UNIVERISTY OF PIRAEUS - DEPARTMENT OF INFORMATICS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΜΣ «Προηγμένα Συστήματα Πληροφορικής – Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	Ευφυείς Agents σε Mobile Εφαρμογές Smart Agents for Mobile Applications
Ονοματεπώνυμο φοιτητή:	Κωνσταντίνος Κουμπανάκης
Πατρώνυμο:	Στυλιανός
Αριθμός Μητρώου:	ΜΠΣΠ21024
Επιβλέπων:	Αλέπης Ευθύμιος, Αναπληρωτής Καθηγητής

Ημερομηνία Παράδοσης 05/2023

Τριμελής Εξεταστική Επιτροπή

Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής	Μαρία Βίρβου Καθηγήτρια	Κωνσταντίνος Πατσάκης Αναπληρωτής Καθηγητής

Περίληψη

Η παρούσα διπλωματική εργασία ασχολείται με την διερεύνηση των δυνατοτήτων των ευφυών agents για την αναγνώριση γλώσσας, εξαγωγή παραμέτρων και προθέσεων του χρήστη σε mobile εφαρμογές.

Η χρήση των agents εμφανίζεται όλο και πιο συχνά στις mobile εφαρμογές με σκοπό την αυτοματοποίηση εργασιών όπου σε αντίθετη περίπτωση ο χρήστης θα εκτελούσε χειροκίνητα στο κινητό του.

Ένας agent μπορεί να προσφέρει πληθώρα πλεονεκτημάτων σε μια mobile εφαρμογή όπως η ταχύτητα εκτέλεσης εργασιών, η δυνατότητα η εφαρμογή να είναι προσβάσιμη από άτομα με περιορισμένη όραση και άλλες ειδικές ανάγκες, καλύτερη εμπειρία χρήσης της εφαρμογής και εκτέλεση περίπλοκων περιπτώσεων χρήσης στα πλαίσια της εφαρμογής.

Παρόλα αυτά, η ενσωμάτωση ενός ευφυούς agent σε μια mobile εφαρμογή δεν είναι πάντοτε μια εύκολη διαδικασία. Οι παράμετροι που πρέπει να ληφθούν υπόψιν κατά την σχεδίαση του λογισμικού είναι αρκετές και πρέπει να οριστούν εξ' αρχής τα πλαίσια πάνω στα οποία θα λειτουργεί ο agent.

Στην παρούσα εργασία θα αναλύσουμε την χρησιμότητα και τις δυνατότητες που προσφέρει ένας agent σε μια mobile εφαρμογή, τον τρόπο ένταξής του σε μια mobile εφαρμογή και τις προκλήσεις που θα εμφανιστούν σχεδιάζοντας ένα λογισμικό όπου θα χρησιμοποιεί έναν ευφυή agent.

Επιπλέον, θα γίνει παρουσίαση και ανάλυση της mobile εφαρμογής, που υλοποιήθηκε στα πλαίσια της διπλωματικής εργασίας, όπου γίνεται χρήση ενός ευφυούς agent μέσω της υπηρεσίας DialogFlow της Google για την πλήρη αυτοματοποίηση εργασιών και πλοήγησης σε application με σκοπό την παραγγελία προϊόντων ενός καταστήματος από τον πελάτη.

The purpose of this paper is to analyze the capabilities of intelligent agents regarding the identification of user intents and extraction of parameters via the user input in the scope of mobile applications.

The use of agents in mobile applications is becoming even more frequent by utilizing their ability to adapt depending on the user and as a result personalize and automate various tasks.

The agents can bring many advantages in a mobile application such as automated tasks, personalization, better user experience and the ability to build an app that can be used by users with disabilities.

That said, it is not always an easy process to properly integrate and use an agent in a mobile application. There are many parameters that must be considered while designing the software that will dictate its quality.

We will try to describe the proper practices to integrate an agent, the challenges that come along, as well as all the capabilities that an agent can provide.

Lastly, we will present and analyze the mobile application that was created, called Store Agent, that will demonstrate how we can take advantage of an agent in order to create an application that can automate tasks easily and efficiently.

1 Εισαγωγή στους ευφυείς agents

Ένας ευφυής agent είναι ένα σύστημα το οποίο μπορεί να λάβει αποφάσεις ή να εκτελέσει μια εργασία με βάση την αλληλεπίδραση με τον χρήστη του συστήματος.

Συνήθως, ένας agent προσπαθεί να αναλύσει το user input που δέχεται, στην περίπτωση μας μέσω mobile εφαρμογών, ώστε είτε να εκτελέσει απευθείας μια ενέργεια ή να αναλύσει και να καταλήξει σε συμπεράσματα για το user input.

Επιπροσθέτως, σε πολλές περιπτώσεις μαθαίνει τις προτιμήσεις του εκάστοτε χρήστη μέσω του input που δέχεται από εκείνον με αποτέλεσμα να έχουμε έναν personalized agent που γνωρίζεις τις ανάγκες του κάθε χρήστη και προσαρμόζει τις αποφάσεις του αναλόγως.

Σε γενικές γραμμές, ένας agent δέχεται συγκεκριμένες παραμέτρους υπό τις οποίες θα προσπαθήσει να οδηγηθεί σε συμπεράσματα για το user input που δέχθηκε.

Για παράδειγμα, σε μια εφαρμογή παραγγελιοληψίας προϊόντων, ο agent πρέπει να γνωρίζει τις παραμέτρους του περιβάλλοντος με βάση τις οποίες λειτουργεί. Δηλαδή, πρέπει να γνωρίζει τα προϊόντα που ενδέχεται να ζητήσει ο χρήστης, τις διαθέσιμες ποσότητες και άλλες σχετικές παραμέτρους, έτσι ώστε ο agent να μπορεί να αποφασίσει αν ένα προϊόν που ζήτησε ο χρήστης υπάρχει σε απόθεμα.

Κάποια βασικά χαρακτηριστικά που πρέπει να διαθέτει ένας agent είναι, μεταξύ άλλων, η ικανότητα να λύνουν προβλήματα που αφορούν την εφαρμογή στην οποία εντάσσονται, το να είναι ικανοί να προσαρμόζονται δυναμικά σε νέα user inputs που δεν έχουν συναντήσει έως εκείνη τη χρονική στιγμή και το να είναι ικανοί να εξάγουν παραμέτρους και συμπεράσματα με βάση το user input που δέχθηκαν.

1.1 Κατηγορίες agents

Υπάρχουν πολλές κατηγορίες ευφυούς agents, που η κάθε μια μπορεί να χρησιμοποιηθεί συνδυαστικά με κάποια άλλη ή μεμονωμένα, για την χρήση σε μια εφαρμογή.

- **Reflex Agents**: Ο συγκεκριμένος τύπος agent επικεντρώνεται στο να λαμβάνει αποφάσεις με βάση την τρέχουσα κατάσταση που έχει διαμορφωθεί εκείνη τη στιγμή στην εφαρμογή. Δεν λαμβάνει υπόψιν αποφάσεις που έχουν παρθεί στο παρελθόν και κατά συνέπεια είναι ικανός να αντιδράει σε συμβάντα που εκείνη τη στιγμή προκαλούνται από την αλληλεπίδραση με τον χρήστη.
- **Model Based Agents**: Οι Model Based Agents αποφασίζουν για το πως θα αντιδράσουν σε ένα συμβάν που προκαλείται από τον χρήστη με παρόμοιο τρόπο όπως οι Reflex Agents. Επιπροσθέτως, λαμβάνουν υπόψιν ένα μοντέλο του περιβάλλοντος υπό το οποίο δρουν και για το οποίο υπεύθυνος ορισμού του είναι ο εκάστοτε προγραμματιστής.
- **Goal Based Agents**: Ο συγκεκριμένος τύπος agent επεκτείνει την λειτουργία των Model Based Agents με το να λαμβάνει υπόψιν τους στόχους και τα επιθυμητά αποτελέσματα του χρήστη για κάθε ενέργεια.
- **Utility Based Agents**: Οι Utility Based Agents βασίζονται πάνω στους Goal Based Agents και επεκτείνουν την λειτουργικότητά τους με το να προσπαθούν να αποφασίσουν ποιο θα ήταν το ιδανικό αποτέλεσμα για μια συγκεκριμένη ενέργεια που προκαλεί ο χρήστης. Με αυτό τον τρόπο, μπορούν να αποφασίσουν ποιο αποτέλεσμα θα ήταν επιθυμητό, ανάμεσα από πολλές επιλογές, για τον συγκεκριμένο χρήστη.
- **Learning Agents**: Οι Learning Agents έχουν την ικανότητα να βελτιώνονται σταδιακά ώστε να έχουν καλύτερη επίγνωση του περιβάλλοντος υπό το οποίο δρουν. Η διαδικασία μάθησης βασίζεται σε μεγάλο βαθμό σε feedback μηχανισμούς.

2 Google Dialogflow

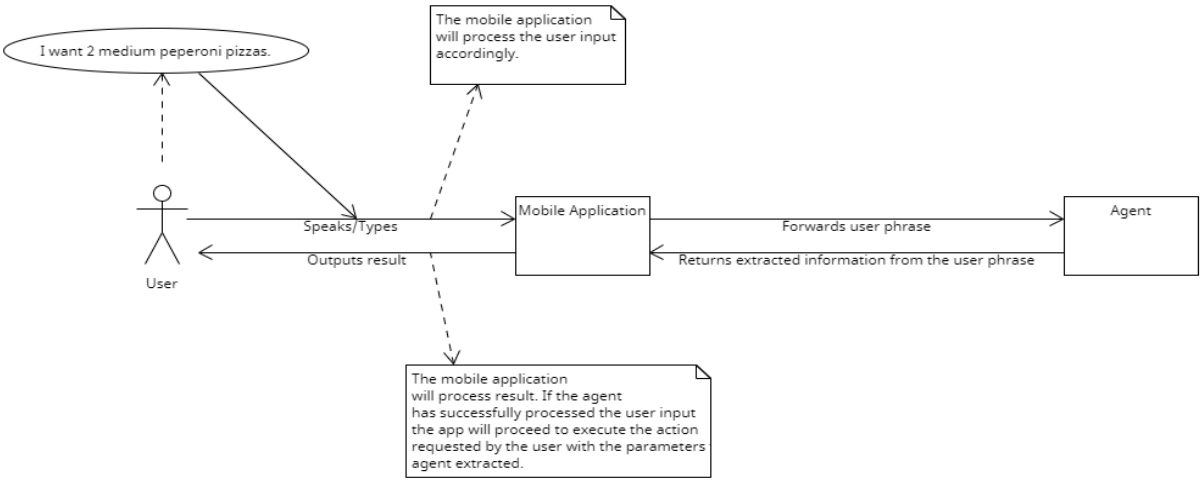
Στα πλαίσια της διπλωματικής εργασίας, έχει μελετηθεί, αναλυθεί και χρησιμοποιηθεί στην εφαρμογή που υλοποιήθηκε, και θα παρουσιάσουμε παρακάτω, ο Dialogflow Agent της Google.

Ο συγκεκριμένος agent είναι συμβατός με Android και Web εφαρμογές καθώς δίνεται η δυνατότητα να επικοινωνεί η εφαρμογή με τον agent είτε μέσω WEB API calls είτε μέσω platform-specific SDK.

Το Dialogflow χρησιμοποιείται για την δημιουργία agents που εντάσσονται σε πληθώρα εφαρμογών για την εξυπηρέτηση πολλών διαφορετικών σκοπών. Υπάρχουν integrations με το Facebook Messenger, Slack, Google Assistant και άλλες.

Είναι επίσης διαθέσιμος σε πληθώρα γλωσσών, ανάμεσα τους και τα Ελληνικά.

Ένα τυπικό flow της αλληλεπίδρασης με τον agent είναι η απαίτηση του χρήστη να εκτελεστεί μια ενέργεια στα πλαίσια της εφαρμογής, είτε γραπτώς είτε προφορικώς. Κατόπιν, η εφαρμογή προωθεί το αίτημα στον agent σε μορφή κειμένου και τέλος ο agent επεξεργάζεται την φράση του user ώστε να εξαγάγει απαραίτητες πληροφορίες που θα επιστρέψει πίσω στην εφαρμογή ώστε να μπορέσει να εκτελεστεί η ενέργεια που ζητήθηκε από τον χρήστη.



User – Mobile - Agent Interaction

Αναλύοντας τις δυνατότητες του Dialogflow agent, θα εστιάσουμε σε δύο βασικές λειτουργίες των οποίων τον ορισμό και την παραμετροποίηση είναι υπεύθυνος ο προγραμματιστής. Η πρώτη λειτουργία είναι τα Entites και η δεύτερη τα Intents.

2.1 Entities

Τα Entities (οντότητες) ορίζονται ώστε ο agent να έχει την δυνατότητα να αναγνωρίσει οντότητες σημαντικές για την λειτουργία της εφαρμογής.

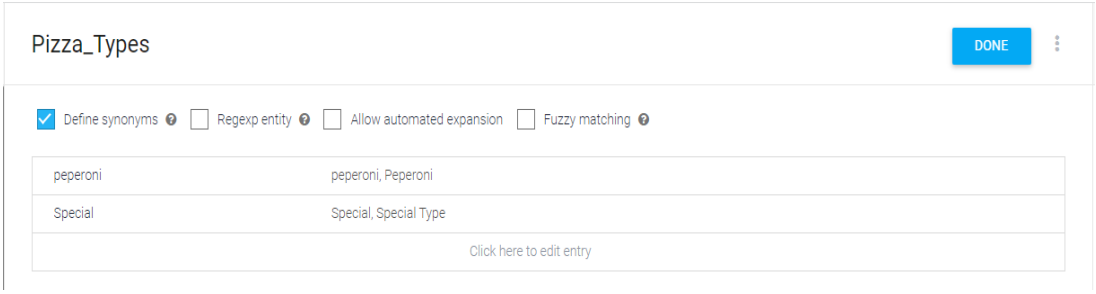
Οι οντότητες αυτές εκφράζουν objects τα οποία ο agent θα μπορέσει να ανιχνεύσει μέσα σε μια φράση, να τα εξαγάγει και να τα παρουσιάσει πίσω στην εφαρμογή ως μέρος του response που επιστρέφει με βάση το request του χρήστη.

Στο παρακάτω παράδειγμα παραγγελειοληψίας φαγητού μέσω ηλεκτρονικής εφαρμογής, θα μπορούσαμε να ορίσουμε 3 οντότητες που μπορεί να αναγνωρίσει ο agent. Τον τύπο του είδους, την ποσότητα που ζητήθηκε και το μέγεθος του. Συνεπώς, θα μπορούσε να συμπεράνει πως μέσα στην φράση του χρήστη "I want 2 medium peperoni pizzas." περιέχονται οι ακόλουθες οντότητες. Η πρώτη οντότητα είναι η ποσότητα και έχει την τιμή δύο, η δεύτερη είναι το μέγεθος και έχει την τιμή medium και η τελευταία είναι ο τύπος και έχει την τιμή peperoni.

Προφανώς, πρέπει να υπάρχει αντιστοιχία μεταξύ των οντοτήτων που έχουμε ορίσει στον agent και αυτών που έχουμε ορίσει στην mobile εφαρμογή. Με αυτόν τον τρόπο η mobile εφαρμογή μπορεί να πάρει τις οντότητες που έγιναν extract από τον agent ως παραμέτρους για να μπορέσει να ολοκληρώσει την ενέργεια που έχει ζητηθεί.

Μέσα από το σχετικό UI interface του agent που παρέχεται από το Dialogflow μπορούμε να δούμε όλες τις οντότητες που ορίσαμε.

Στο παράδειγμα μας, δημιουργήσουμε τις παρακάτω οντότητες για τους τύπους ειδών που γνωρίζει ο agent. Σε αυτή τη περίπτωση ο agent θα μπορούσε να αναγνωρίσει μόνο τους δύο συγκεκριμένους τύπους που φαίνονται παρακάτω.



Agent Entities

Παρατηρούμε πως έχουμε επίσης την δυνατότητα να ορίσουμε για κάθε μια οντότητα που δημιουργούμε και μια σειρά από συνώνυμα ως εναλλακτικές ονομασίες των οντοτήτων μας ώστε ο agent να μπορεί να αντιδράσει ακόμη και στην περίπτωση όπου μια οντότητα ζητηθεί από τον χρήστη με διαφορετική ονομασία.

2.2 Intents

Οδηγούμαστε λοιπόν στο εξής ερώτημα <<Πως καταλαβαίνει ο agent και κατά επέκταση η mobile εφαρμογή, ποια ακριβώς είναι η πρόθεση του χρήστη;>>. Έως τώρα έχουμε περιγράψει την διαδικασία εξαγωγής των οντοτήτων και θα προχωρήσουμε στην διαδικασία εντοπισμού της πρόθεσης του χρήστη.

Σε αυτό το σημείο εκμεταλλευόμαστε τα intents. Τα intents εκφράζουν την πρόθεση του χρήστη. Κάθε ένα intent που δημιουργούμε στα πλαίσια του agent εκφράζει και μια διαφορετική ενέργεια που μπορεί να ζητηθεί. Συνεπώς, κάθε intent που δημιουργείται αντιστοιχίζεται σε μια μόνο πρόθεση.

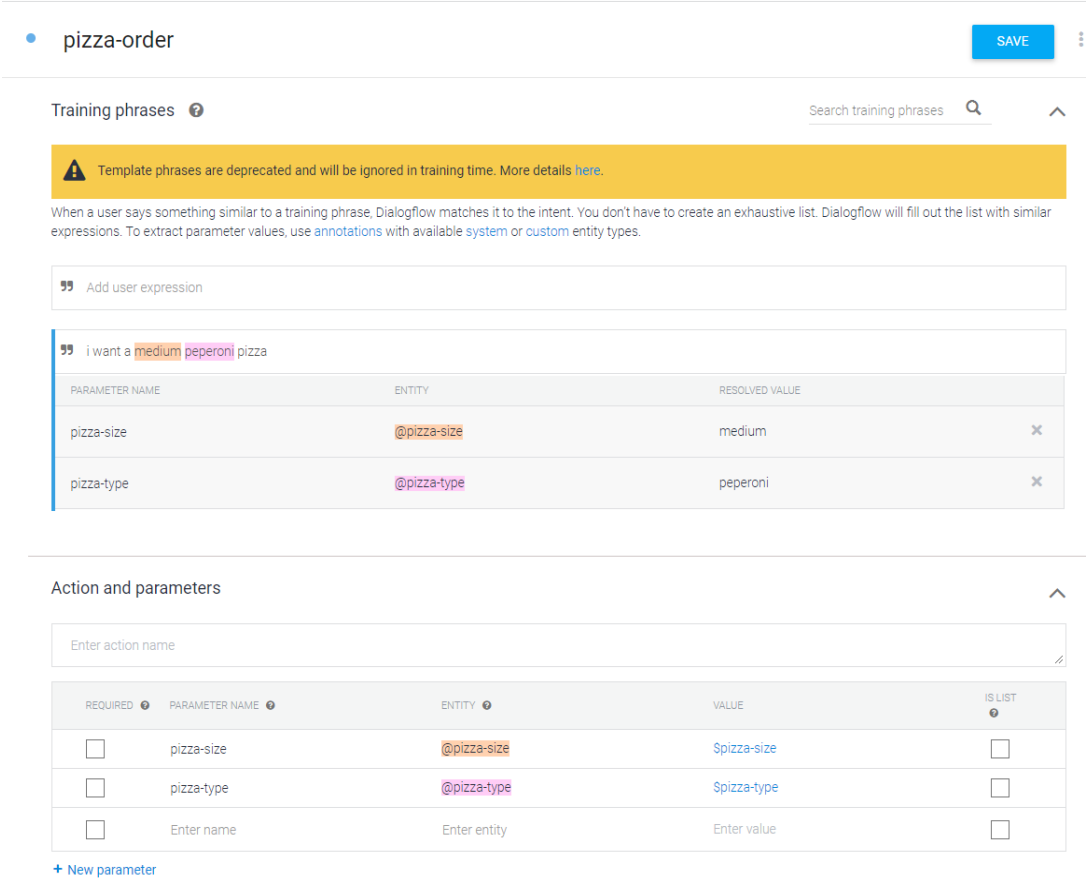
Τα intents αποτελούνται από τρία βασικά μέρη:

- Την ονομασία. Η ονομασία είναι σημαντική καθώς με αυτόν τον τρόπο μπορεί η mobile εφαρμογή να καταλάβει ποιο intent εκτελέστηκε από τον agent ώστε να εκτελεστούν οι απαραίτητες ενέργειες και στην εφαρμογή.
- Τα training phrases. Τα training phrases είναι όλες οι πιθανές φράσεις με τις οποίες ο χρήστης μπορεί να ζητήσει μια συγκεκριμένη ενέργεια που αντιστοιχεί σε ένα intent.
- Οι οντότητες των training phrases. Είναι σημαντικό όταν δώσουμε σε ένα intent κάποια training phrases να ορίσουμε μέσα σε κάθε ένα από αυτά, ποιες είναι οι οντότητες που ο agent θα πρέπει να εξαγάγει. Αυτό το βήμα είναι ιδιαίτερα σημαντικό διότι μια οντότητα ενδέχεται να είναι υποχρεωτική ώστε να μπορέσει να ικανοποιηθεί το συγκεκριμένο intent. Το αν μια οντότητα είναι υποχρεωτική και κατά επέκταση πρέπει να βρίσκεται σε κάθε training phrase μπορούμε να το ορίσουμε στα πλαίσια του συγκεκριμένου intent.
- Τα contexts των οποίων έπεται περαιτέρω ανάλυση.

Σε καμία περίπτωση δεν χρειάζεται να καλύψουμε όλες τις διαφορετικές περιπτώσεις και όλες τις διαφορετικές οντότητες στα training phrases που θα δώσουμε σε ένα intent. Ο agent θα εκτελέσει αυτόματα ένα training process όπου θα επεκτείνει τα training phrases που του δώσαμε για όλες τις οντότητες που έχουμε δηλώσει ότι μπορεί να συναντήσει στα πλαίσια του συγκεκριμένου intent.

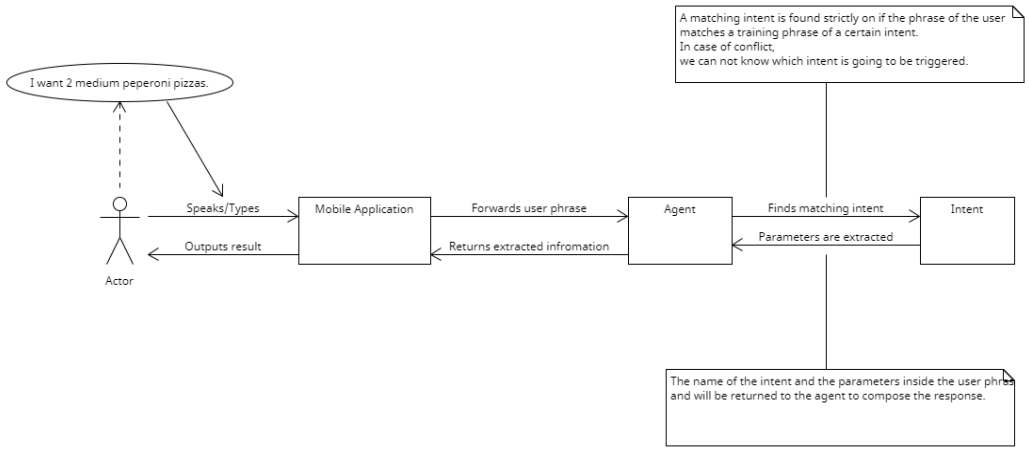
Αν επιστρέψουμε πίσω στο παράδειγμα μας με την φράση “I want 2 medium peperoni pizzas.”, μπορούμε να δημιουργήσουμε ένα νέο intent το οποίο θα ονομάσουμε pizza-order και θα έχει ως training phrases κάποιες ενδεικτικές φράσεις με τις οποίες ο χρήστης μπορεί να παραγγείλει.

Επίσης, παρατηρούμε στην παρακάτω εικόνα πως μέσα στο training phrase που ορίσαμε για το συγκεκριμένο intent, οι οντότητες που δημιουργήσαμε προηγουμένως έχουν ήδη εξαχθεί από τον agent.



Intent Training Phrases

Παρακάτω, φαίνεται το διάγραμμα του παραδείγματος με τις κατάλληλες επεκτάσεις ώστε να αποτυπωθεί η διαδικασία ενεργοποίησης ενός intent και της εξαγωγής των οντοτήτων από την φράση του χρήστη.



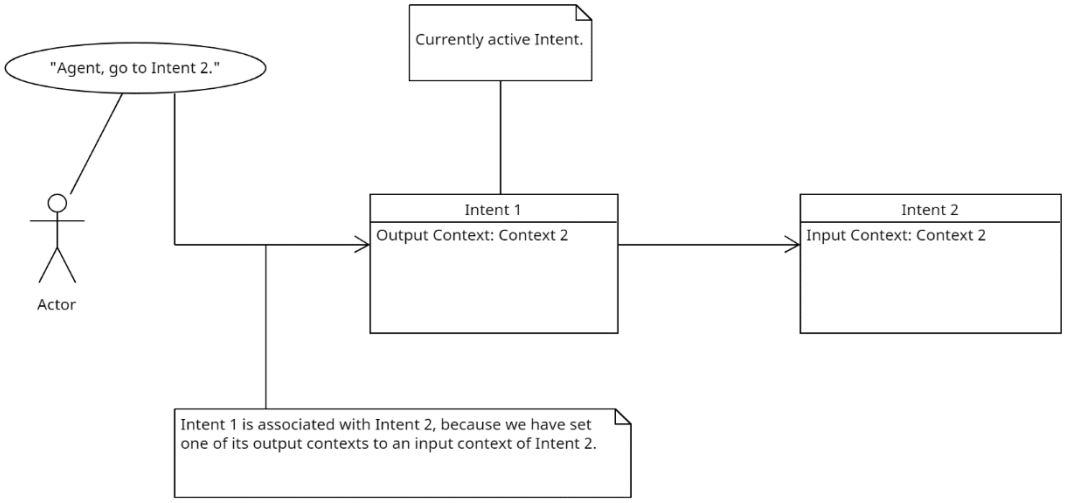
User – Mobile– Agent Interaction

2.3 Contexts

Τέλος, θα αναλύσουμε τα contexts ενός intent ξεκινώντας από τα input contexts. Τα input contexts χρησιμοποιούνται ώστε να μπορέσουμε να ενεργοποιήσουμε ένα intent. Εφόσον δηλώσουμε σε ένα intent ένα ή περισσότερα input contexts, αυτά μπορούν να χρησιμοποιηθούν από άλλα intents ώστε να το ενεργοποιήσουν.

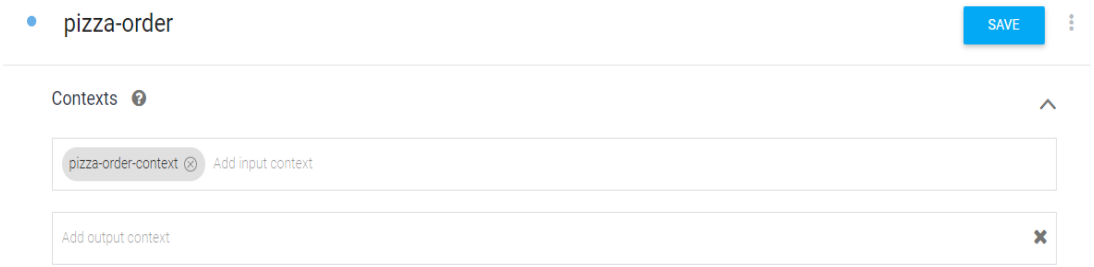
Για να ενεργοποιηθεί ένα intent μέσω ενός άλλου, θα πρέπει ο προγραμματιστής να έχει θέσει ως output context ένα ή περισσότερα από τα input contexts του intent που θέλουμε να ενεργοποιήσουμε. Συνεπώς, το εκάστοτε τρέχον intent μπορεί να ενεργοποιήσει intents των οποίων τα input contexts τα έχουμε θέσει ως output context του τρέχοντος.

Διαγραμματικά, αυτή η σχέση φαίνεται παρακάτω.



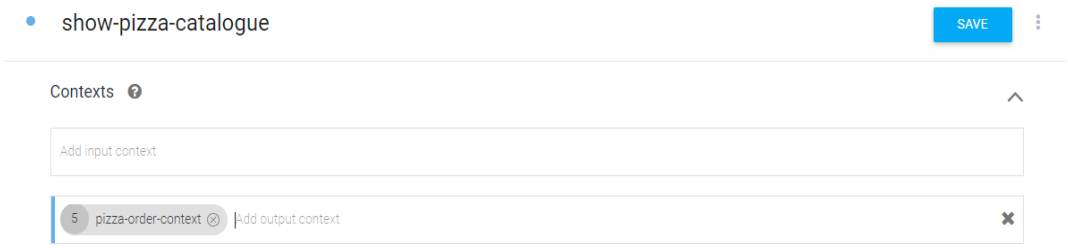
Activating Intent via Context

Στο παράδειγμα μας, μπορούμε να ορίσουμε ως input context του intent pizza-order το pizza-order-context.



Intent Output Contexts

Αν στην συνέχεια θέλαμε να δημιουργήσουμε ένα νέο intent, για παράδειγμα το show-pizza-catalogue ώστε να μπορεί ο χρήστης να δει τον κατάλογο με τα προϊόντα, θα μπορούσαμε να ορίσουμε ως output context του show-pizza-catalogue το input context του intent pizza-order.



Intent Input Contexts

Με αυτόν τον τρόπο ορίζουμε στον agent ότι ο χρήστης μπορεί να μεταβεί από το show-pizza-catalogue στο intent pizza-order. Αν δεν είχαμε θέσει ως output context το pizza-order-context, ο χρήστης δεν θα μπορούσε να εκτελέσει αυτή την μετάβαση.

3 Store Agent

Σε αυτό το σημείο και αφού έχουμε αναλύσει το πως λειτουργεί ένας Google Dialogflow agent, θα παρουσιάσουμε την εφαρμογή που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας.

Η εφαρμογή που αναπτύχθηκε ονομάζεται Store Agent και καλύπτει όλες τις λειτουργίες που μπορεί να εκτελέσει ένας χρήστης ώστε να πλοηγηθεί και να παραγγείλει προϊόντα μέσω ενός ηλεκτρονικού καταστήματος.

Ο Store Agent έχει αναπτυχθεί σε native Android με την χρήση της Kotlin, του Jetpack Compose για την δημιουργία του UI και της MVI αρχιτεκτονικής για την δομή του κώδικα.

Η εφαρμογή δίνει επιγραμματικά τις παρακάτω δυνατότητες στον χρήστη μέσω φωνητικών εντολών:

A. Προβολή των διαθέσιμων κατηγοριών.
B. Προβολή των διαθέσιμων προϊόντων κάθε κατηγορίας.
C. Προσθήκη και αφαίρεση προϊόντων στο καλάθι.
D. Αύξηση και μείωση των ποσοτήτων των προϊόντων που υπάρχουν στο καλάθι.
E. Ολοκλήρωση της παραγγελίας.

Λόγω του ότι η εφαρμογή είναι native, έχουμε εκμεταλλευτεί πλήρως τις δυνατότητες Speech to Text, Text to Speech και άλλων τεχνολογιών του Android.

Ο Store Agent είναι πλήρως παραμετροποιήσιμος υπό την έννοια ότι μπορεί να υποστηριχθεί οποιοσδήποτε τύπος ηλεκτρονικού καταστήματος με οποιεσδήποτε κατηγορίες προϊόντων.

Όλες οι φράσεις (training phrases) με τις οποίες ο χρήστης μπορεί να ζητήσει μια ενέργεια είναι πλήρως παραμετροποιήσιμες και παράγονται για κάθε οντότητα που υποστηρίζει η εφαρμογή. Με αυτόν τον τρόπο κατά την εγκατάσταση της εφαρμογής, κάθε διαφορετικό ηλεκτρονικό κατάστημα να μπορεί να υποστηρίζει τις δικές του οντότητες, δεδομένα και φράσεις με τις οποίες οι χρήστες μπορούν να ζητήσουν ένα προϊόν, μια κατηγορία προϊόντος κτλ.

Οι οντότητες που υποστηρίζονται αυτή τη στιγμή από την εφαρμογή και κατά επέκταση από τον dialogflow agent είναι:

- Master Category. Η βασική κατηγορία, κάτω από την οποία εμφανίζονται διάφορες υποκατηγορίες.
- Subcategory. Υποκατηγορία κάτω από την οποία εμφανίζονται τα προϊόντα της.
- Product. Προϊόν του ηλεκτρονικού καταστήματος το οποίο ανήκει σε μια βασική κατηγορία και υποκατηγορία.
- Basket. Το καλάθι με τα προϊόντα του χρήστη.
- Quantity. Η ποσότητα ενός προϊόντος στο καλάθι.

Επιπλέον, υποστηρίζεται πλήρως παραμετρικά ο ορισμός συνώνυμων ονομασιών για τις οντότητες που ορίζουμε.

Τέλος, τα intents τα οποία υποστηρίζει ο dialogflow agent και τα οποία αντιπροσωπεύουν μια συγκεκριμένη ενέργεια του χρήστη στα πλαίσια της εφαρμογής είναι τα παρακάτω:

- choose-product-from-product-type. Για την επιλογή ενός προϊόντος.
- choose-product-type. Για την επιλογή μιας βασικής κατηγορίας.
- choose-product-type-and-subtype. Για την επιλογή μιας βασικής κατηγορίας συνδυαστικά με μια υποκατηγορία ή την επιλογή μιας υποκατηγορίας.
- choose-product-with-quantity. Για την επιλογή ενός προϊόντος με μια συγκεκριμένη ποσότητα.
- see-basket. Για την προβολή του καλαθιού του χρήστη.
- delete-product. Για την διαγραφή ενός προϊόντος από το καλάθι.
- basket-decrease-product-value. Για την μείωση της ποσότητας ενός προϊόντος που βρίσκεται στο καλάθι κατά ή σε μια συγκεκριμένη ποσότητα.
- basket-increase-product-value. Για την αύξηση της ποσότητας ενός προϊόντος που βρίσκεται στο καλάθι κατά ή σε μια συγκεκριμένη ποσότητα.
- finalize-order. Για την ολοκλήρωση της παραγγελίας.

3.1 Αρχιτεκτονική εφαρμογής

Η αρχιτεκτονική της εφαρμογής σε επίπεδο επικοινωνίας με τον agent και σε επίπεδο δομής του κώδικα είναι πολύ σημαντική ώστε να μπορέσουμε να εκμεταλλευτούμε πλήρως τις δυνατότητες του Android και του Dialogflow agent.

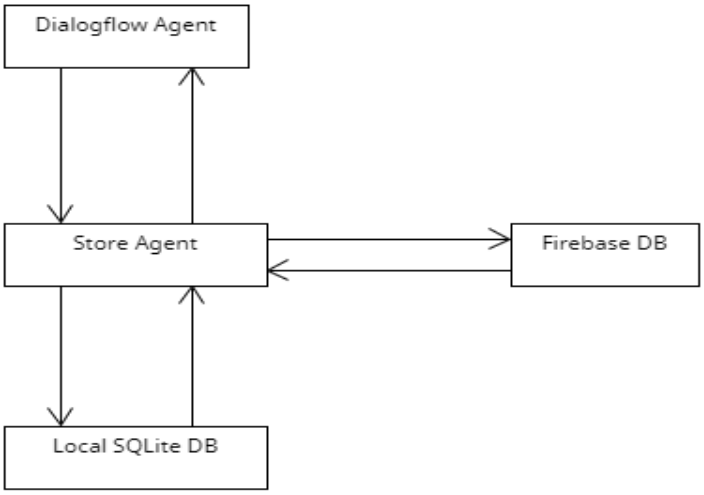
Η επικοινωνία της mobile εφαρμογής με τρίτα συστήματα χωρίζεται σε δύο μέρη. Το πρώτο μέρος είναι η επικοινωνία με την firebase βάση δεδομένων της εφαρμογής και το δεύτερος μέρος είναι η επικοινωνία με τον dialogflow agent.

Επίσης, η εφαρμογή διαθέτει εξ ’αρχής μια τοπική SQLite βάση στην οποία υπάρχουν όλα τα δεδομένα για τους παρακάτω πίνακες, όπου υπενθυμίζουμε πως παράλληλα είναι και οντότητες στα πλαίσια του dialogflow agent:

- Master Category. Η βασική κατηγορία, κάτω από την οποία εμφανίζονται διάφορες υποκατηγορίες.
- Subcategory. Υποκατηγορία κάτω της οποίας εμφανίζονται τα προϊόντα της.
- Product. Προϊόν του ηλεκτρονικού καταστήματος το οποίο ανήκει σε μια βασική κατηγορία και υποκατηγορία.

Σε αυτό το σημείο, αφού το συγκεκριμένο κομμάτι είναι εκτός scope της εφαρμογής και της διπλωματικής εργασίας, θεωρούμε ότι έχει προηγηθεί η δημιουργία της SQLite βάσης της εφαρμογής με τα απαραίτητα δεδομένα έπειτα από επικοινωνία με ένα τρίτο σύστημα.

Παρακάτω, φαίνεται συνοπτικά η αρχιτεκτονική της εφαρμογής:



Mobile App Architecture Overview

3.1.1 Δομή βάσης δεδομένων

Σε αυτό το σημείο θα αναλύσουμε την δομή της βάσης δεδομένων της εφαρμογής. Αποτελείται από 8 πίνακες οι οποίοι στο σύνολό τους απαρτίζουν την δομή μιας εφαρμογής που αναφέρεται σε ηλεκτρονικό κατάστημα.

Ο πρώτος πίνακας είναι ο Master Category. Οι εγγραφές αυτού του πίνακα αφορούν βασικές κατηγορίες προϊόντων οι οποίες μπορούν να έχουν μια ή περισσότερες υποκατηγορίες. Για παράδειγμα, η βασική κατηγορία 'Κονσόλες' μπορεί έχει αρκετές υποκατηγορίες όπως 'Κονσόλες Playstation' κτλ.

Ο Subcategories πίνακας αφορά οντότητες οι οποίες χαρακτηρίζονται ως υποκατηγορίες μιας βασικής κατηγορίας όπως περιγράψαμε παραπάνω. Ιδιαίτερα σημαντικά είναι τα πεδία NumericField και StringField. Αυτά τα πεδία συνδέονται άμεσα με την δυνατότητα παραμετροποίησης της εφαρμογής. Κάθε NumericField πεδίο του πίνακα Subcategories αφορά τον τίτλο ενός αριθμητικού χαρακτηριστικού της υποκατηγορίας και κάθε StringField πεδίο αφορά τον τίτλο ενός αλφαριθμητικού χαρακτηριστικού της υποκατηγορίας.

Για παράδειγμα η υποκατηγορία 'Κονσόλες Playstation' μπορεί να έχει ως αριθμητικό πεδίο την χωρητικότητα σκληρού δίσκου. Συνεπώς, η συγκεκριμένη υποκατηγορία θα είχε σαν τιμή στο πεδίο NumericField1 'HDD'. Επειδή όπως θα αναφέρουμε παρακάτω, στο κάθε προϊόν μπορούμε επίσης να ορίσουμε Numeric και String fields, τα προϊόντα που ανήκουν στην συγκεκριμένη κατηγορία 'Κονσόλες Playstation', θα έχουν σαν τιμή στο πεδίο NumericField1 την χωριτηκότητα του σκληρού δίσκου π.χ. 250.

Με αυτόν τον τρόπο μπορούμε εντελώς παραμετρικά να ορίσουμε τα χαρακτηριστικά της κάθε υποκατηγορίας και να θέσουμε τις τιμές αυτών των χαρακτηριστικών στα προϊόντα που ανήκουν στην συγκεκριμένη υποκατηγορία.

Επίσης, μπορούμε να ορίσουμε όσα Numeric και String fields επιθυμούμε. Αν κάποια υποκατηγορία εκμεταλλεύεται μόνο ορισμένα Numeric και String fields, τότε τα υπόλοιπα θα παραμείνουν κενά και κατά επέκταση δεν θα εμφανιστούν στην εφαρμογή.

Η σύνδεση που μας παρουσιάζει ποιες βασικές κατηγορίες σχετίζονται με ποιες υπόκατηγορίες βρίσκεται στον πίνακα MasterToSubcategory όπου συνδέει μια βασική κατηγορία με μία ή περισσότερες υποκατηγορίες.

Επιπροσθέτως, υπάρχουν οι πίνακες Master Category Alias και Subcategory Alias. Ο πίνακας Master Category Alias συνδέει μια βασική κατηγορία με μια ή περισσότερες συνώνυμες ονομασίες της και αντίστοιχα ο πίνακας Subcategory Alias συνδέει μια υποκατηγορία με μια ή περισσότερες συνώνυμες ονομασίες της.

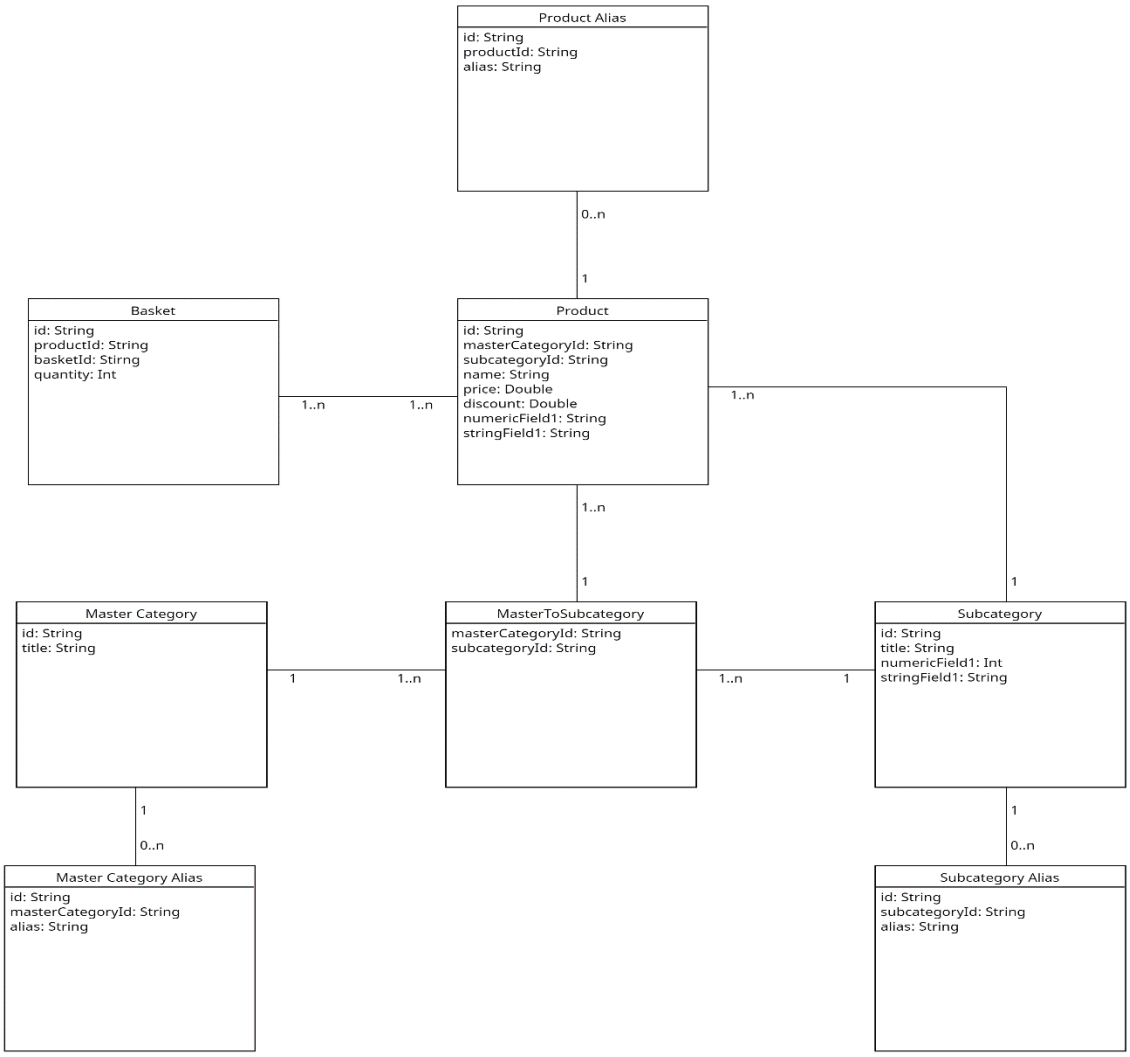
Οι παραπάνω δύο πίνακες είναι ιδιαίτερα σημαντικοί καθώς ο χρήστης μπορεί να ζητήσει μια βασική κατηγορία ή υποκατηγορία με μια συνώνυμη ονομασία της. Μέσω αυτών των πινάκων η εφαρμογή μπορεί να συνδέσει την συνώνυμη ονομασία με την βασική κατηγορία ή την υποκατηγορία στην οποία αναφέρεται.

Ο πίνακας Product αναφέρεται σε οντότητες προϊόντων. Κάθε προϊόν έχει ονομασία, τιμή, έκπτωση, την βασική κατηγορία με την οποία σχετίζεται όπως και την υποκατηγορία με την οποία σχετίζεται. Επιπλέον, για κάθε προϊόν ορίζονται τα Numeric και String fields που αναλύσαμε παραπάνω.

Όπως και για τις κατηγορίες, έτσι και για τα προϊόντα υπάρχει ο πίνακας Product Alias στον οποίο μπορούμε να ορίσουμε συνώνυμες ονομασίες για κάθε προϊόν.

Τέλος, ο πίνακας Basket είναι υπεύθυνος ώστε να διατηρεί την σύνδεση μεταξύ ενός καλαθιού και των προϊόντων που έχουν εισαχθεί σε αυτό.

Παρακάτω φαίνεται η δομή της βάσης δεδομένων σε διάγραμμα.



App Database Structure

3.2 Συγχρονισμός και παραμετροποίηση

Συνεχίζοντας, είναι σημαντικό να αναλύσουμε το βήμα του συγχρονισμού, όπου μέσω αυτού θα αναδείξουμε και την χρησιμότητα της firebase βάσης που διαθέτουμε.

Κατά την πρώτη εγκατάσταση της εφαρμογής πρέπει υποχρεωτικά να εκτελεστεί το βήμα του συγχρονισμού. Το βήμα αυτό είναι υπεύθυνο να δώσει όλα τα απαραίτητα δεδομένα στον dialogflow agent όπως ποιες θα είναι οι οντότητες και για κάθε intent ποια θα είναι τα training phrases.

Το βήμα του συγχρονισμού συνδέεται άμεσα με την δυνατότητα παραμετροποίησης της εφαρμογής αφού σε μια περίπτωση η εφαρμογή μπορεί να υποστηρίζει ηλεκτρονικά προϊόντα ενώ σε άλλη περίπτωση να υποστηρίζει προϊόντα πολλαπλών τύπων.

Στην firebase βάση δεδομένων βρίσκονται όλα τα training phrases που θέλουμε να παράγει η εφαρμογή ανά intent.

Για παράδειγμα, έστω ότι έχουμε ορίσει ένα training phrase στην firebase για το intent choose-product-type. Έστω ότι το training phrase είναι το "Show me the @master_category@.". Η εφαρμογή κατά το βήμα του συγχρονισμού θα παράγει τόσα training phrases όσα και τα master categories.

Συνεπώς, αν τα master categories που υπήρχαν στην βάση της εφαρμογής ήταν:

1. Consoles
2. Smartphones

κατά τον συγχρονισμό η εφαρμογή θα δημιουργούσε 2 training phrases για το intent choose-product-type, τα "Show me the Consoles." και "Show me the Smartphones".

Σαν τελευταίο βήμα, θα έστελνε αυτά τα δύο training phrases στον agent ώστε να εισαχθούν σαν training phrases του intent choose-product-type.

Από εκεί και πέρα, όποτε ο χρήστης θα ζητούσε να δει τις κονσόλες με την φράση "Show me the consoles" ή μια παρεμφερή της, ο agent θα μπορεί να αναγνωρίσει τη φράση ώστε να ικανοποιηθεί το intent choose-product-type.

Η ίδια διαδικασία επαναλαμβάνεται για κάθε training phrase του κάθε intent που έχουμε ορίσει στην firebase.

Επίσης, κατά τον συνγχρονισμό, στέλνονται στον dialogflow agent και οι οντότητες που πρέπει να γνωρίζει.

Στο παραπάνω παράδειγμα, τα δύο master categories που ορίσαμε, Consoles και Smartphones είναι παράλληλα και οντότητες στα πλαίσια του dialogflow agent. Συνεπώς, πρέπει να σταλθούν και αυτές οι πληροφορίες από την εφαρμογή.

Με την ολοκλήρωση του βήματος του συγχρονισμού, έχουμε εξασφαλίσει ότι ο agent έχει πλήρως παραμετροποιηθεί για το εκάστοτε ηλεκτρονικό κατάστημα.

3.3 Επικοινωνία με τον Agent

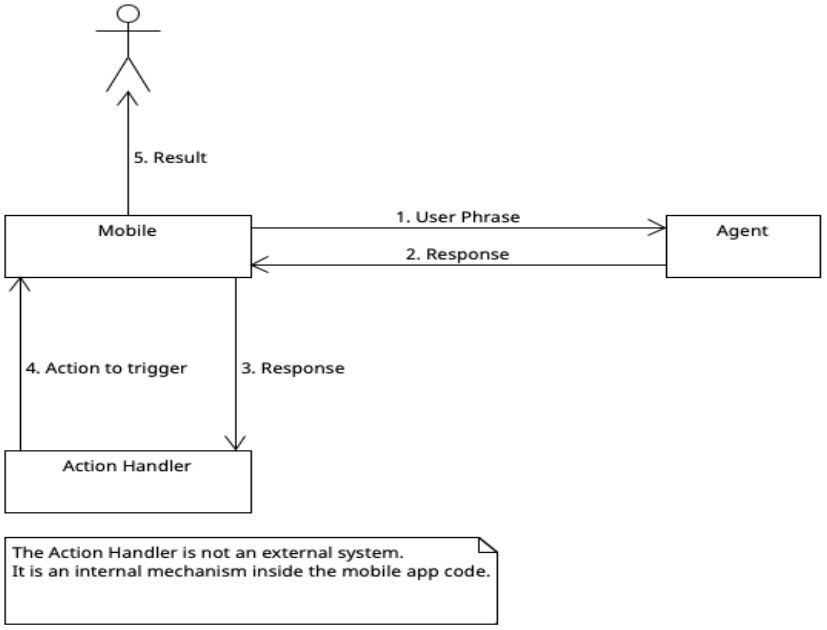
Η επικοινωνία με τον agent γίνεται μέσω http calls με την βοήθεια της βιβλιοθήκης Dialogflow API v2 για Android. Η mobile εφαρμογή προωθεί την φράση που υπαγόρευσε ο χρήστης μέσω του μικροφώνου της συσκευής και περιμένει το response σχετικά με το ποιο intent ικανοποιήθηκε. Μόλις η mobile εφαρμογή λάβει την απάντηση του agent, εκτελεί μια επεξεργασία στο response για να αποφασίσει ποια ενέργεια πρέπει να ενεργοποιηθεί ώστε να πλοηγηθεί ο χρήστης στην κατάλληλη οθόνη ανάλογα με το intent που ικανοποιήθηκε.

Αφού ολοκληρωθεί η πλοήγηση του χρήστη στην κατάλληλη οθόνη, τα data που θα προβάλει η εφαρμογή εξαρτώνται από τις οντότητες που έκανε extract ο agent από την φράση του χρήστη. Οι οντότητες, που για την εφαρμογή είναι δεδομένα που πρέπει να προβάλει στην εκάστοτε οθόνη τα οποία θα αντλήσει από την τοπική SQLite βάση, περιέχονται μεταξύ πολλών άλλων πληροφορίων εντός του response του agent.

Για παράδειγμα, εντός της εφαρμογής ο χρήστης μπορεί να ζητήσει να μπει ένα προϊόν στο καλάθι του. Αφου ο agent επεξεργαστεί την φράση του χρήστη και ικανοποιηθεί το σχετικό intent, στο response που επιστρέφεται στην εφαρμογή περιέχεται σαν οντότητα η ονομασία του προιόντος που ζήτησε ο χρήστης. Η mobile εφαρμογή τότε είναι υπεύθυνη να λάβει αυτή τη παράμετρο, να φορτώσει τις πληροφορίες του συγκεκριμένου προϊόντος από την τοπική βάση και ανάλογα το intent που ικανοποιήθηκε να τις εκμεταλλευτεί κατάλληλα.

Για την συγκεκριμένη περίπτωση η εφαρμογή δεν θα εκτελέσει κάποια πλοήγηση σε άλλη οθόνη, αλλά θα προσθέσει το συγκεκριμένο προϊόν στο καλάθι του χρήστη.

Συνεπώς, σε γενική μορφή η επικοινωνία με τον agent και η επεξεργασία του response από την mobile εφαρμογή φαίνεται στο παρακάτω σχήμα.



App Communication and Response Parsing

Σύμφωνα λοιπόν με την παραπάνω δομή, ο κώδικας της εφαρμογής και συγκεκριμένα ο Action Handler χρειάζεται να γνωρίζει όλα τα intents που έχουμε δημιουργήσει στα πλαίσια του agent.

Ο λόγος για αυτό είναι ότι διαφορετική επεξεργασία θα εκτελέσει ο Action Handler πάνω στο response αν το intent που ικανοποιήθηκε ήταν η προσθήκη προϊόντος στο καλάθι και διαφορετική επεξεργασία θα εκτελέσει στο response αν το intent που ικανοποιήθηκε ήταν η ολοκλήρωση παραγγελίας. Αυτό συμβαίνει διότι κάθε intent εκφράζει και μια διαφορετική ενέργεια του χρήστη συνεπώς αξιοποιούνται με διαφορετικό τρόπο τα δεδομένα του response.

Επίσης, με την παραπάνω δομή μπορούμε εύκολα να επεκτείνουμε την λειτουργικότητα, αρκεί ο Action Handler να γνωρίζει κάθε νέο intent που έχουμε προσθέσει στον agent ώστε να μπορεί να επεξεργαστεί το response και να ειδοποιήσει την υπόλοιπη εφαρμογή κατάλληλα.

4 Android αρχιτεκτονική

Όπως είχε αναφερθεί προηγουμένως, η Android εφαρμογή είναι δομημένη με το μοντέλο MVI (Model-View-Intent). Η συγκεκριμένη αρχιτεκτονική μας επιτρέπει αρκετά εύκολα να εκτελέσουμε ασύγχρονες http κλήσεις χωρίς να μπλοκάρουμε το UI thread. Με εξίσου εύκολο τρόπο μπορούμε να ειδοποιούμε το UI για αλλαγές όταν αυτές οι κλήσεις ή γενικότερα οποιαδήποτε ασύγχρονη λειτουργία ολοκληρωθεί.

Συνεπώς, με την παραπάνω δομή μπορούμε αποδοτικά και γρήγορα να εκτελούμε τις ασύγχρονες κλήσεις στον agent κάθε φορά που ο χρήστης μιλάει στο μικρόφωνο της συσκευής.

Επιπλέον, η MVI αρχιτεκτονική μας επιτρέπει να έχουμε ένα codebase το οποίο πολύ εύκολα μπορεί να επεκταθεί και να συντηρηθεί.

Ένα από τα σημαντικότερα πλεονεκτήματα, όχι μόνο του MVI άλλα κάθε front end αρχιτεκτονικής που επιλέγουμε (MVVM, MVP), είναι πως ο κώδικας του UI είναι πλήρως διαχωρισμένος από τον κώδικα που διαχειρίζεται την λογική της κάθε οθόνης.

Ακόμη, μεταξύ άλλων που θα αναλύσουμε περαιτέρω, έχει γίνει χρήση των Speech To Text και Text To Speech APIs του Android ώστε να έχουμε αναγνώριση φωνής και υπαγόρευσή κειμένου από την εφαρμογή.

4.1 MVI - MvRx

Πιο συγκεκριμένα υλοποιήσαμε το μοντέλο MVI με την χρήση της βιβλιοθήκης MvRx. Η συγκεκριμένη βιβλιοθήκη προσφέρει έναν έτοιμο σκελετό ώστε να δομήσουμε εύκολα και γρήγορα κάθε οθόνη της εφαρμογής με το μοντέλο MVI. Αρχικά, πρέπει να διαχωρίσουμε τον κώδικα που σχετίζεται με την λογική της οθόνης από τον κώδικα που σχετίζεται με το UI και που βρίσκεται στο εκάστοτε fragment.

Ο κώδικας που εμπεριέχει την λογική του κάθε fragment εντάσσεται στην αντίστοιχη ViewModel κλάση της οθόνης. Εκεί γράφουμε τον κώδικα ο οποίος είναι υπεύθυνος να επικοινωνήσει με τον agent, να λάβουμε ή να γράψουμε δεδομένα από και προς την βάση είτε να εκτελέσουμε in memory υπολογισμούς ανάλογα με την ενέργεια που ζητήθηκε.

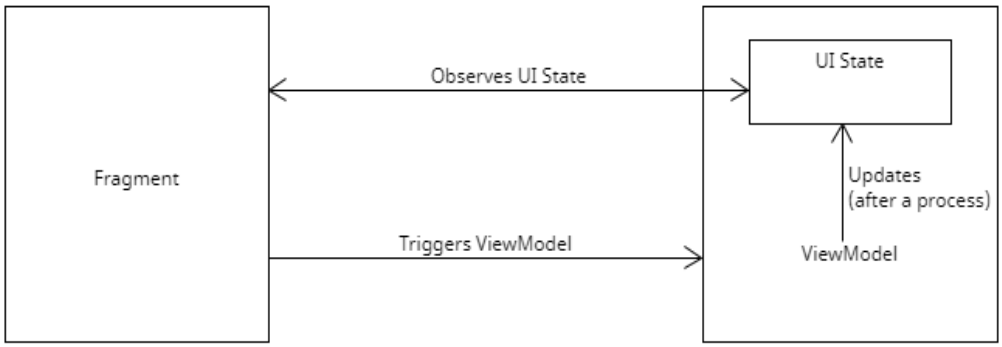
Μέσω του MvRx, κάθε ViewModel κλάση διατηρεί το UI state του fragment με το οποίο σχετίζεται. Πρόκειται για ένα data class το οποίο απαρτίζεται από διάφορα properties που περιγράφουν συνολικά την κατάσταση της οθόνης την συγκεκριμένη στιγμή. Το fragment παρακολουθεί συνεχώς το UI state και περιμένει να εντοπίσει κάποια αλλαγή στα properties η οποία προήλθε από το ViewModel. Μόλις εντοπίσει μια αλλαγή ενεργοποιείται η διαδικασία recompose της οθόνης και ενημερώνεται το fragment και κατά επέκταση η οθόνη που παρουσιάζεται στον χρήστη με το νέο UI.

Για παράδειγμα, θέλουμε για όσο χρονικό διάστημα η εφαρμογή προσπαθεί να επικοινωνήσει με τον agent να παρουσιάζουμε στον χρήστη ένα loading indicator ώστε να είναι ξεκάθαρο πως η εφαρμογή εκτελεί μια χρονοβόρα λειτουργία.

Για να ενημερώσουμε το fragment ότι πρέπει να δείξει το loading indicator πρέπει το ViewModel να ενημερώσει το κατάλληλο property στο UI state που έστω ότι ονομάζεται ShowLoadingIndicator: Boolean. Μόλις αυτό το property αλλάξει τιμή σε True, το fragment που παρακολουθεί το UI state ενημερώνεται με το νέο UI state και δημιουργεί εκ νέου το UI δείχνοντας αυτή τη φορά το loading indicator.

Προφανώς, για λόγους βελτιστοποίησης δεν είναι απαραίτητο κάθε φορά που αλλάζει το UI state ενός fragment να δημιουργείται από την αρχή όλη η οθόνη αλλά μόνο τα κομμάτια της οθόνης που επηρεάστηκαν από την αλλαγή του UI state.

Με αυτή τη δομή έχουμε εξασφαλίσει μια decoupled σχέση μεταξύ του UI της οθόνης και του κώδικα που αφορά την λογική της, δίνοντας την δυνατότητα ο κώδικας να είναι επεκτάσιμος, αποδοτικός και συντηρήσιμος.



MVI Architecture

4.2 Speech To Text – Text To Speech

Τα components Speech To Text (STT) και Text To Speech (TTS) τέθηκαν εξ' αρχής ως απαραίτητα για την ορθή λειτουργία της εφαρμογής. Το STT φροντίζει να ανιχνεύει τις φράσεις που υπαγορεύει ο χρήστης μέσω του μικροφώνου της συσκευής και το TTS υπαγορεύει μια φράση που δίνεται από την εφαρμογή μέσω των ηχείων της συσκευής.

Κατά τον σχεδιασμό, αποφασίστηκε πως το ιδανικό και πιο αποδοτικό για τον χρήστη θα ήταν να υπαγορεύει την ενέργεια που θα ήθελε να εκτελεστεί σε αντίθεση με το να την πληκτρολογούσε. Η χρήση του STT δίνει την δυνατότητα στον χρήστη να μπορέσει να αλληλοεπιδράει με την εφαρμογή χωρίς να χρειαστεί να πατήσει την οθόνη.

Ως αποτέλεσμα, ο χρήστης κερδίζει αρκετό χρόνο, βελτιώνεται κατά πολύ το UX της εφαρμογής και μπορούμε να δώσουμε την δυνατότητα στον χρήστη να συνθέσει μια αλληλουχία ενεργειών που θα ήθελε να εκτελεστούν όπου σε αντίθετη περίπτωση θα έπρεπε να πατήσει μια σειρά από κουμπιά στο UI της εφαρμογής.

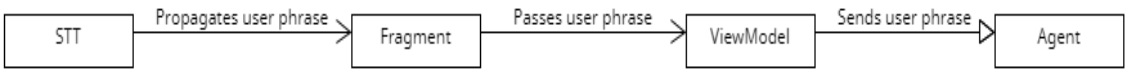
Το Android προσφέρει μια φιλική προς τον προγραμματιστή διεπαφή για να αλληλεπιδράσει με το STT component. Η ενέργεια του STT ενεργοποιείται με το πάτημα του πλήκτρου του μικροφώνου που βρίσκεται σε κάθε οθόνη. Η επιλογή να ενεργοποιείται με το πάτημα του μικροφώνου, σε αντίθεση με το να ήταν μονίμως ενεργοποιημένη, πάρθηκε διότι έχουμε κάνει την παραδοχή πως ενδέχεται να υπάρχει θόρυβος στο περιβάλλον του χρήστη με αποτέλεσμα το STT να λαμβάνει λανθασμένα σήματα. Επιπλέον, το STT καταναλώνει μεγάλα ποσοστά από την μπαταρία της συσκεύης.

Κάθε fragment δημιουργεί ένα δικό του connection με το STT όταν ο χρήστης πατήσει το μικρόφωνο. Αυτό δίνει την δυνατότητα κάθε fragment να μπορεί να χειριστεί όπως επιθυμεί την διασύνδεση με το STT προσφέροντας επεκτασιμότητα. Απαραίτητο είναι το κάθε fragment να υλοποιήσει τις μεθόδους του interface του STT οι οποίες γίνονται invoke ανάλογα με τον αν το STT ανίχνευσε επιτυχώς την φράση που υπαγόρευσε ο χρήστης ή όχι.

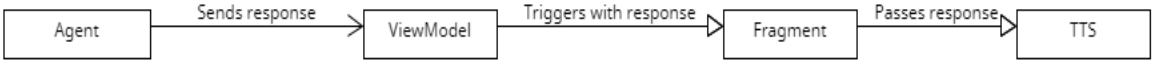
Εφόσον το STT τελειώσει επιτυχώς την διαδικασία ανίχνευσής της φράσης του χρήστη, κάνει invoke την success μέθοδο που έχει ήδη υλοποιήσει το fragment. Παίρνοντας σε string representation την φράση του χρήστη μέσω της success μεθόδου, ειδοποιείται το ViewModel και ξεκινάει η διαδικασία επικοινωνίας με τον agent.

Συνεχίζοντας με το TTS, μας προσφέρει την δυνατότητα να υπαγορεύουμε στον χρήστη την απάντηση του agent σχετικά με το εάν μπόρεσε να ανιχνευθεί επιτυχώς η πρόθεση του. Το παραπάνω βήμα είναι ιδιαίτερα σημαντικό καθώς προσφέρουμε feedback στον χρήστη σχετικά με το αν η ενέργεια που ζήτησε εκτελέστηκε επιτυχώς. Επιπλέον, ενισχύουμε το UX της εφαρμογής καθώς ο χρήστης έχει την αίσθηση ότι αλληλοεπιδρά με έναν πραγματικό άνθρωπο, ξεφεύγοντας από τα τετριμμένα μηνύματα που τείνουν να εμφανίζουν συνεχώς οι mobile εφαρμογές ως feedback.

Για την επικοινωνία με το TTS αρκεί να αρχικοποιηθεί το object που αναφέρεται στο TTS και μέσω μεθόδων που καλούμε στα κατάλληλα σημεία, τροφοδοτούμε το TTS με την απάντηση του agent ώστε να την υπαγορέυεσει.



STT User Phrase to Agent



Agent Response to TTS

4.3 Jetpack Compose

Όσο αφορά την δημιουργία του UI έχει χρησιμοποιηθεί το Jetpack Compose που είναι ο πιο αποδοτικός και σύγχρονος τρόπος να συνθέτεις UI στο Android. Με το Compose μπορούμε να περιγράψουμε το UI της κάθε οθόνης με κώδικα και να ανεξαρτητοποιηθούμε από τα xml αρχεία που χρειάζονταν μέχρι πρόσφατα για να δημιουργήσουμε μια οθόνη.

Πιο συγκεκριμένα, έχουμε την δυνατότητα στο fragment να περιγράψουμε ακριβώς την μορφή των διαφορετικών UI components της οθόνης. Αυτό μας προσφέρει την ευελιξία να έχουμε δυναμικό UI το οποίο μπορούμε άμεσα να συνδέσουμε με το ViewModel μας χωρίς να παρεμβάλλονται xml αρχεία, οι κλήσεις findViewById και τα κοστοβόρα View Bindings.

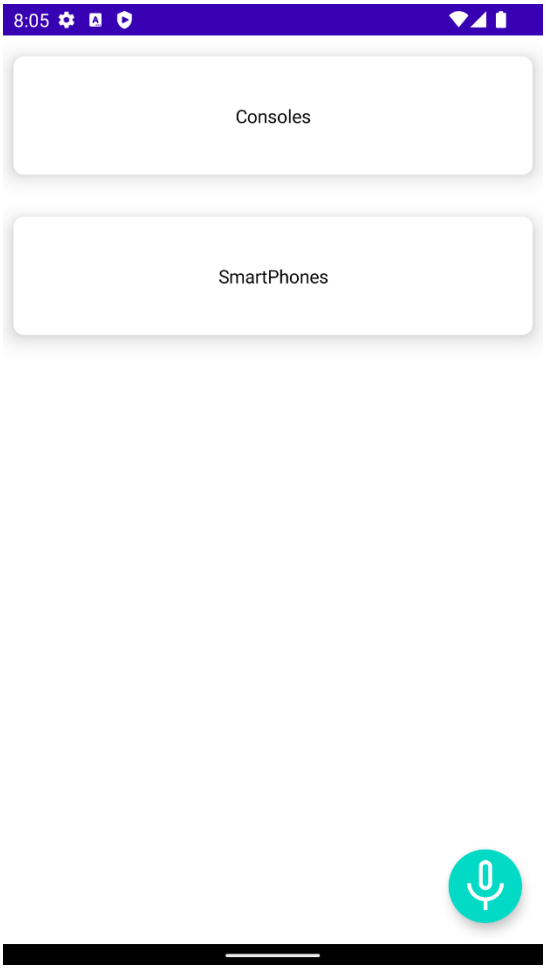
Επιπλέον, με την βοήθεια του Compose έχει απλοποιηθεί αρκετά η χρήση των διαφορετικών UI components και η ένταξή τους στο συνολικό layout της οθόνης. Με απλές δηλώσεις και ελάχιστες γραμμές κώδικα έχουμε την δυνατότητα να δημιουργήσουμε List Views, Grids και πολλά άλλα UI components όπου σε αντίθετη περίπτωση θα απαιτούσε αρκετή δουλεία ώστε να δημιουργηθούν.

Ακόμη, το Compose δεν μας απαγορεύει να έχουμε compatibility με οθόνες που τυχόν χρησιμοποιούν ήδη xml για την δημιουργία του UI. Προσφέρετε ένας αρκετά απλός τρόπος για να μπορέσουμε να επεκτείνουμε το εκάστοτε fragment ώστε να προσθέσουμε επιπλέον UI components με Compose.

5 Παράδειγμα χρήσης Store Agent

Παρακάτω θα παρουσιάσουμε μια τυπική περίπτωση χρήσης της εφαρμογής όπου ο χρήστης πλοηγείτε στα προϊόντα του ηλεκτρονικού καταστήματος, προσθέτει ένα προϊόν με μια συγκεκριμένη ποσότητα στο καλάθι του και ολοκληρώνει την παραγγελία του.

Ο χρήστης ξεκινάει από την οθόνη όπου παρουσιάζονται όλες οι βασικές (master) κατηγορίες του καταστήματος.



Dashboard Page

Πατώντας το πλήκτρο του μικροφώνου μπορεί να μιλήσει στον agent ώστε να δει τις υποκατηγορίες μια συγκεκριμένης κατηγορίας.

Στην περίπτωση μας, ο χρήστης λέει την φράση 'I would like to see the consoles'. Συνοπτικά, η διαδικασία που ακολουθείται είναι πως η φράση αυτή στέλνεται στον agent, ικανοποιείται το intent choose-product-type και παράλληλα εξάγεται η οντότητα consoles.

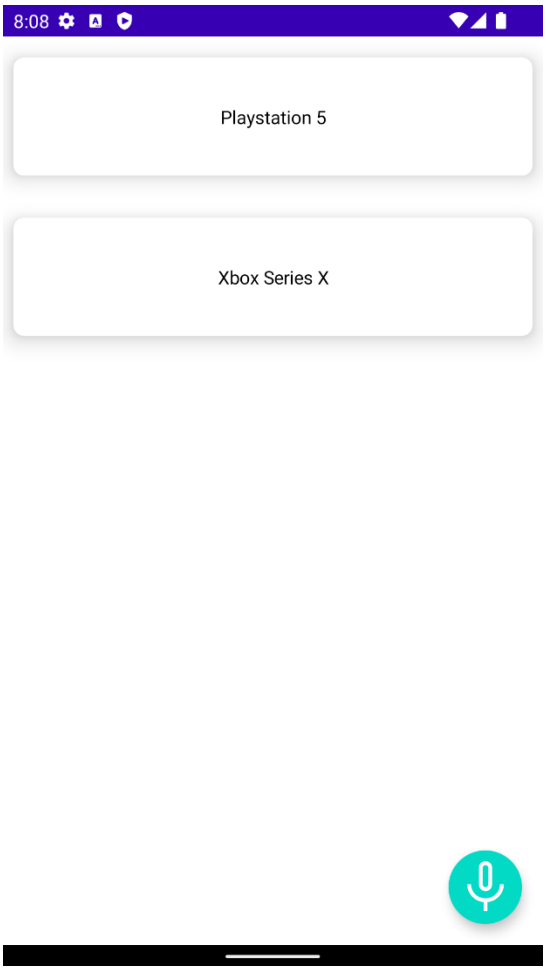
Έπειτα, αυτές οι πληροφορίες, μεταξύ άλλων, επιστρέφονται στην mobile εφαρμογή όπου ο Action Handler είναι υπεύθυνος ώστε να τις μετασχηματίσει σε μια δομή όπου μπορεί να διαχειριστεί η εφαρμογή.

Στην συνέχεια, ο Action Handler προωθεί τις πληροφορίες στο ViewModel της συγκεκριμένης οθόνης. Το ViewModel επεξεργάζεται τα δεδομένα με βάση κάποια συγκεκριμένη λογική και τέλος αποφασίζει αν θα πρέπει να γίνει πλοήγηση σε επόμενη οθόνη.

Στην περίπτωση μας αποφασίζεται πως πρέπει να γίνει πλοήγηση στην οθόνη των υποκατηγοριών. Η μόνη παράμετρος που χρειάζεται να γνωρίζει η οθόνη υποκατηγοριών είναι την βασική κατηγορία της οποίας τις υποκατηγορίες θα προβάλει.

Στην περίπτωση μας, έχουμε αυτή τη πληροφορία οπότε το ViewModel της οθόνης των βασικών κατηγοριών ειδοποιεί το fragment ότι πρέπει να γίνει μια πλοήγηση στην οθόνη υποκατηγοριών με παράμετρο την βασική κατηγορία Consoles.

Αφού η πλοήγηση γίνει με επιτυχία, ο χρήστης καταλήγει στην παρακάτω οθόνη.



Subcategories Page

Στην συνέχεια, ο χρήστης πατώντας το πλήκτρο του μικροφώνου αναφέρει τη φράση 'Show me the Playstation 5.'.

Σε αυτό το σημείο ακολουθείται ακριβώς η ίδια αλληλουχία ενεργειών όπως περιγράψαμε παραπάνω με τις μόνες διαφορές ότι τώρα βρισκόμαστε στην οθόνη των υποκατηγοριών, η οποία έχει το δικό της ViewModel και ικανοποιείται το intent choose-product-type-and-subtype.

Όπως είναι αναμενόμενο, η εφαρμογή αποφασίζει πως πρέπει να γίνει μια πλοήγηση από την οθόνη των υποκατηγοριών στην οθόνη των προϊόντων με παράμετρο το Playstation 5.

Ως αποτέλεσμα, ο χρήστης πλοηγείτε στην παρακάτω οθόνη όπου βλέπει όλα τα διαθέσιμα προϊόντα της υποκατηγορίας Playstation 5.

Products Page

Σε αυτό το σημείο ο χρήστης επιθυμεί να προσθέσει στο καλάθι του το προϊόν Playstation 5 No Disc Model. Συνεπώς, αναφέρει την φράση 'I want 3 Playstation 5 No Disc Model'. Σαν αποτέλεσμα, ικανοποιείται το intent choose-product-from-product-type και ο agent εξάγει τις παραμέτρους Playstation 5 No Disc Model όπου είναι το προϊόν που θέλουμε να προστεθεί στο καλάθι και 3 που είναι η ποσότητα.

Η εφαρμογή λαμβάνει αυτές τις πληροφορίες και το ViewModel ειδοποιεί την οθόνη προβολής προϊόντων πως πρέπει να ενεργοποιηθεί η ενέργεια προσθήκης προϊόντος στο καλάθι.

Επειδή η ενέργεια προσθήκης προϊόντος στο καλάθι μπορεί να εκτελεστεί από οποιοδήποτε σημείο της εφαρμογής, διότι θεωρούμε πως ο χρήστης ενδέχεται να γνωρίζει το προϊόν που επιθυμεί εκ των προτέρων, πρέπει να ειδοποιηθεί ένα global ViewModel το οποίο είναι scoped σε όλη την εφαρμογή (Activity View Model).

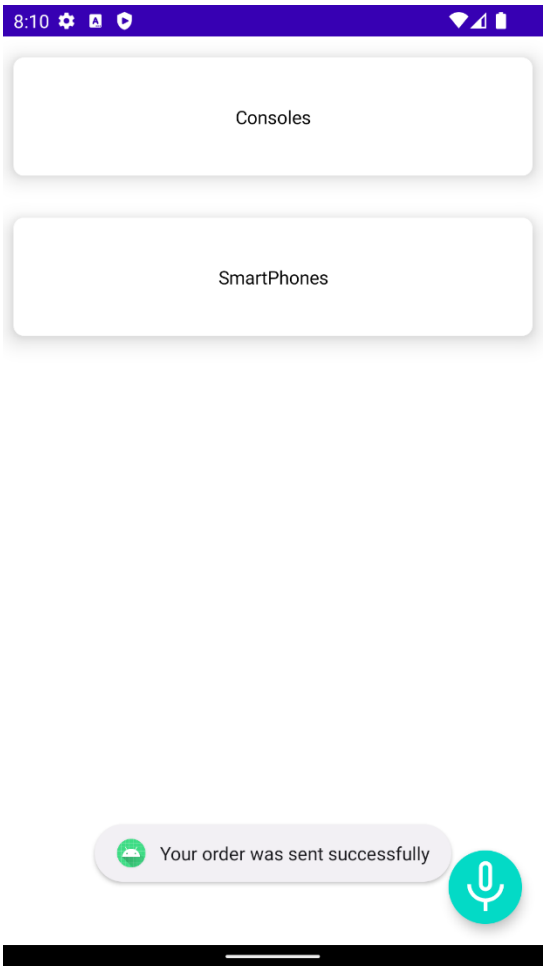
Με αυτόν τον τρόπο, η εφαρμογή εισάγει το προϊόν στο καλάθι με την συγκεκριμένη ποσότητα.

Σε αυτό το σημείο ο χρήστης ζητάει να δει το καλάθι του με την φράση 'Show me my basket.'. Κατά συνέπεια ικανοποιείται το intent see-basket στα πλαίσια του agent και με παρόμοιο τρόπο όπως αναλύσαμε παραπάνω, η εφαρμογή λαμβάνει τις πληροφορίες που της προωθεί ο agent και αποφασίζει ότι πρέπει να πλοηγηθεί στο καλάθι του χρήστη.

Συνεπώς, ο χρήστης βλέπει την παρακάτω οθόνη.

User Basket Page

Στην συνέχεια ο χρήστης ζητάει την ολοκλήρωση της παραγγελίας του και ακολουθεί η εκτέλεση παρόμοιων ενεργειών με αυτές που έχουμε αναφέρει. Η παραγγελία του αποθηκεύεται ενώ ο χρήστης ανακατευθύνεται στην κεντρική σελίδα της εφαρμογής με νέο καλάθι.



Dashboard Page

Σε αυτό το σημείο να υπενθυμίσουμε πως παρουσιάσαμε μια ενδεικτική χρήση της εφαρμογής. Υποστηρίζονται ένα σύνολο από ενέργειες όπως η πλήρης διαχείριση του καλαθιού με μείωση και αύξηση ποσότητας ενός συγκεκριμένου προϊόντος, την διαγραφή συγκεκριμένου προϊόντος από το καλάθι χωρίς να υπάρχει η ανάγκη να βρίσκεται ο χρήστης στην οθόνη του καλαθιού και άλλες.

6 Συμπεράσματα

Κλείνοντας, θα παρουσιάσουμε ορισμένα συμπεράσματα στα οποία οδηγηθήκαμε από την μελέτη και την δημιουργία εφαρμογής που χρησιμοποιεί ευφυή agent.

Είναι φανερό πως η χρησιμότητα τους είναι τεράστια και μπορούν να εφαρμοστούν σε αρκετούς τομείς προσφέροντας πλήρη αυτοματοποίηση σεναρίων χρήσης ενώ παράλληλα δίνεται η εντύπωση στον χρήστη πως αλληλοεπιδρά και εξυπηρετείται από ένα πραγματικό άτομο.

Χαρακτηριστικά παραδείγματα εφαρμογής των ευφυούς agents είναι σε επιχειρήσεις όπως τα καταστήματα, τα ηλεκτρονικά καταστήματα, online αυτοματοποιημένη βοήθεια σε μια ηλεκτρονική υπηρεσία, προσωπικοί assistants στο κινητό εξατομικευμένοι με βάση τον χρήστη και πολλά άλλα.

Επιπλέον, οι ευφυείς agents προσφέρουν λύσεις σε επιχειρήσεις οι οποίες λειτουργούν με ελάχιστους υπαλλήλους, όπου σε περιόδους υψηλού φόρτου χρειάζονται επιπλέον δυναμικό. Επιπροσθέτως, δημιουργείται η δυνατότητα λειτουργίας καταστημάτων τα οποία λειτουργούν εξ' ολοκλήρου με ευφυείς agents για την διεκπεραίωση λειτουργιών και την εξυπηρέτηση των πελατών.

Ένα ακόμη χρήσιμο συμπέρασμα είναι πως με τους ευφυείς agents μπορούμε να εξυπηρετήσουμε άτομα με ειδικές ανάγκες. Για παράδειγμα, άτομα με περιορισμένη όραση τα οποία θα μπορούν με ευκολία να πλοηγηθούν σε μια εφαρμογή με την βοήθεια ενός agent, οι οποίοι μπορούν παράλληλα να εξατομικευτούν για κάθε άτομο ξεχωριστά.

Η εξατομίκευση των agents με βάση τις ανάγκες του χρήστη έρχεται να δώσει πληθώρα λύσεων όταν στόχος μας είναι να αναπτύξουμε μια εφαρμογή η οποία θα είναι ικανή να εξυπηρετεί τον χρήστη ώστε να εκτελέσει σημαντικές για εκείνον ενέργειες.

Ακόμη, οι agents μπορούν να εκπαιδευτούν ώστε να χρησιμοποιηθούν σε συστήματα ασφαλείας, αυτόνομα οχήματα, συστήματα προτάσεων περιεχομένου, ανάλυση δεδομένων και σε πολλές ακόμη κατηγορίες συστημάτων αναδεικνύοντας ακόμη περισσότερο την χρησιμότητά τους.

Καταλήγουμε λοιπόν, πως οι ευφυείς agents μπορούν να δώσουν σημαντικές λύσεις στην εξυπηρέτηση των χρηστών είτε είναι άτομα τα οποία πλοηγούνται σε ένα ηλεκτρονικό κατάστημα, είτε είναι άτομα τα οποία τους χρησιμοποιούν ώστε να ολοκληρώσουν αποτελεσματικά μια συγκεκριμένη εργασία. Σημαντικά επίσης πλεονεκτήματα της χρήσης τους είναι η αίσθηση της αλληλεπίδρασης με πραγματικό άνθρωπο αλλά και η δυνατότητα εξατομίκευσης και προσαρμογής με βάση τον εκάστοτε χρήστη.

Συνεπώς, είναι σίγουρο πως θα παρατηρούμε όλο και περισσότερο να γίνονται κομμάτι της καθημερινότητας μας.

Βιβλιογραφία

Airbnb. (2022). *Mavericks*. Ανάκτηση από Airbnb IO: https://airbnb.io/mavericks/#/

AirBnB. (2022). *Mavericks*. Ανάκτηση από Github: https://github.com/airbnb/mavericks

Google. (2022). *Chatbots Dialogflow*. Ανάκτηση από Google Developers: https://developers.google.com/learn/pathways/chatbots-dialogflow

Google. (2022). *Compose Documentation*. Ανάκτηση από Android Developers: https://developer.android.com/jetpack/compose/documentation

Google. (2022). *Dialogflow Documentation*. Ανάκτηση από Google Cloud: https://cloud.google.com/dialogflow/docs

Google. (2022). *Dialogflow Java Client*. Ανάκτηση από Google Cloud: https://cloud.google.com/dialogflow/es/docs/reference/libraries/java

Google. (2022). *Dialogflow v2 Java Docs*. Ανάκτηση από Google Developers: https://developers.google.com/resources/api-libraries/documentation/dialogflow/v2/java/latest/com/google/api/services/dialogflow/v2/model/GoogleCloudDialogflowV2Intent.html

Google. (2022). *Github*. Ανάκτηση από Java Dialogflow: https://github.com/googleapis/java-dialogflow

Google. (2022). *Read and Write to Firebase*. Ανάκτηση από Google Firebase: https://firebase.google.com/docs/database/android/read-and-write

Google. (2022). *Room*. Ανάκτηση από Android Developer: https://developer.android.com/training/data-storage/room