ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής - Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Chatbot εξατομικευμένης πληροφόρησης για παροχή πληροφοριών για λήψη υπηρεσιών** **Personalized information chatbot providing information for receiving services**
Ονοματεπώνυμο Φοιτητή	**Αλεξανδρόπουλος Αθανάσιος**
Πατρώνυμο	**Αλκιβιάδης**
Αριθμός Μητρώου	**ΜΠΣΠ19002**
Επιβλέπων	**Σακκόπουλος Ευάγγελος, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Νοέμβριος 2022**

Τριμελής Εξεταστική Επιτροπή

Ευάγγελος Σακκόπουλος Αναπληρωτής Καθηγητής	Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής	Διονύσιος Σωτηρόπουλος Επίκουρος Καθηγητής

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον αναπληρωτή καθηγητή, κύριο Ευάγγελο Σακκόπουλο, που παρά όλες τις δυσκολίες που εμφανίστηκαν, συνέχισε να με καθοδηγεί στην εκπόνηση της παρούσας μεταπτυχιακής διατριβής.

Και την οικογένεια μου που παραμένει δίπλα μου και με στηρίζει σε όλα τα βήματα μου.

Περίληψη

Στόχος της παρούσας μεταπτυχιακής διατριβής είναι η δημιουργία ενός chatbot που θα μπορούσε να χρησιμοποιηθεί ως εξατομικευμένο εργαλείο πληροφόρηση. Μέσω της ανάπτυξης μιας εφαρμογής ψηφιακού βοηθού (chatbot), το οποίο θα μπορεί να λάβει εξατομικευμένη πληροφόρηση από ένα χρήστη και να του παρέχει σχετική πληροφόρηση για τη λήψη της συγκεκριμένης υπηρεσίας που αναζητά αλλά και καθοδήγηση για την επίτευξη της. Στο πλαίσιο της εργασίας παρουσιάζεται το σενάριο ανάληψης πτυχιακής εργασίας από φοιτητές του πανεπιστήμιου. Συγκεκριμένα, ο εκάστοτε φοιτητής, θα μπορεί να επισκεφτεί τον ψηφιακό βοηθό και μέσα από ερωτήσεις που θα του κάνει ο ψηφιακός βοηθός και τις απαντήσεις που θα του δώσει, ο χρήστης θα λάβει την απαραίτητη πληροφορία για τη δήλωση του θέματος που του προτείνεται από τον ψηφιακό βοηθό, βάση της επιθυμίας που δηλώνει ο χρήστης, θα μπορεί να λάβει βοήθεια για τη συμπλήρωση της δήλωσης που πρέπει να κάνει στη γραμματεία του τμήματος και τέλος να λάβει ενημέρωση με τη διαδικασία που ισχύει για να προχωρήσει σε αλλαγή θέματος πτυχιακής εργασίας. Για την υλοποίηση της εργασίας χρησιμοποιήθηκαν διάφορες τεχνολογίες διαδικτύου όπως και μηχανικής μάθησης μέσω της τεχνολογίας Rasa Open Source.

Abstract

The aim of this thesis is the implementation of a chatbot application which will be able to be used as a personalized information extraction tool. The application of this virtual assistant chatbot, will be able to get personalized information from a user and provide the appropriate information back, so the user can receive the service he/she requires, but also to guide the user for the finalization of this service. In the scope of this thesis, we will implement the scenario for a student at a university, to choose a thesis topic. In more detail, the university student will be able to visit the virtual assistant and through the questions of the chatbot and his answers, he will be able to receive the appropriate information required to be able to apply for the topic suggested by the chatbot, receive help with filling the application form needed from the university department and to receive information about the process the student has to follow, if there is a need to change the thesis topic. For the implementation of this work, we used different web technologies as well as AI from the Rasa Open Source platform.

Περιεχόμενα

Κατάλογος Εικόνων

Συντομογραφίες

AI	Artificial Intelligence
NLP	Natural Language Processing
NLU	Natural Language Understanding
API	Application Programming Interface
UML	Unified Modeling Language
DB	Database
IDE	Integrated Development Environment
SDK	Software Development Kit
NoSQL	Not Structured Query Language
SQL	Structured Query Language
CLI	Command Line Interface
SSL	Secure Sockets Layer
CDD	Conversation-Driven Development

1 Εισαγωγή

1.1 Chatbot – Ιστορική αναδρομή

Τα chatbot, είτε αλλιώς ψηφιακοί βοηθοί, επεκτείνονται ολοένα και περισσότερο στη καθημερινότητά μας ως χρήστες. Ένα chatbot είναι ένα πρόγραμμα τεχνητής νοημοσύνης που θα μπορούσαμε να πούμε πως απαντάει ως μια νοήμων οντότητα, με χρήση φωνής η κειμένου. Κάθε chatbot μπορεί να κατανοήσει την ανθρώπινη γλώσσα με τη χρήση NLP, το οποίο βασίζεται στη πληροφορία που ο δημιουργός του έχει φροντίσει να του προσθέσει (Khanna et al., 2015).

Θα μπορούσε να πει κανείς ότι η ιδέα των ψηφιακών βοηθών έχει ξεκινήσει από το 1950 που ο Alan Touring είχε αναρωτηθεί αν μπορεί να δημιουργηθεί ένα υπολογιστικό πρόγραμμα το οποίο θα μίλαγε σε μια ομάδα ανθρώπων και αυτοί δε θα μπορούσαν να καταλάβουν αν είναι άνθρωπος ή μηχανή, το όνομα της δοκιμασίας αυτής είναι σήμερα γνωστό και ως δοκιμασία Turing. Το πρώτο chatbot δημιουργήθηκε το 1966 με το όνομα ELIZA που μπορούσε και επέστρεφε τις προτάσεις του χρήστη σε ερωτηματική μορφή και ουσιαστικά προσομοίωνε μια ψυχοθεραπευτική λειτουργία (Adamopoulou & Moussiades, 2020). Όμως, τα chatbot τώρα δε περιορίζονται μόνο στη συνομιλία ενός ανθρώπου και μιας μηχανής, συνέχεια γίνονται προσπάθειες στην προσάρτησή τους σε περισσότερες υπηρεσίες που απευθύνονται σε χρήστες όπως για παράδειγμα, σε γραφεία υποστήριξης, ως συστήματα αυτόματης απάντησης τηλεφώνων, ως εργαλεία εκπαίδευσης, σε πολλές επιχειρήσεις και στο ηλεκτρονικό εμπόριο (Shawar & Atwell, n.d.).

1.2 Σκοπός διατριβής

Από έρευνα που έγινε το 2017, έχει αποδειχθεί ότι ο κυριότερος λόγος (68%) που ένας χρήστης θα χρησιμοποιήσει έναν ψηφιακό βοηθό είναι η παραγωγικότητα που μπορεί να του προσφέρει, για την απόκτηση βοήθειας και πληροφοριών. Συγκεκριμένα η ευκολία, η ταχύτητα και η άνεση χρήσης που προσφέρουν οι ψηφιακοί βοηθοί, ήταν ένα από τους σημαντικότερους λόγους χρήσης ενός chatbot (Brandtzaeg & Følstad, n.d.).

Ο κύριος στόχος της παρούσας μεταπτυχιακής διατριβής είναι η δημιουργία ενός chatbot, που θα μπορούσε να χρησιμοποιηθεί ως εξατομικευμένο εργαλείο πληροφόρησης. Δηλαδή να δράσει ως εργαλείο το οποίο, λαμβάνοντας κάποια πληροφορία από τον εκάστοτε χρήστη, θα μπορούσε να του παρέχει πληροφόρηση ώστε να μπορέσει να λάβει κάποια υπηρεσία που επιθυμεί, καθώς και να τον καθοδηγήσει για την ολοκλήρωση αυτής της υπηρεσίας, αναδεικνύοντας έτσι τη τεχνολογία των chatbot ως ικανή να δράση ως προσωποποιημένος βοηθός στη καθημερινότητα ενός χρήστη. Το παράδειγμα που θα αναπτυχθεί στα πλαίσια αυτής της μεταπτυχιακής διατριβής, αφορά τη διαδικασία που θα χρειαστεί να ακολουθήσει ένας φοιτητής του τμήματος του πανεπιστημίου, για να μπορέσει να επιλέξει ένα θέμα πτυχιακής εργασίας, να ενημερωθεί σχετικά με τη δήλωση της στη γραμματεία του τμήματος και να λάβει πληροφορίες σχετικά με τη διαδικασία αλλαγής του θέματος που έχει ήδη επιλέξει.

Για την υλοποίηση της εφαρμογής του ψηφιακού βοηθού έγινε χρήση της γλώσσας προγραμματισμού Python για άντληση δεδομένων και πληροφοριών, του λογισμικού Rasa Open Source, αλλά και άλλες τεχνολογίες ανάπτυξης εφαρμογών διαδικτύου και βάσεων δεδομένων πιο συγκεκριμένα, Mongo DB και SQLite.

1.3 Αναφορά δομής διατριβής

Η πρώτη ενότητα της παρούσας μεταπτυχιακής διατριβής, αποτελείτε από μια ιστορική αναδρομή στα chatbot, την ανάλυση του σκοπού της διατριβής καθώς και τη περιγραφή της δομής της.

Στη δεύτερη ενότητα, θα γίνει αφορά στις απαιτήσεις του συστήματος και τις προδιαγραφές που προκύπτουν για την υλοποίηση της εφαρμογής chatbot καθώς και τη διάθεσή της στο κοινό. Και τέλος, θα γίνει αναφορά στις τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής.

Η τρίτη ενότητα αποτελείται από την ανάλυση του συστήματος της εφαρμογής, δηλαδή θα γίνει ανάλυση των απαιτήσεων που έχουν προκύψει.

Η τέταρτη ενότητα αφορά τη διαδικασία υλοποίησης του ψηφιακού βοηθού, δηλαδή τα βήματα που ακολουθήθηκαν για την ολοκλήρωση της εφαρμογής, καθώς και μια αναφορά στο μοντέλο υλοποίησης και ελέγχου που ακολουθήθηκε.

Στη πέμπτη ενότητα παραθέτετε ένας συνοπτικός οδηγός χρήσης και επέκτασης της εφαρμογής που υλοποιήθηκε.

Τέλος, ακολουθεί αναφορά στη μελλοντική εξέλιξη που θα μπορούσε να έχει η εφαρμογή chatbot, καθώς και η τεχνολογία των ψηφιακών βοηθών. Και ως τελικό μέρος γίνεται αναφορά στα συμπεράσματα που προέκυψαν από την εκπόνηση της παρούσας μεταπτυχιακής διατριβής.

2 Περιγραφή συστήματος

Σε αυτό το κεφάλαιο θα γίνει αναφορά των προδιαγραφών του συστήματος και των απαιτήσεων που προκύπτουν από αυτές. Στη συνέχεια θα υπάρξει μια περιγραφή των εργαλείων και των τεχνολογιών που χρησιμοποιήθηκαν για την υλοποίηση της παρούσας εφαρμογής ψηφιακού βοηθού.

2.1 Προδιαγραφές - Απαιτήσεις

Το λογισμικό, η εφαρμογή ψηφιακού βοηθού, γνωρίζουμε ότι απευθύνεται σε χρήστες φοιτητές, οι οποίοι θέλουν να ενημερωθούν για τη διαδικασία που πρέπει να ακολουθήσουν σχετικά με την εκπόνηση της πτυχιακής τους εργασίας αλλά και να λάβουν βοήθεια ώστε να μπορέσουν να διαλέξουν το κατάλληλο για αυτούς θέμα. Μέσα από το ψηφιακό βοηθό ένας φοιτητής θα μπορεί να ενημερωθεί για τη διαδικασία που θα πρέπει να ακολουθήσει για αλλαγή του θέματος που έχει ήδη αναλάβει. Θα μπορεί με την απάντηση διαφόρων ερωτήσεων του ψηφιακού βοηθού, να επιλέξει ένα νέο θέμα πτυχιακής εργασίας και να ενημερωθεί σχετικά με τον επιβλέποντα καθηγητή του θέματος για να έρθει σε επικοινωνία με αυτόν Θα μπορεί επίσης να λάβει βοήθεια με την εύρεση αλλά και τη συμπλήρωση του εγγράφου που πρέπει να υποβάλει στη γραμματεία του τμήματός του για να καταχωρηθεί το θέμα που έχει επιλέξει.

Το σύστημα που στεγάζει το λογισμικό της εφαρμογής θα πρέπει να μπορεί να διατεθεί στο ευρύ στο κοινό τόσο για τη χρήση του όσο και για έλεγχο, οπότε θα υλοποιηθεί με τη χρήση του Docker Engine πιο συγκεκριμένα το λογισμικό Docker Compose λόγω της μετατρεψιμότητας που μπορεί να προσφέρει και την ευελιξία σε πακέτα ανοιχτού λογισμικού που μπορούν να χρησιμοποιηθούν μέσα από αυτό για την επίτευξη των βασικών αναγκών μιας τέτοιας εφαρμογής, για παράδειγμα web-server, διαχείριση βάσης δεδομένων.

2.2 Εργαλεία και τεχνολογίες

Τα εργαλεία και οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της παρούσας εφαρμογής θα μπορούσαν να χωριστούν σε δυο κατηγορίες, πρώτα τα εργαλεία που βοήθησαν στην ανάλυση της εφαρμογής, πιο συγκεκριμένα η γλώσσα μοντελοποίησης UML (Unified Modeling Language) και μετά στα εργαλεία που συνέβαλλαν στη ανάπτυξη του τεχνικού μέρους της εφαρμογής, όπως η πλατφόρμα Rasa Open Source, η γλώσσα προγραμματισμού Python η διάφορες βάσης δεδομένων και άλλα.

2.2.1 Ανάλυση απαιτήσεων και λογισμικού

Για την ανάλυση και τη κατανόηση των απαιτήσεων έγινε χρήση της Unified Modeling Language ή αλλιώς UML. Η UML είναι μια γλώσσα μοντελοποίησης γενικής χρήσης στο πεδίο της ανάπτυξης λογισμικού, παρέχοντας ένα τυποποιημένο τρόπο για τον σχεδιασμό ενός συστήματος. Ένα βασικό χαρακτηριστικό της είναι ότι δεν εξαρτάται από κάποια μεθοδολογία ανάπτυξης λογισμικού οπότε μπορεί να ενταχθεί στη διαδικασία του σχεδιασμού και ανάλυσης ανεξαρτήτως της μεθοδολογίας που ακολουθείται. Οι κύριες κατηγορίες διαγραμμάτων που έχει είναι τα δομικά και τα διαγράμματα συμπεριφοράς του συστήματος(Omg, 2009).

2.2.2 Τεχνολογία Server

Για τη φιλοξενία της εφαρμογής επιλέξαμε ως βάση ένα μηχάνημα Linux Ubuntu αν και το λογισμικό δε θα είχε πρόβλημα να λειτουργήσει και σε Windows, η επιλογή αυτή έγινε λόγω της ευελιξίας που προσφέρει το λογισμικό.

Βέβαια προτιμήσαμε να χρησιμοποιήσουμε και τεχνολογία Docker. Το Docker, είναι πλατφόρμα ανοιχτού κώδικα που μπορεί να εικονικοποιησει μια εφαρμογή σε ένα Docker Container, τα Container αυτά συνήθως αποτελούν ένα ολοκληρωμένο μέρος μιας εφαρμογής, όπως για παράδειγμα τη βάση δεδομένων. Κάποια από τα πλεονεκτήματα που μας επιτρέπει η

χρήση του λογισμικού αυτού είναι η ταχύτητα που μπορούμε να “σηκώσουμε” την εφαρμογή μας και να τη διαθέσουμε στο κοινό, αυτό μπορεί να πραγματοποιηθεί επειδή με τη χρήση του η εφαρμογή μας χωρίζεται σε μικρότερα μέρη, η μεταφερσιμότητα, η επεκτασιμότητα αφού τα containers μπορούν να μεταφερθούν αρκετά εύκολα από περιβάλλον σε περιβάλλον και είναι εύκολο να γίνουν διορθώσεις σε αυτά. Η ταχεία παράδοση αποτελεί άλλο ένα πλεονέκτημα αφού τα containers ακολουθούν ένα τυποποιημένο μορφή, οπότε ο υπεύθυνος για το μηχάνημα διακομιστεί πρέπει απλά να φροντίσει στην συντήρηση του μηχανήματος και όχι στην επίλυση τυχόν προβλημάτων της εφαρμογής και τέλος ένα Docker container χρησιμοποιεί μόνο τους πόρους που χρειάζεται, αυτό το καθιστά πιο αποτελεσματικό και αποφεύγεται η σπατάλη περισσότερων υπολογιστικών πόρων (Rad et al., 2017). Η χρήση του Docker έγινε μέσω του λογισμικού Docker Compose, το οποίο προσφέρει ένα τρόπο σύνθεσης μιας εφαρμογής πολλαπλών συστατικών, καθένα από αυτά αποτελεί μια εικόνα και ένα σύνολο οδηγιών που προσδιορίζουν πως πρέπει να συμπεριφέρεται το κάθε συστατικό του (Hasan Ibrahim et al., n.d.).

Τέλος, έγινε χρήση και του λογισμικού Nginx Proxy Manager, το οποίο αποτελεί μια λύση ανοιχτού κώδικα που επιτρέπει την εύκολη πρόσβαση σε σελίδες εφαρμογές χωρίς να χρειάζεται μεγάλη γνώση των παραμέτρων ενός Nginx server (*Nginx Proxy Manager*, n.d.).

2.2.3 Λογισμικό ανάπτυξης chatbot - Rasa open source

Για την παρούσα εργασία όπως αναφέραμε θα χρειαστεί να αναπτυχθεί ένα chatbot. Για την επιλογή του σωστού λογισμικού ανάπτυξης υπήρχαν κάποιες απαιτήσεις που έπρεπε να ληφθούν υπόψιν, αρχικά θα πρέπει η πλατφόρμα να υποστηρίξει και την ελληνική γλώσσα, να είναι λογισμικό ανοιχτού κώδικα, να μην υπάρχει κάποια χρέωση και να είναι εύκολο ως προς τη χρήση του.

Οι πλατφόρμες που επιλέξαμε αρχικά ήταν αποτελούν κάποιες από τις πιο γνωστές στο κλάδο και πληρούν τα περισσότερα αν όχι από τα παραπάνω κριτήρια. Αυτές ονομαστικά ήταν οι: Dialog Flow, Microsoft Bot framework, Wit.ai, BotPress, Intercom και Rasa Open Source.

Το Dialog Flow είναι μια πλατφόρμα κατανόησης φυσικής γλώσσας που διευκολύνει την σχεδίαση και την προσθήκη μιας διεπαφής συνομιλίας χρήστη σε εφαρμογές για κινητά, εφαρμογές διαδικτύου και σε πολλές δημοφιλείς πλατφόρμες όπως Slack, Facebook Messenger, Telegram και άλλες(*Dialogflow Documentation | Google Cloud*, n.d.). Κάποια από τα πλεονεκτήματα του είναι ό,τι παρέχει εργαλεία UI για τη δημιουργία της ροής διαλόγου, μπορεί να γίνει διαθέσιμο μέσα από πληθώρα εφαρμογών και καναλιών, διαθέτει ένα από τα πιο εξελιγμένα μοντέλα NLU που μπορεί να κατανοήσει τις προθέσεις και τα συμφραζόμενα του χρήστη με μεγαλύτερη ακρίβεια, μπορεί να κάνει επαναχρησιμοποίηση των intent, να διαχειριστεί τις μεταβιβάσεις των ροών και να επιτρέψει στο χρήστη να έχει απόκλιση από το κυρίος θέμα και στη συνέχεια να μπορεί να επιστρέψει στη κύρια ροή. Κάποια από τα αρνητικά του και τους λόγους που δεν επιλέχθηκε, είναι ό,τι δε μπορεί να διατεθεί μέσω δικού μας διακομιστή και είναι διαθέσιμο μόνο μέσω της Google, δεν είναι εργαλείο ανοιχτού κώδικα, δεν υπάρχει δωρεάν πακέτο παρά μόνο πακέτο περιορισμένης χρήσης.

Το Microsoft Bot Framework ή αλλιώς Azure bot, αποτελείται από μια συλλογή βιβλιοθηκών εργαλείων και υπηρεσιών που επιτρέπουν στο χρήστη τους να δημιουργήσει ψηφιακούς βοηθούς. Το Microsoft Bot Framework διαθέτει ένα αρθρωτό και επεκτάσιμο SDK που επιτρέπει τη δημιουργία bot και τη σύνδεση τους με υπηρεσίες τεχνητής νοημοσύνης. Πιο συγκεκριμένα κάποια σημαντικά του μέρη είναι:

- Το Bot Framework SDK, το οποίο υπάρχει διαθέσιμο σε διάφορες γλώσσες προγραμματισμού, όπως Python, C#, JavaScript και άλλες
- Εργαλεία CLI για βοήθεια με την ανάπτυξη bot από άκρο σε άκρο.
- Το Bot Connector Service, που είναι υπεύθυνο για την αναμετάδοση των μηνυμάτων και συμβάντων μεταξύ bots και καναλιών

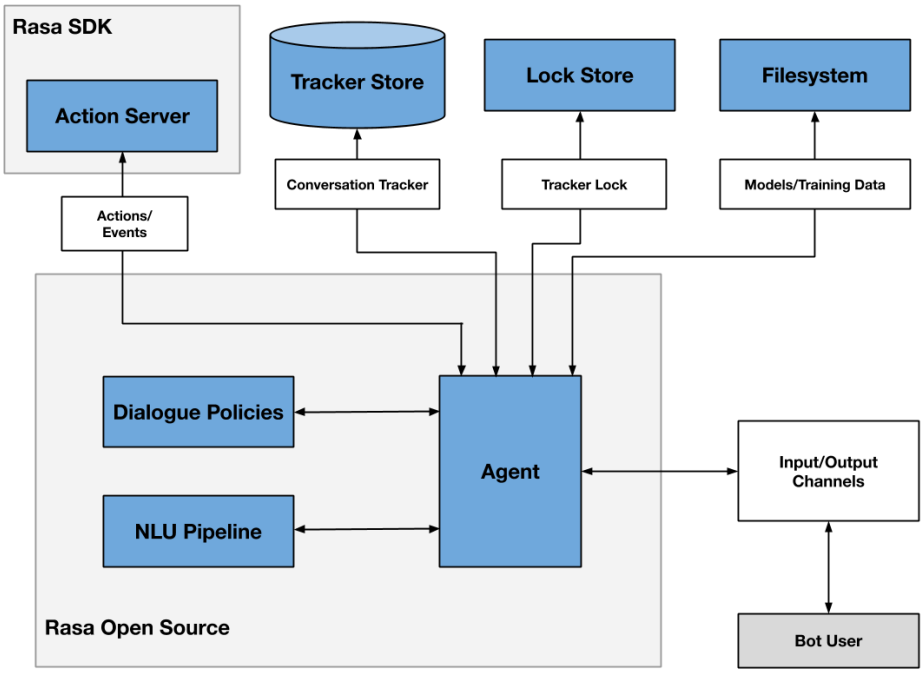
Βέβαια για τη πλήρη λειτουργία του χρειάζεται να χρησιμοποιηθούν και άλλα κομμάτια λογισμικού(*What Is the Bot Framework SDK? - Bot Service | Microsoft Learn*, 2022). Κάποιοι από τους λόγους που δεν επιλέχθηκε ήταν ό,τι τα Ελληνικά δεν υποστηρίζονται στο μέγιστο, και το Microsoft Bot Framework δεν διατίθεται δωρεάν παρά μόνο για περιορισμένη χρήση. Αξίζει να

σημειώσουμε ότι το περιβάλλον ανάπτυξης του που είναι το Azure και αυτό είναι διαθέσιμο επί πληρωμή.

Το BotPress είναι μια πλατφόρμα ανοιχτού λογισμικού για τη δημιουργία bot, είναι πλήρως επεκτάσιμο και υποστηρίζει πολλές βιβλιοθήκες NLU. Έχει σχεδιαστεί για να μπορεί να υλοποιήσει chatbot με τη χρήση εικονικών ροών και με τη χρήση μικρών ποσοτήτων δεδομένων με τη μορφή προθέσεων, οντοτήτων και θέσεων υποδοχής. Διαθέτει ένα πρόγραμμα δημιουργίας οπτικών συνομιλιών και εξομοιωτή για να μπορούν να δοκιμαστούν οι συνομιλίες αυτές. Παρέχει τη δυνατότητα εισαγωγής κώδικα Javascript για τη δημιουργία εξατομικευμένων μεθόδων. Μπορεί να παρέχει διασύνδεση με πληθώρα καναλιών όπως Facebook Messenger, Slack, Telegram και άλλα. Οι λόγοι που δεν επιλέχθηκε ήταν το η μικρή κοινότητα του, δεν δίνει μεγάλη έμφαση στο AI με αποτέλεσμα να λειτουργεί περισσότερο για την απάντηση συχνών ερωτήσεων και βάση του εγχειριδίου χρήσης τα Ελληνικά δεν υποστηρίζονται από το ίδιο παρά μόνο με τη χρήση άλλων βιβλιοθηκών(*What Is Botpress?* | *Botpress Documentation*, n.d.).

Η πλατφόρμα που επιλέξαμε είναι το Rasa Open Source. Το Rasa Open Source είναι μια από τις ποιο δημοφιλής πλατφόρμες ανοιχτού κώδικα για την ανάπτυξη AI βοηθών με τη χρήση κειμένου και φωνής (*Introduction to Rasa Open Source & Rasa Pro*, 2022). Είναι γραμμένο στη γλώσσα προγραμματισμού Python. Ένα σημαντικό πλεονέκτημα του είναι ότι μπορεί να υλοποιηθεί σε τοπικό περιβάλλον, δηλαδή η ανάπτυξη και ο έλεγχος της εφαρμογής, μπορεί να γίνει στον υπολογιστή του προγραμματιστή και η επικοινωνία μπορεί να γίνει μέσω της γραμμής εντολών μέσω του CLI εργαλείου που εμπεριέχει. Επίσης, υποστηρίζει την επικοινωνία μέσω πληθώρας καναλιών όπως Facebook Messenger, Slack, Telegram αλλά και με χρήση WebSocket ή Rest API επιτρέπουν στον προγραμματιστή να το προσθέσει εύκολα στη μια δική του ιστοσελίδα. Τα βασικά στοιχεία που το αποτελούν είναι:

- Το Rasa Open Source, που αποτελείτε από
    - Το NLU, το μέρος που είναι υπεύθυνο να εντοπίζει τις επιθυμίες του χρήστη (intents) να εξάγει τις οντότητες και να ανακτά την απόκριση που θα αποστείλει.
    - Το Dialogue Policies που επιλέγει την επόμενη ενέργεια που θα εκτελέσει ο agent
- Το Rasa SDK που αποτελεί το μέρος του κώδικα Python που μπορούμε να γράψουμε εμείς για να κάνουμε δράσεις όπως επικοινωνία με μια βάση δεδομένων, κλήση σε κάποιο API και άλλα.
- Κανάλια εισόδου εξόδου, που αποτελούν τα το σύνολο καναλιών που μπορεί να ενσωματωθεί το Rasa Open Source
- Το Tracker Store, μια βάση δεδομένων που είναι υπεύθυνη για την αποθήκευση των της κατάστασης των συνομιλιών chatbot και χρήστη.



Εικόνα 1 Αρχιτεκτονική Rasa (πηγή: https://rasa.com/docs/rasa/arch-overview)

2.2.5 Βάση δεδομένων

Υπήρξαν δυο περιπτώσεις που χρειάστηκε η χρήση βάσεων δεδομένων, αρχικά για την αποθήκευση και άντληση πληροφοριών για χρήση από το chatbot, και για το Tracker Store, δηλαδή για την αποθήκευση των της κατάστασης των συνομιλιών chatbot και χρήστη.

Για τη πρώτη περίπτωση επιλέχθηκε η σχεσιακή βάση δεδομένων SQLite. Η SQLite είναι μια δημοφιλή βάση δεδομένων ανοιχτού κώδικα και χρησιμοποιείται πολύ συχνά σε μικρής εμβέλειας εφαρμογές. Χαρακτηριστικά , η επιλογή του συγκεκριμένου λογισμικού έγινε λόγω:

- της ταχύτητας εγκατάστασης που προσφέρει, αφού μπορεί να υπάρχει ως αρχείο στο πηγαίο κώδικα της εφαρμογής
- της ευκολίας χρήσης
- του κόστους, διότι ως λογισμικό ανοιχτού κώδικα δεν υπάρχει κάποια επιβάρυνση
- καθώς και την ευκολία διασύνδεσης με πληθώρα γλωσσών προγραμματισμού όπως η Python

Για την ανάπτυξη του Tracker Store επιλέχθηκε η χρήση της NoSQL βάσης δεδομένων MongoDB, η επιλογή έγινε λόγω της προβολής που παρέχει μέσω του προγράμματος MongoDBCompas σε μορφή collection, καθώς είναι πιο εύκολη η ανάγνωση του tracker store. Αξίζει να σημειωθεί ότι και η MongoDB διατίθεται δωρεάν και αποτελεί λογισμικό ανοιχτού κώδικα.

3 Ανάλυση απαιτήσεων

Σε αυτό το κεφάλαιο θα γίνει ανάλυση των βασικών λειτουργιών του λογισμικού που εξετάζει η παρούσα μεταπτυχιακή διατριβή. Θα δημιουργηθούν διαγράμματα με τη χρήση της ενοποιημένης γλώσσας σχεδιασμού (unified modeling language) (UML) για την ευκολότερη κατανόηση και παρουσίαση των απαιτήσεων του λογισμικού, αλλά και των περιορισμών που δημιουργούν. Συγκεκριμένα θα εξεταστούν οι περιπτώσεις χρήσης της εφαρμογής με χρήση του διαγράμματος περιπτώσεων χρήσης (use case diagram), θα γίνει η εξακρίβωση της συνομιλίας και του μονοπατιού της συζήτησης που θα κάνει ο ψηφιακός βοηθός με τον χρήστη με χρήση διαγράμματος δραστηριοτήτων (activity diagram), θα αποτυπωθούν οι οντότητες που χρειάζεται να δημιουργηθούν στην εσωτερική βάση δεδομένων για την άντληση των απαραίτητων δεδομένων με τη δημιουργία ενός διαγράμματος σχέσεων - οντοτήτων (entity relationship diagram), θα γίνει ανάλυση της συνεργασίας μεταξύ των διαφόρων αντικειμένων που αποτελείτε η εφαρμογή με τη χρήση διαγράμματος συνεργασίας (Collaboration diagram) καθώς και τέλος η σειρά ανταλλαγής των μνημάτων μεταξύ των διάφορων αντικειμένων του λογισμικού με τη δημιουργία διαγράμματος ακολουθίας (sequence diagram).

3.1 Κατανόηση απαιτήσεων

Το λογισμικό, η εφαρμογή ψηφιακού βοηθού, αποτελεί ένα ψηφιακό βοηθό που λαμβάνοντας εξατομικευμένη πληροφορία για ένα χρήστη μπορεί να του παρέχει πληροφορίες για την υπηρεσία που θέλει να λάβει καθώς και να τον καθοδηγήσει για την επίτευξη της. Η εξατομίκευση στη ζωή μας θεωρείτε πανταχού παρούσα και εμπεριέχεται σε πληθώρα εφαρμογών, έχοντας με τη σειρά της επηρεάσει την οπτική μας προς αυτή, ενώ αρχικά μπορεί να θεωρούταν ως μια περιττή πολυτέλεια τώρα οι χρήστες έχουν φτάσει να τη θεωρούν απαραίτητη(Goldenberg et al., 2021).

Η εφαρμογή μας χρειάζεται να μπορεί να υποστηρίξει κάποια πολύ βασικά χαρακτηριστικά. Αρχικά για να επιτύχουμε την εξατομικευμένη πληροφόρηση, δεδομένου ότι δε διαθέτουμε δεδομένα αναφορικά με τους χρήστες, θα δημιουργήσουμε ένα πολύ απλό μοντέλο χρήστη, το οποίο θα κατασκευάζει ο ψηφιακός βοηθός σε κάθε επίσκεψη της εφαρμογής από ένα χρήστη, ο ψηφιακός βοηθός θα πρέπει να μπορεί να πάρει ορισμένες πληροφορίες από το φοιτητή αυτόν, με ερωτήσεις που θα του κάνει, για να μπορέσει να προβάλει ένα σωστό αποτέλεσμα για επιλογή νέου θέματος πτυχιακής εργασίας. Ο μόνος γνώμονας επιλογής προτίμησης για το παράδειγμά μας θα είναι η επιλογή ενός μαθήματος. Με βάση την επιλογή κάποιου μαθήματος από το χρήστη, ο βοηθός θα μπορεί να γνωρίσει μια επιθυμία του χρήστη ως προς το μάθημα αυτό και να του προτείνει σχετικά θέματα πτυχιακής εργασίας. Επίσης, ο ψηφιακός βοηθός θα πρέπει να μπορεί να παρέχει καθοδήγηση προς το χρήστη και για άλλες υπηρεσίες, αυτές όπως προκύπτει από την έρευνα που έγινε, είναι κοινές για όλους του φοιτητές, συγκεκριμένα θα πρέπει ο χρήστης να μπορεί να λάβει ενημέρωση για αλλαγή του θέματος που μπορεί να έχει ήδη δηλώσει αλλά και καθοδήγηση για την υποβολή της απαιτούμενης δήλωσης θέματος στη γραμματεία του τμήματος. Τα δυο τελευταία κομμάτια θα μπορούσαν να θεωρηθούν και ως απλές συχνές ερωτήσεις αλλά δε παύουν να αποτελούν ένα βασικό μέρος της δήλωσης νέου θέματος πτυχιακής εργασίας.

3.1.1 Διαδικασία ανάθεσης θέματος πτυχιακής εργασίας

Όπως προκύπτει, θα χρειαστεί να γίνει άντληση πληροφοριών σχετικά με τη διαδικασία εκπόνησης πτυχιακών εργασιών και συγκεκριμένα με την ανάθεση πτυχιακής εργασίας. Αυτό επιτεύχθηκε με την χρήση του κανονισμού εκπόνησης πτυχιακής εργασίας του τμήματος πληροφορικής του Πανεπιστημίου Πειραιώς[1].

Συγκεκριμένα από εκεί προέκυψε, ό,τι η διαδικασία που πρέπει να ακολουθήσει ένας φοιτητής για να του ανατεθεί μια πτυχιακή εργασία, είναι αρχικά η επιλογή του θέματος που επιθυμεί από ιστότοπο που διαθέτουν οι διδάσκοντες του τμήματος. Στη συνέχεια να γίνει

[1] https://www.cs.unipi.gr/files/kanonismos_ekponysis_ptyxiakis_ergasias.pdf

έγκριση του από το μέλος του ΔΕΠ, ΕΔΙΠ και λοιποί διδάσκοντες του Τμήματος που έχουν ανακοινώσει το θέμα και τέλος, να γίνει δήλωση του θέματος πτυχιακής εργασίας στη γραμματεία του τμήματος. Ο στόχος της εργασίας είναι να επιτρέψει στο φοιτητή να διαλέξει εύκολα ένα θέμα και να του δώσει την απαραίτητη πληροφορία για να έρθει σε επικοινωνία με το επιβλέποντα που έχει οριστεί για το θέμα αυτό και στη συνέχεια, να είναι ικανό να τον βοηθήσει στη δήλωση του θέματος αυτού στη γραμματεία του τμήματος. Για να μπορέσουμε να αναπτύξουμε την εφαρμογή της εργασίας, θα γίνει απόπειρα δημιουργίας μιας βάσης δεδομένων που θα μπορούσε να χρησιμοποιηθεί ως αποθετήριο θεμάτων πτυχιακών εργασιών και από εκεί ο χρήστης, φοιτητής, θα μπορέσει στο τέλος να επιλέξει κάποιο θέμα και να ενημερωθεί σχετικά με την διαδικασία που θα πρέπει να ακολουθήσει για τη δήλωσή του.

3.2 Διάγραμμα αρχιτεκτονικής εφαρμογής

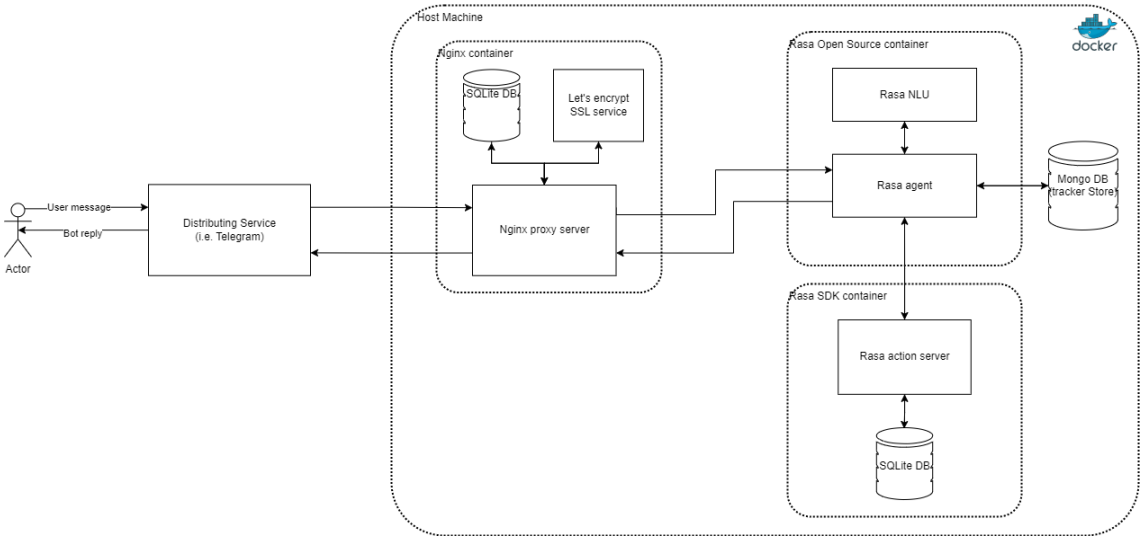
Στο παρακάτω διάγραμμα παρουσιάζεται ένα διάγραμμα υψηλού επιπέδου αρχιτεκτονικής που θα ακολουθήσει η εφαρμογή του ψηφιακού βοηθού. Συγκεκριμένα, βλέπουμε αρχικά ότι για να μπορέσει να επικοινωνήσει με τον ψηφιακό βοηθό ο χρήστης, θα πρέπει ο βοηθός να παρέχεται μέσω κάποιου καναλιού επικοινωνίας, στην προκειμένη περίπτωση θα επιλέξουμε το Telegram, στη συνέχεια υπάρχει η επικοινωνία με το υπόλοιπο σύστημα το οποίο θα στεγαστεί σε έναν Ubuntu Server και για την υλοποίηση του θα χρησιμοποιηθεί η τεχνολογία Docker Engine για να δημιουργήσει τα απαραίτητα υποσυστήματα του περιβάλλοντος της εφαρμογής.

Τα υποσυστήματα αυτά είναι, ένας Web Server NGINX που αναλαμβάνει τη διάθεση της εφαρμογής στο κοινό, χρήστες της, και περιέχει τα υποσυστήματα SQLite DB που είναι η βάση δεδομένων που αποθηκεύονται τα αρχεία διαχείρισής του, όπως και το Let’s Encrypt SSL service που είναι το υποσύστημα υπεύθυνο για την έκδοση SSL πιστοποιητικών για την εφαρμογή, τα οποία είναι απαραίτητα για να μπορέσει η εφαρμογή να είναι διαθέσιμη μέσω του Telegram.

Το επόμενο υποσύστημα που υπάρχει είναι το ίδιο το Rasa Open Source το οποίο λαμβάνει τα μηνύματα του χρήστη και μέσω του NLU που θα δημιουργηθεί για την εφαρμογή, μπορεί να καθορίσει τη διεργασία που πρέπει να γίνει από τα λοιπά υποσυστήματα για να σταλθεί η σωστή απάντηση στο χρήστη και είναι τέλος υπεύθυνο για την αποστολή της απάντησης αυτής.

Το υποσύστημα του Rasa Open Source συνδέεται άμεσα με το Rasa SDK το οποίο αποτελείται από τις βοηθητικές μεθόδους οι οποίες κάνουν επαλήθευση τα δεδομένα του χρήστη είναι υπεύθυνες για την άντληση δεδομένων από τη βάση δεδομένων που έχει δημιουργηθεί για τις ανάγκες τις εφαρμογής, καθώς και την ίδια τη βάση δεδομένων.

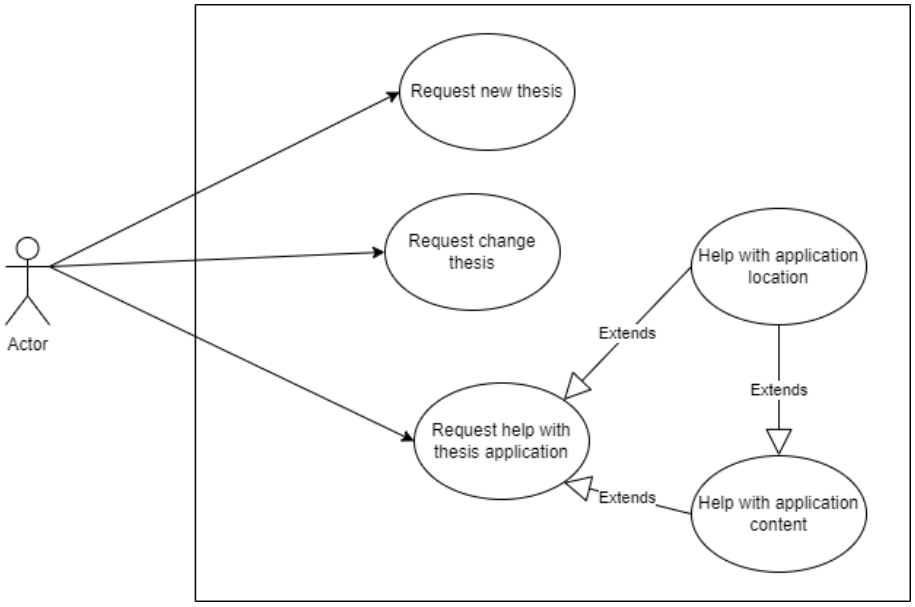
Τέλος, σε άμεση συνεργασία με το Rasa Open Source είναι και το υποσύστημα MongoDB(tracker store) το οποίο είναι υπεύθυνο για την καταγραφή όλων των ενεργειών που γίνονται προς και από το ψηφιακό βοηθό, η χρήση του ωφελεί στην καλύτερη λειτουργία του ψηφιακού βοηθού μιας και μέσα από αυτό μπορούμε να δούμε τις κινήσεις που πραγματοποίησε ένας χρήστης και τις απαντήσεις ή κινήσεις που έκρινε σωστές η εφαρμογή και να προβούμε σε διάφορες διορθωτικές κινήσεις ώστε να γίνει πιο αποτελεσματική η χρήση της εφαρμογής ψηφιακού βοηθού.



Εικόνα 2 Οπτικοποίηση αρχιτεκτονικής εφαρμογής

3.3 Περιπτώσεις χρήσης εφαρμογής

Παρακάτω θα εξετάσουμε και θα αναλύσουμε με τη βοήθεια διαγράμματος περιπτώσεων χρήσης (use case diagram) τις περιπτώσεις χρήσης της εφαρμογής από τους χρήστες, φοιτητές.



Εικόνα 3 Διάγραμμα περιπτώσεων χρήσης

Στην παραπάνω απεικόνιση του διαγράμματος περιπτώσεων χρήσης, φαίνονται οι κύριες λειτουργίες που θα μπορεί να κάνει ο φοιτητής, χρήστης της εφαρμογής. Οι λειτουργίες αυτές είναι, πρώτα ότι θα μπορεί να μάθει για τη διαδικασία που μπορεί να ακολουθήσει αν θέλει να αλλάξει το θέμα της πτυχιακής εργασίας που έχει αναλάβει, θα μπορεί να λάβει καθοδήγηση επιλογής νέου θέματος πτυχιακής εργασίας και τέλος να λάβει βοήθεια όσο αναφορά την αίτηση που θα πρέπει να υποβάλει στη γραμματεία του τμήματός του, σχετικά με τα περιεχόμενα που θα πρέπει να έχει η αίτηση αυτή αλλά και για το μέρος από το οποίο θα μπορούσε να κάνει λήψη του εγγράφου της αίτησης για να προβεί στη συμπλήρωση του.

3.3.1 Μετά-συνθήκες

Εάν ο φοιτητής ολοκληρώσει τη διαδικασία επιλογής θέματος πτυχιακής εργασίας τότε θα πληροφορηθεί με τα στοιχεία του επιβλέποντα καθηγητή, ώστε να μπορέσει να έρθει σε επικοινωνία μαζί του όπως προβλέπεται από τον κανονισμό.

Εάν ο φοιτητής ζητήσει αλλαγή του θέματος πτυχιακής εργασίας που έχει αναλάβει να εκπονήσει θα ενημερωθεί με τη διαδικασία που πρέπει να ακολουθήσει για την πραγματοποίηση της διαδικασίας αυτής.

Στην περίπτωση ο φοιτητής ζητήσει βοήθεια με τη συμπλήρωση του εντύπου της δήλωσης θέματος πτυχιακής εργασίας, θα ενημερωθεί με τα πεδία που θα πρέπει να γνωρίζει όπως και με το μέρος που μπορεί να προμηθευτεί το έγγραφο αυτό.

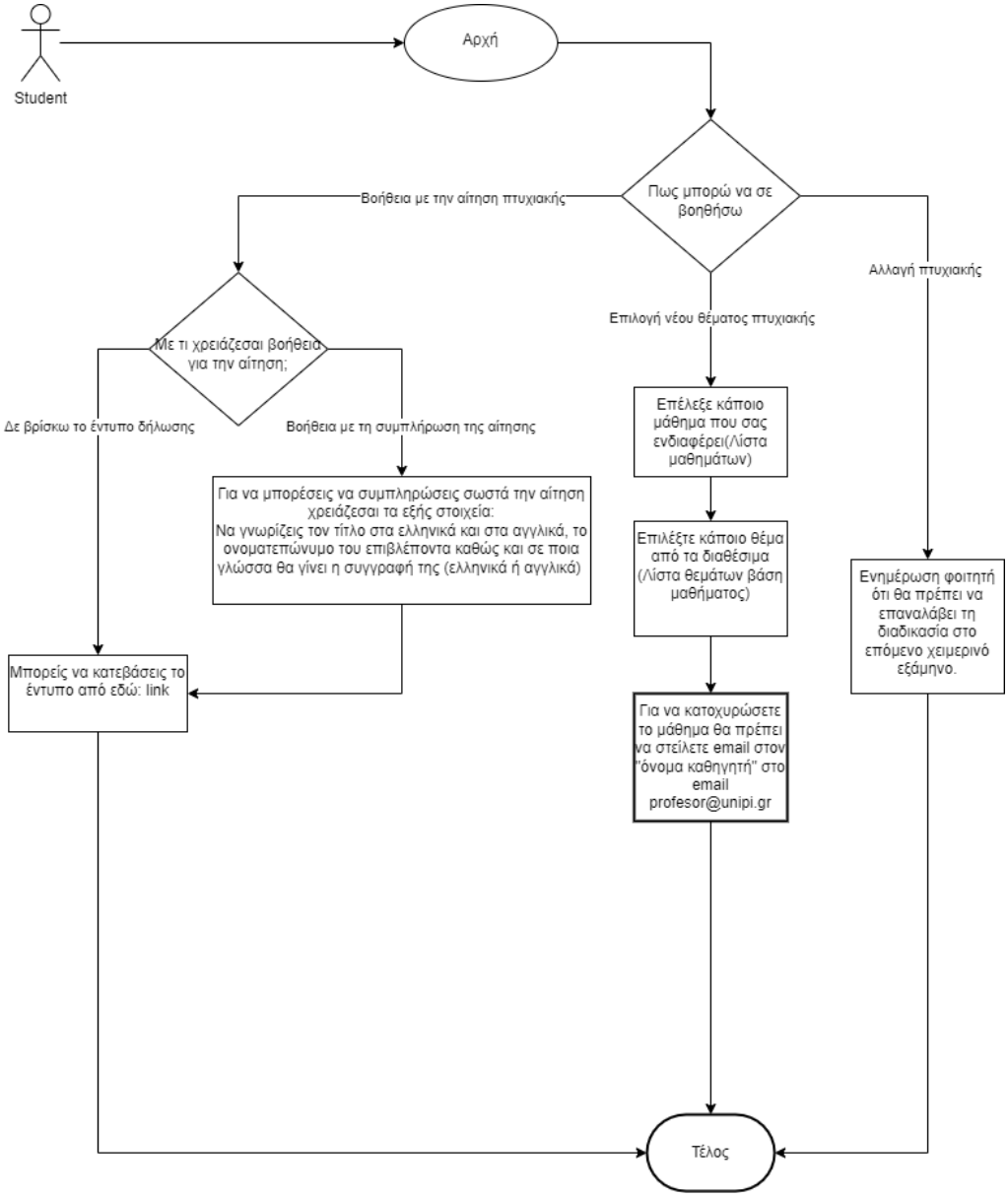
Σε κάθε περίπτωση με το πέρας της χρήσης της εφαρμογής, ο ψηφιακός βοηθός ζητάει από το χρήστη να συμπληρώσει μια φόρμα αξιολόγησης.

Στο παρακάτω διάγραμμα φαίνονται οι ροές που θα μπορεί να ακολουθήσει ένας χρήστης του ψηφιακού βοηθού οι οποίες προκύπτουν ως εξέλιξη του παραπάνω διαγράμματος περιπτώσεων.

3.4 Conversation flow diagram

Στο παρακάτω διάγραμμα φαίνονται οι ροές που μπορεί να ακολουθήσει ένας χρήστης της εφαρμογής, ψηφιακού βοηθού, οι οποίες προκύπτουν ως επέκταση του παραπάνω διαγράμματος περιπτώσεων χρήσης. Οι βασικοί συμμετέχοντες των ροών που προκύπτουν είναι ο χρήστης, φοιτητής και ο ψηφιακός βοηθός. Για την απεικόνιση του έγινε υλοποίηση ενός flowchart διαγράμματος, το οποίο συνίσταται για την απεικόνιση μιας εφαρμογής chatbot, ψηφιακού βοηθού(Bohl & Rynn, 2008).

Επίσης, πρέπει να σημειωθεί ότι στο τέλος των ροών θα δίνεται η δυνατότητα στο χρήστη να επαναλάβει κάποια διαδικασία, όπως και μέσα από την εφαρμογή δίνονται και άλλες απαντήσει χαρακτηριζόμενες ως συχνές ερωτήσεις που μπορεί να κάνει ο χρήστης σε ένα ψηφιακό βοηθό σε σχέση με την εκπόνηση της πτυχιακής του εργασίας, όπως για παράδειγμα η ερώτηση «Είναι υποχρεωτικό να παραδώσω πτυχιακή εργασία;» και άλλες παρόμοιες, που αν και στην αρχική υλοποίηση δε κρίθηκαν σημαντικές, μέσα από δοκιμές και ανατροφοδότηση που υπήρξε από χρήστες κρίθηκε καλό να προστεθούν.



Εικόνα 4 Conversation flow diagram

3.4.1 Κύρια ροή

Η κύρια ροή, που είναι και η βασική λειτουργία της εφαρμογής μας, περιγράφει την περίπτωση που ο φοιτητής θέλει να επιλέξει νέο θέμα πτυχιακής εργασίας. Αυτό αποτελεί και το βασικό σενάριο χρήσης της εφαρμογής.

1. Ο χρήστης ξεκινάει τη συνομιλία με το chatbot
2. Το chatbot τον καλωσορίζει και του δίνει τρεις επιλογές που μπορεί να τον βοηθήσει, να επιλέξει θέμα πτυχιακής εργασίας, να ενημερωθεί σχετικά με την αλλαγή υπάρχουσας πτυχιακής εργασίας και να λάβει πληροφορίες σχετικά με την αίτηση της πτυχιακής εργασίας
3. Ο φοιτητής απαντάει ό,τι θέλει να επιλέξει νέο θέμα πτυχιακής εργασίας
4. Το chatbot αποθηκεύει την επιθυμία του φοιτητής, δείχνει μια λίστα με μαθήματα από το τμήμα και παρακινεί τον μαθητή να διαλέξει ένα από αυτά.
5. Ο φοιτητής διαλέγει ένα μάθημα από τη λίστα

6. Το chatbot αποθηκεύει την επιλογή του και δείχνει στον φοιτητή τα θέματα που είναι σχετικά με το μάθημα που επέλεξε.
7. Ο φοιτητής επιλέγει ένα από τα θέματα
8. Το chatbot ψάχνει τα στοιχεία του καθηγητή βάση των αποθηκευμένων επιλογών, στη συνέχεια ενημερώνει τον φοιτητή ότι θα πρέπει να έρθει σε επικοινωνία με τον επιβλέποντα καθηγητή στέλνοντας του email και του δείχνει τη διεύθυνση email.
9. Η ροή τελειώνει με επιτυχία.

3.4.2 Εναλλακτική ροή 3α

Η πρώτη εναλλακτική ροή που προέκυψε κατά τη μελέτη της εφαρμογής είναι η περίπτωση που ο φοιτητής θέλει να αλλάξει ήδη υπάρχων θέμα πτυχιακής εργασίας.

1. Ο φοιτητής απαντάει ότι θέλει να αλλάξει την πτυχιακή που έχει ήδη με μια νέα
2. Το chatbot του απαντάει ότι θα πρέπει να επαναλάβει τη διαδικασία επιλογής νέας πτυχιακής στο επόμενο χειμερινό εξάμηνο.
3. Η ροή τελειώνει με επιτυχία

3.4.3 Εναλλακτική ροή 3β

Η δεύτερη εναλλακτική ροή που προκύπτει αφορά την περίπτωση που ο φοιτητής έχει επιλέξει ήδη κάποιο θέμα πτυχιακής εργασίας και χρειάζεται κάποια βοήθεια με τη συμπλήρωση της φόρμας δήλωσης θέματος πτυχιακής εργασίας στη γραμματεία του τμήματος και δε γνωρίζει που μπορεί να βρει το έντυπο της δήλωσης αυτής.

1. Ο φοιτητής απαντάει ότι θέλει βοήθεια με την αίτηση πτυχιακής εργασίας
2. Το chatbot τον ρωτάει με τι χρειάζεται βοήθεια και του δίνει τις επιλογές: Δε βρίσκω το έντυπο της δήλωσης και Βοήθεια με τη συμπλήρωση της αίτησης
3. Ο φοιτητής επιλέγει την επιλογή “Δε βρίσκω το έντυπο της δήλωσης “
4. Το chatbot του στέλνει το έντυπο της αίτησης σε μορφή PDF μέσω συνδέσμου
5. Η ροή τελειώνει με επιτυχία

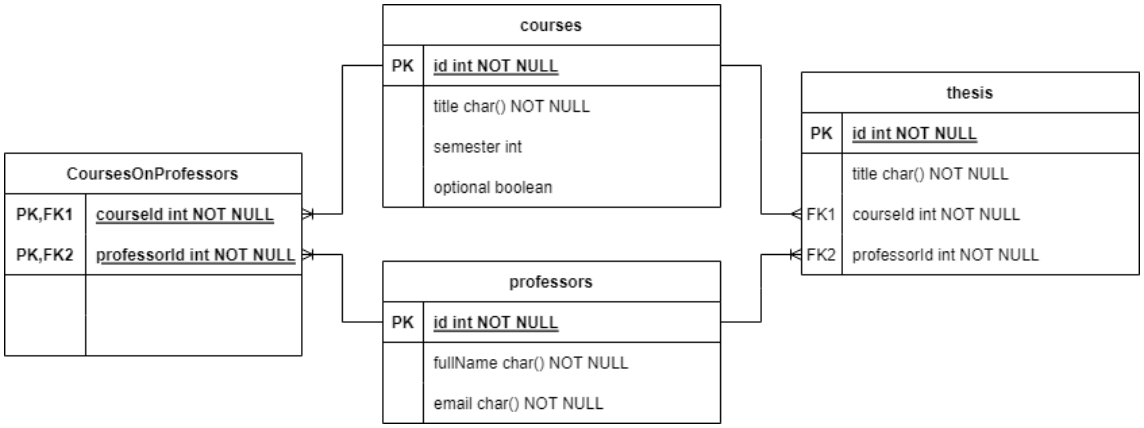
3.4.4 Εναλλακτική ροή 3β2α

Από την εναλλακτική ροή 3β προκύπτει το σενάριο που ο φοιτητής έχει επιλέξει θέμα πτυχιακής εργασίας, χρειάζεται βοήθεια με την αίτηση δήλωσης της πτυχιακής του εργασίας, τόσο στη συμπλήρωση του όσο και στο μέρος εύρεσης του εγγράφου.

1. Ο φοιτητής επιλέγει την επιλογή “Βοήθεια με τη συμπλήρωση της αίτησης “
2. Το chatbot του στέλνει τα απαραίτητα στοιχεία που χρειάζεται, τίτλος πτυχιακής σε ελληνικά και αγγλικά, ονοματεπώνυμο επιβλέποντα και γλώσσα συγγραφής πτυχιακής εργασίας
3. Συνεχίζει από το σημείο 4 της ροής 3β.

3.5 Διάγραμμα οντοτήτων (για τη βάση)

Στο παρακάτω διάγραμμα φαίνεται η μορφή της SQLite βάσης δεδομένων που χρησιμοποιεί η εφαρμογή για την αναζήτηση των διάφορων πληροφοριών που χρειάζονται για να απαντηθούν τα ερωτήματα του χρήστη.



Εικόνα 5 Διάγραμμα οντοτήτων βάσης δεδομένων

Όπως φαίνεται από το διάγραμμα οντοτήτων, η βάση δεδομένων περιέχει τρεις βασικές οντότητες και μια βοηθητική.

Συγκεκριμένα η οντότητα courses αναφέρεται στα διαθέσιμα μαθήματα από τα οποία μπορεί να επιλέξει ο φοιτητής, χρήστης της εφαρμογής, αυτή η οντότητα περιέχει πληροφορίες όπως το τίτλο του μαθήματος, το εξάμηνο στο οποίο ανήκει και αν αυτό το μάθημα είναι υποχρεωτικό.

Η οντότητα professors περιέχει στοιχεία για τους καθηγητές του τμήματος, τα οποία θα σταλούν στο τέλος στο φοιτητή όταν διαλέξει κάποιο από τα θέματα πτυχιακής εργασίας, συγκεκριμένα περιέχει ένα πρωτεύων κλειδί, το ονοματεπώνυμο τους και το email τους.

Για να αντιστοιχιστούν οι καθηγητές με τα μαθήματα χρειάστηκε μια νέα βοηθητική οντότητα CoursesonProfessors, η οποία περιέχει το πρωτεύων κλειδί του μαθήματος από την οντότητα courses και το πρωτεύον κλειδί από την οντότητα professors στην οποία ανήκει. Η δημιουργία αυτής της οντότητας έγινε για τη δημιουργία της σχέσεις πολλά προς πολλά ανάμεσα σε καθηγητές και μαθήματα.

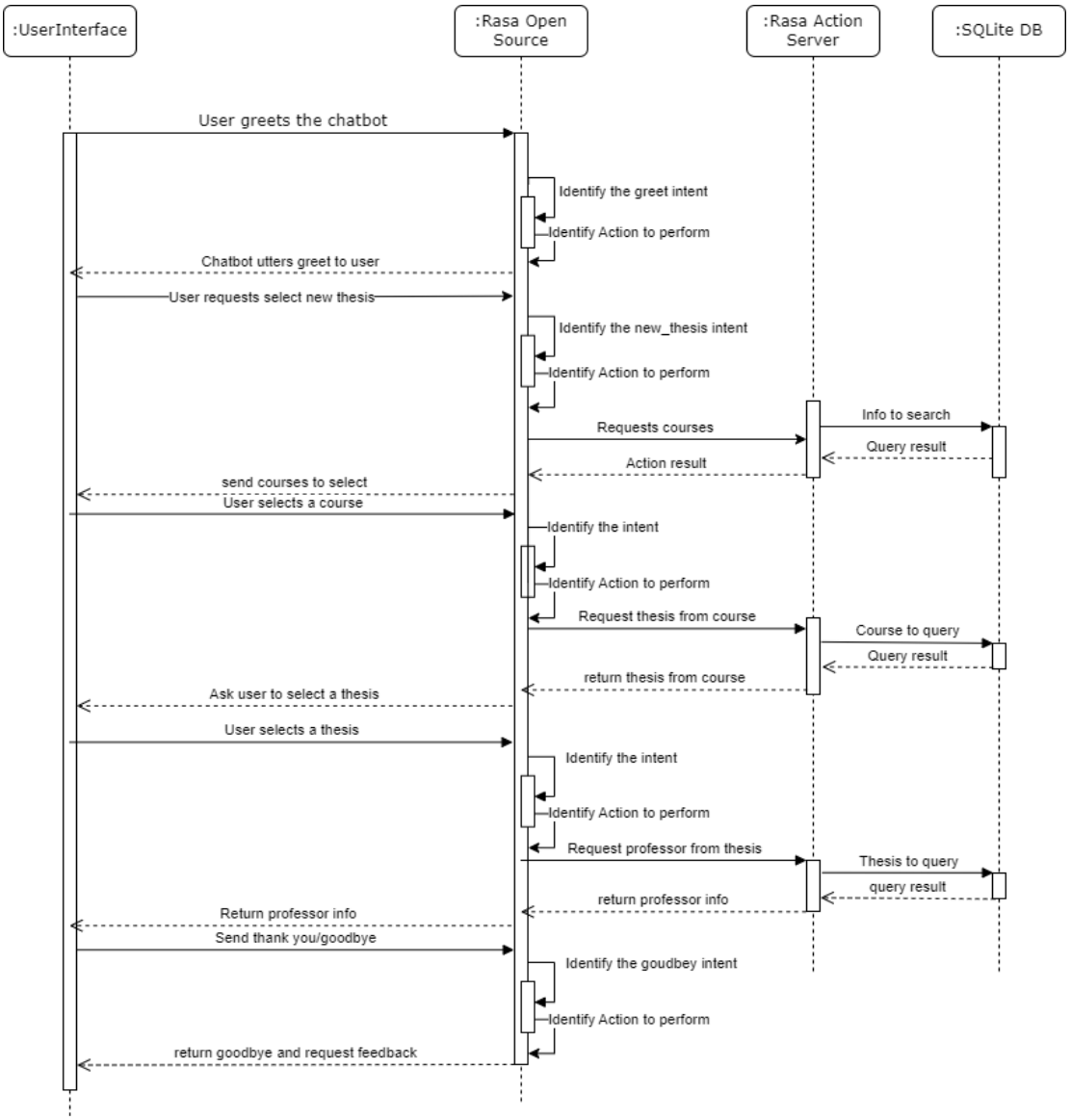
Τέλος, δημιουργήθηκε και η οντότητα thesis η οποία περιέχει όλα τα διαθέσιμα θέματα, από τα οποία θα κληθεί να διαλέξει ο φοιτητής, πιο συγκεκριμένα αυτή αποτελείται από ένα πρωτεύων κλειδί, το τίτλο του θέματος, το πρωτεύων κλειδί του μαθήματος στο οποίο ανήκει και το πρωτεύων κλειδί του καθηγητή ο οποίος έχει καταχωρήσει το θέμα.

3.6 Διάγραμμα ακολουθίας

Στο διάγραμμα ακολουθίας φαίνεται η σειρά με την οποία γίνεται η αλληλεπίδραση ανάμεσα στο ψηφιακό βοηθό και στο χρήστη, φοιτητή, για το κύριο σενάριο της εφαρμογής, δηλαδή την επιλογή του νέου θέματος πτυχιακής εργασίας.

Ο χρήστης αρχικά επικοινωνεί με το ψηφιακό βοηθό, το μήνυμα το λαμβάνει το υποσύστημα Rasa Open Source το οποίο κάνει εντοπισμό της πρόθεσης του χρήστη και εντοπίζει την κατάλληλη ενέργεια στην οποία πρέπει να προβεί, βλέπου ότι αρχικά ο χρήστης πρέπει να ξεκινήσει τη συνομιλία του με το ψηφιακό βοηθό, αυτός στη συνέχεια να εντοπίσει την πρόθεση του χρήστη ώστε να καταλήξει στην κατάλληλη ενέργεια και να μπορέσει να στείλει πίσω στο χρήστη χαιρετισμό και τις επιλογές για τις οποίες μπορεί να το χρησιμοποιήσει. Ο χρήστης με τη σειρά του στέλνει στο ψηφιακό βοηθό μήνυμα επιθυμίας επιλογής θέματος πτυχιακής εργασίας, το Rasa Open Source εντοπίζει την επιθυμία αυτή του χρήστη και στη συνέχεια επικοινωνεί με το Rasa Action Server ζητώντας τα διαθέσιμα μαθήματα, το Rasa Action Server στη συνέχεια ψάχνει στη SQLite βάση δεδομένων τα διαθέσιμα μαθήματα τα οποία γυρίζει ως απάντηση στο Rasa Open Source και αυτό με τη σειρά του στο χρήστη, στη συνέχεια ο χρήστης ενημερώνει το Rasa Open Source με το μάθημα ενδιαφέροντος, το Rasa Open Source πρέπει πάλι να εντοπίσει την επιθυμία μαθήματος και να το στείλει στο Rasa Action Server αυτό με τη σειρά του πρέπει να ψάξει στη βάση δεδομένων τα διαθέσιμα θέματα πτυχιακών εργασιών που υπάρχουν για το μάθημα αυτό και αφού τα βρει θα τα στείλει πίσω στο Rasa Open Source και αυτό με τη σειρά του θα τα στείλει στο χρήστη, έπειτα ο χρήστης θα επιλέξει ένα από αυτά τα θέματα και θα το

στείλει πίσω στο Rasa Open Source και αυτό πάλι θα κάνει εντοπισμό της επιθυμίας αυτής του χρήστη και θα το στείλει στο Rasa Action Server για να λάβει πληροφορίες για τον επιβλέποντα καθηγητή του θέματος που διάλεξε ο χρήστης, το Rasa Action Server θα αναζητήσει στη βάση δεδομένων πληροφορίες για τον καθηγητή που αντιστοιχεί το θέμα και θα τις στείλει πίσω στο Rasa Open Source για να σταλούν από εκεί πίσω στο χρήστη, ο χρήστης τέλος θα ενημερώσει τον ψηφιακό βοηθό για το τέλος της διαδικασίας και το Rasa Open Source λαμβάνοντας αυτό το μήνυμα θα προχωρήσει πάλι σε αναγνώριση επιθυμίας του χρήστη για το πέρας της διαδικασίας και θα του στείλει μήνυμα αποχαιρετισμού καθώς και θα του ζητήσει τη συμπλήρωση μιας φόρμας αξιολόγησης, εκεί η διαδικασία τελειώνει.



Εικόνα 6 Διάγραμμα ακολουθίας

Σχετικά με τα υπόλοιπα σενάρια μπορούν να διατυπωθούν με την επανάληψη του πρώτου κομματιού του κύριου σεναρίου αφού αποτελούν απλή επικοινωνία ανάμεσα στον χρήστη και το Rasa Open Source χωρίς τη χρήση κάποιου άλλου υποσυστήματος.

Τέλος, θα πρέπει να συμπληρώσουμε ότι σε κάθε επικοινωνία χρήστη και Rasa Open Source αλλά και σε κάθε κίνηση που πραγματοποιεί το Rasa Open Source υπάρχει και επικοινωνία με τη βάση δεδομένων Tracker Store που κρατάει ιστορικό για όλες τις κινήσεις του Rasa Open Source.

4 Υλοποίηση και Έλεγχος

Σε αυτή την ενότητα θα γίνει ανάλυση του τρόπου υλοποίησης της εφαρμογής. Θα γίνει ανάλυση του μοντέλου που ακολουθήθηκε, για το τρόπο υλοποίησης και ποιος ήταν ο λόγος πίσω από αυτή την επιλογή, θα παρουσιαστούν τα βασικά μέρη που αποτελούν την εφαρμογή καθώς και τα βασικά υποσυστήματα που είναι αναγκαία για την ομαλή λειτουργία. Τέλος θα γίνει αναφορά στον έλεγχο που πραγματοποιήθηκε για την αποτελεσματικότητα της εφαρμογής καθώς και τις βελτιώσεις που έγιναν βάση των αποτελεσμάτων του ελέγχου αυτού.

4.1 Conversation-Driven Development

Για την υλοποίηση και την ανάπτυξη του ψηφιακού βοηθού, για τις ανάγκες της παρούσας μεταπτυχιακής διατριβής χρησιμοποιήθηκε η μεθοδολογία “Conversation-Driven Development” ή εν συντομία CDD.

Όπως μας γίνεται γνωστό από το Rasa Technologies αυτός ο τρόπος ανάπτυξης εφαρμογών σχετίζεται με το να παίρνουμε άμεση ανατροφοδότηση από τους χρήστες του ψηφιακού βοηθού και να χρησιμοποιούμε άμεσα το περιεχόμενο της ανατροφοδότησης αυτής για την καλυτέρευση της εφαρμογής. Στο ίδιο σημείο μας αναφέρεται πως για να έχουμε καλύτερα αποτελέσματα με τη χρήση αυτής της μεθοδολογίας είναι να μπορέσουμε να μοιραστούμε όσο πιο άμεσα γίνεται το ψηφιακό μας βοηθό με το κοινό που απευθύνεται, για να έχουμε ας αποτέλεσμα και πιο γρήγορη ανατροφοδότηση. Επίσης, είναι σημαντικό να επισημάνουμε ότι η μεθοδολογία αυτή καταλήγει να είναι κυκλική, τα μέρη που θα μπορούσε κανείς να πει ότι αποτελείτε είναι αρχικά η διάθεση της εφαρμογής, μετά η συνεχή επικοινωνία των χρηστών που έχουμε ορίσει με αυτή, η χρήση των δεδομένων που λαμβάνουμε από την επικοινωνία των χρηστών ως δεδομένα εκπαίδευσης του ψηφιακού βοηθού, η επαλήθευση ότι ο ψηφιακός βοηθός ανταποκρίνεται όπως πρέπει και εκτελεί τις ενέργειες που θέλουμε μετά την εισαγωγή των νέων δεδομένων, η συνεχή παρακολούθηση των σφαλμάτων που μπορεί να δημιουργηθούν και τέλος η επίλυση αυτών και όπως είπαμε παραπάνω η μετά γίνεται επανάληψη της διαδικασίας(*Conversation-Driven Development*, 2022).

Θα μπορούσε κανείς να παρουσιάσει το CDD και ως μια μορφή CI/CD (continuous integration/ continuous deployment), όπως το CDD έτσι και το CI/CD βασίζεται στη γρήγορη διάθεση των εφαρμογών στους χρήστες και στη συνεχή βελτίωση των εφαρμογών μέσω μιας διαδικασίας. Αλλά το πιο σωστό θα ήταν να πούμε ότι το ένα συμπληρώνει το άλλο, μιας και το CI/CD περιγράφει τη διαδικασία της αυτοματοποίησης της διάθεσης των εφαρμογών στους τελικούς χρήστες δίνοντας λύση στο πρόβλημα της ενσωμάτωσης και ενημέρωσης λογισμικού που μπορεί να προκαλέσει πολλά προβλήματα στις ομάδες συντήρησης και ανάπτυξής του(*What Is CI/CD?*, 2022), αυτό μας υποδηλώνει ότι η χρήση της μεθοδολογίας CI/CD θα μπορούσε να αποτελέσει πολύ καλή ιδέα για την ανάπτυξη μιας εφαρμογής ψηφιακού βοηθού με τη χρήση της πλατφόρμας Rasa Open Source.

Κατά την υλοποίηση της εφαρμογής του ψηφιακού βοηθού χρησιμοποιήθηκε η τεχνική CDD όπως προτείνεται από το εγχειρίδιο χρήσης του Rasa, αυτό έγινε διαθέτοντας τον υπερσύνδεσμο για την εφαρμογή, που οδηγούσε σε κάποιο κανάλι επικοινωνίας, σε συμφοιτητές του προγράμματος μεταπτυχιακών σπουδών του Πανεπιστημίου Πειραιώς, αλλά και φοιτητές άλλον πανεπιστημιακών ιδρυμάτων και φίλους. Στη συνέχεια, μαζεύοντας είτε προφορική ανατροφοδότηση είτε μέσω της χρήσης φόρμας Google Forms, σε κάθε σενάριο που ολοκληρωνόταν στην εφαρμογή, μπορούσαμε να δούμε την αποτελεσματικότητά του με το χρήστη και τον εμπλουτισμό των σεναρίων βάση της ανατροφοδότησης που υπήρξε. Αυτό είχε ως αποτέλεσμα να φτάσουμε σε ένα σημείο που το chatbot, ψηφιακός βοηθός, μπορούσε να ανταποκριθεί σωστά στα περισσότερα ζητούμενα που θα μπορούσε να είχε ο εκάστοτε χρήστης και θα αφορούσαν την επιλογή νέου θέματος πτυχιακής εργασίας, την αλλαγή του ή ακόμα και αν ζητούσε βοήθεια σχετικά με το έντυπο που θα χρειαζόταν να συμπληρώσει και να παραδώσει στη γραμματεία του τμήματος.

4.2 Βάση δεδομένων

Πρώτα από όλα κατά την υλοποίηση της εφαρμογής κρίθηκε αναγκαία η δημιουργία μιας βάσης δεδομένων, που θα μπορούσε η εφαρμογή του ψηφιακού βοηθού να ανατρέξει για πληροφορίες σχετικά με τα διαθέσιμα μαθήματα, θέματα πτυχιακών εργασιών αλλά και τους επιβλέποντες καθηγητές αυτών. Για τη δημιουργία της βάσης δεδομένων χρησιμοποιήθηκε η τεχνολογία SQLite, επειδή ο όγκος των δεδομένων ήταν σχετικά μικρός, της ευκολίας εγκατάστασης και χρήσης κρίναμε ότι θα ήταν η κατάλληλη βάση για τη παρούσα διατριβή.

Για τη δημιουργία του σχήματος και την εισαγωγή των δεδομένων στη βάση έγινε χρήση του ORM Prisma, με τη χρήση της αυτού μπορέσαμε να κατασκευάσουμε γρήγορα το σχήμα της βάσης όπως αυτό φαίνεται στην Εικόνα 5 της προηγούμενης ενότητας.

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "sqlite"
  url      = env("DATABASE_URL")
}

model Course {
  id Int @id @default(autoincrement())
  title String @unique
  semester Int
  optional Boolean
  professors CoursesOnProfessors[]
  thesises Thesis[]

  @@map("courses")
}

model Professor {
  id Int @id @default(autoincrement())
  fullName String @unique
  email String @unique

  courses CoursesOnProfessors[]
  thesis Thesis[]

  @@map("professors")
}

model Thesis {
  id Int @id @default(autoincrement())
  title String @unique
  courseId Int
  course Course @relation(fields: [courseId], references: [id])
  professorId Int
  professor Professor @relation(fields: [professorId], references: [id])
  students Student[]

  @@map("thesis")
}

model CoursesOnProfessors {
  course Course @relation(fields: [courseId],references: [id])
  courseId Int
  professor Professor @relation(fields: [professorId], references: [id])
  professorId Int

  @@id([professorId, courseId])
}
```

Εικόνα 7 Δημιουργία σχήματος βάσης με χρήση του λογισμικού Prisma

Η επιλογή του συγκεκριμένου εργαλείου έγινε για προσωπική ευκολία μιας και αποτελεί γνωστικό αντικείμενο από άλλες εφαρμογές που έχουν υλοποιηθεί με τη χρήση του. Το Prisma

μας έδωσε τη δυνατότητα να κατασκευάσουμε γρήγορα το σχήμα της βάσης αλλά και με τη χρήση απλού κώδικα JavaScript να εισάγουμε στη βάση τις απαραίτητες πληροφορίες που χρειαζόμασταν όπως αυτές περιεγράφηκαν προηγουμένως. Στη συνέχεια περάσαμε τη βάση ως αρχείο στον υπό-φάκελο “actions” στην εφαρμογή του Rasa Open Source.

4.3 Σύνδεση βάσης δεδομένων και εφαρμογής

Για τη σύνδεση της SQLite βάσης δεδομένων με την υπόλοιπη εφαρμογή χρειάστηκε η δημιουργία ενός συνόλου βοηθητικών μεθόδων που θα είναι υπεύθυνες για τη σύνδεση και την αποσύνδεση από τη βάση δεδομένων που θα χρησιμοποιήσει η εφαρμογή με τη χρήση της “sqlite3” βιβλιοθήκης που είναι διαθέσιμη στη Python.

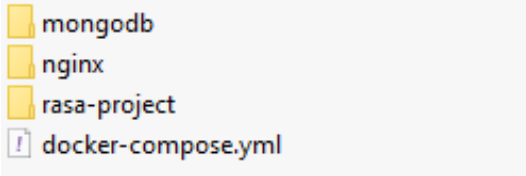
actions > db_helpers.py

```python
import sqlite3

def connect_to_db():
    try:
        sqliteConnection = sqlite3.connect('./actions/rasa.sqlite')
        return sqliteConnection
    except sqlite3.Error as error:
        print("Error while connecting to sqlite", error)

def close_connection(sqliteConnection):
    if sqliteConnection:
        sqliteConnection.close()
```

Εικόνα 8 Βοηθητικές μέθοδοι βάσης δεδομένων

4.4 Δομή εφαρμογής

Στη παρακάτω εικόνα φαίνεται η δομή του συνόλου της εφαρμογής του ψηφιακού βοηθού, μαζί με τα βοηθητικά υποσυστήματα.

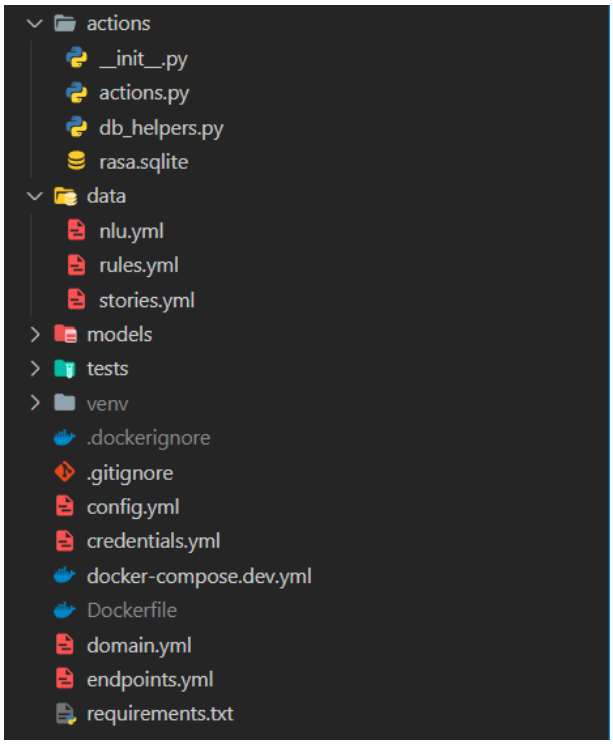


Εικόνα 9 Δομή εφαρμογής και υποσυστημάτων

Στο φάκελο “mongodb” υπάρχει η βάση δεδομένων του Tracker Store που χρησιμοποιείτε από το Rasa Open Source για να αποθηκεύει τις συνομιλίες που κάνει ο ψηφιακός βοηθός, στο φάκελο “nginx” υπάρχουν οι ρυθμίσεις που χρειάζεται να γίνουν στον NGINX proxy server για να μπορέσουμε να διαθέσουμε την εφαρμογή μας στο κοινό, το αρχείο “docker-compose.yml" περιγράφει της κινήσεις που πρέπει να γίνουν από το Docker Engine για να «σηκωθεί» η εφαρμογή μας και να γίνει διαθέσιμη στο κοινό και τέλος στο φάκελο “rasa-project” βρίσκεται το κύριο μέρος της εφαρμογή που αποτελείτε από το Rasa Open Source και το Rasa Action Server.

Εδώ θα ήταν καλό να αναφέρουμε ότι η χρήση του λογισμικού NGINX Proxy Manager όπως και της βάσης δεδομένων MongoDB δεν είναι αναγκαία για τη δημιουργία και τη λειτουργία της εφαρμογής, συγκεκριμένα το λογισμικό NGINX Proxy Manager χρησιμοποιήθηκε για τη διευκόλυνση έκδοσης SSL πιστοποιητικού και να μπορέσει η εφαρμογή μας να είναι διαθέσιμη μέσω του Telegram, και η βάση MongoDB χρησιμοποιήθηκε ως Tracker Store για την ευκολότερη πρόσβαση στα περιεχόμενα της, αν θέλαμε να χρησιμοποιήσουμε την εφαρμογή του ψηφιακού βοηθού θα μπορούσαμε χωρίς τα παραπάνω από άλλα κανάλια επικοινωνίας όπως το socker io

και το Rest API που παρέχονται από την εφαρμογή καθώς και για τη βάση το Rasa Open Source παρέχει την “InMemoryTrackerStore” που εκτελεί την ίδια διαδικασία(*Tracker Stores*, 2022).



Εικόνα 10 Δομή βασικής εφαρμογής

Στη παραπάνω εικόνα φαίνονται τα βασικά στοιχεία της εφαρμογής, πιο συγκεκριμένα στο κεντρικό φάκελο βλέπουμε το αρχείο “domain.yml” το οποίο είναι και ένα από τα πιο σημαντικά της εφαρμογής καθώς περιλαμβάνει όλα τα intent, entities, slots, actions, forms και responses που χρησιμοποιεί η εφαρμογή του ψηφιακού βοηθού (*Domain*, 2022). Στη συνέχεια στο αρχείο “requirements.txt” υπάρχουν όλες η απαραίτητες βιβλιοθήκες της Python που πρέπει να εγκατασταθούν στο εικονικό περιβάλλον για να μπορέσει να λειτουργήσει η εφαρμογή του Rasa Open Source. Το αρχείο “endpoints.yml” χρησιμοποιείτε για να δηλώσουμε τα διάφορα endpoints που μπορεί να χρησιμοποιήσει το Rasa Open Source, δηλαδή σε αυτό το αρχείο θα δηλωθεί η διεύθυνση του Rasa Action Server αφού λειτουργεί σαν ξεχωριστή υπηρεσία, η διεύθυνση επικοινωνίας με το Tracker Store που χρησιμοποιούμε, αν χρησιμοποιούμε το “InMemoryTrackerStore” τότε δε χρειάζεται να δηλώσουμε κάτι στο αρχείο αυτό. Το αρχείο “credentials.yml” περιέχει τις ρυθμίσεις που χρειάζονται να γίνουν για τη σύνδεση με εξωτερικά κανάλια επικοινωνίας, εκεί δηλώνουμε δηλαδή τα διάφορα Token που μπορεί να λάβουμε από κανάλια όπως το Facebook Messenger, το Telegram, το Slack αλλά και τα βασικά κανάλια του Rasa Open Source όπως το socket io και το Rest API. Το αρχείο “config.yml” είναι το σημείο που ορίζουμε το τρόπο και τις πολιτικές που θα ακολουθήσει το Rasa Open Source για την εκπαίδευση του μοντέλου μας(*Model Configuration*, 2022). Όσο αναφορά τα αρχεία “Dockerfile” και το “.dockerignore” χρησιμοποιήθηκαν για να αυτοματοποιήσουμε την εκπαίδευση ενός νέου μοντέλου και την δοκιμή του και δεν αποτελούν βασικά αρχεία της εφαρμογής. Το αρχείο “docker-compose.dev.yml” μπορεί να χρησιμοποιηθεί αν θέλουμε να λειτουργήσουμε μόνο τη βασική εφαρμογή χωρίς τον NGINX Proxy Manager και τη Mongo DB ως Tracker Store με την εντολή “docker compose -f docker-compose.dev.yml up -d”.

Στον επιμέρους φάκελο “venv” υπάρχει το εικονικό περιβάλλον που θα χρειαστεί να δημιουργήσουμε για την εγκατάσταση των απαραίτητων βιβλιοθηκών της Python.

Ο φάκελος “tests” περιέχει τα test που έχουμε πραγματοποιήσει ή πραγματοποιούμε πριν διαθέσουμε την εφαρμογή στο ευρύ κοινό.

Ο φάκελος “models” αποθηκεύονται τα μοντέλα που δημιουργούνται κάθε φορά που εκπαιδεύουμε το AI του ψηφιακού βοηθού, και χρησιμοποιείται από το NLU του ψηφιακού βοηθού

κάθε φορά για τη κατηγοριοποίηση των μηνυμάτων που λαμβάνει αλλά και για την εξαγωγή των απαραίτητων πληροφοριών από τα μηνύματα αυτά.

Στο φάκελο “data” υπάρχουν τρία βασικά αρχεία για την εφαρμογή του ψηφιακού βοηθού, το “nlu.yml”, το “rules.yml” και το “stories.yml”. Στο αρχείο “nlu.yml” ορίζουμε τα intents και παραδείγματα χρήσης του κάθε intent και συμβάλει σε μεγάλο βαθμό στην εκπαίδευση του μοντέλου που θα χρησιμοποιήσει η εφαρμογή. Το αρχείο “stories.yml” περιέχει τα stories που μπορεί να διαχειριστεί ο ψηφιακός βοηθός, δηλαδή τη ροή που θα μπορούσε να έχει η επικοινωνία του ψηφιακού βοηθού με ένα χρήστη του, κάθε πιθανότητα διαλόγου αποτελεί ένα story. Το αρχείο “rules.yml” είναι παρόμοιο με το αρχείο “stories.yml”, με τη διαφορά πως ένας rule περιγράφει ένα μικρό μέρος διαλόγου και ακολουθάει πάντα ένα συγκεκριμένο μονοπάτι.

Ο τελευταίος φάκελος, ο φάκελος “actions” θα αποτελέσει το Rasa Action Server, αυτός ο φάκελος περιέχει τα custom Actions, δηλαδή είναι το μέρος που δημιουργούμε τις δικές μας βοηθητικές κλάσεις που βρίσκονται στο αρχείο “actions.py”, και τις χρησιμοποιούμε για την εύρεση πληροφοριών στη βάση δεδομένων, για την επαλήθευση των τιμών που δίνει ο χρήστης καθώς και για την αποστολή των απαντήσεων αυτών των αποτελεσμάτων στο χρήστη. Επίσης, στο φάκελο αυτό περιέχεται και η βάση δεδομένων “rasa.sqlite” που περιέχει τις πληροφορίες σχετικά με τα μαθήματα τους καθηγητές και τα διαθέσιμα θέματα πτυχιακών εργασιών που είναι διαθέσιμα προς τους φοιτητές, καθώς και το αρχείο “db_helpers.py”, το οποίο περιέχει βοηθητικές μεθόδους για τη σύνδεση και την αποσύνδεση με τη βάση δεδομένων.

4.5 Υλοποίηση

Στην ενότητα αυτή θα προσδιορίσουμε τις διαδικασίες που εφαρμόστηκαν για την υλοποίηση της εφαρμογής του chatbot, ψηφιακού βοηθού. Πιο συγκεκριμένα θα γίνει αναφορά στη πολιτική που ακολουθήθηκε για την εκπαίδευση των μοντέλων, των intent που χρειάστηκε να δημιουργηθούν, των δεδομένων εκπαίδευσης, αλλά και των μεθόδων που χρειάστηκε να δημιουργηθούν με τη χρήση της γλώσσας Python.

4.5.1 Chatbot Configurations

Για να μπορέσουμε να κατανοήσουμε τις ρυθμίσεις που χρειάστηκαν να εφαρμοστούν θα πρέπει αρχικά να κατανοήσουμε το NLU pipeline και τα μέρη του. Το NLU pipeline υπάρχει στο αρχείο “config.yml” της εφαρμογής μας και περιγράφει τα βήματα που θα ακολουθήσει η εφαρμογή του Rasa Open Source για να μπορέσει να εντοπίσει τα διάφορα intents και entities που μπορεί να του στείλει ο χρήστης. Κάποια από τα κύρια συστατικά που αποτελείτε το NLU pipeline είναι: Tokenizers, Featurizers, Intent Classifiers, Entity Extractors και ResponseSelectors.

Αρχικά οι Featurizers είναι υπεύθυνοι για να χωρίσουν το κείμενο του χρήστη σε μικρότερα μέρη, αυτό συνίσταται να γίνει πρώτού περάσει το κείμενο περάσει στο επίπεδο του NLP, καθώς το νόημα του κειμένου μπορεί εύκολα να γίνει κατανοητό από τις λέξεις που υπάρχουν σε αυτό(Singh, 2019). Για την επίτευξη του διατίθενται πολλές διαδικασίες αλλά και βιβλιοθήκες, όπως ο WhiteSpaceTokenizer, η βιβλιοθήκες Spacy και άλλοι, στη περίπτωση μας χρησιμοποιήσαμε τον WhiteSpaceTokenizer ο οποίος χρησιμοποιεί τα κενά που υπάρχουν σε μια λέξη για να τη χωρίσει σε μικρότερα κομμάτια.

Στη συνέχεια θα χρειαστεί να δηλώσουμε έναν ή περισσότερους Featurizers. Οι Featurizers είναι υπεύθυνοι για τη μετατροπή χαρακτήρων κειμένου σε αριθμητικά διανύσματα ώστε να μπορεί να γίνει πιο αποτελεσματική η χρήση μοντέλων μηχανικής μάθησης(Yassar, 2019). Στην εφαρμογή μας χρησιμοποιήσαμε τον LexicalSyntacticFeaturizer που δημιουργεί λεξιλογικά και συντακτικά χαρακτηριστικά για ένα μήνυμα για να υποστηρίξει την εξαγωγή οντοτήτων από αυτό, τον RegexFeaturizer ο οποίος δημιουργεί μια διανυσματική αναπαράσταση του μηνύματος του χρήστη με τη χρήση κανονικών εκφράσεων και τέλος χρησιμοποιήθηκε ο CountVectorsFeaturizer που δημιουργεί μια συλλογή τύπου bag-of-words με τις προθέσεις(intents), τα μηνύματα του χρήστη και την απάντηση.

Το επόμενο κομμάτι στο NLU pipeline θα είναι οι Intent Classifiers, τα μοντέλα αυτά αναλύουν τις προθέσεις πίσω από το κείμενο και τις κατηγοριοποιεί ως αιτήματα η εγκρίσεις(Karatas, 2022). Όπως προτείνεται από το Rasa Open Source εδώ χρησιμοποιούμε τον

DIETClassifier. Μετά από διάθεση της εφαρμογής στους χρήστες όμως φάνηκε ότι το chatbot δε μπορούσε να καταλάβει κάποιες φορές το χρήστη με αποτέλεσμα να μην του απαντάει, για να λύσουμε το πρόβλημα αυτό χρειάστηκε να προσθέσουμε τον FallbackClassifier ο οποίος κατηγοριοποιεί ένα με το intent “nlu_fallback” όταν τα υπόλοιπες κατηγοριοποιήσεις intent δε ξεπεράσουν ένα επιθυμητό επίπεδο.

Επίσης, θα μπορούσαμε να προσθέσουμε και κάποιο Entity Extraction μοντέλο αν χρησιμοποιούσαμε αλλά για τις ανάγκες της εφαρμογής δε κρίθηκε αναγκαίο, αφού μπορεί να ολοκληρωθεί η υλοποίηση με τη χρήση του DIETClassifier.

Τέλος, χρειάστηκε να ορίσουμε ResponseSelectors, οι ResponseSelectors παράγουν ένα λεξικό με κλειδί το intent ανάκτησης του επιλογέα απάντησης και ως τιμή περιέχει προβλεπόμενες απαντήσεις, το όριο εμπιστοσύνης που έχει δώσει το μοντέλο και το κλειδί απάντησης του intent απόκρισης(Warmerdam, 2021).

Επόμενο στάδιο στο αρχείο “config.yml” είναι οι πολιτικές που χρησιμοποιεί η εφαρμογή για να επιλέξει ποια λειτουργεία θα εκτελέσει σε κάθε βήμα της συζήτησης με το χρήστη(*Policies*, 2022). Εδώ χρησιμοποιήθηκαν η TEDPolicie(Transformer Embedding Dialogue) η οποία είναι μια αρχιτεκτονική πολλών επιπέδων που χρησιμοποιείται για τη πρόβλεψη της επόμενης ενέργειας και την αναγνώριση οντοτήτων. Η UnexpecTEDIntentPolicy που είναι παρόμοια με τη TEDPolicie αλλά βοηθάει στην αναγνώριση των επιθυμιών που είναι απρόσμενες να ζητηθούν από τη μεριά του χρήστη και χρησιμοποιείται συνήθως ως βοηθητική της TEDPolicie. Επίσης σε χρήση είναι και το MemoizationPolicy, η πολιτική αυτή μπορεί και θυμάται τα stories που έχουν γραφεί από τα δεδομένα εκπαίδευσης που έχουμε δώσει στην εφαρμογή, και ελέγχοντας αν η επικοινωνία που κάνει ο ψηφιακός βοηθός με το χρήστη μπορεί να προβλέψει την επόμενη ενέργεια από τις αντίστοιχες ιστορίες(stories) των δεδομένων εκπαίδευσης αποδίδοντας τους βαθμό σιγουριάς. Και τέλος έγινε χρήση και της πολιτικής RulePolicy η οποία μπορεί να διαχειριστεί κομμάτια επικοινωνίας, εφαρμογής και χρήστη, που ακολουθούν μια συγκεκριμένη συμπεριφορά, οι προβλέψεις αυτού του τύπου πολιτικής προέρχονται από τα rules που έχουμε εισάγει στην εφαρμογή.

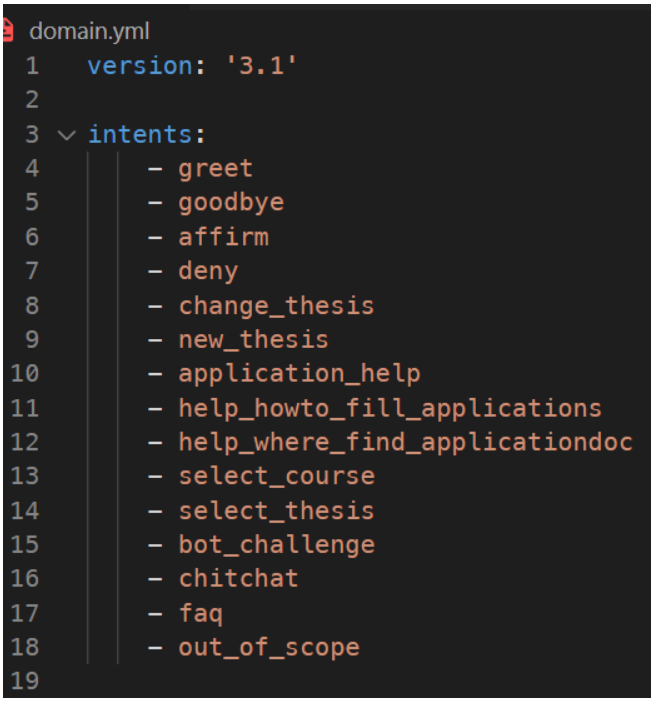
config.yml

```yaml
# The config recipe.
# https://rasa.com/docs/rasa/model-configuration/
recipe: default.v1

# Configuration for Rasa NLU.
# https://rasa.com/docs/rasa/nlu/components/
language: el

pipeline:
    # # No configuration for the NLU pipeline was provided. The following default pipeline was used to train your model.
    # # If you'd like to customize it, uncomment and adjust the pipeline.
    # # See https://rasa.com/docs/rasa/tuning-your-model for more information.
    - name: WhitespaceTokenizer
    - name: RegexFeaturizer
    - name: LexicalSyntacticFeaturizer
    - name: CountVectorsFeaturizer
    - name: CountVectorsFeaturizer
      analyzer: char_wb
      min_ngram: 1
      max_ngram: 4
    - name: DIETClassifier
      epochs: 100
      constrain_similarities: true
    - name: EntitySynonymMapper
    - name: ResponseSelector
      epochs: 100
      constrain_similarities: true
    - name: ResponseSelector
      epochs: 100
      retrieval_intent: faq
    - name: ResponseSelector
      epochs: 100
      retrieval_intent: chitchat
    - name: FallbackClassifier
      threshold: 0.7
      ambiguity_threshold: 0.7

# Configuration for Rasa Core.
# https://rasa.com/docs/rasa/core/policies/
policies:
    # # No configuration for policies was provided. The following default policies were used to train your model.
    # # If you'd like to customize them, uncomment and adjust the policies.
    # # See https://rasa.com/docs/rasa/policies for more information.
    - name: MemoizationPolicy
    - name: RulePolicy
      core_fallback_threshold: 0.3
    - name: UnexpecTEDIntentPolicy
      max_history: 5
      epochs: 100
    - name: TEDPolicy
      max_history: 5
      epochs: 100
      constrain_similarities: true
```

Εικόνα 11 Αρχείο config.yml

4.5.2 Intents

Τα intents θα μπορούσαν να προσδιοριστούν ως το πράγμα που προσπαθεί να επιτύχει ένας χρήστης με τη χρήση του ψηφιακού βοηθού. Το NLU του Rasa Open Source θα ομαδοποιήσει τα μηνύματα του χρήστη σε ένα ή περισσότερα intents, για να μπορέσει αυτό να γίνει με αποτελεσματικότητα θα πρέπει να αποφευχθούν ορισμένα συνηθισμένα λάθη, αρχικά θα πρέπει να δοθούν αρκετά δεδομένα εκπαίδευσης ώστε να μπορεί ο ψηφιακός βοηθός να κατηγοριοποιήσει τα μηνύματα του χρήστη με κάποια επιθυμία που έχει δηλωθεί από εμάς. Θα πρέπει να προσέξουμε να μη χρησιμοποιούμε παρόμοια intents, θα πρέπει να προσέξουμε τον όγκο των δεδομένων εκπαίδευσης που εισάγουμε στα intent, θα πρέπει να είναι ισορροπημένες οι διαφορές μεταξύ τους, αν και όσο περισσότερα παραδείγματα δίνουμε τόσο μεγαλύτερη αξιοπιστία θα έχουμε, δίνοντας περισσότερα παραδείγματα σε ένα intent μπορεί να οδηγήσει σε ένα προκατειλημμένο ταξινομητή που με τη σειρά του θα επηρεάσει αρνητικά την ακρίβεια που θα μπορούσε να δώσει το μοντέλο μας(Wochinger, 2019). Τα intent στο Rasa Open Source πρέπει να δηλωθούν σε δυο μέρη, στο “domain.yml” που είναι και η αρχική τους δήλωση και στο “nlu.yml” που είναι το σημείο που δίνουμε τα παραδείγματα εκπαίδευσης για το κάθε intent.



Εικόνα 12 Δήλωση intents

4.5.3 Training Data

Τα δεδομένα εκπαίδευσης αποτελούνται από παραδείγματα φράσεων που θα μπορούσε να επικοινωνήσει ένας χρήστης στο ψηφιακό βοηθό κατηγοριοποιημένα βάση των intents που σχετίζονται. Αυτά ορίζονται στο αρχείο “nlu.yml”, γενικά είναι ένα από τα σημεία που θα πρέπει να προσεχθεί κατά την υλοποίηση της εφαρμογής καθώς η έλλειψη ικανών ποσοτήτων παραδειγμάτων μπορεί να προκαλέσει δυσλειτουργία στο τελικό μας μοντέλο. Γενικότερα αυτό το σημείο βοηθάει περισσότερο η τεχνική CDD που αναφέραμε προηγουμένως, καθώς οι χρήστες όπως προέκυψε και από την εφαρμογή που υλοποιήθηκε είναι η καταλληλότερη πηγή πληροφορίας για την απόκτηση νέων παραδειγμάτων εκπαίδευσης.

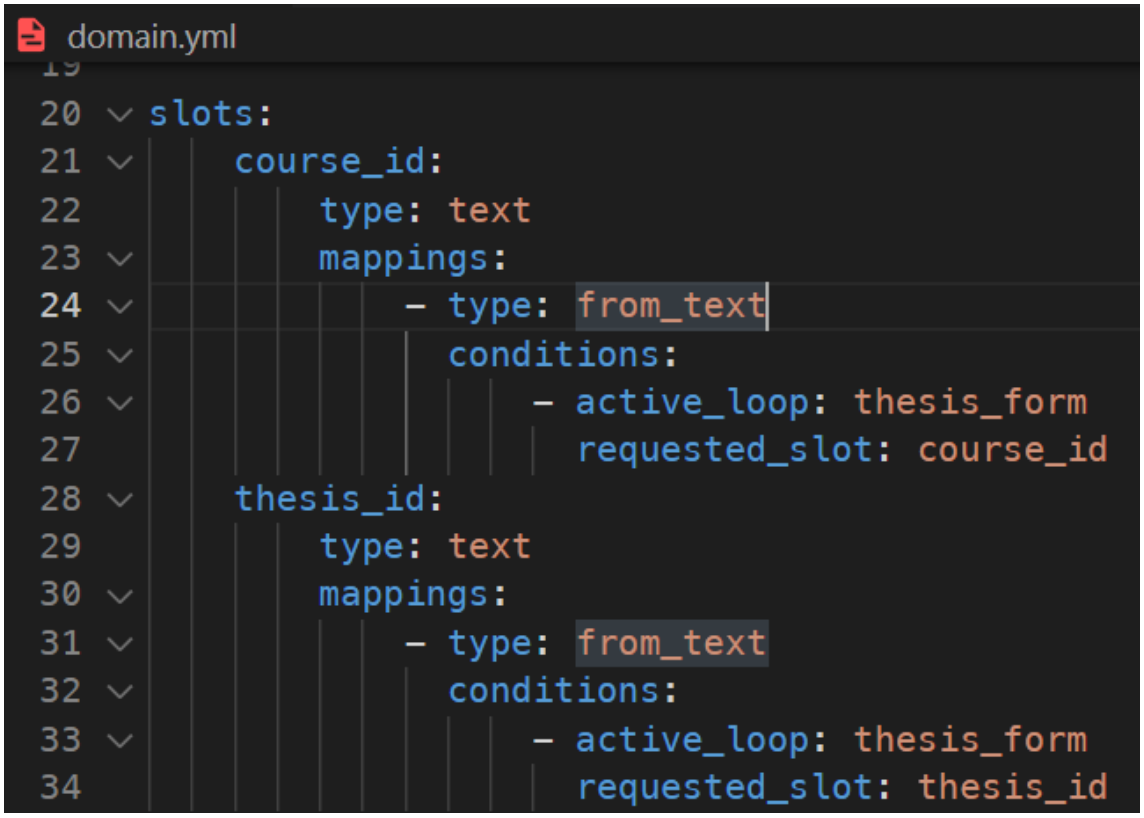
data > nlu.yml

```yaml
version: '3.1'

nlu:
  - intent: greet
    examples: |
      - γεια
      - γεια σου
      - γιά χαρά
      - καλημέρα
      - καλησπέρα

  - intent: goodbye
    examples: |
      - αντίο
      - σε ευχαριστώ
      - να είσαι καλά
      - τα λέμε
      - καληνύχτα
      - καλό βράδυ
      - καλή συνέχεια

  - intent: affirm
    examples: |
      - ναι
      - φυσικά
      - βέβαια
      - βεβαίως
      - αμέ
      - σίγουρα
      - εννοείται

  - intent: deny
    examples: |
      - όχι
      - ποτέ
      - δε χρειάζομαι κάτι άλλο
      - φτανει ολα καλα
      - οχι ενταξει
      - δεν
      - δε μπορεις

  - intent: change_thesis
    examples: |
      - θέλω να αλλάξω θέμα
      - θέλω να αλλάξω πτυχιακή
      - Αλλαγή θέματος
      - Αλλαγή πτυχιακής
      - Αλλαγή πτυχιακής εργασίας
      - θέλω άλλο θέμα πτυχιακής

```

Εικόνα 13 Παράδειγμα training data

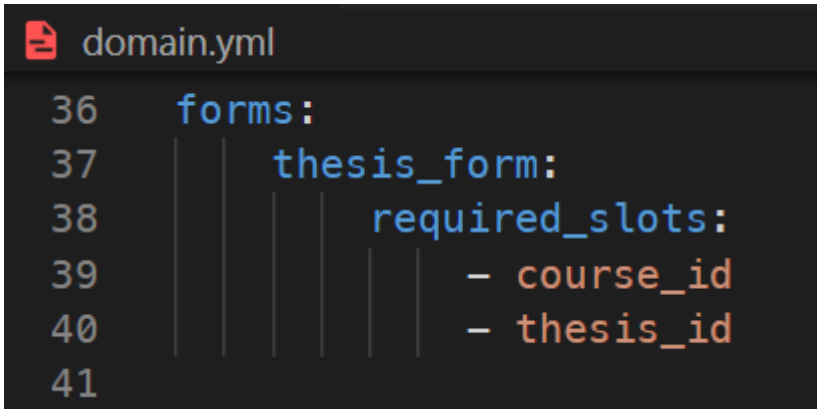
4.5.4 Slots and forms

Τα slots αποτελούν ένα είδους μνήμης της εφαρμογής, δρουν ως μια μορφή αποθήκευσης κλειδιού και τιμής και αποθηκεύουν τη πληροφορία που έχει δώσει ο χρήστης για μπορέσει αυτή να χρησιμοποιηθεί από την εφαρμογή για παράδειγμα σε μια φόρμα (*Slots*, 2022).



Εικόνα 14 Δήλωση slots

Τα forms είναι μια μορφή ενέργειας που μπορεί να βοηθήσει το ψηφιακό βοηθό να αντλήσει πληροφορίες από το χρήστη, στη δική μας περίπτωση κατά την εκτέλεση ενός form χρησιμοποιούνται τα slots για τη αποθήκευση των πληροφοριών που δίνει ο χρήστης και στη συνέχεια γίνεται επεξεργασία τους σε ένα custom action.



Εικόνα 15 Δήλωση form

4.5.5 Stories και rules

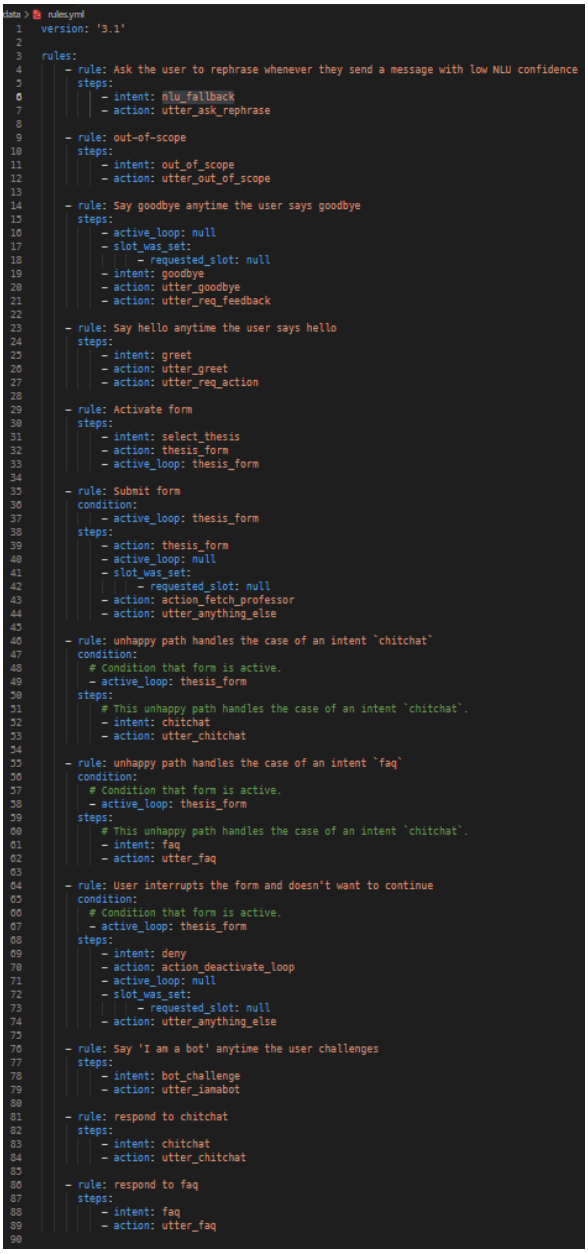
Τα stories αποτελούν ένα είδος δεδομένων εκπαίδευσης και χρησιμοποιούνται για την εκπαίδευση της εφαρμογής ως προς τη διαχείριση του διαλόγου με τον χρήστη. Είναι μια αποτύπωση ενός μονοπατιού συζήτησης δομημένο σε βήματα που αποτελούνται από προθέσεις του χρήστη και ενέργειες ή απαντήσεις του chatbot και δηλώνονται στο αρχείο “stories.yml” στο φάκελο “data”(*Stories*, 2022).

data > stories.yml

```yaml
version: '3.1'

stories:
    - story: change thesis
      steps:
          - intent: change_thesis
          - action: utter_change_thesis
          - action: utter_anything_else

    - story: Help for thesis application not finding the application doc
      steps:
          - intent: application_help
          - action: utter_application_help
          - intent: help_where_find_applicationdoc
          - action: utter_help_where_find_applicationdoc
          - action: utter_anything_else

    - story: Help for thesis application how to fill the application doc
      steps:
          - intent: application_help
          - action: utter_application_help
          - intent: help_howto_fill_applications
          - action: utter_help_howto_fill_applications
          - action: utter_help_where_find_applicationdoc
          - action: utter_anything_else

    - story: thesis negative
      steps:
          - intent: new_thesis
          - action: utter_select_thesis
          - intent: deny
          - action: utter_anything_else

    - story: nothing else
      steps:
          - action: utter_anything_else
          - intent: deny
          - action: utter_goodbye
          - action: utter_req_feedback

```

Εικόνα 16 Εισαγωγή stories

Τα rules είναι και αυτά είδος δεδομένων εκπαίδευσης που χρησιμοποιούνται για την εκπαίδευση της εφαρμογής ως προς τη διαχείριση του διαλόγου με τον χρήστη. Τα rules όμως περιγράφουν μικρά κομμάτια διαλόγου που πάντα ακολουθούν το ίδιο μονοπάτι. Τα rules όπως και τα stories δηλώνονται στο φάκελο “data” στο αρχείο “rules.yml”.



```yaml
version: '3.1'

rules:
  - rule: Ask the user to rephrase whenever they send a message with low NLU confidence
    steps:
      - intent: nlu_fallback
      - action: utter_ask_rephrase

  - rule: out-of-scope
    steps:
      - intent: out_of_scope
      - action: utter_out_of_scope

  - rule: Say goodbye anytime the user says goodbye
    steps:
      - active_loop: null
      - slot_was_set:
          - requested_slot: null
      - intent: goodbye
      - action: utter_goodbye
      - action: utter_req_feedback

  - rule: Say hello anytime the user says hello
    steps:
      - intent: greet
      - action: utter_greet
      - action: utter_req_action

  - rule: Activate form
    steps:
      - intent: select_thesis
      - action: thesis_form
      - active_loop: thesis_form

  - rule: Submit form
    condition:
      - active_loop: thesis_form
    steps:
      - action: thesis_form
      - active_loop: null
      - slot_was_set:
          - requested_slot: null
      - action: action_fetch_professor
      - action: utter_anything_else

  - rule: unhappy path handles the case of an intent `chitchat`
    condition:
      # Condition that form is active.
      - active_loop: thesis_form
    steps:
      # This unhappy path handles the case of an intent `chitchat`.
      - intent: chitchat
      - action: utter_chitchat

  - rule: unhappy path handles the case of an intent `faq`
    condition:
      # Condition that form is active.
      - active_loop: thesis_form
    steps:
      # This unhappy path handles the case of an intent `chitchat`.
      - intent: faq
      - action: utter_faq

  - rule: User interrupts the form and doesn't want to continue
    condition:
      # Condition that form is active.
      - active_loop: thesis_form
    steps:
      - intent: deny
      - action: action_deactivate_loop
      - active_loop: null
      - slot_was_set:
          - requested_slot: null
      - action: utter_anything_else

  - rule: Say 'I am a bot' anytime the user challenges
    steps:
      - intent: bot_challenge
      - action: utter_iamabot

  - rule: respond to chitchat
    steps:
      - intent: chitchat
      - action: utter_chitchat

  - rule: respond to faq
    steps:
      - intent: faq
      - action: utter_faq
```

Εικόνα 17 Εισαγωγή rules

4.5.6 Responses

Τα responses αποτελούν το σύνολο απαντήσεων που μπορεί να στείλει η εφαρμογή του ψηφιακού βοηθού προς τον χρήστη. Συνήθως είναι κείμενο και κάποιες φορές μπορεί να έχουν και κουμπιά. Οι απαντήσεις δηλώνονται στο αρχείο “domain.yml”. Συνίσταται κάθε απάντηση να έχει το λεκτικό “utter_” ως πρώτο κομμάτι.

4.5.7 Custom Actions

Τα custom actions αποτελούν το σημείο που μπορούμε να γράψουμε δικό μας κώδικα, όπως για παράδειγμα μέσα από αυτά μπορούμε να κάνουμε ερωτήματα στη βάση δεδομένων που έχουμε δημιουργήσει, να κάνουμε επαλήθευση των slots κάποιας φόρμας που χρησιμοποιεί η εφαρμογή μας και άλλα. Η χρήση των custom actions στη παρούσα εφαρμογή έγινε στην ανάκτηση πληροφοριών από τη βάση δεδομένων, δημιουργήθηκαν τρία τέτοια custom actions, το action

“actions_ask_cource_id” που ήταν υπεύθυνο κατά την εκκίνηση του σεναρίου επιλογής νέου θέματος να δείχνει στο χρήστη τα διαθέσιμα μαθήματα που υπάρχουν για να διαλέξει θέμα, αφού ο χρήστης διάλεγε κάποιο μάθημα τότε εκτελούταν το action “action_ask_thesis_id” που ως σκοπό είχε βάση την επιθυμία του χρήστη στο μάθημα να φέρνει από τη βάση πληροφορίες για τα διαθέσιμα θέματα που υπήρχαν και στη συνέχεια εκτελούταν το action “action_fetch”professor” το οποίο λάμβανε την επιθυμία του φοιτητή ως προς το θέμα πτυχιακής εργασίας και τον ενημέρωνε με τα στοιχεία του καθηγητή που θα πρέπει να έρθει σε επικοινωνία.

actions > actions.py

```python
from rasa_sdk import Action, Tracker, FormValidationAction
from rasa_sdk.events import AllSlotsReset
from rasa_sdk.executor import CollectingDispatcher
from rasa_sdk.types import DomainDict
from actions.db_helpers import connect_to_db, close_connection


class ActionFetchCourses(Action):

    def name(self) -> Text:
        return "action_ask_course_id"

    def run(self,
            dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        db = connect_to_db()
        cursor = db.cursor()

        cursor.execute("SELECT * FROM courses;")
        courses = cursor.fetchall()

        close_connection(db)

        course_buttons = [
            {
                "title": course[1],
                "payload": f"{course[0]}"
            }
            for course in courses
        ]

        dispatcher.utter_message(
            text="Διάλεξε μάθημα για να δεις τα διαθέσιμα θέματα πτυχιακών που υπάρχουν:",
            buttons=course_buttons,
            button_type="vertical"
        )

        return []
```

Εικόνα 18 action_ask_course_id

Οι κλάσεις που αποτελούν το κώδικα για τα custom actions, βρίσκονται στο φάκελο “actions” στο αρχείο “actions.py” και όλα τα custom actions πρέπει να δηλωθούν και στο αρχείο “domain.yml”.

4.5.8 Κανάλια επικοινωνίας

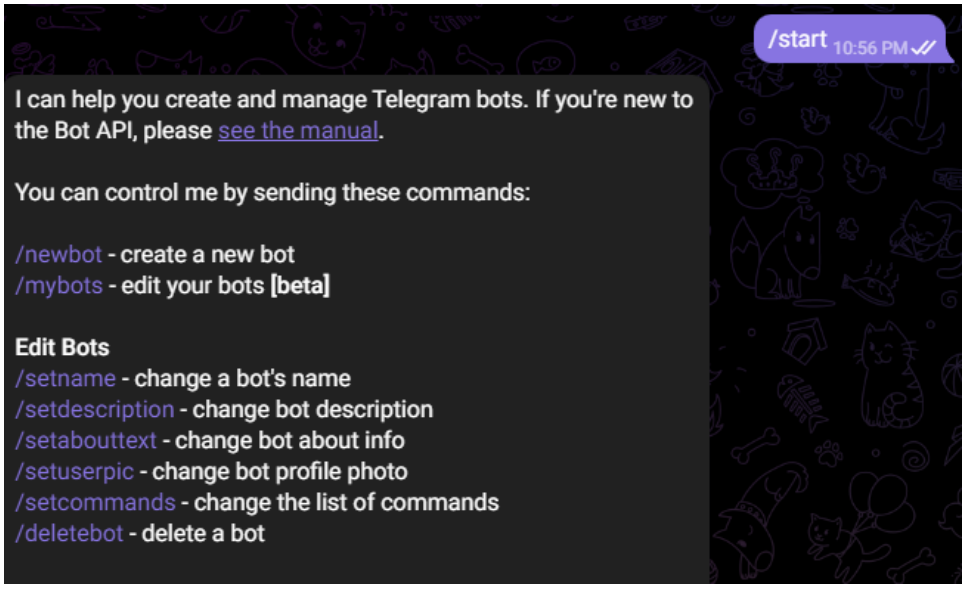
Ως κανάλια επικοινωνίας επιλέχθηκε το Socket IO το οποίο κάνει χρήση WebSocket για επικοινωνία σε πραγματικό χρόνο, με τη χρήση αυτού του καναλιού μας δίνεται η επιλογή να δημιουργήσουμε μια δική μας εφαρμογή web που να επιτρέπει την επικοινωνία του χρήστη με την εφαρμογή μας. Για την υλοποίηση αυτού έγινε χρήση του Rasa Chat Widget σε μια html σελίδα(*Your Own Website*, 2022).

```html
index.html > ...
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
  <div>
    The widget will won't show here, instead it will attach itself to the main page
  </div>
  <div
    data-initial-payload="Γειά"
    data-root-element-id="storybook-preview-wrapper"
    data-websocket-url="http://104.248.243.238:5005/socket.io"
    id="rasa-chat-widget"
  />
  <script src="https://unpkg.com/@rasahq/rasa-chat" type="application/javascript"></script>
</body>
</html>
```

Εικόνα 19 HTML κώδικας με χρήση Rasa Webchat Widget

Ένα άλλο κανάλι που μας προσφέρει το Rasa είναι το Rest. Το Rest κανάλι λειτουργεί όπως και ένα Rest API, έρχεται ήδη ενεργοποιημένο από το Rasa Open Source.

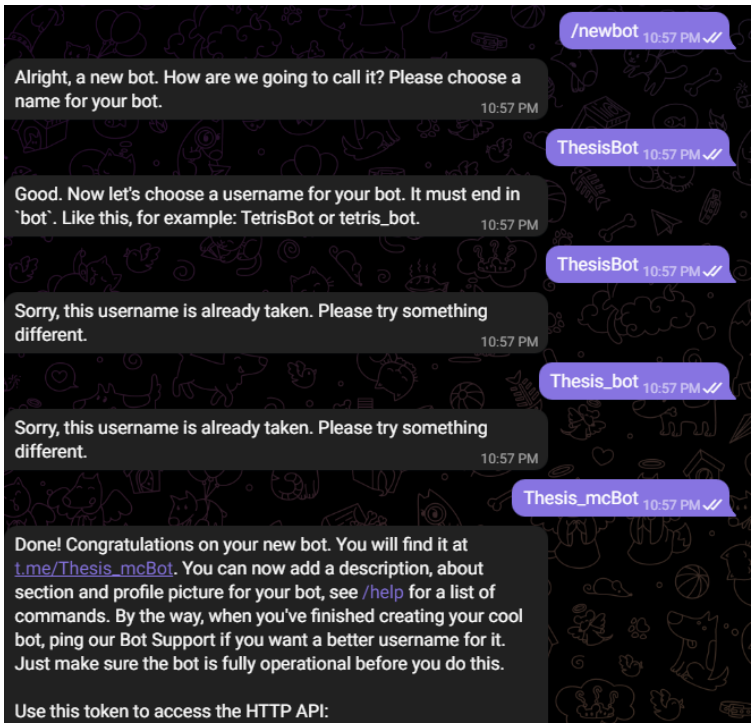
Τέλος έγινε και η δημιουργία του καναλιού Telegram. Το Telegram είναι μια εφαρμογή ανταλλαγής μηνυμάτων, επιλέχθηκε επειδή σε σύγκριση με το Facebook Messenger, Google hangouts και άλλα, η διαμόρφωσή της ήταν αρκετά απλή. Αρχικά θα πρέπει να έχουμε ένα λογαριασμό Telegram, στη συνέχεια θα χρειαστεί να δημιουργήσουμε ένα bot στο Telegram, αυτό μπορεί να γίνει με τη χρήση του BotFather, ο BotFather είναι μια εφαρμογή του Telegram που μας επιτρέπει να δημιουργήσουμε ένα νέο bot ή και ένα παιχνίδι. Η όλη διαδικασία γίνεται από το περιβάλλον ανταλλαγής μηνυμάτων του Telegram.



Εικόνα 20 Εκκίνηση δημιουργίας νέου bot από το Telegram

Ακολουθώντας τις οδηγίες της εφαρμογής του Telegram έχουμε δημιουργήσει ένα νέο bot και του έχουμε δώσει όνομα, στη συνέχεια το Telegram μας ενημερώνει με το σύνδεσμο για

να επικοινωνήσουμε με το bot αλλά και το token που θα χρειαστεί να βάλουμε στις ρυθμίσεις του Rasa Open Source για να μπορέσει η εφαρμογή μας να χρησιμοποιήσει το Telegram.



Εικόνα 21 Δημιουργία Telegram Bot

Στη συνέχεια χρειάζεται να δηλώσουμε στο Telegram τη διεύθυνση της εφαρμογής μας, αυτό γίνεται με τη χρήση του HTTP API που διαθέτει το Telegram. Κάνοντας μια απλή κλήση στον σύνδεσμο με το token που λάβαμε από το Telegram και τον σύνδεσμο επικοινωνίας με το Rasa «https://api.telegram.org:443"GET /bot< token>/setWebhook?url=<Rasa_websocket_url>».

4.5.8 Server Configuration

Για την διαμόρφωση του server χρησιμοποιήθηκε η τεχνολογία Docker, πιο συγκεκριμένα το εργαλείο Docker Compose, αυτό έγινε για τη μεταφερσιμότητα που μπορεί να προσφέρει. Με τη χρήση του δημιουργήθηκε ένα περιβάλλον που με ευκολία μπορούσαμε να ξεκινήσουμε να σταματήσουμε και να μορφοποιήσουμε τις υπηρεσίες που είχαμε ανάγκη με ευκολία. Τα βασικά στοιχεία που περιγράφουν την εφαρμογή είναι τα υποσυστήματα που φαίνεται στη παρακάτω εικόνα.

Εικόνα 22 Περιγραφή διαμόρφωσης server με Docker

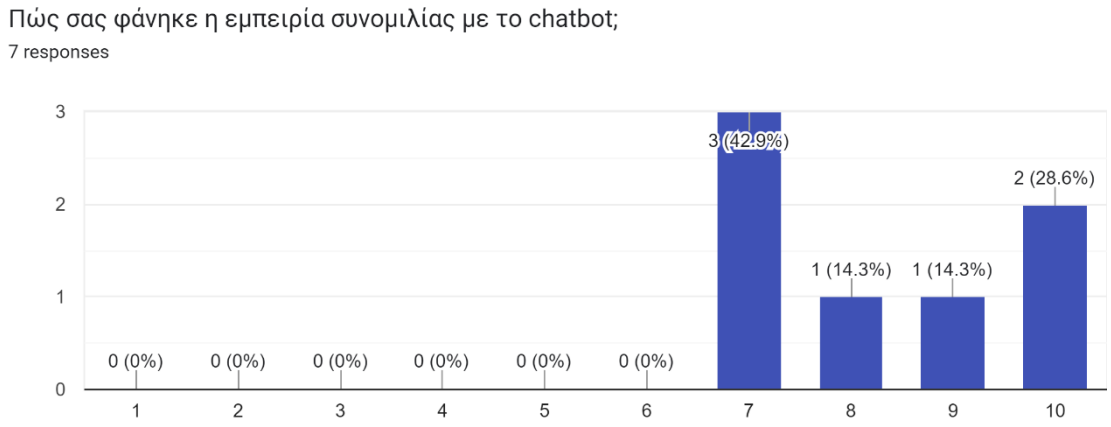
Τα συστήματα αυτά είναι, η κεντρική μας εφαρμογή το Rasa Open Source με όνομα “rasa”, αυτό περιέχει τα μοντέλα του NLU τα intents τα responses τα stories και τα rules που έχουμε δηλώσει στην εφαρμογή μας. Το service “action_server” περιέχει τα custom actions που αναφέραμε προηγουμένως, εδώ περιέχονται ο κώδικας που έχουμε γράψει για την επεξεργασία της βάσης δεδομένων μας. Και τέλος η βάση MongoDB που αποτελεί το Tracker Store, αυτό χρησιμοποιείτε από το Rasa Open Source για την αποθήκευση των διαλόγων της εφαρμογής και των χρηστών. Στη συνέχει προστέθηκε και ένας NGINX proxy server για να μπορέσουμε να διαθέσουμε την εφαρμογή μας με πιστοποιητικό SSL ώστε να μπορεί να επικοινωνήσει με το Telegram μιας και είναι μια απαίτηση του η επικοινωνία να γίνεται μέσω HTTPS πρωτοκόλλου.

4.6 Έλεγχος - αξιολόγηση

Για τον έλεγχο και την αξιολόγηση του συστήματος έγινε δημιουργία ενός απλού ερωτηματολογίου. Το ερωτηματολόγιο αποτελούταν από τρία τμήματα, το πρώτο τμήμα είχε δυο ερωτήσεις αρχικά για την εμπειρία του χρήστη με το chatbot, με απάντηση κλίμακας δέκα σημείων με τελικά σημεία το 0 για χείριστη έως το 10 για άριστη, και στη συνέχεια μια ερώτηση αν υπήρξε σημείο που το chatbot δεν απάντησε κάτι σωστά που δεχόταν μια απάντηση ναι ή όχι. Το δεύτερο τμήμα αποτελούταν από ερωτήσεις ανοιχτού κειμένου οι οποίες ρωτούσαν το χρήστη ποια ερώτηση έκανε προς το ψηφιακό βοηθό, τι του απάντησε αυτός και τι θα περίμενε να απαντήσει, το τμήμα αυτό βοήθησε στην διόρθωση λογικών σφαλμάτων στην εφαρμογή και ως σκοπό είχε την εκπλήρωση του της μεθοδολογίας που περιγράφει το CDD, λαμβάνοντας άμεση ανατροφοδότηση. Και το τελευταίο τμήμα αποτελούταν από μια ερώτηση ανοιχτού κειμένου όπου

ο χρήστης μπορούσε να αφήσει κάποιο γενικό σχόλιο για μελλοντική εξέλιξη που θα μπορούσε να έχει η εφαρμογή. Η υλοποίηση της φόρμας έγινε με χρήση Google Forms.

Από τα αποτελέσματα είδαμε ότι οι χρήστες ήταν ικανοποιημένοι με τη λειτουργεία του chatbot, και εντοπίστηκαν σφάλματα που υπήρχαν λόγω των διαφορετικών καναλιών που χρησιμοποιήθηκαν. Για παράδειγμα στο Telegram όταν ο χρήστης ξεκίναγε μια νέα συνομιλία το chatbot δεν απαντούσε γιατί δε μπορούσε να καταλάβει το “/start” που στέλνει αυτόματα το Telegram. Βέβαια το δείγμα χρηστών ήταν μικρό, εφτά άτομα, αποτελούμενο από διαφορετικές βαθμίδες φοιτητών διαφόρων πανεπιστημίων.



Εικόνα 23 Βαθμολόγηση εμπειρίας chatbot (Πηγή: Google Forms)

5 Οδηγίες εγκατάσταση και χρήσης

Στο παρόν κεφάλαιο θα δημιουργηθεί ένας οδηγός χρήσης της εφαρμογής. Ποιο συγκεκριμένα, θα γίνει αναφορά και ανάλυση στις ενέργειες που θα χρειαστεί να πραγματοποιηθούν για να λειτουργήσει αρχικά η εφαρμογή, καθώς και πως μπορούμε να προχωρήσουμε στην επέκτασή της. Η διαδικασία που θα ακολουθηθεί θα αρχίζει από την εγκατάσταση των βασικών λογισμικών που χρειαζόμαστε για την εκκίνηση της εφαρμογής αλλά και την επέκτασή της, στη συνέχεια θα γίνει αναφορά στα αντικείμενα που μπορούν να προστεθούν ή να αλλαχτούν στην εφαρμογή ώστε να μπορέσουμε να προβούμε σε επέκταση η παραμετροποίηση αυτής και τέλος, θα δείξουμε τις διαδικασίες που πρέπει να ακολουθηθούν για την διάθεσή της στο κοινό.

5.1 Εγκατάσταση λογισμικού

Ο ψηφιακός βοηθός που κατασκευάστηκε για την παρούσα μεταπτυχιακή διατριβή είναι εγκατεστημένος σε έναν Ubuntu Linux Server. Αν θελήσουμε να το μεταφέρουμε σε κάποιο άλλο περιβάλλον μπορεί να γίνει πολύ απλά λόγω του τρόπου που έχει γίνει η εγκατάσταση του και το μόνο που θα χρειαστεί είναι ένα περιβάλλον που να μπορεί να υποστηρίξει το Docker Engine και συγκεκριμένα το εργαλείο Docker Compose.

Αν θελήσουμε όμως να προβούμε σε επέκταση ή ακόμα και σε αλλαγή αυτού τότε θα χρειαστούμε και κάποια επιμέρους πράγματα όπως, Python, PIP, και έναν text editor ή IDE, για να μπορέσουμε να προβούμε σε αλλαγές του κώδικα της εφαρμογής.

Πιο συγκεκριμένα για την απλή χρήση της εφαρμογής θα χρειαστεί η εγκατάσταση του Docker Engine και Docker Compose στον υπολογιστή που έχουμε στη διάθεσή μας αυτό μπορεί να γίνει πολύ απλά, ακολουθώντας τις οδηγίες που είναι διαθέσιμες στη σελίδα του λογισμικού Docker Engine (https://docs.docker.com/engine/install/), αρχικά θα πρέπει να βεβαιωθούμε αν υπάρχει εγκατεστημένο το Docker Engine και το Docker Compose στο μηχάνημα πηγαίνοντας στη γραμμή εντολών και γράφοντας αυτές τις εντολές, “docker --version” που μας δείχνει την έκδοση του λογισμικού Docker Engine και την εντολή “docker compose version” που μας δείχνει την έκδοση του λογισμικού Docker Compose, αν αυτά δεν είναι διαθέσιμα μπορούμε να προχωρήσουμε στην εγκατάσταση τους με τη διαδικασία που προτείνεται παραπάνω. Στη συνέχεια έχοντας διαθέσιμα τα αρχεία της εφαρμογής, στο φάκελο που υπάρχει το αρχείο “docker-compose.yml” μπορούμε να εκτελέσουμε την εντολή “docker compose up -d”, αυτή η εντολή αναλαμβάνει να ανατρέξει στις ρυθμίσεις που έχουμε ορίσει στο “docker-compose.yml” αρχείο, να δημιουργήσει τα απαραίτητα “containers” όπως ονομάζονται τα υποσυστήματα στην εννοιολογικά του Docker Engine, να κατεβάσει τις απαραίτητες εφαρμογές στο καθένα από αυτά από το Docker Hub, το οποίο αποτελεί ένα είδους αποθετηρίου κώδικα για ανοιχτό λογισμικό, και τέλος να τις ενεργοποιήσει. Αυτό μετά φροντίζει για τις σωστές εκδόσεις στο λογισμικό που χρησιμοποιούμε καθώς και την επανεκκίνηση τους αν αυτές τερματιστούν με κάποιο σφάλμα συστήματος, όπως και καθιστά τη μεταφερσιμότητα όλης της εγκατάστασης πολύ εύκολη και γρήγορη.

Στην περίπτωση που θέλουμε να προβούμε σε επέκταση ή και αλλαγή του πηγαίου κώδικα της εφαρμογής τότε θα χρειαστούμε παραπάνω στοιχεία, τα οποία θα βοηθήσουν στην εκπαίδευση του μοντέλου μας που θα χρησιμοποιηθεί από το Rasa Open Source και πιο συγκεκριμένα από το NLU του. Θα πρέπει να φροντίσουμε να έχουμε στο εκάστοτε μηχάνημα Python έκδοσης 3.7 ή 3.8 καθώς και το πρόγραμμα PIP. Όπως είπαμε θα χρειαστούμε έκδοση Python 3.7 ή 3.8 όπως αναφαίρετε στο εγχειρίδιο χρήσης του Rasa Open Source(*Setting up Your Environment*, 2022), για να ελέγξουμε αν έχουμε εγκατεστημένη την Python και σε ποια έκδοση είναι μπορούμε να χρησιμοποιήσουμε την εντολή “python --version” αν η Python δεν είναι διαθέσιμη ή είναι σε λάθος έκδοση μπορούμε να προχωρήσουμε στην εγκατάσταση της ακολουθώντας τις οδηγίες εγκατάστασης της ανάλογα με το λογισμικό που χρησιμοποιούμε στο διαθέσιμο μηχάνημα. Στη συνέχεια θα χρειαστεί να βεβαιωθούμε ότι έχουμε το λογισμικό PIP αυτό μπορούμε να το κάνουμε αντίστοιχα δίνοντας την εντολή “pip --version”, αν δεν είναι διαθέσιμο θα πρέπει να προχωρήσουμε στην εγκατάστασή του αντίστοιχα. Αφού επιβεβαιώσουμε ότι υπάρχει η Python και το PIP στο σύστημα μας, δημιουργούμε ένα εικονικό περιβάλλον της Python, με αυτό το τρόπο μπορούμε να απομονώσουμε όσο το δυνατόν περισσότερο τα στοιχεία της εφαρμογής μας ώστε να μη δημιουργηθούν σφάλματα όσο αναφορά τις εκδόσεις διαφόρων

βιβλιοθηκών που αυτό μπορεί να χρησιμοποιήσει και μπορεί να χρησιμοποιούνται και σε άλλες Python εφαρμογές που υπάρχουν στο σύστημα μας. Για να δημιουργήσουμε το εικονικό αυτό περιβάλλον αρκεί να δώσουμε την εντολή “python -m venv ./venv” και για να το ενεργοποιήσουμε δίνουμε την εντολή “.\venv\bin\activate”, τώρα μπορούμε να εγκαταστήσουμε τα απαραίτητα πακέτα που μπορεί να χρειάζεται η εφαρμογή μας χωρίς να δημιουργηθεί κάποιο σφάλμα με άλλη εφαρμογή, επίσης σημειώνουμε ότι για να τερματίσουμε το εικονικό περιβάλλον χρειάζεται να πληκτρολογήσουμε την εντολή “deactivate”. Στη συνέχεια θα πρέπει να εγκαταστήσουμε τα απαραίτητα πακέτα για τη λειτουργία της εφαρμογής μας, αυτά στη συγκεκριμένη περίπτωση είναι το “Rasa” το οποίο υπάρχει και στο “requirements.txt” αρχείο δίνοντας την εντολή “pip install rasa” ή “pip install -r requirements.txt”, αυτό θα κάνει εγκατάσταση των απαιτήσεων του συστήματος και θα μπορούμε να προχωρήσουμε στην επέκταση της εφαρμογής.

5.2 Τρόπος επέκτασης εφαρμογής

Η επέκταση της εφαρμογής χωρίζεται σε πολλαπλά μέρη, αφού πρώτα έχουμε εγκαταστήσει τα απαραίτητα στοιχεία μπορούμε να προβούμε σε αλλαγές και προσθήκες στην εφαρμογή του ψηφιακού βοηθού

5.2.1 Εισαγωγή νέου Intent

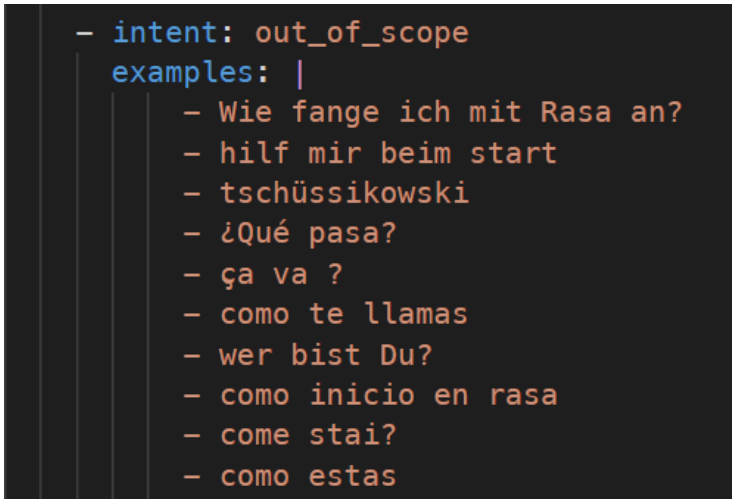
Τα intent που έχουν ήδη δημιουργηθεί βρίσκονται στο αρχείο “domain.yml” στο πάνω μέρος του, αν θέλουμε να προσθέσουμε ένα νέο intent θα πρέπει να βρούμε την οντότητα intents στο αρχείο αυτό και να το προσθέσουμε με το επιθυμητό όνομα, εδώ θα πρέπει να είμαστε πολύ προσεκτικοί για το τρόπο γραφής μας, διότι τα YAML αρχεία λειτουργούν όπως και η Python με εσοχές, οπότε αν δεν έχουμε ορίσει σωστό εσοχή κατά τη προσθήκη του κειμένου μας θα μπορούσαμε να καταλήξουμε σε σφάλμα κατά την εκπαίδευση του μοντέλου και χρειάζεται να προστεθεί μια παύλα πριν το όνομα του Intent πχ “- out_of_scope”.

Στη συνέχεια θα πρέπει να πάμε στο αρχείο “data/nlu.yml” και να δηλώσουμε και εκεί το νέο intent μέσα στην οντότητα “nlu” προσθέτουμε σε μια κενή γραμμή “- intent: out_of_scope”, όπως και πριν πάλι θα πρέπει να προσέξουμε το τρόπο γραφής γιατί πρόκειται για αρχείο YAML και αλλιώς θα έχουμε πρόβλημα κατά την εκπαίδευση του μοντέλου.

Με παρόμοιο τρόπο μπορούν να δηλωθούν και entities αλλά και slots για την εφαρμογή του ψηφιακού βοηθού, με τη μόνη διαφορά ότι αυτά χρειάζεται να δηλωθούν στο “domain.yml” αρχείο και στη συνέχεια τα entities μόνο στο αρχείο “data/nlu.yml” αν και όπου αυτό είναι αναγκαίο.

5.2.2 Προσθήκη εκπαιδευτικού υλικού

Το εκπαιδευτικό υλικό αναφέρεται στις περιπτώσεις χρήσης, που ο ψηφιακός βοηθός θα χρησιμοποιήσει ένα συγκεκριμένο intent, το υλικό αυτό θα το περάσουμε στην εφαρμογή μας ως παραδείγματα. Συγκεκριμένα, στο αρχείο “data/nlu.yml” κάτω από το intent που έχουμε προσθέσει θα γράψουμε μια εσοχή μετά το λεκτικό “examples: |” αυτό βοηθάει το Rasa Open Source να καταλάβει ότι ακολουθούν παραδείγματα σχετικά με το intent αυτό. Για παράδειγμα το “out_of_scope” intent θα μπορούσε να γίνει όπως στην εικόνα παρακάτω:



Εικόνα 24 Παράδειγμα εισαγωγής εκπαιδευτικού υλικού σε intent

Στη παραπάνω εικόνα φαίνεται ότι έχουν προστεθεί λέξεις από ξένα λεξικά, αυτό μπορεί να γίνει για να καταλαβαίνει ο ψηφιακός βοηθός ότι ο χρήστης που επικοινωνεί μαζί του δε μιλάει κάποια γλώσσα στην οποία μπορεί να αντιληφθεί και να τον βοηθήσει. Θα πρέπει να προσέξουμε κατά την εισαγωγή των παραδειγμάτων να μπορέσουμε να πιάσουμε όσο μεγαλύτερο φάσμα μπορούμε από τα πιθανά παραδείγματα που μπορεί να υπάρχουν για το συγκεκριμένο intent ώστε να μπορέσουμε να επιτύχουμε και μεγαλύτερες πιθανότητες για σωστή κατηγοριοποίηση του μηνύματος του χρήστη.

Επίσης, αξίζει να σημειωθεί ότι εδώ θα μπορούσαμε να δηλώσουμε και κάποια entities τα οποία θα μπορεί να θέλαμε να καταλάβει ο ψηφιακός βοηθός και ίσως και να χρησιμοποιήσει στη συνέχεια της συζήτησής του με το χρήστη, αυτό μπορεί να γίνει βάζοντας τη λέξη που θα μπορούσε να αναφερθεί σε entity σε αγκύλες και το τύπο entity που θα μπορεί να αναφέρεται σε παρενθέσεις.

5.2.3 Προσθήκη νέων φορμών

Οι φόρμες που μπορεί να χρησιμοποιήσει ο ψηφιακός βοηθός και αυτές θα πρέπει να δηλωθούν στο αρχείο “domain.yml”, θα πρέπει να βρούμε ή να δηλώσουμε την οντότητα “forms” και κάτω από αυτή να ορίσουμε ένα δικό μας όνομα φόρμας και στη συνέχεια να δηλώσουμε τα entities που αυτή η φόρμα θα χρησιμοποιήσει. Οι φόρμες αποτελούν μια σχετικά νέα ιδιότητα του Rasa Open Source και μπορεί να φανούν αρκετά χρήσιμες στη καθοδήγηση του χρήστη και στην απόσπαση των απαραίτητων πληροφοριών από αυτόν για τη καλύτερη εξυπηρέτηση του από το ψηφιακό βοηθό.

5.2.4 Προσθήκη νέων απαντήσεων

Όπως έχουμε πει και στο προηγούμενο κεφάλαιο οι απαντήσεις που μπορεί να δώσει ο ψηφιακός βοηθός υπάρχουν στο αρχείο “domain.yml”, συγκεκριμένα στην οντότητα responses, μια καλή πρακτική για τη προσθήκη απαντήσεων είναι να δίνουμε ως όνομα το λεκτικό “utter_” ακολουθούμενο από το intent στο οποίο θέλουμε να μπορεί να απαντήσει, για παράδειγμα στη περίπτωση του “out_of_scope” intent το όνομα θα ήταν “utter_out_of_scope”, έτσι είναι εύκολο να κατανοήσει ο προγραμματιστής για πιο intent προορίζεται αυτή η απάντηση και να μπορέσει εύκολα να δημιουργήσει τα stories και τα rules χωρίς να χρειάζεται να ανατρέξει κάθε φορά σε άλλα αρχεία για να δει το όνομα που θα πρέπει να προσθέσει. Αφότου δώσουμε το επιθυμητό όνομα στην απάντηση προσθέτουμε, ακολουθώντας και εδώ τη σύνταξη των YAML αρχείων, το τύπο απάντησης πχ “text” για περιεχόμενο κειμένου, “button” για κουμπί και ότι άλλο μπορεί να υποστηριχθεί από το κανάλι που μπορεί να χρησιμοποιούμε, ένα παράδειγμα με κείμενο μπορεί να φανεί στη παρακάτω εικόνα(Εικόνα 7) για την εισαγωγή της απάντησης του intent “greet”.

```
responses:
    utter_greet:
        - text: 'Γεια σου! Είμαι ο ψηφιακός βοηθός
        για τις πτυχιακές εργασίες.'
```

Εικόνα 25 Παράδειγμα εισαγωγής απάντησης

Ένα άλλο παράδειγμα με κουμπιά φαίνεται στη παρακάτω εικόνα(Εικόνα 8) που αναφέρεται στο intent “application_help” που μπορεί να ζητηθεί από το χρήστη για να λάβει βοήθεια σχετικά με το έντυπο για τη δήλωση του θέματος της πτυχιακής του εργασίας στη γραμματεία της σχολής.

```
utter_application_help:
    - text: 'Αφού έχεις βρει θέμα και έχεις μιλήσει με τον επιβλέποντα
    καθηγητή, χρειάζεται να δηλώσεις το θέμα σου στη γραμματεία του
    τμήματος, μπορώ παρακάτω είναι οι τρόποι με τους οποίους μπορώ να σου
    παρέχω βοήθεια.'
      buttons:
        - title: 'Βοήθεια στη συμπλήρωση της αίτησης'
          payload: '/help_howto_fill_applications'
        - title: 'Βοήθεια με την εύρεση του εντύπου αίτησης'
          payload: '/help_where_find_applicationdoc'
      button_type: vertical
```

Εικόνα 26 Παράδειγμα εισαγωγής απάντησης με κουμπιά

5.2.5 Εισαγωγή νέου story

Τα story μπορούν να εντοπιστούν στο φάκελο data στο αρχείο “stories.yml”, και αυτό το αρχείο ακολουθεί τον ίδιο τρόπο γραφής με τα προηγούμενα, με χρήση αρχείων YAML, κάθε story πρέπει να περιγράφει ένα σενάριο επικοινωνίας ανάμεσα σε χρήστη και ψηφιακό βοηθό, συγκεκριμένα για να γράψουμε ένα σωστό story θα πρέπει να γνωρίζουμε και να έχουμε δηλώσει τα intents που θα πρέπει να δείξει ο χρήστης στον ψηφιακό βοηθό αλλά να έχουμε και μια εικόνα για τις απαντήσεις και άλλες δράσεις που θα μπορούσε να εκτελέσει ο ψηφιακός βοηθός και αυτές να έχουν δηλωθεί, για να απαντήσει στο χρήστη.

Αφού επιβεβαιώσουμε τη δήλωση των intent και των απαντήσεων και action που μπορεί να κάνει ο ψηφιακός βοηθός, μπορούμε στο αρχείο “data/stories.yml” κάτω από την οντότητα stories να προσθέσουμε μια νέα ή να αλλάξουμε μια ήδη υπάρχουσα, γενικά η μορφή που πρέπει να έχουν τα stories είναι αρχικά ένας τίτλος που δηλώνεται με το λεκτικό “story:” και το τίτλο που επιθυμούμε να του δώσουμε, αυτό υπάρχει για δική μας διευκόλυνση ώστε να έχουμε γνώση σε τι αναφέρεται το story αυτό. Στη συνέχεια θα πρέπει να δηλώσουμε τα βήματα που ακολουθούνται στο story, αυτό γίνεται δημιουργώντας μια οντότητα steps μέσα στο story που δηλώσαμε και προσθέτοντας εκεί intents και actions ανάλογα με το σκοπό που θέλουμε να πετύχουμε ένα παράδειγμα τέτοιου story φαίνεται στην εικόνα 9 που ακολουθεί, όταν ο χρήστης μας ζητήσει να κάνει αλλαγή της πτυχιακής του εργασίας.

```
stories:
    - story: change thesis
      steps:
          - intent: change_thesis
          - action: utter_change_thesis
          - action: utter_anything_else
```

Εικόνα 27 Παράδειγμα story

5.2.6 Εισαγωγή νέου rule

Παρόμοια με τα stories είναι και τα rules, με τη διαφορά ότι ένα rule εκτελείται κάθε φορά και έχει προτεραιότητα από τα stories, τείνει να αποτελεί μια μικρότερη συλλογή intent και action και να είναι πιο ευθύ.

Για τη δημιουργία ενός νέου rule, ακολουθούμε την ίδια τακτική με τα story, πρώτα επιβεβαιώνουμε ότι έχουμε δηλώσει τα απαραίτητα intent και action ή απαντήσεις και στη συνέχει στο φάκελο data και στο αρχείο “rules.yml” μπορούμε να προσθέσουμε το νέο rule που επιθυμούμε. Και εδώ όπως και στα υπόλοιπα αρχεία πρέπει να είμαστε προσεκτικοί με το τρόπο γραφής μας γιατί πρόκειται για αρχεία YAML, όπως και στα stories αρχικά δηλώνουμε το όνομα “rule:” και στη συνέχεια το τίτλο που θέλουμε να δώσουμε στο rule, ακολουθούμενο από τα βήματα που το περιγράφουν. Ένα παράδειγμα rule μπορεί να αποτελέσει η εισαγωγή του “out_of_scope” intent που προσθέσαμε προηγουμένως, συγκεκριμένα θέλουμε όταν ο ψηφιακός βοηθός καταλαβαίνει ότι ο χρήστης δηλώνει αυτό το intent να του στέλνει μια σχετική απάντηση που να τον ενημερώνει πως δε μπορεί να τον βοηθήσει. Ο τρόπος επίτευξης του παραπάνω μπορεί να φανεί στην εικόνα 10 που ακολουθεί.

```
- rule: out-of-scope
  steps:
      - intent: out_of_scope
      - action: utter_out_of_scope
```

Εικόνα 28 Παράδειγμα εισαγωγής νέου rule

5.2.7 Εισαγωγή νέου action

Για τη δημιουργία ενός νέου action, θα πρέπει να κάνουμε αρχικά δήλωσή του στο αρχείο “domain.yml” στην οντότητα “actions”, εκεί δηλώνουμε απλά το όνομα που επιθυμούμε να έχει το action που θα δημιουργήσουμε και που θα χρησιμοποιούμε στα stories και τα rules για να αναφερθούμε σε αυτό.

Για να γράψουμε το κώδικα για το νέο αυτό action θα πρέπει να πάμε στο φάκελο actions και συγκεκριμένα στο αρχείο “actions.py”, για να δημιουργήσουμε εκεί ένα action, δημιουργούμε μια νέα υπό-κλάση της κλάσης Action, αυτή πρέπει να υλοποιήσει τουλάχιστον δυο μεθόδους, τη μέθοδο name που τη χρησιμοποιούμε για να δηλώσουμε το όνομα του action και είναι το ίδιο με το όνομα που έχουμε δώσει στο “domain.yml” αρχείο, και τη μέθοδο run που είναι εμπεριέχει το κώδικα που θέλουμε να εκτελεστεί από το συγκεκριμένο action, μέσα σε αυτή τη μέθοδο μπορούμε να φτιάξουμε το πρόγραμμα μας να κάνει διάφορες κινήσεις, όπως να κάνει κάποια αναζήτηση στη βάση δεδομένων, να λάβει ή και να στείλει πληροφορίες με τη χρήση κάποιου API, να καλέσει άλλες μεθόδους που έχουμε δηλώσει, να μεταβάλουμε τις τιμές από τα διαθέσιμα

slots που μπορεί να υπάρχουν, ή να πάρουμε πληροφορίες από το “tracker_store” το οποίο είναι διαθέσιμο στα actions.

Ένα παράδειγμα ενός action αποτελεί το παρακάτω που αναφέρεται στην αναζήτηση των στοιχείων του καθηγητή για να σταλούν στο καθηγητή που ενδιαφέρεται για θέμα πτυχιακής εργασίας στο οποίο είναι υπό την επίβλεψή του. Βλέπουμε πως αρχικά δηλώνουμε ένα όνομα για τη κλάση μας που συνηθίζετε να ακολουθεί το pascal case τρόπο γραφής, δηλαδή κάθε πρώτο γράμμα μια λέξης να είναι κεφαλαίο συμπεριλαμβανομένης και της πρώτης λέξης και οι λέξεις να μη χωρίζονται με κενά ή άλλα σημεία στίξης πχ “ActionFetchProfessor), στη συνέχεια στη μέθοδο run δηλώνουμε το όνομα του action όπως αυτό έχει δηλωθεί και στο αρχείο “domain.yml”, εδώ συνηθίζεται να ακολουθούμε το snake case τρόπο γραφής, δηλαδή να υπάρχει μια κάτω παύλα ανάμεσα στις λέξεις πχ “action_fetch_professor”, όπως φαίνεται και από το παράδειγμα τα ονόματα που διαλέγουμε τείνουν να είναι τα ίδια και το μόνο που τα διαφοροποιεί είναι ο τρόπος γραφής τους. Στη συνέχεια δημιουργούμε τη μέθοδο run η οποία όπως είπαμε και παραπάνω εμπεριέχει το κύριο κομμάτι κώδικα για το action που θέλουμε να δημιουργήσουμε, στη συγκεκριμένη περίπτωση βλέπουμε ότι γίνεται μια σύνδεση στη διαθέσιμη βάση δεδομένων, κάνουμε μια αναζήτηση για τα στοιχεία του επιβλέποντα καθηγητή και στο τέλος στέλνουμε το αποτέλεσμα στο χρήστη, δημιουργώντας το κείμενο που θα σταλθεί μέσω του Rasa Open Source.

Θεωρείτε αναγκαία η βασική γνώση προγραμματισμού με τη γλώσσα Python για να μπορέσουμε να δημιουργήσουμε κάποιο action.

```python
class ActionFetchProfessor(Action):

    def name(self) -> Text:
        return "action_fetch_professor"

    def run(self,
            dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        db = connect_to_db()
        cursor = db.cursor()

        thesis_id = tracker.get_slot('thesis_id')

        cursor.execute(
            f'SELECT * FROM thesis INNER JOIN professors ON thesis.professorId =
            professors.id WHERE thesis.id = {thesis_id};')
        thesis = cursor.fetchone()

        close_connection(db)

        dispatcher.utter_message(
            text=f"Επικοινώνησε με τον/την {thesis[5]} στο email {thesis[6]} για
            το θέμα: {thesis[1]}"
        )

        # Reset all slots so that the conversation can start over
        return [AllSlotsReset()]
```

Εικόνα 29 Παράδειγμα δημιουργίας action

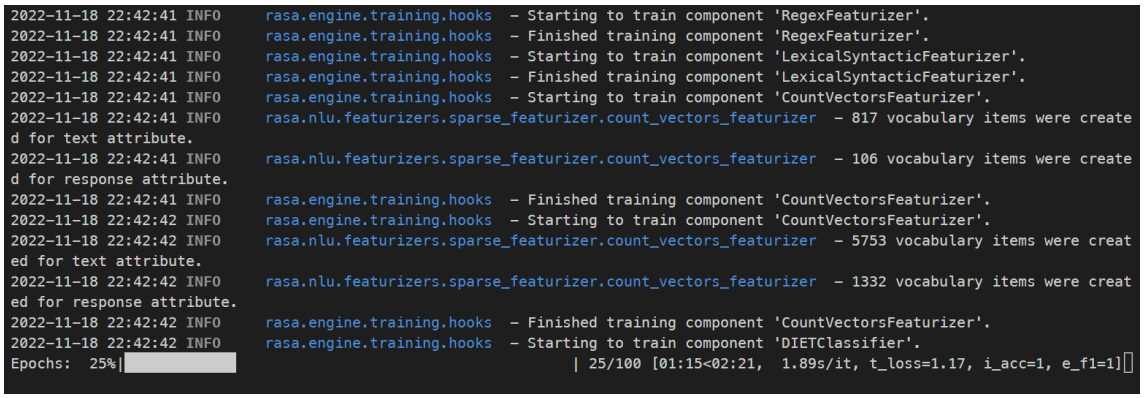
5.3 Εκπαίδευση μοντέλου

Για να προχωρήσουμε σε εκπαίδευση του μοντέλου που έχουμε δημιουργήσει, θα πρέπει να είμαστε στον αρχικό φάκελο της εφαρμογής, στο ίδιο επίπεδο με το φάκελο που περιέχει το τοπικό

εικονικό περιβάλλον που έχουμε δημιουργήσει, εκεί ανοίγοντας τη γραμμή εντολών, ή και το PowerShell για τα Windows, εκτελούμε αρχικά την εντολή “./venv/bin/activate” για να ενεργοποιήσουμε το εικονικό περιβάλλον αν δε το έχουμε ήδη κάνει σε προηγούμενο βήμα, και στη συνέχεια εκτελούμε την εντολή “rasa train”, αυτή η εντολή θεωρείτε αρκετά χρονοβόρα γιατί χρειάζεται συνήθως αρκετούς υπολογιστικούς πόρους για την εκτέλεση της, βέβαια αυτό εξαρτάται από τις ρυθμίσεις που έχουμε συμπεριλάβει στο αρχείο “config.yml” στο οποίο ενημερώνουμε το Rasa Open Source ποιες διαδικασίες θα εκτελέσει και με ποιον τρόπο θα κάνει το train.

Η διαδικασία της εκπαίδευσης νέου μοντέλου, θα πρέπει να επαναλαμβάνεται κάθε φορά που κάνουμε κάποια αλλαγή στο μοντέλο μας, δηλαδή προσθέτουμε ή αλλάζουμε κάποιο intent, story, rule και ούτω καθεξής. Επίσης, μετά το πέρας της εκπαίδευσης του μοντέλου ενημερωνόμαστε στη γραμμή εντολών, με το όνομα του νέου εκπαιδευμένου μοντέλου και το μέρος στο οποίο έχει αποθηκευτεί.

Μια άλλη λειτουργία που μας παρέχει για την εκπαίδευση του μοντέλου είναι η εντολή “rasa interactive” η συγκεκριμένη εντολή είναι αρκετά βοηθητική κατά τη διάρκεια της ανάπτυξης της εφαρμογής, αφού όταν τελειώσει την εκπαίδευση του μοντέλου ανοίγει στη γραμμή εντολών ένα κανάλι επικοινωνίας με το ψηφιακό βοηθό, στο οποίο εκτός από την απλή επικοινωνία μας δείχνει και τις κινήσεις που κάνει το Rasa Open Source βάση αυτών που γράφει ο χρήστης και τις πληροφορίες που του δίνει.



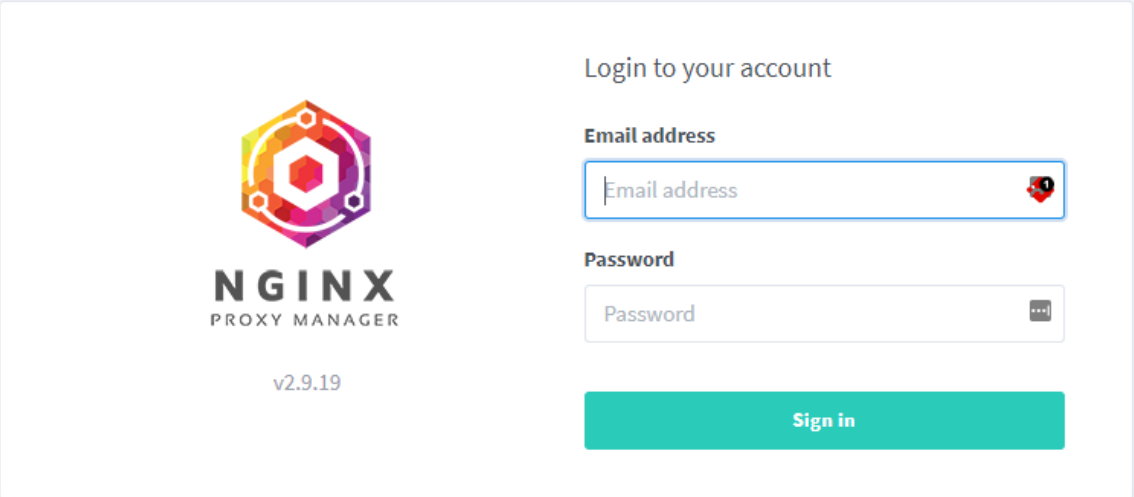
Εικόνα 30 Παράδειγμα εκπαίδευσης μοντέλου

5.4 Χρήση Εφαρμογής

Αφότου ολοκληρωθεί η εκπαίδευση του νέου μοντέλου μπορούμε να διαθέσουμε την εφαρμογή μας στα διαθέσιμα κανάλια, αν θέλουμε να το τρέξουμε τοπικά μπορούμε να εκτελέσουμε την εντολή “rasa run actions” η οποία θα τρέξει ένα server που θα είναι υπεύθυνος για τα actions που έχουμε δημιουργήσει και μετά σε ένα νέο παράθυρο της γραμμής εντολών να εκτελέσουμε την εντολή “rasa shell” η οποία θα μας επιτρέψει να επικοινωνήσουμε με το ψηφιακό βοηθό μέσω της γραμμής εντολών.

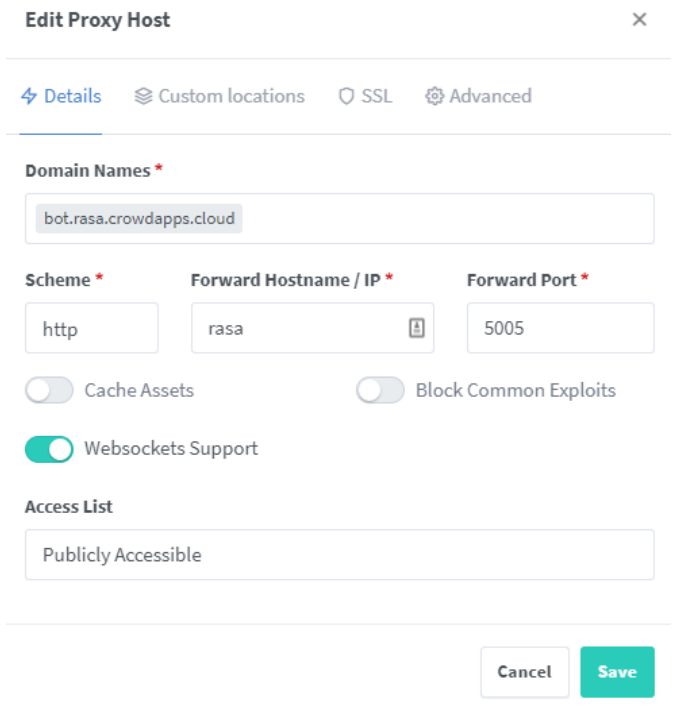
Αν θέλουμε να διαθέσουμε το ψηφιακό βοηθό στο κοινό όμως θα πρέπει να κάνουμε και κάποιες άλλες ενέργειες, αρχικά θα πρέπει να είμαστε στον αρχικό φάκελο που περιέχει το “docker-compose.yml” αρχείο, το οποίο είναι υπεύθυνο και για τις άλλες υπηρεσίες που είναι απαραίτητες για τη πλήρη διάθεση του ψηφιακού βοηθού στο κοινό, όταν είμαστε σε αυτό το σημείο εκτελούμε την εντολή “docker compose up -d” αυτή η εντολή θα δημιουργήσει έναν nginx web server, που είναι υπεύθυνος να προωθήσει τις κλήσεις που θα κάνει ο εκάστοτε χρήστης της εφαρμογής προς την εφαρμογή, τη βάση δεδομένων “tracker store” , τον Rasa Action Server καθώς και το ίδιο το Rasa Open Source το οποίο θα ξεκινήσει με το νέο μοντέλο που έχουμε εκπαιδεύσει. Στη συνέχεια θα πρέπει να έχουμε στη διάθεσή μας κάποιο δηλωμένο όνομα χώρου στο διαδίκτυο(domain name), συγκεκριμένα θα χρειαστούμε ένα κεντρικό π.χ. example.com στο οποίο θα έχουμε την ευχέρεια να δηλώσουμε υπό-ονόματα(sub-domains), πχ nlu.example.com. Αφού έχουμε στη διάθεσή μας αυτό το όνομα θα πρέπει να κάνουμε είσοδο στο πακέτο του nginx που χρησιμοποιούμε το οποίο θα είναι η IP του μηχανήματος μας στη πόρτα 81, εκεί θα μας ζητήσει να κάνουμε είσοδο με τα στοιχεία μας, αν είναι η πρώτη φορά μπορούμε να κάνουμε

είσοδο με τα στοιχεία email: “admin@example.com” και για κωδικό το “changeme” όπως φαίνεται στη παρακάτω εικόνα, αμέσως μετά τη πρώτη είσοδο θα μας ζητηθεί η αλλαγή του κωδικού.



Εικόνα 31 Είσοδος στο Nginx Proxy Manager

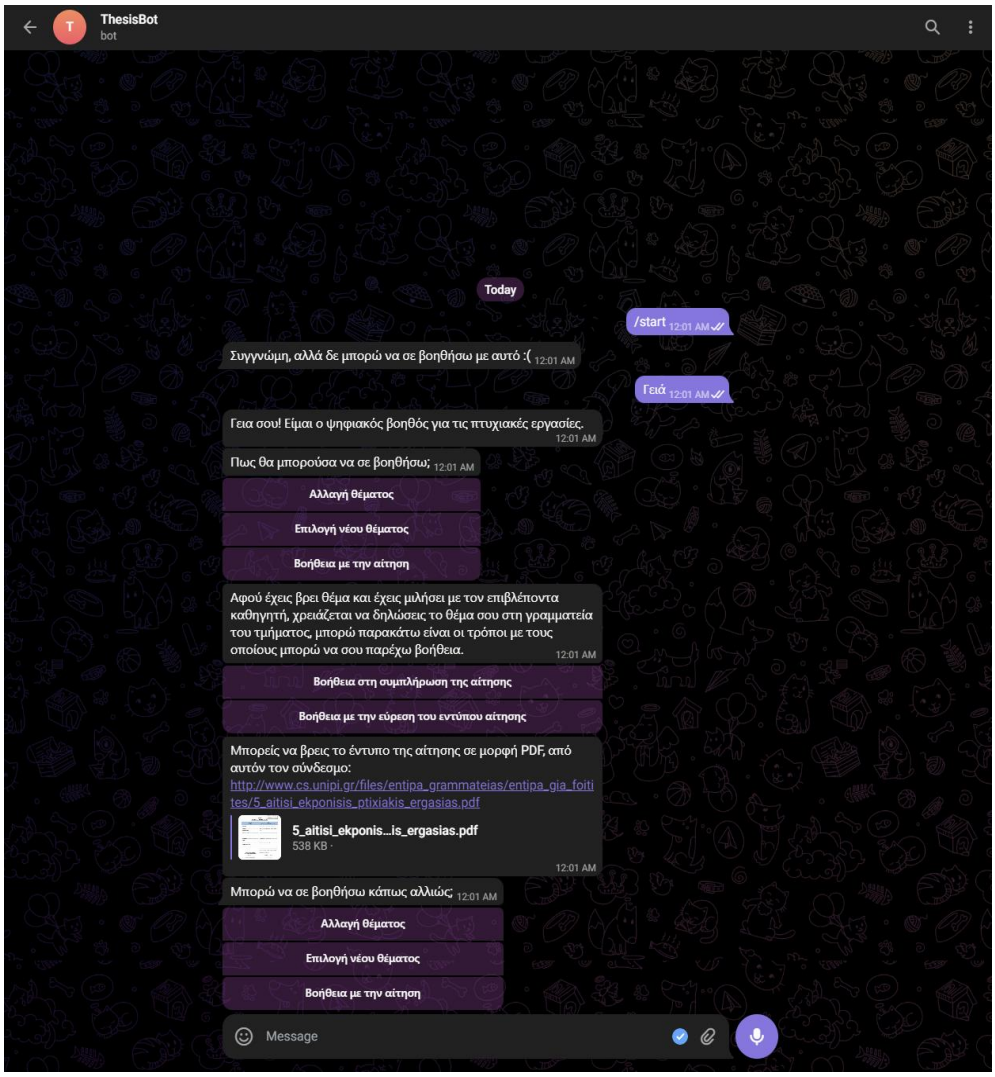
Στη συνέχεια επιλέγουμε την εισαγωγή νέου Proxy Host (Add Proxy Host) και συμπληρώνουμε τα απαραίτητα στοιχεία, το όνομα που έχουμε θέσει στα κανάλια επικοινωνίας ότι θα έχει το bot μας και επειδή η εφαρμογή χρησιμοποιεί το Docker Engine δε χρειάζεται να δηλώσουμε κάποια IP παρά μόνο το όνομα που έχει δοθεί στο υποσύστημα στο “docker-compose.yml" αρχείο και τη πόρτα, όπως φαίνεται στη παρακάτω εικόνα



Εικόνα 32 Παράδειγμα προσθήκης proxy host

Στη συνέχεια μπορούμε να προσθέσουμε με αυτόματο τρόπο ένα SSL πιστοποιητικό για να μπορεί η εφαρμογή μας να χρησιμοποιεί το πρωτόκολλο HTTPS που είναι αναγκαίο αν θέλουμε να χρησιμοποιήσουμε κάποιο κανάλι επικοινωνίας όπως το Facebook Messenger, Telegram και άλλα. Αφότου ολοκληρώσουμε και αυτή τη διαδικασία ο ψηφιακός βοηθός είναι πλέον διαθέσιμος

από το κανάλι που έχουμε ορίσει, όπως μπορούμε να δούμε με το παράδειγμα τις παρακάτω εικόνας.

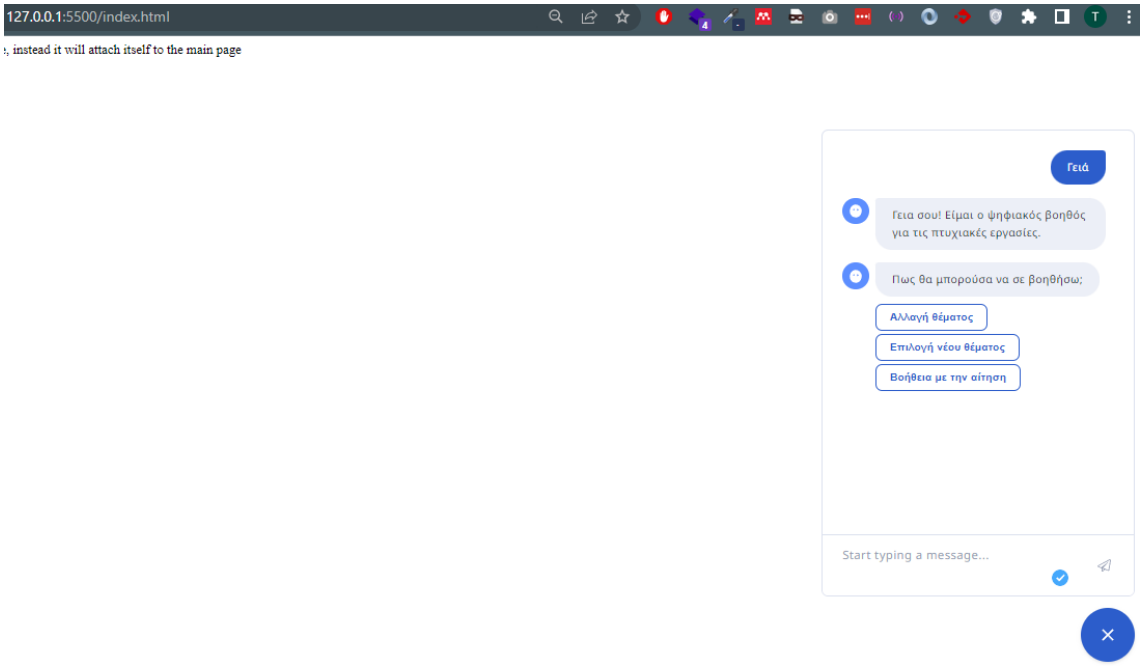


Εικόνα 33 Παράδειγμα συνομιλίας με τον ψηφιακό βοηθό

Θα μπορούσαμε να αποφύγουμε το βήμα του Nginx Proxy Manager αν δε χρειαζόμασταν κάποιο κανάλι που μας αναγκάζει να κάνουμε την επικοινωνία μαζί του μέσω πρωτοκόλλου HTTPS και μπορούμε να δουλέψουμε με την υπηρεσία Socket Io στη διεύθυνση “http://IP:port/socketio” ή το Rest API που παρέχεται. Αν για παράδειγμα έχουμε ενεργοποιήσει το Rest API τότε μπορούμε να επικοινωνήσουμε με τον ψηφιακό βοηθό μέσω της διεύθυνσης “http://IP:port/webhooks/rest/webhook” στέλνοντας μια κλήση όπως αυτή της παρακάτω εικόνας.

```
curl --location --request POST 'http://
    ip:5005/webhooks/rest/webhook' \
--header 'Content-Type: application/json' \
--data-raw '{
    "sender": "test",
    "message": "γεια"
}'
```

Εικόνα 34 Παράδειγμα κλήσης Rest API



Εικόνα 35 Παράδειγμα επικοινωνίας με Rasa Chat Widget και Socket IO

6 Συμπεράσματα και μελλοντική εξέλιξη

6.1 Συμπεράσματα

Μέσα από την εκπόνηση της παρούσας εργασίας, έγινε εμφανές το πόσο σημαντικό είναι η διάθεση του συνόλου της πληροφορίας από ένα μέρος και της σωστής πληροφόρησης, ακόμα και για ένα απλό σχετικά θέμα, όπως αυτό της διαδικασίας ανάθεσης πτυχιακής εργασίας. Παρατηρήσαμε ότι η πληροφορία για τα θέματα πτυχιακής εργασίας πολλές φορές είναι διάσπαρτη σε διαφορετικούς ιστότοπους, ακόμα και όταν πρόκειται και για ένα τμήμα, πρόγραμμα σπουδών ή ακόμα μπορεί και να μην είναι και προσβάσιμη, μέχρις ότου ένας φοιτητής να έρθει σε επικοινωνία με ένα καθηγητή του τμήματος. Επίσης, παρατηρήσαμε ότι όπου είναι διαθέσιμα τα θέματα παρουσιάζονται ως σύνολο στον εκάστοτε φοιτητή, συνεπώς δεν υπάρχει μια εξατομικευμένη πληροφόρηση ως προς το αντικείμενο που θα φαινόταν ενδιαφέρον σε αυτόν.

Ο στόχος της εργασίας ήταν να προσφέρει ένα τρόπο εξατομίκευσης της πληροφορίας σχετικά με τα θέματα πτυχιακής εργασίας, αλλά και να παρέχει επιπλέον πληροφόρηση σχετικά με τις διαδικασίες που θα πρέπει να ακολουθήσει ένας φοιτητής για την τελική καταχώρηση του θέματος του, μέσω της δημιουργίας μιας εφαρμογής chatbot, ή αλλιώς ψηφιακού βοηθού. Η εφαρμογή μας μέσα από το διάλογο και τα στοιχεία που λαμβάνει από το χρήστη, μπόρεσε να παρέχει εξατομικευμένη πληροφόρηση ως προς τα θέματα που υπάρχουν διαθέσιμα λαμβάνοντας υπόψιν την επιθυμία του χρήστη σε κάποιο μάθημα επιλογής του.

Για την υλοποίηση της εφαρμογής χρειάστηκε να γίνει η μελέτη και η κατανόηση του κανονισμού εκπόνησης πτυχιακής εργασίας του πανεπιστημίου και πιο συγκεκριμένα της διαδικασίας ανάθεσης θέματος πτυχιακής εργασίας.

Διαπιστώθηκε στη συνέχεια, ότι θα πρέπει να δημιουργηθεί μια βάσης δεδομένων, που θα μπορούσε να κρατήσει τις σχετικές πληροφορίες για τα μαθήματα του τμήματος, τους καθηγητές αλλά και τα θέματα που αυτοί θα μπορούσαν να είχαν διαθέσει. Πρέπει να σημειώσουμε ότι η μορφή του τρόπου παρουσίασης των δεδομένων είναι αρκετά απλή και η επιλογή της έγινε, για να μπορέσει να λειτουργήσει η εφαρμογή αποτελεσματικά, αλλά με ικανότητα εύκολης επέκτασης των διαθέσιμων πληροφοριών που θα μπορούσε να δεχτεί.

Με τη χρήση του λογισμικού Rasa Open Source, δημιουργήσαμε το κεντρικό υποσύστημα της εφαρμογής μας, το οποίο κάλυπτε όλες τις απαιτήσεις που θα μπορούσαμε να έχουμε όπως η υποστήριξη της Ελληνικής γλώσσας και η εύκολη εκμάθηση. Δημιουργήθηκαν δυο κανάλια επικοινωνίας χρήστη και εφαρμογής, με χρήση της πλατφόρμας Telegram, αλλά και μέσω σελίδας ιστού με τη χρήση του widget που παρέχει το ίδιο το Rasa.

Ένα από τα προβλήματα που κληθήκαμε να αντιμετωπίσουμε κατά την υλοποίηση ήταν η διάθεση της εφαρμογής στο κοινό μέσω των διαθέσιμων καναλιών. Αρχικά τα περισσότερα διαθέσιμα κανάλια που προσφέρονται, χρειάζονται μια μακροσκελή διαδικασία, δημιουργίας μικρό εφαρμογής σε δικό τους ιστότοπο και πολλές φορές αυτό μπορεί να χρειαστεί να λάβει και κάποιου είδους έγκριση, η οποία μπορεί να είναι αρκετά χρονοβόρα. Τα κανάλια που επιλέχθηκαν ήταν αυτά που χρειάστηκαν τη μικρότερη διαδικασία ρυθμίσεων, με ευκολότερο να είναι το widget που προσφέρετε από το Rasa για προσθήκη σε δική μας ιστοσελίδα.

Τέλος, λαμβάνοντας υπόψιν και την αξιολόγηση της εφαρμογής, παρατηρήσαμε ότι η εφαρμογή θεωρήθηκε χρήσιμη και επί το πλείστων μπόρεσε να καθοδηγήσει τον εκάστοτε χρήστη στη λήψη εξατομικευμένης πληροφόρησης για την επιλογή θέματος πτυχιακής εργασίας, να λάβει καθοδήγηση για τη συμπλήρωση της δήλωσης που θα πρέπει να κάνει στη γραμματεία αλλά και γενικευμένη πληροφόρηση στο θέμα των πτυχιακών εργασιών.

6.2 Μελλοντική εξέλιξη εφαρμογής

Ως πρόταση μελλοντικής εξέλιξης της εφαρμογής θα μπορούσε να είναι η παροχή περισσότερων κριτηρίων από το χρήστη πέρα από το επιθυμητό μάθημα. Για να επιτευχθεί με αποτελεσματικότητα όμως κάτι τέτοιο, θα χρειαστεί να υπάρξει ένα κοινό αποθετήριο θεμάτων, που θα μπορεί να συλλέγει όλα τα διαθέσιμα θέματα από τους καθηγητές, τα οποία να έχουν τα

σχετικά κριτήρια, που θα πρέπει να μπορεί να καλύψει ο φοιτητής, για την επιλογή του θέματος. Έτσι ο εκάστοτε φοιτητής θα μπορούσε να απαντήσει σε αρκετά περισσότερες ερωτήσεις μέσω του ψηφιακού βοηθού και να αποκτήσει μια πολύ πιο προσωποποιημένη πληροφόρηση σχετικά με το ιδανικό για αυτόν θέμα πτυχιακής εργασίας.

Βέβαια, υπάρχουν και άλλου είδους εφαρμογές στις οποίες μπορεί να χρησιμοποιηθεί ένας ψηφιακός βοηθός με αποτελεσματικότητα, όπως η εφαρμογής που αναλύεται στο 2020 11th International Conference on Information, Intelligence, Systems and Applications με τίτλο “HappyCruise: An architecture for Personalized Secure Boarding on Cruises” η οποία στοχεύει στην ανάπτυξη λογισμικού για τη καθοδήγηση επιβατών αλλά και μελών του πληρώματος στην επιβίβαση τους σε ένα πλοίο. Ένας ψηφιακός βοηθός θα μπορούσε γνωρίζοντας τον επιβάτη να τον καθοδηγήσει αποτελεσματικά σε σχέση με τα απαραίτητα έγγραφα που θα έπρεπε να έχει στη διάθεσή του και να τον βοηθήσει για παράδειγμα με τα πράγματα που θα μπορούσε να φέρει , αλλά και να του παρέχει και άλλες πληροφορίες σχετικές με την επιβίβαση του όπως και να του στείλει και κάποια υπενθύμιση αν αυτός το επιθυμεί. Ο ψηφιακός βοηθός γνωρίζοντας από πριν τη διαδικασία που πρέπει να ακολουθηθεί θα μπορούσε να βελτιώσει και διευκολύνει τη διαδικασία που εκτελείτε και από τα μέλη του πληρώματος δίνοντας οδηγίες για τις διαδικασίες που πρέπει να εκτελέσει το κάθε μέλος(Viennas et al., 2020).

Ακόμα εφαρμογές όπως αυτές της διαχείρισης ταυτοτήτων και πορτοφολιών θα μπορούσαν να χρησιμοποιηθούν σε συνδυασμό με τη τεχνολογία των chatbot για την ταυτοποίηση του χρήστη αλλά και για τη άντληση απαραίτητων πληροφοριών από το ψηφιακό βοηθό σε σχέση με τον χρήστη αυτόν. Μια τέτοια εφαρμογή που στοχεύει στην άντληση μόνο των απαραίτητων πληροφοριών για ένα χρήστη ώστε να ληφθεί μια απόφαση ή να δοθεί πληροφόρηση στο χρήστη αποτελεί η αρχιτεκτονική OLYMPUS Architecture(Moreno et al., 2020). Ένα άλλο σχετικό παράδειγμα αυτού θα μπορούσε να είναι η εφαρμογή που υλοποιήθηκε στα πλαίσια του 2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA) “ Mobile student information services: A case study in Greek Open University” η οποία ως σκοπό είχε τη δημιουργία ενός συστήματος ταυτοποίησης μαθητών και διάθεσης υλικού σχετικού με τη πληροφόρηση ενός φοιτητή για το πανεπιστήμιο του (Kastanas & Sakkopoulos, 2022).

Με την επίτευξη της διάθεσης περισσότερων στοιχείων, θα μπορούσαμε να κάνουμε και τη μετάβαση από έναν ψηφιακό βοηθό που βασίζεται σε στοιχεία που δίνει ο χρήστης σχετικά με την υπηρεσία που θέλει να λάβει πληροφόρηση, σε πλήρως προσωποποιημένους ψηφιακούς βοηθούς. Ως τώρα η εξατομίκευση της πληροφορίας μπορεί να επιτευχθεί μέσω της συλλογής πληροφοριών από τον χρήστη, για την υπηρεσία που αναζητάει τώρα. Το επόμενο βήμα θα είναι ένας πλήρως εξατομικευμένος βοηθός, ο οποίος μέσω της συνομιλίας με το χρήστη θα μπορεί να τον μάθει καλύτερα με το χρόνο, να μάθει πότε θα μπορεί να έρθει σε επικοινωνία με το χρήστη, θα μπορεί να μάθει και να θυμάται τις επιθυμίες του και έτσι να μπορέσει να προσφέρει μια πλήρως προσωποποιημένη διεπαφή. Όπως μας λένε και από το Rasa πιστεύουν ότι υπάρχουν πέντε στάδια έξυπνων βοηθών(Nichol, 2018), με αυτά να είναι:

- Επίπεδο 1: Βοηθοί ειδοποιήσεων
- Επίπεδο 2: Βοηθοί συχνών ερωτήσεων
- Επίπεδο 3: Βοηθοί με βάση τα συμφραζόμενα
- Επίπεδο 4: Εξατομικευμένοι Βοηθοί
- Επίπεδο 5: Αυτόνομη Οργάνωση Βοηθών

Τώρα, με τη χρήση του CDD, φαίνεται να διανύουμε το τρίτο επίπεδο, αλλά υπάρχει πολύ μεγάλος χώρος για εξέλιξη και πολλά πράγματα που πρέπει να γίνουν για να κατορθωθεί η εξέλιξη αυτή (Nichol, 2020).

Βιβλιογραφία

Adamopoulou, E., & Moussiades, L. (2020). Chatbots: History, technology, and applications. *Machine Learning with Applications*, *2*, 100006. https://doi.org/10.1016/J.MLWA.2020.100006

Bohl, Marilyn., & Rynn, M. (2008). *Tools for structured and object-oriented design : an introduction to programming logic*. Pearson Prentice Hall.

Brandtzaeg, P. B., & Følstad, A. (n.d.). *Why people use chatbots*. https://doi.org/10.1007/978-3-319-70284-1_30

Conversation-Driven Development. (2022). https://rasa.com/docs/rasa/conversation-driven-development/

Dialogflow Documentation | Google Cloud. (n.d.). Retrieved November 28, 2022, from https://cloud.google.com/dialogflow/docs/

Domain. (2022). https://rasa.com/docs/rasa/domain/

Goldenberg, D., Albert, J., Mizrachi, S., Horowitz, A., Teinemaa, I., & Kofman, K. (2021). *Personalization in Practice: Methods and Applications 1 BACKGROUND*. 1123. https://doi.org/10.1145/3437963.3441657

Hasan Ibrahim, M., Sayagh, M., & Hassan, A. E. (n.d.). *A Study of How Docker Compose is Used to Compose Multi-component Systems*. Retrieved November 22, 2022, from https://github.com/SAILResearch/replication-21-ibrahim-dockercompose

Introduction to Rasa Open Source & Rasa Pro. (2022). https://rasa.com/docs/rasa/#rasa-open-source

Karatas, G. (2022, January 7). *Intent Classification: What it is and How it Works?* https://research.aimultiple.com/intent-classification/

Kastanas, G., & Sakkopoulos, E. (2022). Mobile student information services: A case study in Greek Open University. *13th International Conference on Information, Intelligence, Systems and Applications, IISA 2022*. https://doi.org/10.1109/IISA56318.2022.9904343

Khanna, A., Pandey, B., Vashishta, K., Kalia, K., Pradeepkumar, B., & Das, T. (2015). A Study of Today's A.I. through Chatbots and Rediscovery of Machine Intelligence. *International Journal of U-and e-Service*, *8*(7), 277–284. https://doi.org/10.14257/ijunesst.2015.8.7.28

Model Configuration. (2022). https://rasa.com/docs/rasa/model-configuration/

Moreno, R. T., Bernabe, J. B., Rodríguez, J. G., Frederiksen, T. K., Stausholm, M., Martínez, N., Sakkopoulos, E., Ponte, N., & Skarmeta, A. (2020). The OLYMPUS Architecture—Oblivious Identity Management for Private User-Friendly Services. *Sensors 2020, Vol. 20, Page 945*, *20*(3), 945. https://doi.org/10.3390/S20030945

Nginx Proxy Manager. (n.d.). Retrieved November 22, 2022, from https://nginxproxymanager.com/guide/

Nichol, A. (2018, August 20). *The next generation of AI assistants in enterprise – O'Reilly*. https://www.oreilly.com/radar/the-next-generation-of-ai-assistants-in-enterprise/

Nichol, A. (2020, June 17). *5 Levels of Conversational AI - 2020 Update | The Rasa Blog | Rasa*. https://rasa.com/blog/5-levels-of-conversational-ai-2020-update/

Omg. (2009). *An OMG ® Unified Modeling Language ® Publication OMG ® Unified Modeling Language ® (OMG UML ®) OMG Document Number: Date*. https://www.omg.org/spec/UML/20161101/PrimitiveTypes.xmi

Policies. (2022). https://rasa.com/docs/rasa/policies/

Rad, B. B., Bhatti, H. J., & Ahmadi, M. (2017). An Introduction to Docker and Analysis of its Performance Metamorphic Malware CLassification using MLP Neural Network View project Cloud Computing View project An Introduction to Docker and Analysis of its Performance. *IJCSNS International Journal of Computer Science and Network Security*, *17*(3). https://www.researchgate.net/publication/318816158

Setting up your environment. (2022). https://rasa.com/docs/rasa/installation/environment-set-up/

Shawar, B. A., & Atwell, E. (n.d.). *Chatbots: Are they Really Useful Chatbots: Are they Really Useful?*

Singh, S. (2019, July 18). *What is Tokenization | Methods to Perform Tokenization*. https://www.analyticsvidhya.com/blog/2019/07/how-get-started-nlp-6-unique-ways-perform-tokenization/

Slots. (2022). https://rasa.com/docs/rasa/domain/#slots

Stories. (2022). https://rasa.com/docs/rasa/stories/

Tracker Stores. (2022). https://rasa.com/docs/rasa/tracker-stores/

Viennas, E., Ioannou, Z. M., Pavlidis, G., Tzimas, G., & Sakkopoulos, E. (2020). HappyCruise: An architecture for Personalized Secure Boarding on Cruises. *11th International Conference on Information, Intelligence, Systems and Applications, IISA 2020*. https://doi.org/10.1109/IISA50023.2020.9284375

Warmerdam, V. (2021, March 29). *Intents & Entities: Understanding the Rasa NLU Pipeline | The Rasa Blog | Rasa*. https://rasa.com/blog/intents-entities-understanding-the-rasa-nlu-pipeline/

What is Botpress? | Botpress Documentation. (n.d.). Retrieved November 28, 2022, from https://v12.botpress.com/overview/what-is-botpress

What is CI/CD? (2022). https://www.redhat.com/en/topics/devops/what-is-ci-cd

What is the Bot Framework SDK? - Bot Service | Microsoft Learn. (2022, November 16). https://learn.microsoft.com/en-us/azure/bot-service/bot-service-overview?view=azure-bot-service-4.0

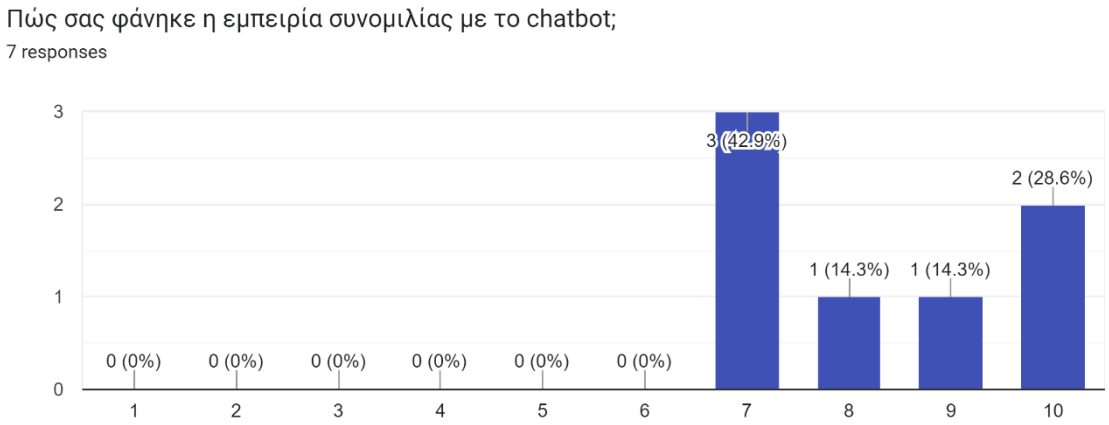
Wochinger, T. (2019, February 21). *Rasa NLU in Depth: Intent Classification | The Rasa Blog | Rasa*. https://rasa.com/blog/rasa-nlu-in-depth-part-1-intent-classification/

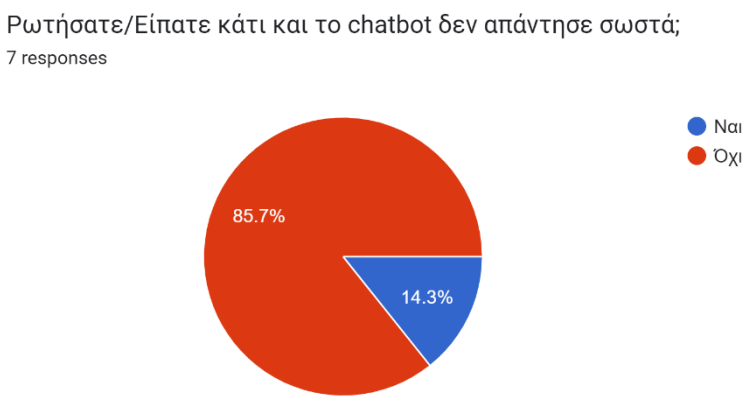
Yassar. (2019, November 4). *Featurization. What is Featurization?* . Medium. https://medium.com/@aliyaser78691/featurization-f63be523644

Your Own Website. (2022). https://rasa.com/docs/rasa/connectors/your-own-website/#chat-widget

Παράρτημα Ι: Αποτελέσματα ερωτηματολογίου «Φόρμα αξιολόγησης Chatbot»

Στη πρώτη ερώτηση η βαθμολόγηση ξεκινάει με 0 και τελειώνει στο 10, με το 0 να σημαίνει πολύ κακή εμπειρία, και 10 άριστη.





Ποιες ήταν οι ερωτήσεις που κάνατε και κρίνεται ό,τι απαντήθηκαν λάθος;

1 response

δεν μου ήρθε μήνυμα χαιρετισμού, δεν απαντάει σωστά στο αν έχω δηλώσει πτυχιακή όταν του πω οχι

Ποιες ήταν οι απαντήσεις του chatbot στις ερωτήσεις σας;

1 response

την 1η φορά μου εμφάνισε ξανά τις ίδιες επιλογές και την 2η φορά δεν με κατάλαβε

Τι κρίνεται ότι θα έπρεπε να απαντήσει;

1 response

να πει την διαδικασία ή να του προτείνει μερικές εναλλακτικές

Θέλετε να προτείνεται κάποια βελτίωση που θα μπορούσε να γίνει στο chatbot;

6 responses

Θα προτιμούσα να μου λεει τις απαντήσεις που στέλνω και όχι το "slug" "key" της απάντησης.

Θα προτιμούσα να καταλαβαίνει πχ την έκφραση θεμα για εργασία κάπως με τη λέξη "θεμα" και να σου προτείνει και εκεί αν θές νέο θέμα η αλλαγή

Στο telegram να αναγνωρίζει το /start message και να σου λεει καλωσηρθες γιατί δεν το αναγνωρίζει το μήνυμα και πετάει μήνυμα οτι δε το ξέρει.

Όταν ανοίγει η σελίδα να είναι ήδη ανοιχτό ή ν εμφανίζει ένα μήνυμα πατά το chatbot πχ

να γίνονται hide οι επιλογές μου

Θα ήταν ιδανικό να μπορεί να μάθει περισσότερα για ένα χρήστη προτού του δείξει θέματα