```
const skillBuilder = Alexa.SkillBuilders.custom();
exports.handler = skillBuilder
  .addRequestHandlers(
    LaunchRequest_Handler,
    getCarStatus_Handler,
    AMAZON_CancelIntent_Handler,
    AMAZON_HelpIntent_Handler,
    AMAZON_StopIntent_Handler,
    RepeatIntent,
    AMAZON_NavigateHomeIntent_Handler,
    AMAZON_FallbackIntent_Handler,
    AMAZON_YesIntent_Handler,
    AMAZON_NoIntent_Handler,
    getComponentStatus_Handler,
    InProgressgetInfo_Handler,
    getComponent_Handler,
    getInfo_Handler,
    getSolution_Handler,
    getTagliando_Handler,
    SessionEndedHandler
  )
  .addErrorHandlers(ErrorHandler)
  .addRequestInterceptors(InitMemoryAttributesInterceptor)
  .addRequestInterceptors(RequestHistoryInterceptor)
  .lambda();
```

Figura 3.17: SkillBuilder

Capitolo 4

Funzionamento dell'applicazione

L'obiettivo del progetto è quello di realizzare un applicativo basato su Alexa Auto SDK, integrando al suo interno le funzionalità di un assistente virtuale automobilistico tramite la Skill precedentemente spiegata. Dunque, tale applicativo rappresenterà l'interfaccia tra l'utente e il veicolo, fornendo informazioni sullo stato di questo ultimo e, se necessario, assistenza. Per fare ciò dovrà essere in grado di recuperare i dati dal veicolo e renderli accessibili alla Skill, in modo tale da poter rilevare potenziali malfunzionamenti.

Come accennato in precedenza, una Skill è automaticamente attiva su tutti i dispositivi alimentati da Alexa collegati allo stesso account. Dopo che l'utente ha effettuato l'accesso con il profilo su cui è abilitata la Skill, le funzionalità principali di Alexa e la Skill stessa sono immediatamente utilizzabili. Per lo sviluppo dell'applicativo si è partiti dalla *Sample App* che fornisce l'SDK, tuttavia questa non restituisce un'interfaccia grafica per mostrare le funzionalità di Alexa perciò è stata implementata una GUI per integrare tali funzionalità e quelle nuove fornite dalla custom Skill *Car Maintenance*. L'IDE multiplatforma che ha semplificato l'implementazio-

ne di una GUI è stato *Qt Creator*, tale ambiente ha permesso lo sviluppo dell'applicativo utilizzando come linguaggio il *C++*.



Figura 4.1: QT Creator

4.1 Struttura

Come già enunciato durante la presentazione di Alexa Auto SDK, all'interno della Sample App sono presenti svariati moduli che permettono a l'utente di interagire con le funzionalità principali di Alexa. Prima di passare all'esposizione di queste ultime, è doveroso dare una panoramica su come è stata strutturata l'interfaccia grafica dell'applicativo.

Esso si presenta attraverso una finestra di dimensioni 1200x700 pixel, suddiviso in tre *BoxLayout* uno sopra l'altro e contenenti all'interno differenti *Widget*. Questi *Widget* [55] sono gli elementi primari per la creazione di interfacce utente in Qt. Essi possono visualizzare dati e informazioni sullo stato, ricevere input dall'utente e fornire un contenitore per altri widget che dovrebbero essere raggruppati insieme. Mentre un *BoxLayout* prende lo

spazio e lo divide in una riga di riquadri, facendo in modo che ogni widget gestito occupi un riquadro.

Il **primo BoxLayout**, situato più in alto di tutti, possiede al suo interno il widget di *Autenticazione* che è sostanzialmente collegato al modulo dell' *Autorizzazione CBL* di Alexa, necessario al fine di eseguire il login da parte dell'utente. A sua volta l'Autenticazione possiede altri tre widget:

- *Non Autenticato*, che possiede un solo bottone per iniziare il processo di autorizzazione;
- *Autenticazione in corso*, nel quale viene mostrato o un codice da inserire ad uno specifico indirizzo o un QR code da scannerizzare;
- *Autenticato*, si accede a questo solo in caso di autorizzazione concessa e possiede un pulsante per la lingua, uno per eseguire il logout e uno che quando viene premuto, invoca e attiva Alexa, accendendo il microfono e lasciando parlare il conducente. Vedi Figura 4.2.

Solo uno di questi widget alla volta verrà reso visibile all'utente, in base a quale è il suo stato di autenticazione.

Il **secondo BoxLayout**, in posizione centrale, ospita la pagina *Home* del sistema e, a seguito di una navigazione o di un'interazione con Alexa, le varie schermate delle applicazioni contemplate nel progetto dimostrativo: *Meteo*, *MediaPlayer*, *Clima bizona*, *Diagnosi Vettura*, *Alerting Notification* e *Local Search*.

Questi widget sono gestiti tramite il **terzo BoxLayout**, il quale è composto da cinque bottoni che permettono di passare da una schermata all'altra. Tali bottoni permettono di raggiungere la *Home page*, *Weather page*,

Media page, *Clima page* e *Diagnostic page*. Tuttavia, questo non è l'unico metodo per navigare nell'applicativo, poiché tramite l'interazione vocale con Alexa è possibile raggiungere le stesse pagine. In questo ultimo BoxLayout è presente anche un widget chiamato *voiceChrome*, in breve è un indicatore visivo del sistema di attenzione di Alexa e viene visualizzato ogni volta che il conducente interagisce vocalmente con Alexa.

Oltre alla struttura di backend, si è progettato anche il frontend dell'applicativo, essendo tutte le schermate del dimostratore progettate congiuntamente dal sottoscritto e dal Dott. David Berti, che ne ha curato lo stile e l'approccio stilistico-visivo.

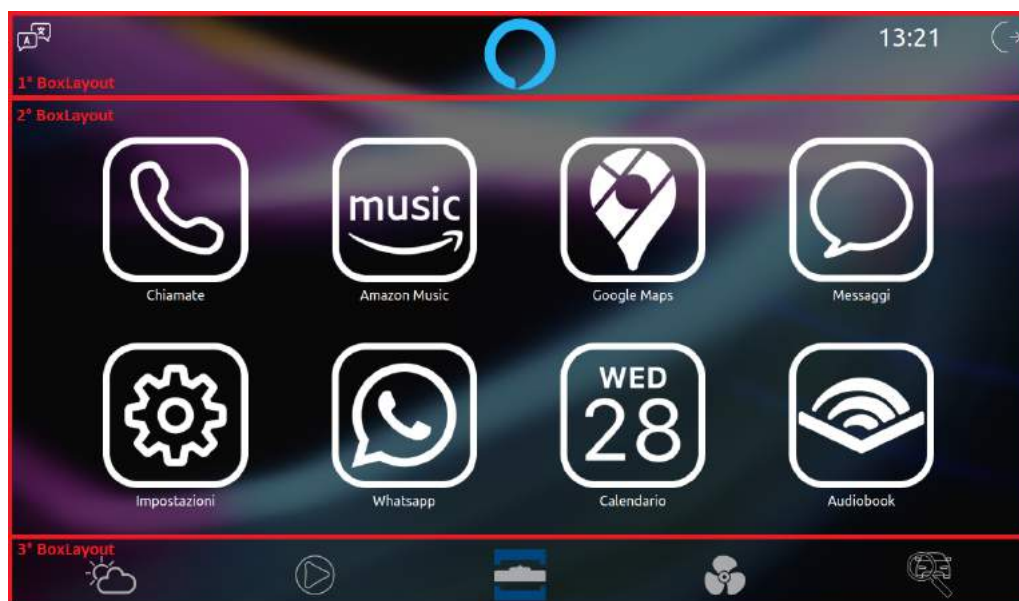


Figura 4.2: Struttura interfaccia grafica, HomePage

4.2 Fase di Autenticazione

Per permettere all'utente di eseguire l'accesso al proprio account e utilizzare così i servizi di Alexa è richiesto un token di accesso. Il modulo *CBL* implementa il meccanismo per l'acquisizione di tali token e attraverso l'interfaccia *Authorization* è possibile avviare, annullare e disconnettersi dal servizio.

Per avviare l'autorizzazione CBL viene utilizzato *startAuthorization()* e come parametri vengono passati "alexacbl" che ne indica il tipo e se l'applicazione sta avviando una nuova sessione di autorizzazione CBL, viene passata una stringa vuota come parametro dati. Altrimenti se la sessione viene avviata con un token di aggiornamento esistente, questo viene fornito tramite *getAuthorizationData()* con "refreshToken" come chiave.

Lo stato di autorizzazione passa così da "UNAUTHORIZED" a "AUTHORIZING". L'auto SDK richiede l'autorizzazione al servizio di login di Amazon restituendo la coppia "code" e "url" all'applicazione tramite il parametro "event" del metodo "eventReceived". A questo punto l'utente viene o reindirizzato all'indirizzo url dove dovrà inserire il codice fornito, oppure scannerizzare un QRcode, il quale evita di far inserire il codice all'utente. Successivamente viene invitato ad inserire le proprie credenziali, se queste risultano essere corrette allora l'SDK imposta i dati di autenticazione tramite *setAuthorizationData()*. L'applicazione si salva il token in modo che al prossimo avvio non vada rifatto nuovamente l'accesso. Inoltre se il profilo utente è configurato vengono passati anche i relativi dati affinché l'applicazione li utilizzi. Infine viene cambiato lo stato in "AUTHORIZED". Vedi Figura 4.3.

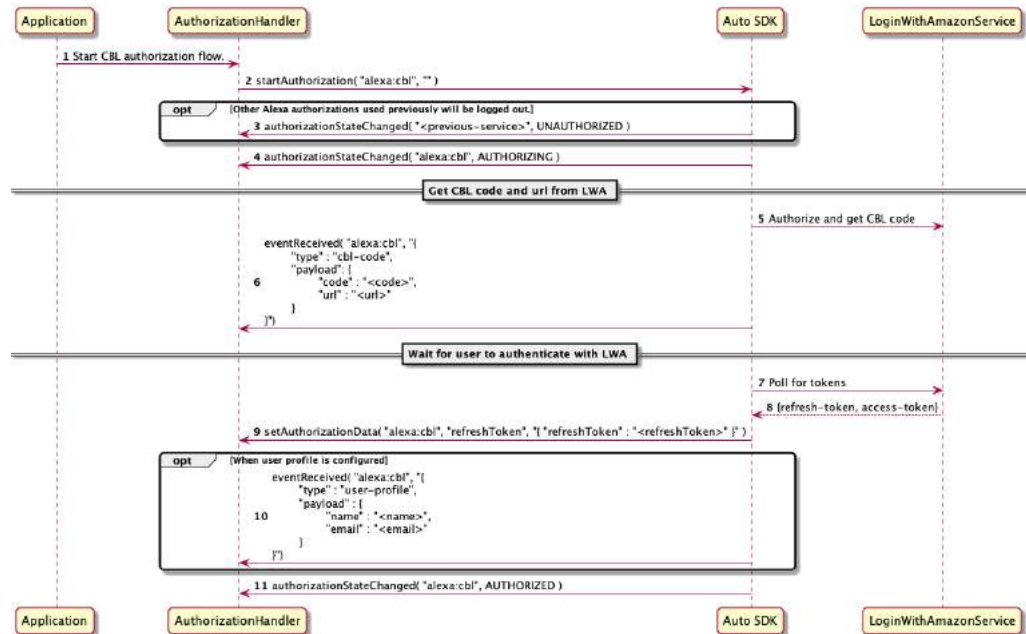


Figura 4.3: Flusso di avvio dell'autorizzazione CBL

4.3 Panoramica delle pagine

Una volta eseguito l'accesso viene restituita all'utente la *Home Page* come mostrato in Figura 4.2. In questa sono presenti otto *ImageButton* strutturati in due file da quattro tramite un *GridLayout* e rappresentano rispettivamente le *Chiamate*, le *Impostazioni*, i *Messaggi*, il *Calendario*, *Google Maps*, *Amazon Music* e *Audiobook*. Tuttavia, questi servizi sono stati aggiunti per simulare esteticamente i sistemi di infotainment esistenti e quindi non vengono realmente implementati.

Per dare una panoramica delle restanti pagine sviluppate è necessario introdurre la classe *TemplateRuntimeHandler* [56] del modulo *Alexa*, la quale permette di ottenere un payload in formato JSON che se accuratamente analizzato, in base al tipo, restituisce i dati per popolare le rispettive pa-

gine. I principali metodi pubblici al suo interno sono *renderTemplate()* che fornisce metadati visivi associati a una richiesta dell'utente ad Alexa (vedi Figura 4.4) e *renderPlayerInfo()* il quale fornisce sempre metadati visivi solo che in questo caso riguardano una richiesta per la riproduzione audio.

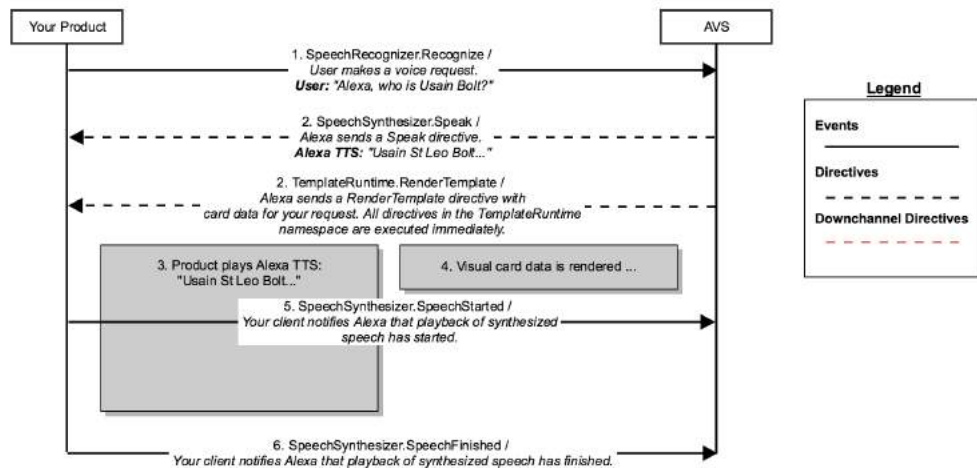


Figura 4.4: Data Flow tra prodotto e AWS

A questo punto vengono illustrate le pagine sviluppate:

1. **Weather Page**, la quale permette all'utente di ottenere le informazioni metereologiche. Dunque, quando viene eseguita una richiesta che comprende informazioni sul meteo, la classe *TemplateRuntimeHandler* fornisce le relative informazioni tramite il metodo *renderTemplate()* nel quale viene eseguito un "parse" del payload. In questo modo i dati sono nel giusto formato per poter essere utilizzati e mostrati a schermo tramite il widget *WeatherCard*, nel quale vengono fornite informazioni come la temperatura corrente, massima, minima e le previsioni per i giorni a venire. Vedi Figura 4.5



Figura 4.5: Weather Page

2. **Media Page**, questa rende possibile la riproduzione di musica tramite radio o altri provider come Amazon Music e Spotify. Come detto precedentemente i metadati riguardanti una richiesta di riproduzione audio vengono recuperati dalla classe *TemplateRuntimeHandler* tramite *renderPlayerInfo()* per poi essere passati al widget *mediaPlayerCard*. Tuttavia, questo non basta poiché è necessaria anche la classe *AudioPlayerHandler* che informa il *mediaPlayer* dei cambiamenti nello stato. Questo può essere utilizzato per aggiornare la GUI della relativa pagina, ad esempio quando lo stato del *mediaPlayer* passa in "PLAYING", il metodo *playerActivityChanged()* rileva tale cambiamento, aggiornando la pagina da visualizzare e notificando ogni secondo l'evento *onGetPlayerPositionAndDuration* che permette di aggiornare la barra di avanzamento del brano. Infine per controllare la riproduzione

multimediale tramite gli appositi pulsanti viene sfruttata l'interfaccia *PlaybackController*. Se un utente preme il pulsante di pausa o semplicemente pronuncia "pausa", allora il mediaPlayer richiama un metodo dell'interfaccia per segnalare il comando. Vedi Figura 4.6.

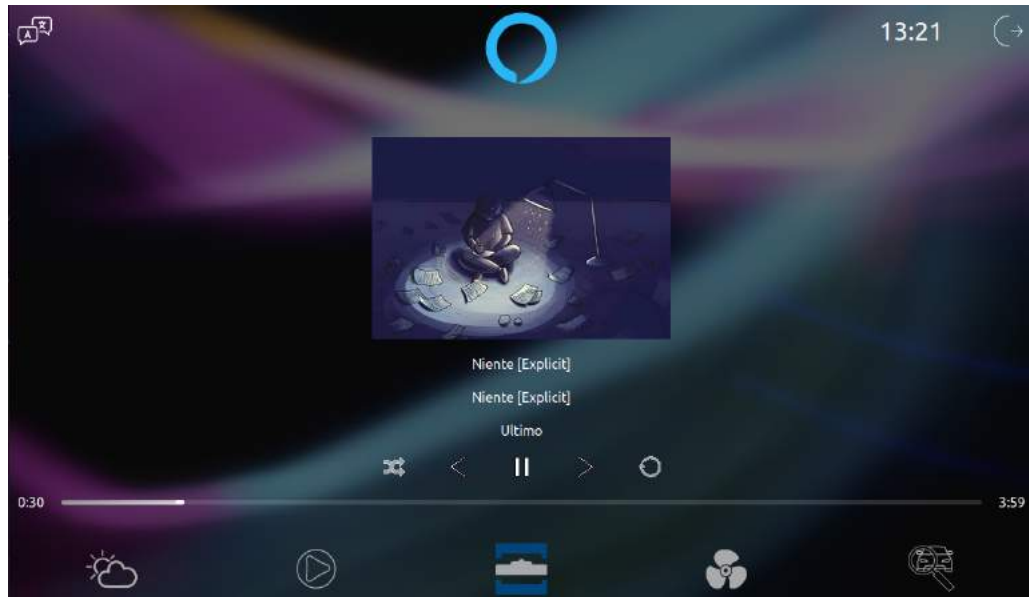


Figura 4.6: Media Page

3. **Clima Page**, in questa sezione l'utente è in grado di regolare la *temperatura* generale del veicolo, la *ventilazione*, scegliere se accendere l'*aria condizionata*, lo *sbrinatori* o il *riscaldamento dei sedili*. Per controllare le funzioni del veicolo tramite Alexa, l'Auto SDK fornisce un modulo chiamato *Car Control*. Qui è possibile definire degli *Endpoints* che rappresentano una parte controllabile del veicolo, come ad esempio la temperatura. Per stabilire come possono essere controllati questi endpoint e quindi quali sono le rispettive funzionalità, il modulo supporta quattro interfacce:

- *Power Controller*, controlla lo stato di alimentazione generale di un endpoint;
- *Toggle Controller*, controlla una particolare proprietà di un endpoint che può essere attivata e disattivata;
- *Mode Controller*, controlla una particolare proprietà di un endpoint che può essere impostata su un valore discreto da un insieme definito di valori;
- *Range Controller*, controlla una particolare proprietà di un endpoint che può essere impostata su un valore numerico all'interno di un intervallo.

Inoltre è possibile utilizzare differenti controller per gestire lo stesso endpoint, ad esempio per accendere e spegnere la ventilazione o per impostare una certa velocità.

Ogni endpoint può appartenere a zero, uno o più *Zone*. Le zone, configurate con gli endpoint membri, definiscono le regioni denominate del veicolo e consentono agli utenti di scegliere come target gli endpoint in base alla posizione. Ad esempio, la definizione delle zone "conducente" e "passeggero" e l'assegnazione degli endpoint "sedile" distinti a ciascuna consente il corretto controllo del "sedile conducente" e del "sedile passeggero" in modo indipendente.

Infine le definizioni di endpoint, controller e zone includono gli *Assets*. Questi sono identificati da ID univoci, raggruppano un nome descrittivo come "aria condizionata" in un gruppo denominato di sinonimi e traduzioni per tutte le lingue supportate.

Dunque, tramite questo modulo è possibile aggiornare l'interfaccia utente, fornendo dati aggiornati per il widget *CarControlCard*.



Figura 4.7: Clima Page

4. **Diagnostic Page**, questa è la pagina che viene visualizzata quando si richiama la custom Skill "Car Maintenance". Tramite questa pagina è possibile verificare le condizioni del veicolo e di componenti specifici, inoltre in caso di problemi viene ricevuta assistenza da parte di Alexa. Il sistema così permette di superare le funzionalità di base fornite da Amazon introducendo nuove abilità di diagnosi, consentendo il miglioramento dell'esperienza utente in auto. Come da Figura 4.8. La sua struttura è formata da un `BoxLayout` verticale posizionato sulla sinistra della schermata dentro il quale sono presenti: una serie di spie che segnalano la presenza di un problema al veicolo, una label per la distanza totale e una per la distanza parziale percorse, il livello del

carburante, della pressione dell'olio e infine il numero di chilometri restanti, allo scadere dei quali è consigliabile effettuare un intervento di manutenzione o tagliando del veicolo. Quando si passa in questa sezione viene aggiunta una macchina stilizzata nello sfondo e vengono associate quattro label rappresentanti la pressione delle rispettive ruote.

Le principali funzionalità sono:

- Verificare se un singolo componente presenta malfunzionamenti, mostrando a schermo la rispettiva spia e fornendo informazioni sulla possibile fonte del problema e sulle possibili soluzioni;
- Controllare l'intero veicolo per verificare che sia tutto funzionante, in caso contrario elencare i componenti che hanno riscontrato un problema;
- Chiedere maggiori informazioni per un messaggio di avviso;
- Aiutare a trovare una soluzione per alcuni problemi dell'auto, dunque fornendo assistenza al conducente;
- Ottenere informazioni dal veicolo come il numero di chilometri restanti prima del prossimo tagliando;
- Fornire informazioni relative su come utilizzare la custom Skill.

I dati presenti nella relativa pagina e che rappresentano i valori dei singoli componenti sono stati simulati direttamente all'interno dell'applicativo non avendo a disposizione una centralina di un'auto. Tuttavia, i dati della macchina in realtà girano su un *bus CAN* (Controller

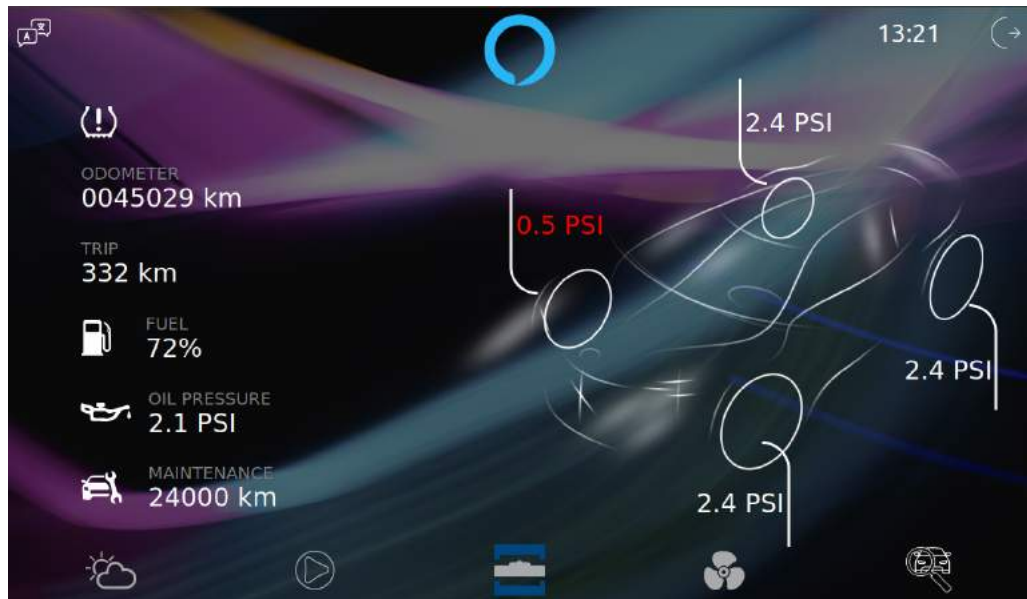


Figura 4.8: Diagnostic Page

Area Network) sotto forma di pacchetti chiamati "frame". Ogni frame ha un payload, da cui si individua che tipo di dato rappresenta e quale è il suo valore. Questi metadati ottenuti randomicamente vengono utilizzati per aggiornare l'interfaccia utente passandoli al widget *DiagnosticCarCard* e per informare Alexa sullo stato del veicolo. Per mantenere l'assistente vocale aggiornato sulle condizioni dell'auto viene sfruttato *DynamoDB*, un database da cui attingere tali informazioni. In caso di criticità, Alex notifica con un messaggio vocale la presenza di uno o più alert e consiglia di aprire Car Maintenance per avere maggiori informazioni al riguardo.

Una volta aperta la custom Skill e identificato il problema, viene fornita una card rappresentante la spiegazione approfondita di questo ultimo, come rappresenta la Figura 4.9, e se richiesto viene restitui-

ta un'altra card con la relativa soluzione al problema, come mostra la Figura 4.10. Tali avvisi vengono percepiti come un singolo segnale, poiché Alexa legge il problema o la soluzione, accompagnandoli visivamente con una card, permettendo così di ridurre il carico cognitivo. Quando il conducente richiama la Skill, le risposte di Alexa vengono recuperate dalla classe *TemplateRuntimeHandler* tramite il solito metodo *renderTemplate()*. Questo ha permesso sia di identificare la richiesta di apertura della pagina, sia la costruzione della card di informazione e della card di soluzione.



Figura 4.9: Information Card in Diagnostic Page

In caso in cui il conducente esegua una *ricerca locale*, per trovare ad esempio una stazione di servizio o un meccanico, viene restituita una pagina contenente il widget *LocalSearch* in cui vengono illustrati i risultati della ricerca tramite una lista. Le informazioni sono sta-



Figura 4.10: Solution Card in Diagnostic Page

te ottenute sempre mediante il metodo *renderTemplate()* della classe *TemplateRuntimeHandler*, nel quale i metadati recuperati vengono sottoposti al parsing e poi utilizzati per riempire gli elementi del widget.

4.4 DynamoDB

Per poter condividere i dati del veicolo con la custom Skill di Alexa e per permettere che questo ultimo possa comunicare con l'applicativo, viene implementato il *DynamoDB*, che è un database serverless NoSQL fornito da AWS. Viene definito tramite una struttura di archivio chiave-valore e adottata un'architettura distribuita per un'elevata disponibilità e scalabilità, permettendo di eseguire applicazioni ad alte prestazioni su qualsiasi scala. Un

database NoSQL, o non relazionale, è costruito per modelli di dati specifici e memorizza i dati in altri formati, non solo tabelle relazionali. I modelli NoSQL consentono di annidare tutti i dati correlati all'interno di un'unica struttura, senza la necessità di dividerli tra tabelle. Dunque, tali modelli sono personalizzati per il caso d'uso specifico, quindi possono supportare meglio carichi di lavoro più grandi, fornendo prestazioni migliori rispetto a quelli relazionali per lo stesso caso d'uso. Tali dati sono organizzati in tabelle, che contengono elementi, dove ognuno di essi contiene una serie di coppie chiave-valore di attributi.

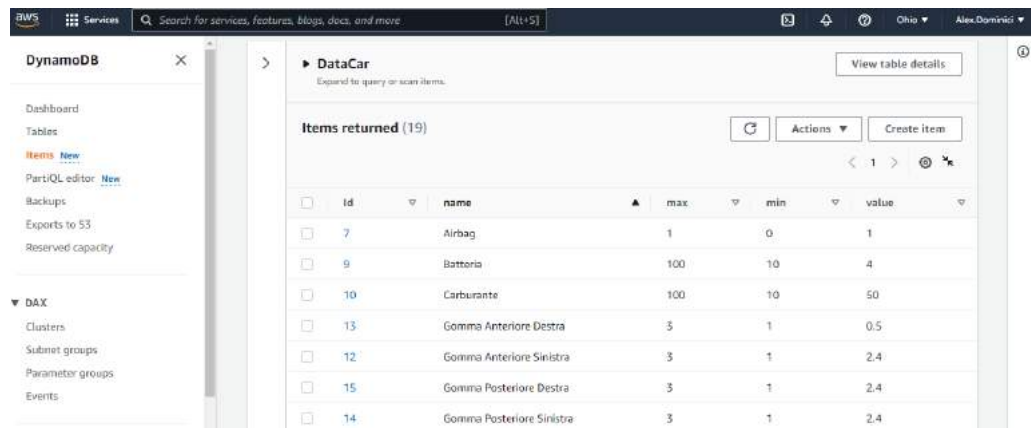
Esistono due tipi speciali di attributi: il *primary-key*, che funziona in modo simile a un ID, e il *sort-key*, che consente di ordinare gli elementi.

Dunque, i concetti principali in un DynamoDB sono:

- *Table*: una collezione che può contenere un numero virtualmente infinito di elementi;
- *Item*: l'unità più basilare in Dynamo, contiene gli attributi dei dati strutturati in un JSON;
- *Attribute*: una coppia chiave-valore che contiene dati informativi su un elemento della tabella;
- *Primary Key*: una forma speciale di attributo che viene utilizzata per fare riferimento agli elementi, in modo simile a un ID;
- *Sort Key*: un'altra forma speciale di attributo che viene utilizzata per organizzare gli elementi in un diverso ordine;
- *Query*: operazione per recuperare un particolare elemento (o insieme di elementi);

- *Scan*: operazione per scansionare l'intera tabella o una sua sezione;
- *Filter*: regole da applicare dopo l'esecuzione di una query o di una scansione, ma prima che i risultati vengano restituiti al richiedente.

Attraverso la console DynamoDB, le tabelle possono essere facilmente gestite, dalla creazione, all'aggiunta, eliminazione e interrogazione dei dati. In Figura 4.11 è possibile vedere la tabella con i componenti dell'auto come elementi.



	id	name	max	min	value
<input type="checkbox"/>	7	Airbag	1	0	1
<input type="checkbox"/>	9	Batteria	100	10	4
<input type="checkbox"/>	10	Carburante	100	10	50
<input type="checkbox"/>	13	Gomma Anteriore Destra	3	1	0.5
<input type="checkbox"/>	12	Gomma Anteriore Sinistra	3	1	2.4
<input type="checkbox"/>	15	Gomma Posteriore Destra	3	1	2.4
<input type="checkbox"/>	14	Gomma Posteriore Sinistra	3	1	2.4

Figura 4.11: Tabella DynamoDB

Inoltre fornisce anche strumenti per connettere altri tipi di applicazioni ad esso tramite *SDK AWS*. Questo ultimo è una libreria open source, multiplatforma per il linguaggio C++ che viene utilizzata per connettersi ad Amazon Web Services.

4.4.1 Connessione tra applicativo e DynamoDB

Prima di poter accedere alle risorse AWS e quindi alla tabella del database per poterla aggiornare, devono essere stabilite le credenziali utente

AWS nel relativo ambiente. Per utilizzare l'SDK, viene creato un utente *AWS Identity and Access Management* (IAM), ottenendo le credenziali (*Access Key ID*, *Secret Access Key*) e rendendole disponibili per l'SDK nell'ambiente di sviluppo. Le applicazioni che utilizzano l'SDK AWS per C++ devono inizializzarlo chiamando *Aws::InitAPI*. Allo stesso modo, prima che l'applicazione venga terminata, l'SDK deve essere arrestato chiamando *Aws::ShutdownAPI*. Entrambe le operazioni accettano opzioni di configurazione tramite un argomento di *Aws::SDKOptions*, il quale influisce sui processi di inizializzazione, arresto e sulle successive chiamate all'SDK. Un esempio delle opzioni disponibili è l'attivazione dell'accesso utilizzando il logger predefinito. Una volta inizializzato l'SDK AWS viene dichiarata la configurazione del client mediante *ClientConfiguration* tramite la quale è possibile definire per esempio la regione di utilizzo. Infine per effettuare la connessione con il servizio DynamoDB viene inizializzato il rispettivo client tramite *DynamoDBClient* passando come parametri le credenziali ottenute tramite AWS IAM e la configurazione del client. Vedi Figura 4.12.

```
Aws::SDKOptions options;
options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;
Aws::InitAPI(options);
Aws::Client::ClientConfiguration clientConfig;

auto key_id = "AKIAUTIOER7BATJS3WRH";
auto key = " ";
Aws::Auth::AWSCredentials credentials = Aws::Auth::AWSCredentials(key_id, key);
clientConfig.region = "us-east-2";
Aws::DynamoDB::DynamoDBClient dynamoClient(credentials, clientConfig);
```

Figura 4.12: Configurazione client DynamoDB

A questo punto è possibile aggiornare un attributo per un elemento che esiste già in una tabella utilizzando il metodo *UpdateItem* di DynamoDB-

Client, fornendo il nome della tabella, il valore della Primary Key , il valore della Sort Key, i campi da aggiornare e il valore corrispondente. Figura 4.13.

```
//Update all attributes|
for(int a=0;a<length;a++){
    Aws::DynamoDB::Model::UpdateItemRequest request;
    request.SetTableName("DataCar");
    // Define KeyName argument
    Aws::DynamoDB::Model::AttributeValue attribValue;
    attribValue.SetS(std::to_string(a));
    std::cout << "*****" << std::to_string(a)<<std::endl;
    request.AddKey("Id", attribValue);

    Aws::DynamoDB::Model::AttributeValue attribValue1;
    attribValue1.SetS(name[a]);
    std::cout << "*****" << std::to_string(a)<<std::endl;
    request.AddKey("name", attribValue1);

    // Construct the SET update expression argument
    Aws::String update_expression("SET #a = :valueA");
    request.SetUpdateExpression(update_expression);

    // Construct attribute name argument
    Aws::Map<Aws::String, Aws::String> expressionAttributeNames;
    expressionAttributeNames["#a"] = "value";
    request.SetExpressionAttributeNames(expressionAttributeNames);

    // Construct attribute value argument
    Aws::DynamoDB::Model::AttributeValue attributeUpdatedValue;
    // Set random value for all components
    if(a<9){
        int random= distr2(generator);
        attributeUpdatedValue.SetN(random);

        value.push_back(random);
        min.push_back(0);
        max.push_back(1);
    }else if(a>8&&a<12){ ... }
    }else if(a>11&&a<17){ ... }
    }else if(a>16){ ... }

    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> expressionAttributeValues;
    expressionAttributeValues[":valueA"] = attributeUpdatedValue;
    request.SetExpressionAttributeValues(expressionAttributeValues);

    // Update the item
    const Aws::DynamoDB::Model::UpdateItemOutcome& result2 = dynamoClient.UpdateItem(request);
    if (!result2.IsSuccess())
    {
        std::cout << result2.GetError().GetMessage() << std::endl;
    }
    std::cout << "Item was updated" << std::endl;
};
```

Figura 4.13: Update tabella DynamoDB

4.4.2 Connessione tra custom Skill e DynamoDB

Per recuperare i dati del DynamoDB tramite la Skill, è necessario dare alla rispettiva lambda function i permessi per accedere alla tabella, vedi Figura 4.14. Questi vengono rilasciati creando un ruolo in IAM AWS al quale viene associata una policy che consente l'accesso alla tabella del DynamoDB, nello specifico "AmazonDynamoDBFullAccess". Come si può evincere dalla Figura 4.15.



Figura 4.14: Configurazione permessi Lambda Function

A questo punto per leggere gli elementi della tabella viene definita un'istanza della classe *AWS.DynamoDB.DocumentClient* nella lambda function. Tale classe semplifica l'utilizzo degli elementi in Amazon DynamoDB astruendo la nozione di AttributeValues. Questa astrazione annota i tipi JavaScript nativi forniti come parametri di input e converte i dati di risposta in tipi JavaScript. Attraverso il metodo *Scan()* della classe stessa vengono restituiti uno o più elementi e attributi di una tabella, la quale viene passata come parametro al metodo. Inoltre possono essere passati ulteriori parametri al metodo, come ad esempio un'espressione per filtrare la scansione della tabella. Come è possibile vedere in Figura 4.16 sono state definite due funzioni per recuperare i dati dal DynamoDB. La prima permette di recu-

Descrizione e Funzionamento applicativo

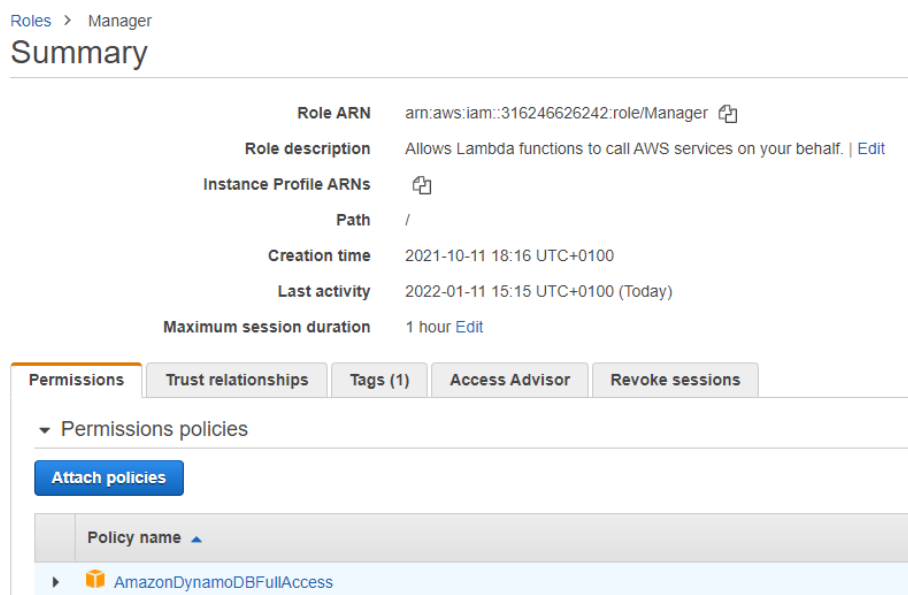


Figura 4.15: Policy applicata ruolo

perare tutti gli elementi della tabella, così da averli disponibili in caso di perdita di connessione. La seconda invece esegue una scansione della tabella secondo il nome dell'elemento richiesto, evitando di recuperare tutti valori della tabella.

4.4.3 Alternativa: MongoDB

DynamoDB non è l'unico ad offrire un servizio di database compatibile con Alexa Skills Kit. Ad esempio, esistono valide alternative, tra cui *MongoDB* [57], database NoSQL distribuito basato su documenti per uso generale. Le differenze più significative tra questi due sono che il MongoDB è indipendente dalla piattaforma, con la possibilità di configurare il database in modo che possa essere eseguito praticamente ovunque. Gli utenti che utilizzano MongoDB sono tenuti a gestire tutta l'infrastruttura e le confi-

```
async function getFromDb() {
  var docClient = new AWS.DynamoDB.DocumentClient({apiVersion: '2012-08-10'});
  var params = {
    TableName: 'DataCar';
  };
  let promise = await docClient.scan(params);
  let result = await promise.promise();
  data = result.Items;
  return data;
}

async function getValueFromDbByName(name) {
  var docClient = new AWS.DynamoDB.DocumentClient({apiVersion: '2012-08-10'});
  var promise = await docClient.scan({
    TableName: 'DataCar',
    FilterExpression: '#name = :nm',
    ExpressionAttributeValues: {' :nm': name},
    ExpressionAttributeNames: {"#name": "name"};
  });
  let result = await promise.promise();
  let lista= [result.Items[0].value, result.Items[0].min, result.Items[0].max];
  return lista;
}
```

Figura 4.16: Metodi per leggere la tabella del DynamoDB

gurazioni, comportando una maggiore complessità. Lo stesso discorso vale per la robustezza del database, gli utenti sono responsabili delle pratiche di sicurezza e sebbene ciò offra un maggiore controllo, è un'attività complessa e dispendiosa in termini di tempo.

Dall'altra parte il DynamoDB è fortemente limitato ad AWS e può essere utilizzato solo tramite questo ultimo. Essendo completamente gestito consente agli utenti di iniziare a utilizzare il database immediatamente, senza dover occuparsi della sicurezza del database, poiché il servizio viene gestito da AWS basandosi su IAM che offre un controllo dettagliato su ruoli e policy utilizzati. In conclusione, selezionare il database giusto non è una scelta semplice poiché dipende da una moltitudine di fattori. Anche quando si confrontano MongoDB e DynamoDB, non è possibile farlo direttamente poiché sono destinati a casi d'uso diversi. Ad esempio, DynamoDB è un servizio di database NoSQL gestito, mentre MongoDB è un software di database NoSQL. Pertanto, il confronto diretto più vicino sarebbe con il database cloud

MongoDB Atlas. La scelta finale è ricaduta su DynamoDB poiché offre il meglio in *compatibilità, facilità d'uso e integrazione*.

4.5 Casi d'uso

In questa ultima sezione del capitolo vengono esposti i principali casi d'uso dell'applicativo e della rispettiva Skill.

- *Autenticazione utente*: Dopo aver visto nel dettaglio la logica che c'è nella fase di autenticazione dell'utente vengono mostrate le interfacce con cui interagisce questo ultimo per portare a termine tale fase. In Figura 4.17 viene riportata la pagina contenente il link e il codice per eseguire l'accesso, in alternativa viene fornito il QRcode in cui è associato il link con il codice già integrato.



Figura 4.17: Interfaccia autenticazione utente

Descrizione e Funzionamento applicativo

Viene quindi richiesto l'inserimento delle proprie credenziali e se non è stato scansionato il QRcode, va fornito anche il codice restituito dall'applicativo, esempio in Figura 4.18. Questo permetterà di identificare il dispositivo per poterlo registrare al proprio account Amazon. Nel caso in cui la registrazione viene portata a termine correttamente, il dispositivo si aggiornerà automaticamente, mostrando a schermo la *Home Page*, vedi Figura 4.2.



Figura 4.18: Login utente

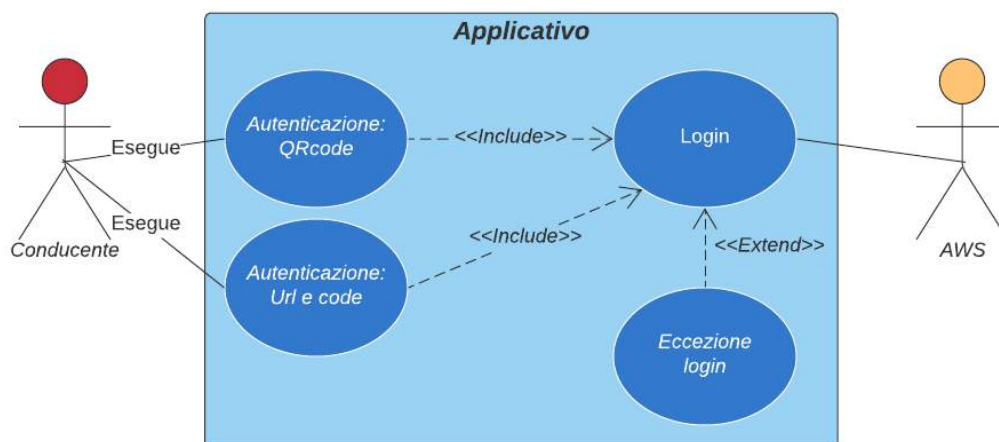


Figura 4.19: Caso d'uso di Autenticazione

- Riproduzione Skill Car Maintenance:** In questo caso il conducente può aprire la Skill come risposta della notifica vocale di Alexa, la quale segnala la presenza di qualche malfunzionamento, oppure per proprio interesse personale. Se è la prima volta che l'utente usufruisce del servizio di diagnostica, la Skill fornisce delle informazioni sul come utilizzarla al meglio. Altrimenti l'utente può direttamente richiedere lo stato generale della macchina come in Figura 4.20, in questo modo Alexa controlla il database e riporta la lista dei componenti che hanno riscontrato un possibile malfunzionamento. Nel caso in cui il conducente fosse già a conoscenza di quale sia il componente che ha riportato un problema, può ottenere maggiori informazioni per questo ultimo (vedi Figura 4.9) o richiedere subito una soluzione da parte di Alexa (vedi Figura 4.10).

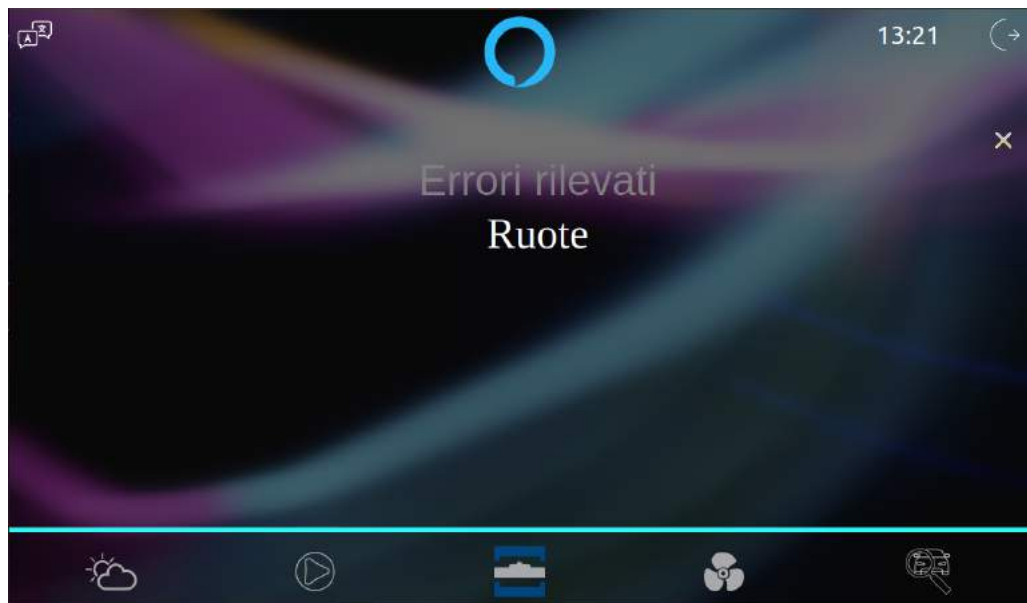


Figura 4.20: Interfaccia controllo generale auto

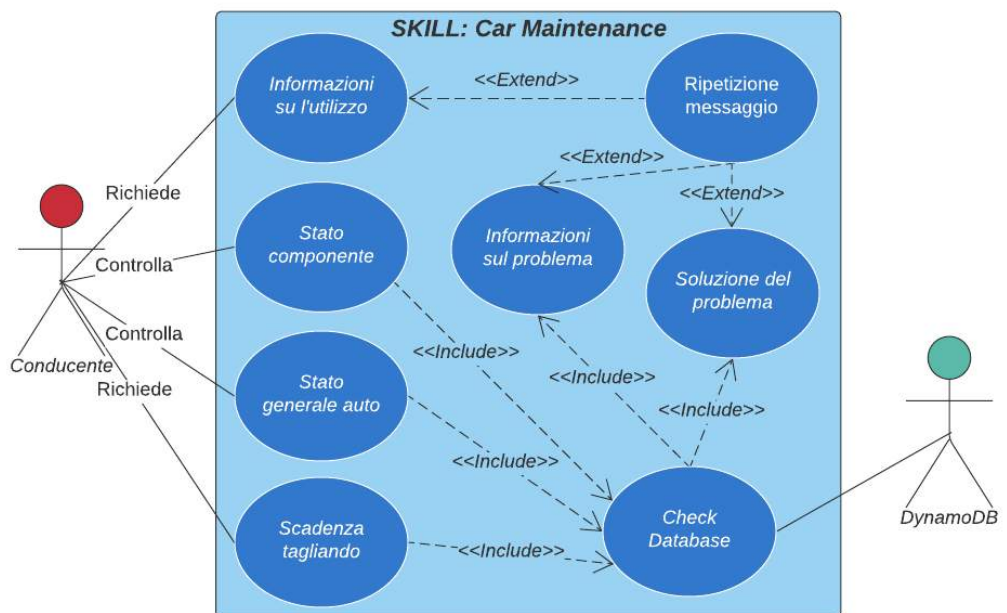


Figura 4.21: Caso d'uso Skill Car Maintenance

- *Ricerca Locale*: L'ultimo caso d'uso preso in considerazione è la ricerca locale di un meccanico da parte del conducente. Questo caso specifico può essere scaturito a seguito dell'esecuzione della Skill, tramite la quale si è venuti a conoscenza di un problema tale per cui Alexa ha consigliato di recarsi in un'autofficina. Nello specifico viene eseguita una ricerca locale, basandosi sulla posizione geografica dell'abitacolo e vengono restituiti successivamente a schermo i primi quattro risultati. L'aggiornamento di tale posizione avviene tramite l'interfaccia *LocationProvider* fornita da Alexa Auto SDK.



Figura 4.22: Interfaccia ricerca locale

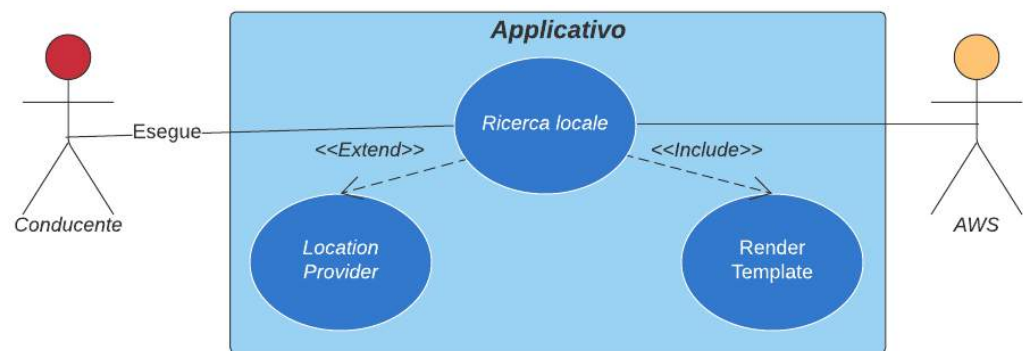


Figura 4.23: Caso d'uso ricerca locale

Conclusioni

In questa tesi sono state presentate le attività svolte per lo sviluppo di un'applicazione in grado di fornire diagnostica e assistenza, basata sul servizio Amazon Alexa.

Partendo dalla Sample App di *Alexa Auto SDK* è stata costruita un'interfaccia grafica che rendesse l'interazione tra l'utente e il veicolo il più semplice possibile. Inoltre per ovviare alla mancanza di un servizio di diagnostica del veicolo e assistenza del conducente da parte di Alexa, viene sviluppata e abilitata la *custom Skill* "Car Maintenance". La creazione di questa ultima è stata resa possibile tramite la combinazione della console per sviluppatori di Alexa e la *lambda function*. Tuttavia, per far sì che la Skill potesse recuperare ed elaborare i dati della macchina viene sfruttato *DynamoDB*, un database NoSQL in cui vengono memorizzati i valori dei vari componenti dell'auto. Adesso oltre a fornire informazioni sul meteo, a riprodurre contenuti multimediali o a gestire il clima della vettura, è anche in grado di fornire una vera e propria assistenza al conducente in caso di guasti.

Dunque, si può affermare di aver raggiunto gli obiettivi preposti, ovvero la possibilità di realizzare un'applicazione in grado di diminuire il carico cognitivo del conducente e migliorando così la *User Experience* a bordo vei-

colo. L'introduzione di un sistema di diagnostica nell'ambiente di Alexa va a soddisfare quella che è la richiesta dei clienti, ovvero avere un'assistente vocale che non solo permetta di mantenere coesa l'esperienza dentro e fuori l'auto, ma che si occupi sempre più anche della sua gestione. Tutto ciò tende a mitigare le criticità prodotte da *driver distraction* o avarie della vettura. Un ulteriore vantaggio risiede nel fatto che in questo modo gli utenti hanno già un'idea di come interagire con l'assistente, poiché già utilizzato in altri ambiti e quindi non è necessario imparare nuovamente ad utilizzare il sistema di bordo. Infine questo gioca un ruolo importante anche per l'interesse delle Case Automobilistiche che vogliono aggiudicarsi una buona fetta di mercato. Esse potrebbero semplicemente offrire soluzioni di Amazon o Google (o entrambi) per allinearsi alle preferenze dei consumatori. La maggioranza dei marchi però non vuole cedere gran parte dell'esperienza di guida in auto a terzi. Dunque, per rispondere a questa esigenza ora forniscono anche opzioni parallele o ibride. I sistemi paralleli includono la possibilità di eseguire, ad esempio Apple CarPlay o un assistente integrato ma non contemporaneamente. Le soluzioni ibride in genere hanno un manager di riconoscimento vocale che soppesa la volontà dell'utente e quale tra più assistenti può soddisfare al meglio la richiesta che poi lascia assumere la responsabilità della risposta.

Tuttavia, a valle di quanto esposto in dettaglio nei precedenti capitoli, alcuni sviluppi futuri di questa tesi potrebbero essere:

- *Predizioni Intelligenti*: per migliorare il sistema di diagnostica un'attività futura che potrebbe essere approfondita è la costruzione di un modello di rete neurale, il quale prendendo come input i dati della

macchina sia in grado di fornire una predizione sui possibili guasti o problemi a cui il conducente può andare incontro. Amazon, garantendo la rispettiva privacy, potrebbe allenare tale modello su una quantità di dati molto vasta, così da poter garantire previsioni più accurate e un'assistenza migliore.

- *Modalità Offline*: oggi Alexa Auto SDK offre, oltre alle funzionalità online, anche una suite completa di funzionalità offline tramite l'estensione "Local Voice Control" (LVC). Questa garantisce che i clienti ricevano una risposta affidabile da Alexa indipendentemente dalla connettività, poiché durante un normale tragitto giornaliero, i conducenti possono perdere la connessione o sperimentare una connettività intermittente durante la guida. Dunque, le Case Automobilistiche incorporano una versione ridotta del Automatic Speech Recognition (ASR) e del Natural Language Understanding (NLU) basati sul cloud di Alexa nel sistema di infotainment del veicolo in modo da aumentarne le capacità. Questo supporto dual-mode non solo riduce l'impatto delle condizioni di rete variabili, ma darà agli utenti del veicolo la sicurezza di pensare sempre più alla voce come un mezzo efficace ed efficiente per l'interazione e la personalizzazione del veicolo.

I moduli disponibili tramite questa estensione sono diversi e aggiungono la capacità di base al SDK per abilitare le funzionalità offline. Tuttavia, permette di far funzionare solo ciò che non ha bisogno della connessione come ad esempio il controllo del clima, la riproduzione della radio o operazioni telefoniche. Dunque, escluse da questa lista rimangono fuori le Alexa Skills che essendo su un server non posso-

no essere utilizzate in assenza di connessione. Una soluzione potrebbe essere l'introduzione di un modulo in Alexa Auto SDK in cui viene approfondita ed estesa la complessità di un'applicazione intesa per la diagnostica. Tale soluzione risulterebbe molto conveniente e utile sia per chi sviluppa, sia per le Case Automobilistiche, che garantiscono con maggior copertura ed efficacia un supporto sempre costante ed attento all'esperienza di bordo dei guidatori e dei passeggeri dei loro veicoli.

- *Ricerca nell'IA conversazionale*: è l'insieme di tecnologie alla base della messaggistica automatizzata e delle applicazioni abilitate al riconoscimento vocale che offrono interazioni simili a quelle umane tra computer e utenti. Il successo di Alexa non significa che l'IA conversazionale sia un problema risolto. Al contrario, è stata appena scalfita la superficie di ciò che è possibile fare [38]. Amazon, grazie ad un Team di ricerca dedicato, sta rendendo Alexa:

1. Più "self-learning": ovvero più intelligente e più veloce riducendo la dipendenza dall'apprendimento supervisionato. Recentemente è stata migliorata la precisione dei riconoscitori vocali di Alexa utilizzando il paradigma insegnante-studente come addestratore. Nel paradigma un modello insegnante potente ma poco pratico viene addestrato su una piccola quantità di dati etichettati manualmente e, a sua volta, annota un corpo molto più ampio di dati non etichettati per addestrare un modello studente più snello ed efficiente.

2. Più usabile: Amazon sta lavorando per rendere più naturale l'invocazione di una Skill. Per adesso è stata rilasciata in inglese una funzionalità che permette agli utenti di interagire con Alexa senza invocare il nome specifico della Skill. Questo aiuta molto i clienti poiché non sempre sanno quale Skill è appropriata per portare a termine un determinato compito. Quando Alexa riceve una richiesta senza un nome specifico, vengono cercate le Skill che potrebbero soddisfare tale richiesta.
3. Più colloquiale: il completamento di attività complesse che richiedono un'interazione avanti e indietro e l'anticipazione degli obiettivi del cliente è ancora un problema impegnativo. Attualmente si sta testando una nuova tecnologia basata sul deep-learning, chiamata "Alexa Conversations", con un piccolo gruppo di sviluppatori di Skills che la stanno utilizzando per creare esperienze di conversazioni di alta qualità con il minimo sforzo. In breve, Alexa Conversations utilizza una serie di dialoghi di esempio per addestrare un modello di deep learning all'avanguardia per prevedere le azioni di dialogo, senza la necessità di creare manualmente a priori delle regole.

Ringraziamenti

Arrivato alla fine di questo percorso di studi mi sento di ringraziare alcune persone, innanzitutto il Prof. Gervasi e il Dott. Perri per avermi aiutato in questi ultimi mesi e per avermi dato la possibilità di lavorare insieme a loro.

Un ringraziamento speciale va al Dott. Berti per avermi guidato con tutta la professionalità che lo contraddistingue sia durante il tirocinio in ART S.p.A. che in fase di sviluppo del progetto di tesi.

Ringrazio la mia famiglia per non avermi mai fatto mancare nulla, in particolare modo l'affetto. Li ringrazio perché mi hanno sempre aiutato, in tutto, mi hanno sostenuto e mi hanno dato la possibilità di arrivare dove sono arrivato.

È solamente grazie a loro che oggi sono quello che sono.

Infine un grazie va ai miei amici che sono e saranno sempre la mia seconda famiglia.

Bibliografia

- [1] Cisco. Consumers desire more automated automobiles, according to cisco study. <https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1184392>, 2013. Accessed, 20-10-2021.
- [2] Alan Baddeley. Working memory. *Current biology*, 20(4):R136–R140, 2010. doi: "<https://doi.org/10.1016/j.cub.2009.12.014>".
- [3] George A Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956. doi: "<https://doi.org/10.1037/h0043158>".
- [4] Derek M Jones. The 7 ± 2 urban legend. In *MISRA C 2002 conference*, 2002.
- [5] National Highway Traffic Safety Administration et al. Overview of motor vehicle crashes in 2019. <https://crashstats.nhtsa.dot.gov/Api/Public/Publication/813060>. Accessed, 26-10-2021.
- [6] Istat. Incidenti stradali in italia. <https://www.istat.it/it/archivio/245757>. Accessed, 26-10-2021.

- [7] Anh Son Le, Tatsuya Suzuki, and Hirofumi Aoki. Evaluating driver cognitive distraction by eye tracking: From simulator to driving. *Transportation research interdisciplinary perspectives*, 4:100087, 2020. doi: "<https://doi.org/10.1016/j.trip.2019.100087>".
- [8] Yulan Liang, John D Lee, and Michelle L Reyes. Nonintrusive detection of driver cognitive distraction in real time using bayesian networks. *Transportation research record*, 2018(1):1–8, 2007. doi: "<https://doi.org/10.3141/2018-01>".
- [9] Effie Lai-Chong Law, Virpi Roto, Marc Hassenzahl, Arnold POS Vermeeren, and Joke Kort. Understanding, scoping and defining user experience: a survey approach. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 719–728, 2009. doi: "<https://doi.org/10.1145/1518701.1518813>".
- [10] Amazon J.D. Power. Voice services in vehicles, a deciding factor in the customer purchase decision. *Report of Amazon*, 2019.
- [11] N Kinnear L Lloyd S Helman P Husband J Scoons S Jones S Stradling F McKenna J Broughton. Novice drivers: Evidence review and evaluation. *Report of Trabsoirt Research Laboratory*, 2013.
- [12] Mark Webster. Voice trends and statistics: What designers need to know about the new tech boom. <https://blog.adobe.com/en/2019/07/22/voice-assistant-statistics-trends-2019.html#gs.eu271u>, 2019. Accessed, 16-11-2021.

- [13] Bret Kinsella and Ava Mutchler. In-car voice assistant consumer adoption report, 2020.
- [14] Commission of the European Communities. Commission recommendation of 26 may 2008 on safe and efficient in-vehicle information and communication systems: Update of the european statement of principles on human-machine interface. *Official Journal of the European Communities*, 216:1–42, 2008.
- [15] JL Campbell, JL Brown, JS Graving, CM Richard, MG Lichty, T Sanquist, and J Morgan. Human factors design guidance for driver-vehicle interfaces. *National Highway Traffic Safety Administration, Washington, DC, DOT HS*, 812:360, 2016.
- [16] John D Lee, Daniel V McGehee, Timothy L Brown, and Dawn Marshall. Effects of adaptive cruise control and alert modality on driver performance. *Transportation research record*, 1980(1):49–56, 2006. doi: "https://doi.org/10.1177/0361198106198000108".
- [17] Arianne Walker. J.d. power: Customers want consistency in voice experience at home and in the vehicle. <https://amzn.to/35N1MTh>, 2020. Accessed, 12-11-2021.
- [18] David Berti. Verso la guida autonoma: design, hud, digital instrument clusters e hmi. <http://blog.davidberti.com/verso-la-guida-autonoma>, 2017. Accessed, 20-12-2021.
- [19] Copeland, B.J. artificial intelligence. <https://www.britannica.com/technology/artificial-intelligence>, 2021. Accessed, 27-10-2021.

- [20] Michael H Cohen, Michael Harris Cohen, James P Giangola, and Jennifer Balogh. *Voice user interface design*. Addison-Wesley Professional, 2004.
- [21] J Norberto Pires. Interface devices and systems. *Industrial Robots Programming: Building Applications for the Factories of the Future*, pages 173–223, 2007. doi: "https://doi.org/10.1007/978-0-387-23326-0_4".
- [22] Sara Reese Hedberg. Dictating this article to my computer: Automatic speech recognition is coming of age. *IEEE Expert*, 12(6):9–11, 1997. doi: "<https://doi.org/10.1109/64.642953>".
- [23] Alexander Maedche, Stefan Morana, Silvia Schacht, Dirk Werth, and Julian Krumeich. Advanced user assistance systems. *Business & Information Systems Engineering*, 58(5):367–370, 2016. doi: "<https://doi.org/10.1007/s12599-016-0444-2>".
- [24] Inma Martínez. Mission control. In *The Future of the Automotive Industry*, pages 21–43. Springer, 2021. doi: "https://doi.org/10.1007/978-1-4842-7026-4_2".
- [25] Mukund Ghangurde. Ford sync and microsoft windows embedded automotive make digital lifestyle a reality on the road. *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, 3(2010-01-2319):99–105, 2010. doi: "<https://doi.org/10.4271/2010-01-2319>".

- [26] James W Jenness, Linda Ng Boyle, John D Lee, Chun-Cheng Chang, Vindhya Venkatraman, Madeleine Gibson, Kaitlin E Riegler, and Daniel Kellman. In-vehicle voice control interface performance evaluation. Technical report, National Highway Traffic Safety Administration, 2016.
- [27] Fabio Arena, Giovanni Pau, and Alessandro Severino. An overview on the current status and future perspectives of smart cars. *Infrastructures*, 5(7):53, 2020. doi: "<https://doi.org/10.3390/infrastructures5070053>".
- [28] Ava Mutchler Bret Kinsella. In-car voice assistant consumer adoption report. *Report of Voicebot.ai*, 2020.
- [29] Allan de Barcelos Silva, Marcio Miguel Gomes, Cristiano André da Costa, Rodrigo da Rosa Righi, Jorge Luis Victoria Barbosa, Gustavo Pessin, Geert De Doncker, and Gustavo Federizzi. Intelligent personal assistants: A systematic literature review. *Expert Systems with Applications*, 147:113193, 2020. doi: "<https://doi.org/10.1016/j.eswa.2020.113193>".
- [30] Erik Cambria and Bebo White. Jumping nlp curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, 9(2):48–57, 2014. doi: "<https://doi.org/10.1109/MCI.2014.2307227>".
- [31] Dirk Schnelle-Walka, Stefan Radomski, Benjamin Milde, Chris Bieermann, and Max Mühlhäuser. Nlu vs. dialog management: To whom

- am i speaking? In *Joint Workshop on Smart Connected and Wearable Things (SCWT'2016)*, co-located with *IUI*, volume 2, 2016. doi: "<https://doi.org/10.13140/RG>".
- [32] Paul Taylor. *Text-to-speech synthesis*. Cambridge university press, 2009. doi: "<https://doi.org/10.1017/CB09780511816338>".
- [33] Maximilian Bisani and Hermann Ney. Joint-sequence models for grapheme-to-phoneme conversion. *Speech communication*, 50(5):434–451, 2008. doi: "<https://doi.org/10.1016/j.specom.2008.01.002>".
- [34] Victor Leitman. Embedded, cloud, and hybrid connectivity solutions for custom voice assistants. <https://bit.ly/35MSTGQ>, 2020. Accessed, 29-11-2021.
- [35] What is alexa? <https://developer.amazon.com/en-US/alexa/>, 2020. Accessed, 14-10-2021.
- [36] Irene Lopatovska, Katrina Rink, Ian Knight, Kieran Raines, Kevin Cosenza, Harriet Williams, Perachya Sorsche, David Hirsch, Qi Li, and Adrianna Martinez. Talk to me: Exploring user interactions with the amazon alexa. *Journal of Librarianship and Information Science*, 51(4):984–997, 2019. doi: "<https://doi.org/10.1177/0961000618759414>".
- [37] Dieter Bohn. Amazon says 100 million alexa devices have been sold, what's next? <https://www.theverge.com/2019/1/4/18168565/>

- amazon-alexa-devices-how-many-sold-number-100-million-dave, 2019. Accessed, 2-12-2021.
- [38] Rohit Prasad. Conversationl ai - alexa at five: looking back, looking forward. <https://www.amazon.science/blog/alexa-at-five-looking-back-looking-forward>, 2019. Accessed, 3-12-2021.
- [39] Alexa auto software development kit. <https://developer.amazon.com/en-US/alexa/devices/alexa-built-in/development-resources/auto-sdk>, 2020. Accessed, 20-11-2021.
- [40] Overview of the alexa auto sdk. <https://github.com/alexa/alexa-auto-sdk>, 2020. Accessed, 20-11-2021.
- [41] What is the alexa skills kit? <https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html>, 2020. Accessed, 21-11-2021.
- [42] About voice interaction models. <https://developer.amazon.com/en-US/docs/alexa/ask-overviews/voice-interaction-models.html>, 2020. Accessed, 21-11-2021.
- [43] Create and manage skills in the developer console. <https://developer.amazon.com/en-US/docs/alexa/devconsole/about-the-developer-console.html>, 2020. Accessed, 22-11-2021.
- [44] Host a custom skill as a web service. <https://developer.amazon.com/en-US/docs/alexa/custom-skills/>

- `host-a-custom-skill-as-a-web-service.html`, 2020. Accessed, 23-11-2021.
- [45] Host a custom skill as an aws lambda function. <https://developer.amazon.com/en-US/docs/alexa/custom-skills/host-a-custom-skill-as-an-aws-lambda-function.html>, 2020. Accessed, 23-11-2021.
- [46] Get started with the alexa skills toolkit for visual studio code. <https://developer.amazon.com/en-US/docs/alexa/ask-toolkit/get-started-with-the-ask-toolkit-for-visual-studio-code.html>, 2020. Accessed, 24-11-2021.
- [47] Alexa skills kit sdks. <https://developer.amazon.com/en-US/docs/alexa/sdk/alexa-skills-kit-sdks.html>, 2020. Accessed, 25-11-2021.
- [48] Alexa skills kit command line interface (ask cli) overview. <https://developer.amazon.com/en-US/docs/alexa/smapi/ask-cli-intro.html>, 2020. Accessed, 25-11-2021.
- [49] Choose the invocation name for a custom skill. <https://developer.amazon.com/en-US/docs/alexa/custom-skills/choose-the-invocation-name-for-a-custom-skill.html>, 2020. Accessed, 26-11-2021.
- [50] Create the interaction model for your skill. <https://developer.amazon.com/en-US/docs/alexa/custom-skills/>

`create-the-interaction-model-for-your-skill.html`, 2020.
Accessed, 26-11-2021.

[51] Identify the slots for the intent. <https://developer.amazon.com/en-US/docs/alexa/custom-skills/create-intents-utterances-and-slots.html>, 2020. Accessed, 26-11-2021.

[52] Delegate the dialog to alexa. <https://developer.amazon.com/en-US/docs/alexa/custom-skills/delegate-dialog-to-alex.html>, 2020. Accessed, 27-11-2021.

[53] About managing the conversation with the user. <https://developer.amazon.com/en-US/docs/alexa/custom-skills/dialog-interface-reference.html#scenario-full-control>, 2020. Accessed, 27-11-2021.

[54] Request and response json reference. <https://developer.amazon.com/en-US/docs/alexa/custom-skills/request-and-response-json-reference.html#session-object>, 2020. Accessed, 28-11-2021.

[55] Qt widgets. <https://doc.qt.io/qt-5/qtwidgets-index.html>, 2020. Accessed, 17-10-2021.

[56] Templateruntime 1.2. <https://developer.amazon.com/en-US/docs/alexa/alexa-voice-service/templateruntime.html>, 2020. Accessed, 18-10-2021.

- [57] Shanika Wickramasinghe. Mongodb vs dynamodb: Comparing nosql databases. <https://www.bmc.com/blogs/mongodb-vs-dynamodb/>, 2021. Accessed, 5-12-2021.