This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 822215

REBUILD
ICT-enabled integration facilitator and life rebuilding guidance
Project start date: 01/01/2019 | Duration: 36 months

Deliverable: D4.8 The digital companion prototype

DUE DATE OF THE DELIVERABLE: 30-04-2021
ACTUAL SUBMISSION DATE: 30-06-2021

Project	REBUILD – ICT-enabled integration facilitator and life rebuilding guidance
Call ID	H2020-SC6-MIGRATION-2018-2019-2020 – DT-MIGRATION-06-2018
Work Package	*WP4 – The Digital Companion*
Work Package Leader	*CERTH*
Deliverable Leader	*CERTH*
Deliverable coordinator	Theodoros Semertzidis (*CERTH*) – theosem@iti.gr
Deliverable Nature	Demonstrator
Dissemination level	Public (PU)
Version	1.0
Revision	Final

DOCUMENT INFO

AUTHORS

Author name	Organization	E-Mail
Theodoros Semertzidis Nikolaos Konstantinou Panagiotis Stalidis Michalis Lazaridis	CERTH	theosem@iti.gr nkonstantinou@iti.gr stalidis@iti.gr Michalis.Lazaridis@iti.gr

DOCUMENT HISTORY

Version #	Author name	Date	Changes
0.1	Panagiotis Stalidis (CERTH)	01-03-2021	TOC
0.2	Nikolaos Konstantinou (CERTH)	20-03-2021	Section 1
0.3	Nikolaos Konstantinou (CERTH)	14-05-2021	Section 2
0.4	Panagiotis Stalidis (CERTH)	11-06-2021	Section 3
0.5	Nikolaos Konstantinou (CERTH)	17-06-2021	Section 4
0.6	Michalis Lazaridis (CERTH)	22-06-2021	Executive summary, Section 5
0.7	Theodoros Semertzidis (CERTH)	22-06-2021	First complete draft, integrated with references
0.8	Panagiotis Stalidis (CERTH)	25-06-2021	Integration of peer review suggestions
1.0	Theodoros Semertzidis (CERTH)	29-06-2021	Reviewed and approved version

DOCUMENT DATA

Keywords	*Chatbot, Conversation agent, rule-based, deep learning, AI*
Editor Address data	Name: Theodoros Semertzidis Partner: CERTH Address: 6th km Charilaou Thermi rd, Thessaloniki, Greece Phone: +302310464160 (ext. 130) Email: theosem@iti.gr
Delivery Date	30-06-2021
Peer Review	*Name, Surname, ENG* *Pilar, Orero, UAB*

EXECUTIVE SUMMARY

This deliverable is reporting the work done in REBUILD project's WP4 on the design and implementation of a digital companion. This will complementary guide migrants --upon their arrival at their new society-- to interacting with the public authorities and public or private services. To this aim a rule-based chatbot undertakes the communication between the immigrants and the authorities. The chatbot replies to common questions and provides migrants with the necessary information. Discussions or requests that cannot be handled by the chatbot, can be routed to local service providers' (LSP) operators or translators. Furthermore, an additional agent is provided to help migrants develop language skills by practicing conversations in the host country's language.

The deliverable describes various aspects of the prototype, including architecture, implementation, technologies, interfaces and integration in the REBUILD usage scenarios. After a brief introduction, the first core chapter of the report focuses on the framework and its components. Next, the implementation, technologies and the tools used are presented. Finally, the integration of the framework to the overall REBUILD framework is described.

This deliverable builds upon D4.4. Nevertheless, while D4.4 dealt with the design of the framework, the current deliverable deals with the implementation and description of the prototype. As expected, all design decisions were reviewed and, when necessary, updated.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 822215

TABLE OF CONTENTS

DOCUMENT INFO **2**
  AUTHORS 2
  DOCUMENT HISTORY 2
  DOCUMENT DATA 2
EXECUTIVE SUMMARY **4**
TABLE OF CONTENTS **5**
INDEX OF TABLES **6**
INDEX OF FIGURES 6
INTRODUCTION **8**
APPROACHES AND FRAMEWORKS **9**
  Rule Based Chatbot 9
  Conversational Agent 10
    Constructing Language Models and Datasets for Dialogue Modelling 11
    Using a Translation Framework on the English Conversational Agent 13
IMPLEMENTATION **15**
  Chat Server 16
  Rule Based Chatbot 18
    UI for the Definition of a new Conversation Tree 19
  Recommender Conversational Agent 23
INTEGRATION **28**
  Job Seeking Scenario 28
  Education Scenario 30
  Healthcare Scenario 31
  Social Mentoring Scenario 33
  Legal Support Scenario 34
CONCLUSION **34**
REFERENCES **35**

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 822215

INDEX OF TABLES

Table 1. Information gathered and used for creating matches in the social mentoring scenario. 33

INDEX OF FIGURES

Figure 1. The conversation tree that was created for the health scenario. 9

Figure 2. An example from the PERSONA-CHAT dataset. 11

Figure 3. An example for a Spanish-to-English translation using Transformers library. The translated text in this example is 'Hey, how are you today?' 12

Figure 4. Example of the English Conversational Agent using MarianNMT, a neural machine translation framework for Spanish. 14

Figure 5. The Digital Companion's Architecture 15

Figure 6. Swagger representation of the available endpoints of the Chat Server API. 17

Figure 7. Structure of the expected JSON for the conversation API. 17

Figure 8. Structure of the expected "envelope" JSON for the message API. 17

Figure 9. Structure of the message JSON. 18

Figure 10. Structure of the expected JSON for the history API. 18

Figure 11. Create a new blank diagram. 19

Figure 12. Add a start node. 20

Figure 13. Change the color and add a label. 20

Figure 14. Add properties by clicking "edit data…" 20

Figure 15. Add the property name. 21

Figure 16. Add the caption to display for "english". 21

Figure 17. Add captions for the other languages and an "image" for the pictogram. 21

Figure 18. Add two possible outcomes for this state: the user wants help with something and the user continues the conversation. 22

Figure 19. By dragging from the arrow to a node, a connecting adge can be created. 22

Figure 20. The same as with nodes, properties can be added using the "Edit Data…" option. 22

Figure 21. Add responses for English, Spanish and Greek. 23

Figure 22. Iterative loops can be created by dragging the edge to the originating node. 23

Figure 23. In the third turn of dialogue it is displayed a message and a dropdown form (with cities as options) that asks the user to select a city of interest. 25

Figure 24. Following the selection of the city ("Thessaloniki") from the user, the recommendation message with all available information for the event is displayed. After the recommendation message, the Agent continues the open-domain conversation with the user. 26

Figure 25. Through the dialogue, the conversational agent collects data for the user's preferences and the city of his/her choice. 27

Figure 26. The conversational Agent suggests the social event with the highest cosine similarity score to the user as a personalized recommendation. 27

Figure 27. Conversation tree for the Job Seeking scenario. 28

Figure 28. Example conversation to call the job seeking scenario recommender through the chatbot. 29

Figure 29. Example response of the chatbot with the most relevant job offers. 30

Figure 30. A part of the Conversation tree for the Training Seeking scenario. 31

Figure 31. A part of the Conversation tree for the healthcare scenario. 32

Figure 32. The chatbot is using the conversation tree to gather information to assist the immigrant in completing an appointment. 33

Figure 33. Depending on the needs of the migrant, the chatbot can guide him/her on how to visit a doctor or get access to medicine. 34

1. INTRODUCTION

For the purposes of the REBUILD project, a communication chatbot serves as a Digital Companion for the migrant. The Digital Companion's role is to respond to common questions posed by either migrants or LSPs. The Digital Companion is developed using two types of chatbots: open-domain and rule based chatbots. Open-domain chatbots can talk about general topics and respond appropriately, while rule-based chatbots are focused on a particular knowledge domain. The main distinction between these two types of chatbots is the method of input processing and response generation.

Rule-based chatbots lie in the domain of communication and can provide services such as scheduling a hospital appointment. They are responsible for getting information and passing them on to the user. Rule-based chatbots choose the system response based on a fixed predefined set of rules, based on recognizing the lexical form of the input text without creating any new text answers. The knowledge used in the chatbot is humanly hand-coded and is organized and presented with conversational patterns. Chat-based/conversational chatbots (agents), on the other hand, converse with the user as if they were a human being, and their goal is to correctly respond to the sentence they have been given. The conversational agent generates answers based on current and previous user messages. These kinds of chatbots are more human-like and use machine learning algorithms and deep learning techniques.

The first section of this document focuses on the approaches and frameworks employed in the development of the Digital Companion. It explains the methods used to create the Rule Based Chatbot and Conversational Agent. Following that, the deliverable describes the implementation of both types of chatbots as well as a simple messaging exchange server (Chat Server) that was created by developing three RESTful API endpoints to facilitate all communications. Finally, the report presents the integration of the Digital Companion for all REBUILD project scenarios, including screenshots of conversation trees and the chatbot on the REBUILD mobile application.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 822215

2. APPROACHES AND FRAMEWORKS

2.1. RULE BASED CHATBOT

Rule-based chatbots are also known as conversation-tree bots. As the name implies, they adhere to a set of rules. These rules serve as the foundation for the types of issues that the chatbot is familiar with and can solve. The algorithm for a rule-based chatbot is pretty simple: receive a message, check the set of rules to establish the status of the discussion, extract the answer for the current state, and respond by sending a message back.

Receiving and sending messages in the REBUILD platform are handled by a platform-developed communication layer. Endpoints for sending and retrieving messages are provided by this layer of communication. The messages are delivered in JSON objects, with the fields "text", "language" and "media" containing the message. The "text" field, which is of the string type, is used for the message's text. The "language" field is also a string type. The "media" field contains a list of URLs from which media files relevant to the message can be downloaded. These media can include photographs or videos submitted by the user to pose a question, as well as images and videos sent back by the representative to visually answer a query. Furthermore, because the REBUILD project takes special care to enable communication even for illiterate migrants, attachments are used to deliver pictograms alongside the text portion of the message. The endpoints used are described in greater detail in Section 3.1.

A key issue for the chatbot is to be able to recognize the state that a discussion has reached in order to offer replies in a consistent manner and minimize the irritation that a user may feel when the bot does not understand a message. One critical decision in that direction is to avoid free text replies from the users. As a result, for each of the conversational states, a set of possible messages has been defined from which the user can choose. The system can easily recognize these user responses and provide a stable transition from one state to the next.

For entering and exiting the conversation tree, a variety of specific states have been specified. The "START NODE" state is the first of them. This is the starting point for the conversation tree and the chatbot's first response in each new discussion. The "END NODE" state, on the other hand, denotes the end of a dialogue. This enables the platform to release any temporary resources associated with the chat.

Furthermore, at any point during the interaction, the user can declare that there is a communication difficulty and that a representative is required in order to progress further. The "HANDOVER" state provides this capability. When a conversation is routed to a live representative, the chatbot is notified to enter this state. When the live representative completes the communication, the chatbot can either finish or continue the conversation by receiving a notification from the representative indicating what the chatbot's next state should be. During the conversation with the representative, messages are disregarded from the chatbot. Finally, if the user wishes to participate in a social conversation, the chatbot enters a special state called "SOCIAL", during which the conversation is passed to the Conversational Agent handled by a different service.

As shown in Figure 1, the conversation tree for the health scenario, the "START NODE," designated as START, is a green ellipse, allowing the graph viewer to clearly identify the start of the dialog. The red ellipses are used for the "END NODE" to indicate the end of the dialogue. Forwarding the chat to a human agent, as well as forwarding the chat to the social bot, is indicated by an orange ellipse. The blue boxes reflect the chatbot's inquiries, while the white wavy boxes indicate the user's possible responses.

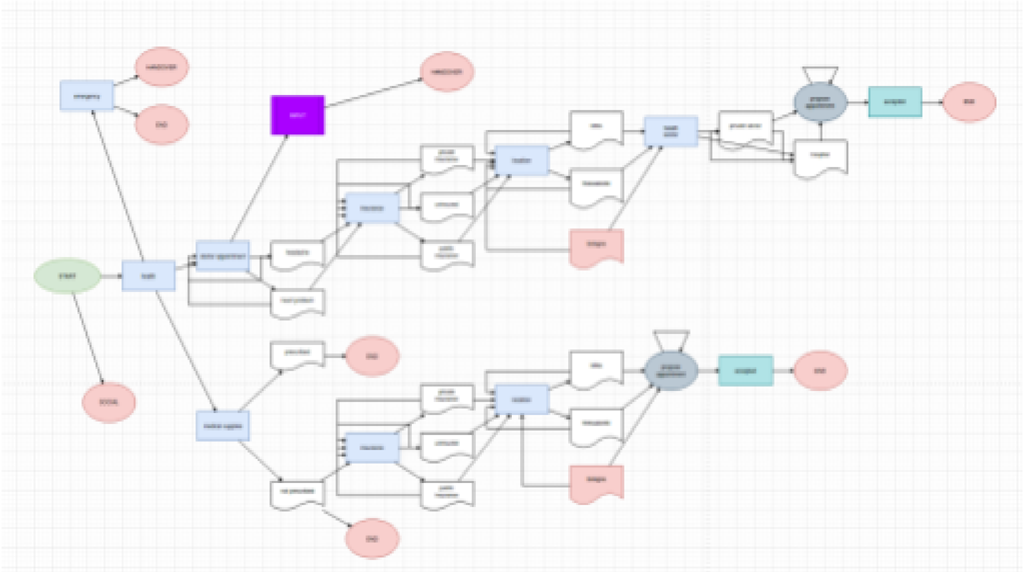


Figure 1. The conversation tree that was created for the health scenario.

2.2 CONVERSATIONAL AGENT

For the purposes of the REBUILD project, a multi-language (English, Greek, Spanish, Italian and Catalan) open-domain Conversational Agent is required. Language Models based on pre-trained transformer architectures, such as BERT *(Devlin et al., 2019)* and GPT-2 *(Radford et al., 2018)*, have significantly improved a wide range of NLP tasks, including chatbots.

Our open-domain chatbot implementation is based on *(Wolf et al. 2019)*, in which the GPT-2 is used as a pre-trained model. The Conversational Agent uses the pre-trained model DIALOGPT (Dialogue Generative Pre-trained Transformer) *(Zhang et al., 2020)*, which has been fine-tuned to the PERSONA-CHAT dataset *(Zhang et al., 2018)*. More information on related work on Open Domain Conversational Agents and the fine-tuning procedure can be found in Deliverable 4.4.

Following the same strategy for all languages, the two main factors required for the development of the conversational agent are: i) pre-trained models as GPT-2 in the remaining languages and ii) appropriate datasets with dialogues for the fine-tuning procedure. Two pre-trained language models for English and Italian are available for use with open sourced-weights, as shown below.

<u>Available pre-trained language models:</u>

- English: DialoGPT, is trained on large-scale dialogue pairs/sessions extracted from Reddit discussion chains. More specifically, the model was trained on 147M multi-turn dialogue from a Reddit discussion thread.
- Italian: The recently open-sourced pre-trained model weights by GePpeTto *(Mattei et al., 2020)* are proposed for Italian. GePpeTto is based on the original GPT-2 and is trained on a large corpus of Italian text drawn from two main sources: Italian Wikipedia and the ItWac *(Baroni et al., 2009)* corpus.

So, in order to implement a conversational agent in the required languages, we tried two main approaches. The first approach was to create language models and datasets in languages where none were available. The second approach, which we chose as the best fit for the REBUILD project, was to use a translation pipeline system for the English DIALOGPT model. The following two sections go into greater detail about both solutions.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 822215

2.2.1 CONSTRUCTING LANGUAGE MODELS AND DATASETS FOR DIALOGUE MODELLING

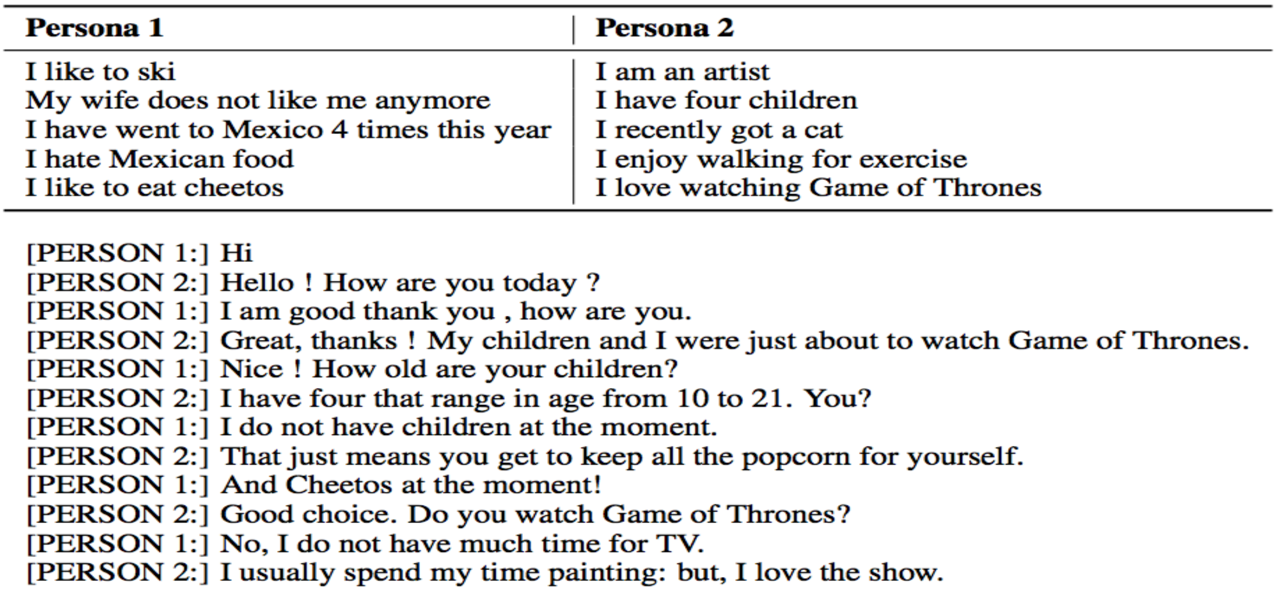
The first step of this approach was the construction of language models, if they were not available. The GPT-2 generative Transformer model is trained from scratch on a large corpus of Greek text so that the model can generate long stretches of contiguous coherent text. The model was based on the *Transformers* library and trained using a collection of nearly 5GB Greek texts, with Greek Wikipedia serving as the primary source. More information about the training of the model can be found in deliverable *D4.4*.

The next step was the creation of appropriate datasets in order to fine-tune the language models in the dialogue task. The first dataset we created consisted of 6369 total dialogues extracted from Greek TV shows and movie subtitles from OpenSubtitles *(https://opus.nlpl.eu/OpenSubtitles-v2018.php)*. OpenSubtitles is a rapidly growing online collection of subtitles for movies and television shows in a variety of languages. Conversational examples are created by using consecutive lines from the subtitle data. There's no guarantee that different lines correspond to different speakers, or that consecutive lines are from the same scene, or even the same show. Despite this, the data contains a plethora of intriguing examples for modeling the mapping from conversational contexts to responses.

Long lines and some text, such as character names and auditory description text, are filtered. The dataset was split to 5866 dialogues for the training and 503 dialogues for the validation set. The Greek GPT-2 model was fine-tuned to this dataset based on the implementation of *(Wolf et al. 2019)*. The model finetunes quickly and 15 epochs of training was sufficient. A batch size of 4 and accumulated gradients over 8 iterations are used, resulting in the effective batch size of 32. In addition, the Adam optimization scheme with a learning rate of 5.7e-5 is used. However, in the end, the interaction with this chatbot was unsatisfying. The chatbot's responses were sometimes irrelevant, and it could only respond to very simple questions. This could be due to the dataset's composition (subtitles from movies and shows).

In order to improve the chatbot's efficiency, we created a Greek synthetic dataset that is identical to the PERSONA-CHAT dataset (Figure 2 shows an example of a dialogue in PERSONA-CHAT dataset), with each dialogue having a personality corresponding to the second speaker. This dataset contains 120 short dialogues with 3-4 sentences as personality for each dialogue. The following are the main rules for creating this dataset:

- Personality phrases are short sentences (maximum 15 words) that can be used to describe a person, such as hobbies, jobs, or different types of preferences, for each dialogue.
- The dialogues are brief, with six turns (12 lines of dialogue) as shown in the figure.
- Each line of dialogue should be no more than 15 words long and both speakers should ask questions to get to know each other.



Persona 1	Persona 2
I like to ski	I am an artist
My wife does not like me anymore	I have four children
I have went to Mexico 4 times this year	I recently got a cat
I hate Mexican food	I enjoy walking for exercise
I like to eat cheetos	I love watching Game of Thrones

[PERSON 1:] Hi
[PERSON 2:] Hello ! How are you today ?
[PERSON 1:] I am good thank you , how are you.
[PERSON 2:] Great, thanks ! My children and I were just about to watch Game of Thrones.
[PERSON 1:] Nice ! How old are your children?
[PERSON 2:] I have four that range in age from 10 to 21. You?
[PERSON 1:] I do not have children at the moment.
[PERSON 2:] That just means you get to keep all the popcorn for yourself.
[PERSON 1:] And Cheetos at the moment!
[PERSON 2:] Good choice. Do you watch Game of Thrones?
[PERSON 1:] No, I do not have much time for TV.
[PERSON 2:] I usually spend my time painting: but, I love the show.

Figure 2. An example from the PERSONA-CHAT dataset.

In this Greek synthetic dataset, we used a data augmentation technique known as Back Translation, which allowed us to double our data to 240 dialogues. Data augmentation is an effective technique for reducing overfitting that involves creating a slightly modified version of the available data. Back Translation is one such NLP augmentation technique that works as follows:

- given a text input in a source language (e.g. English)
- Translate this text into a temporary destination language (e.g. English -> French)
- Translate the previously translated text back into the source language (e.g. French -> English).

Furthermore, we translated the Persona-Chat dataset into Greek using a translation framework described in more detail in the following section and the *Transformers* library. This dataset contains 19319 short dialogues, as shown in Figure 2. In this dataset, we combined the Greek synthetic dataset with the 240 dialogues to provide more information about how a Greek dialogue might look. As a result, the total number of dialogues in the dataset is 19559, with 18521 for training and 1038 for validation.

Using the Greek GPT-2 model weights described in deliverable 4.4, we trained the model on this Greek version of translated Persona-Chat dataset for 3 epochs until there is no progress in validation loss. The model's input is customized to the Greek version of the PERSONA-CHAT dataset to perform the fine-tuning procedure. A two-speaker environment is required for the input representation of the chatbot, in addition to personality sentences like those in Figure 2. To be more specific, for each expression, a sequence of input tokens for the model is generated by combining all of the current speaker persona's sentences with the history of the dialog's previous expressions. Separation tokens (the symbols <BOS> and <EOS> denote the beginning and end of a sentence, respectively) are also used to further separate each expression of the dialog, which is a common strategy in Transformer inputs. Again, a batch size of 4 is used, and gradients are accumulated over 8 iterations, resulting in a total batch size of 32. The Adam optimization scheme is used, with a learning rate of 5.7e-5.

Testing the chatbot trained on this synthetic dataset revealed that it performs significantly better in terms of interaction than the chatbot trained on the Open Subtitles dataset. The responses are more relevant than before, and the "persona" sentences make the dialogue more engaging. However, it still lacks the "depth of knowledge" of the English DIALOGPT, which was pre-trained on much larger datasets.

2.2.2 USING A TRANSLATION FRAMEWORK ON THE ENGLISH CONVERSATIONAL AGENT

The main idea behind this approach for Conversational Agent implementation is to use the pre-trained DIALOGPT model fine-tuned in the PERSONA-CHAT dataset as a base, along with a translation system for all required languages. This solution was chosen for the REBUILD project for two main reasons:

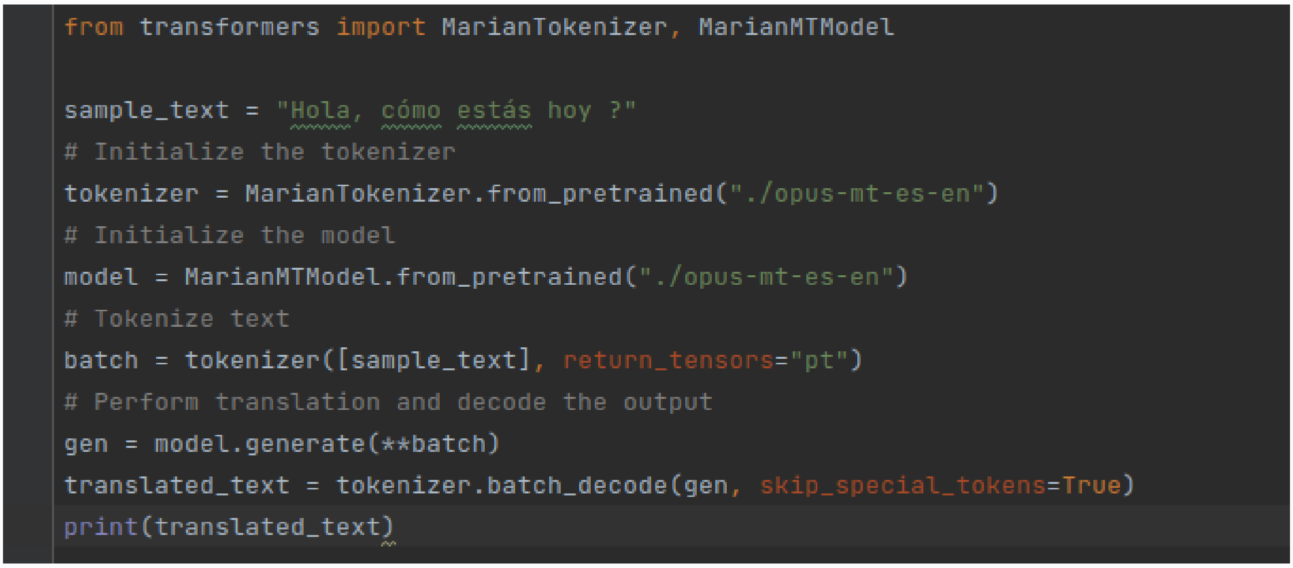
i) Use of a powerful pre-trained model as DIALOGPT that is trained on really large-scale dialogue pairs/sessions.

ii) There is no need to create any additional pre-trained models and datasets.

We used the *HuggingFace Transformers* library to create this translation system. This library makes a large number of pre-trained Transformer-based models available. In addition, it exposes a set of Pipelines that make it very simple to build NLP-based applications. The *HuggingFace* platform includes a repository of pretrained models that have been fine-tuned on a wide range of language tasks (including machine translation) using a variety of Transformer architectures. Filtering for translation models reveals that over 1370 models are available for translation in April 2021. Jörg Tiedemann of the Language Technology Research Group at the University of Helsinki trained the majority of these models. They were trained on the Open Parallel Corpus (OPUS) using MarianNMT, a neural machine translation framework. These models' implementation notes include the following:

- Each model is about 298 MB on disk, there are more than 1,000 models.
- Models were originally trained by Jörg Tiedemann using the Marian C++ library, which supports fast training and translation.
- All models are transformer encoder-decoders with 6 layers in each component. Each model's performance is documented in a model card.
- The list of supported language pairs can be found here https://huggingface.co/Helsinki-NLP .

Using the transformers library to apply a pre-trained model for translation is simple. Figure 4 shows how it works for a simple Spanish-to-English translation. The main steps are: i) Initializing the Tokenizer. For initializing the tokenizer, the Helsinki-NLP pretrained/fine-tuned OpusMT Spanish to English model is used. Textual inputs are converted into tokens using this tokenizer. ii) Initializing the model. In order to generate translations, the same pretrained/fine-tuned model is used. iii) Finally, generate a translation for each element in the batch, decode the batch, and take the first element, which is the translated text.

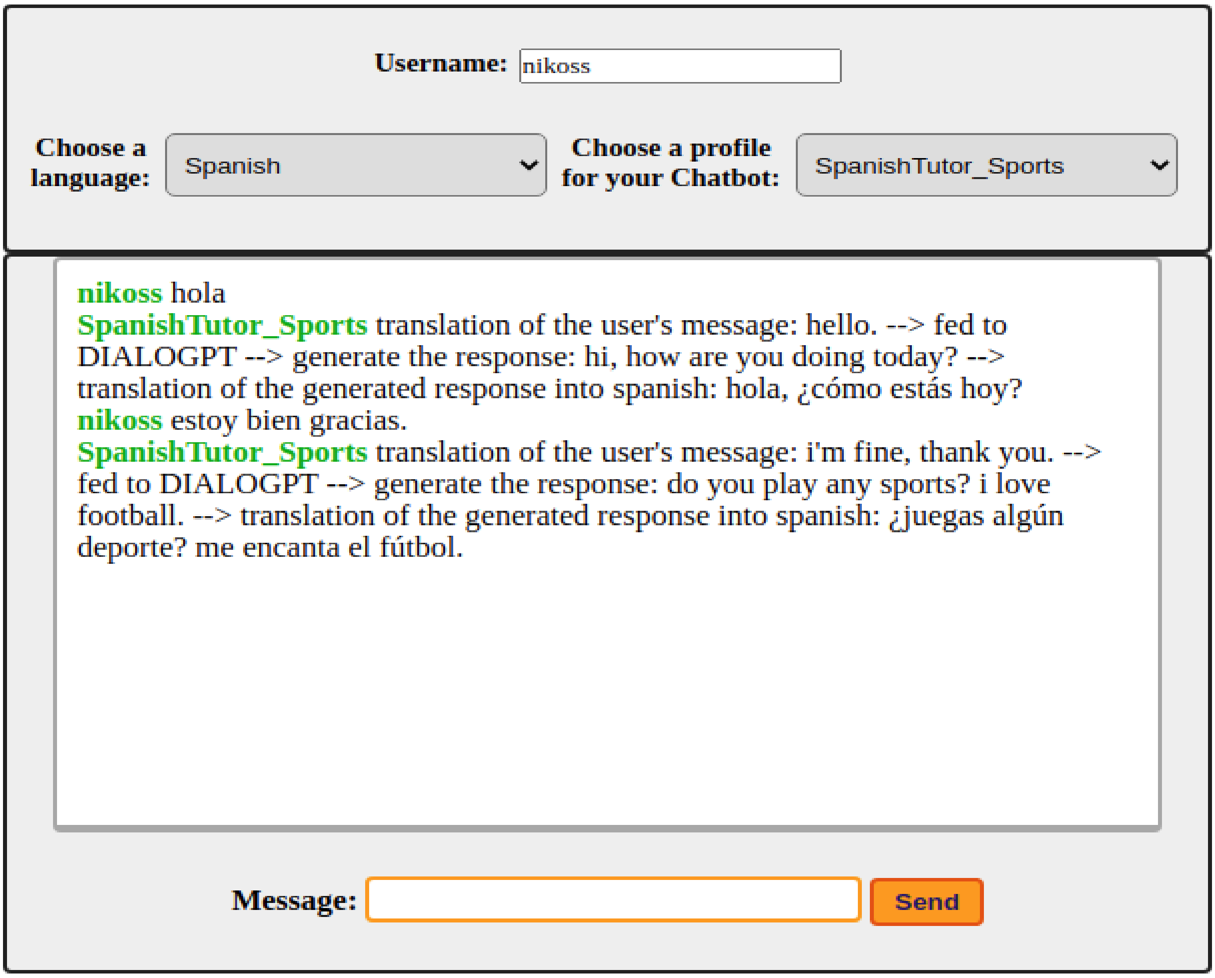


```
from transformers import MarianTokenizer, MarianMTModel

sample_text = "Hola, cómo estás hoy ?"
# Initialize the tokenizer
tokenizer = MarianTokenizer.from_pretrained("./opus-mt-es-en")
# Initialize the model
model = MarianMTModel.from_pretrained("./opus-mt-es-en")
# Tokenize text
batch = tokenizer([sample_text], return_tensors="pt")
# Perform translation and decode the output
gen = model.generate(**batch)
translated_text = tokenizer.batch_decode(gen, skip_special_tokens=True)
print(translated_text)
```

Figure 3. An example for a Spanish-to-English translation using Transformers library. The translated text in this example is 'Hey, how are you today?' .

The implementation of the chatbot using this approach requires two translations in each user message. The message is translated into English first so that the DIALOGPT understands the context, and then the model's generated response is translated into the targeted language, which is the language chosen by the user. Figure 4 illustrates an example of this procedure.



Username: nikoss

Choose a language: Spanish

Choose a profile for your Chatbot: SpanishTutor_Sports

nikoss hola
SpanishTutor_Sports translation of the user's message: hello. --> fed to DIALOGPT --> generate the response: hi, how are you doing today? --> translation of the generated response into spanish: hola, ¿cómo estás hoy?
nikoss estoy bien gracias.
SpanishTutor_Sports translation of the user's message: i'm fine, thank you. --> fed to DIALOGPT --> generate the response: do you play any sports? i love football. --> translation of the generated response into spanish: ¿juegas algún deporte? me encanta el fútbol.

Message:

Figure 4. Example of a conversation with the English Conversational Agent using MarianNMT, a neural machine translation framework for Spanish.

3. IMPLEMENTATION

The Digital Companion in the REBUILD project has been implemented as a chatbot that can communicate with the migrant and help in the completion of tasks. The basic building blocks for this architecture are: the chatbot on the one end, the migrant using the REBUILD application on the other end and a messaging exchange system for the communication of the 2 ends (Figure 5).

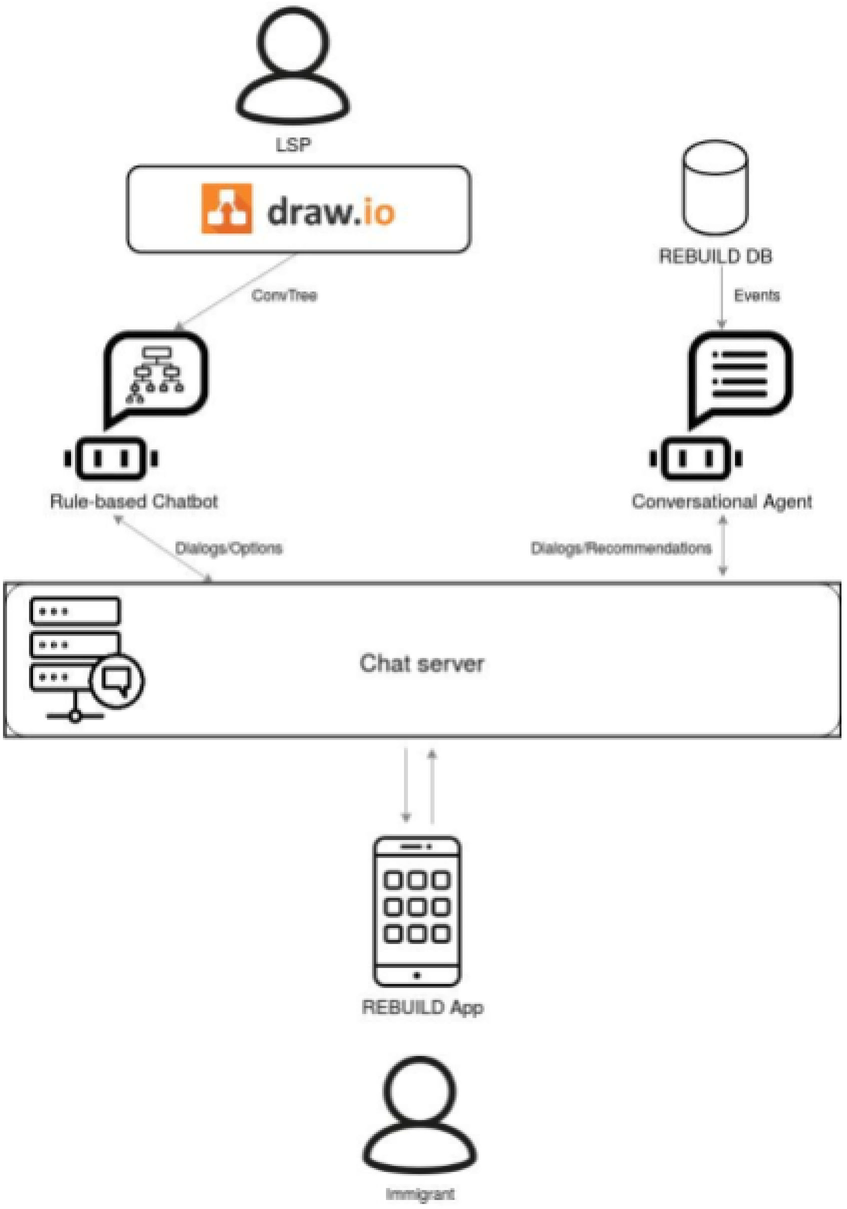


Figure 5. The Digital Companion's Architecture

Depending on the nature of the question that needs to be solved, there are 2 types of chatbots that are used in the REBUILD project. The main usage is for a chatbot that can help a migrant gather information and navigate to a solution for information he/she might be having. Depending on the nature of the problem, a set of domains have been defined for easier navigation. For this type of chatbot it is necessary that all messages are very consistent. Therefore a rule based chatbot is selected to be used in this case. The second application of the chatbot is to act as a language coach for the migrant, to help in the improvement of language skills in an unofficial and fun way. For this use case an open ended conversational agent has been defined.

3.1 CHAT SERVER

The chat server has been implemented by developing 3 RESTful API endpoints, as shown in Figure 6, as well as a *socket.io* endpoint:

1. the *conversation* endpoint: is used to start and manage conversations.

   By performing a POST request to this endpoint, a new conversation can be started between any number of users. The request needs to include a JSON object that defines the initiator (sender) of the conversation and all the other users that will participate (the recipients). The response is the identifier of the conversation that is created. The structure of the expected JSON is shown in Figures 6-7.

   Of course, members can be added or removed in a conversation. For this purpose, a PUT request can be sent with the modified list of members. The response of this request is either an "OK" message or an error code if something went wrong.

   By sending a GET request, one can read the information of a particular conversation. This information includes the members of the conversation as well as when the conversation was started, the current status (ongoing/finished) and when it was last modified. The response is in the form of a JSON object.

   Finally, a DELETE request can be sent to signify the end of a conversation.

2. the *message* endpoint: is used to send and receive a message.

   A new message can be sent by performing a POST request to this endpoint. The request is sent with a JSON object as the one shown in Figure 9, including the *sender*, *recipient* and *conversation* as well as the actual *message* itself. The message can include any or both of *text* and *media*. For the chatbot to work properly, some additional fields are also allowed:

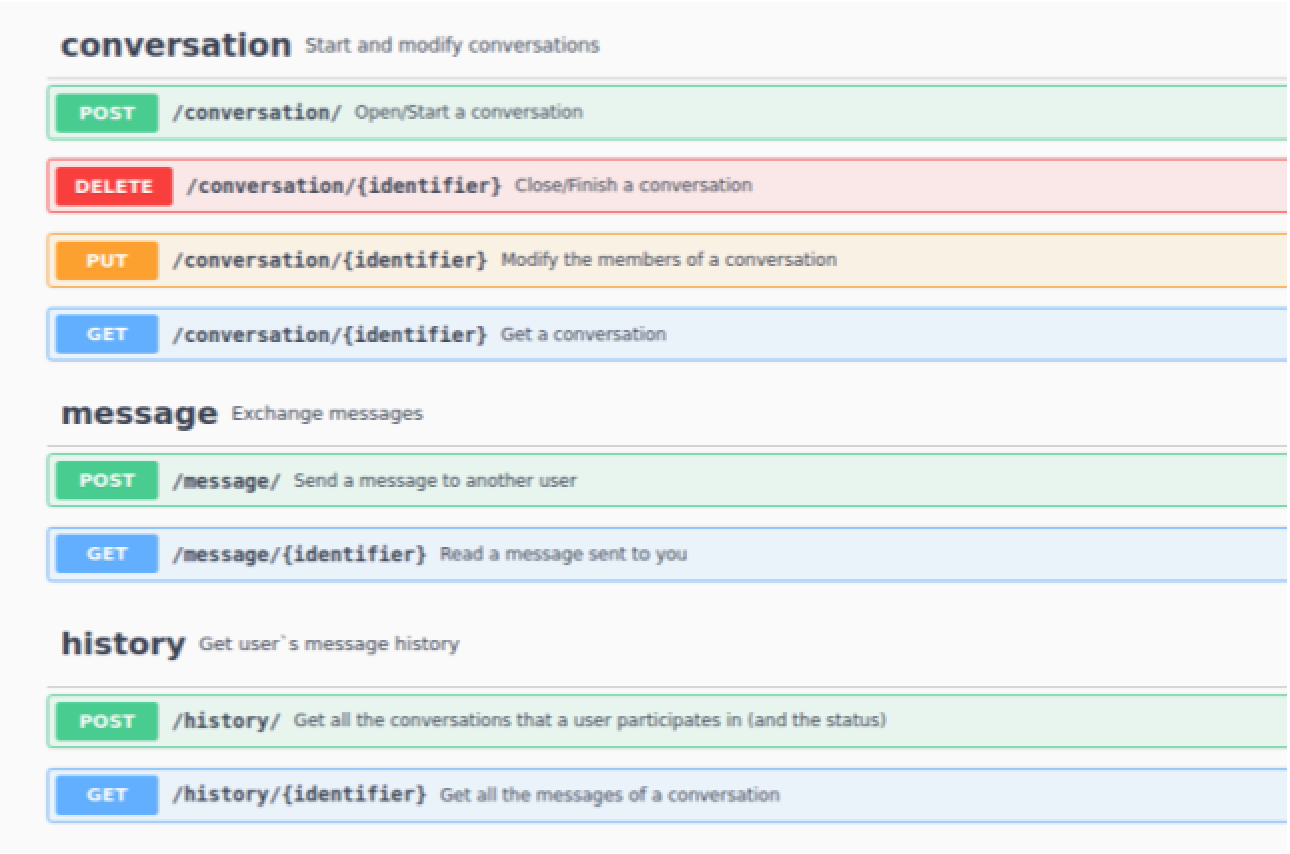
   - language: to define in which language is the text part of the message
   - replies: to provide predefined answers to the user
   - payload: to notify which is the current state of the chatbot
   - key: to define where the information will be stored in the temporary memory
   - action: to define if the application should perform an action
   - token: the authentication token of the user in the IDM so that the chatbot can perform actions on behalf of the user

   Whenever a new message is created through the POST request, all of the recipients of the message that are connected to the *socket.io* endpoint are notified about the message immediately. If a recipient is not connected then the notification is kept in a buffer until the client is connected. Then, using the identifier of the message, it can be retrieved through a GET request. The GET request returns the identified message in a JSON object.

3. the *history* endpoint: can retrieve all the conversations that a user is a member of.

   A POST request to this endpoint returns the identifiers of all the conversations that a user participates in. Optionally, other members of the conversations can also be defined. This option can be used when the application wants to retrieve all of the interaction between the user and a specific chatbot for example. Also optionally, only the ongoing or the finished conversations can be retrieved. The full JSON object expected from this endpoint is depicted in Figure 10.

A GET request can be used to retrieve all of the messages in a conversation by using the conversation identifier. The messages are returned in the same order as they were sent. In order to limit the possible number of calls necessary for a client to retrieve a whole conversation, the service returns the actual messages and not just the message identifiers.



conversation Start and modify conversations

POST `/conversation/` Open/Start a conversation

DELETE `/conversation/{identifier}` Close/Finish a conversation

PUT `/conversation/{identifier}` Modify the members of a conversation

GET `/conversation/{identifier}` Get a conversation

message Exchange messages

POST `/message/` Send a message to another user

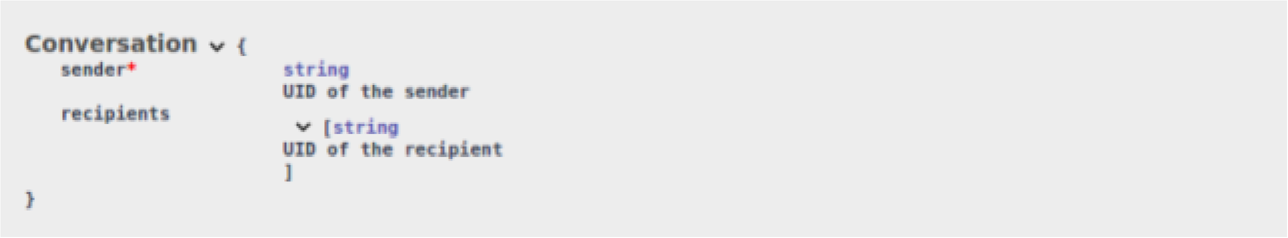
GET `/message/{identifier}` Read a message sent to you

history Get user`s message history

POST `/history/` Get all the conversations that a user participates in (and the status)

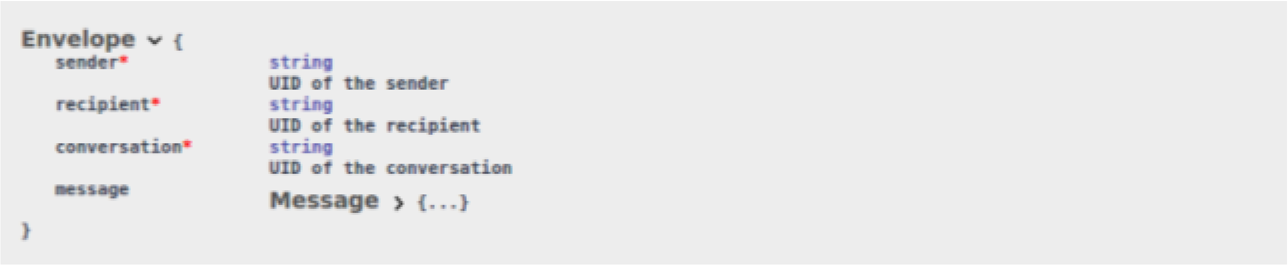
GET `/history/{identifier}` Get all the messages of a conversation

Figure 6. Swagger representation of the available endpoints of the Chat Server API.



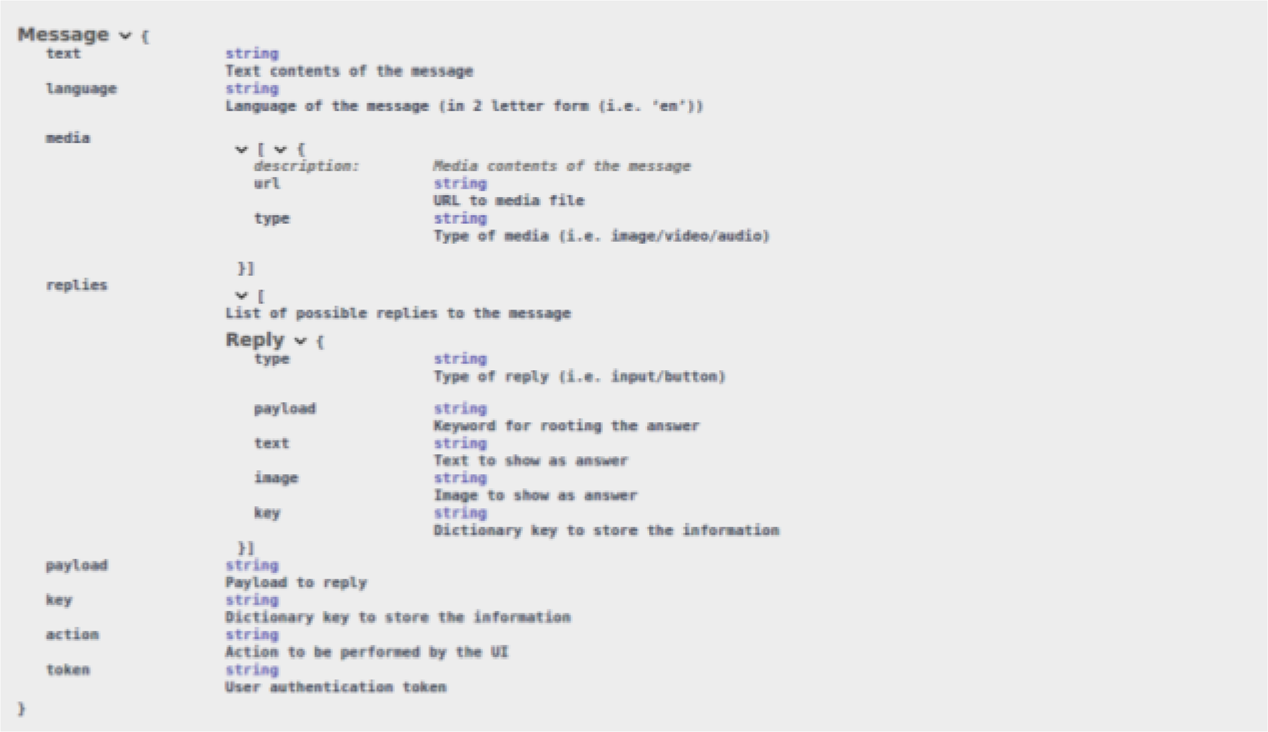
```
Conversation ⌄ {
   sender*              string
                        UID of the sender
   recipients
                         ⌄ [string
                        UID of the recipient
                        ]
}
```

Figure 7. Structure of the expected JSON for the conversation API.



```
Envelope ⌄ {
   sender*              string
                        UID of the sender
   recipient*           string
                        UID of the recipient
   conversation*        string
                        UID of the conversation
   message              Message › {...}
}
```

Figure 8. Structure of the expected "envelope" JSON for the message API.



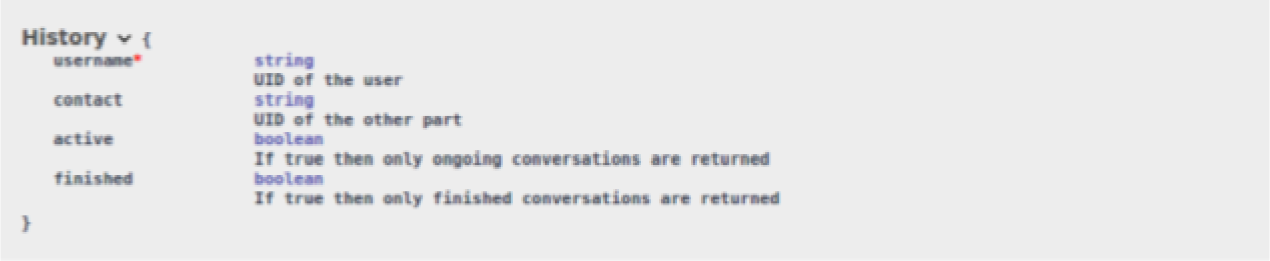
```
Message ⌄ {
   text                   string
                          Text contents of the message
   language               string
                          Language of the message (in 2 letter form (i.e. 'en'))

   media
                           ⌄ [ ⌄ {
                             description:          Media contents of the message
                             url                   string
                                                   URL to media file
                             type                  string
                                                   Type of media (i.e. image/video/audio)

                           }]
   replies
                           ⌄ [
                          List of possible replies to the message
                          Reply ⌄ {
                             type                  string
                                                   Type of reply (i.e. input/button)

                             payload               string
                                                   Keyword for rooting the answer
                             text                  string
                                                   Text to show as answer
                             image                 string
                                                   Image to show as answer
                             key                   string
                                                   Dictionary key to store the information
                           }]
   payload                string
                          Payload to reply
   key                    string
                          Dictionary key to store the information
   action                 string
                          Action to be performed by the UI
   token                  string
                          User authentication token
}
```

Figure 9. Structure of the message JSON.



```
History ⌄ {
   username*              string
                          UID of the user
   contact                string
                          UID of the other part
   active                 boolean
                          If true then only ongoing conversations are returned
   finished               boolean
                          If true then only finished conversations are returned
}
```

Figure 10. Structure of the expected JSON for the history API.

3.2 RULE BASED CHATBOT

The rule-based chatbot was built using the Python programming language. An endpoint was created using the flask micro web services framework to communicate with the platform's user interface and other services. The service monitors the chatbot for new messages, and when one arrives, it extracts the user who sent it as well as the contents of the message. A payload object is enclosed within the message content, and the context to the prior state is supplied to the chatbot. The appropriate state is chosen based on the context.

The networkx library is used to build, maintain, and traverse the various states of the conversation tree. Because the conversation state can be recognized by the graph's node's name as payload, this facilitates usability. Following the identification of the selected node, all of the child nodes are retrieved in order to determine the next state of the conversation. The user receives a reply-message containing the selected next state. Other benefits of using networkx include the ability to easily merge multiple trees into a single conversation and the ability to dynamically edit the conversation tree by adding, deleting, or updating the graph.

3.2.1. UI FOR THE DEFINITION OF A NEW CONVERSATION TREE

The networkx graph can be defined in a variety of ways, but the rules must always be specified in a machine-readable language and added to the system. W3C recommends XML (eXtensible Markup Language) as a software and hardware neutral language for storing and transmitting data. Aside from being software independent, the XML language has no predefined tags and, as the name implies, is extensible, allowing for the addition of new tags at any time. In addition, an utility function for reading an XML file and constructing a new graph was also added. If the XML is not syntactically accurate, the function fails and the user is informed that the file contained issues.

As stated in deliverable *D4.4*, the draw.io[1] online application was chosen to provide users with a user interface for generating and maintaining conversation trees in order to provide an even more user-friendly experience. The primary reasons for choosing draw.io as a user interface are that it is free to use, that it allows users to share graphs in many formats such as Google Drive, and that it offers the option to export the created graph in XML. Furthermore, draw.io allows you to create an infinite number of nodes and edges with predefined shapes. While the predefined shapes have connotations for the user, these connotations are ignored during the translation to a conversation tree for our purposes. The only difference is whether the object is a node or an edge. Node objects are translated into bot responses, whereas edges are translated into user responses. Furthermore, each node or edge can be assigned an unlimited number of properties.

An example usage of draw.io to create the basic blocks of a conversation tree is:

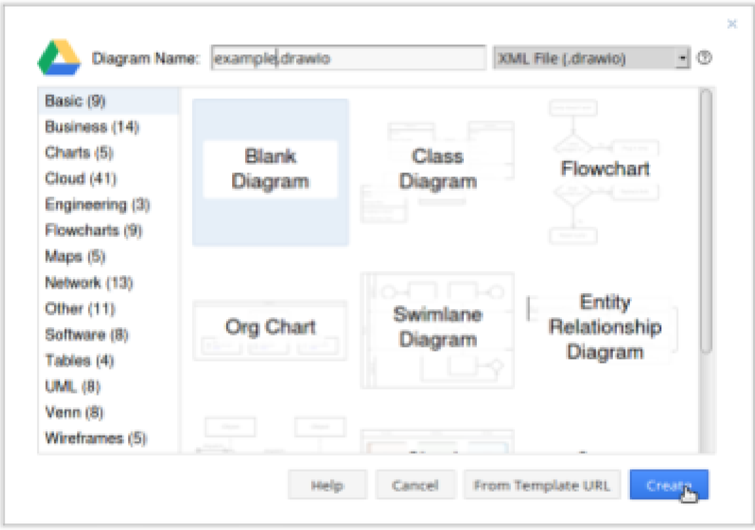


Diagram Name: example.drawio
XML File (.drawio)

Figure 11. Create a new blank diagram.

---

[1] The draw.io application has recently changed its name to diagrams.net and can be reached at https://www.diagrams.net/

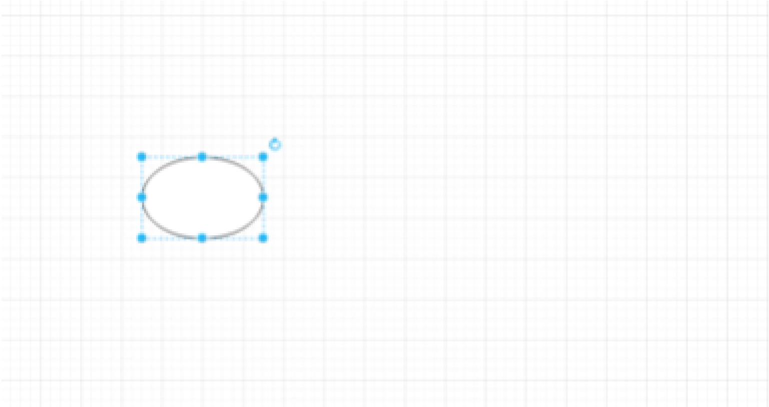


Figure 12. Add a start node.

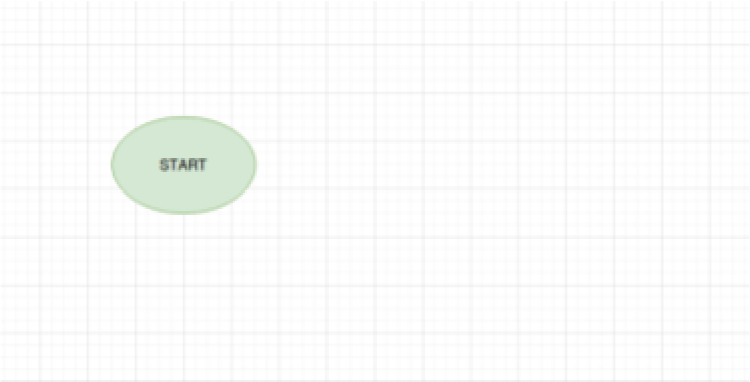


Figure 13. Change the color and add a label.

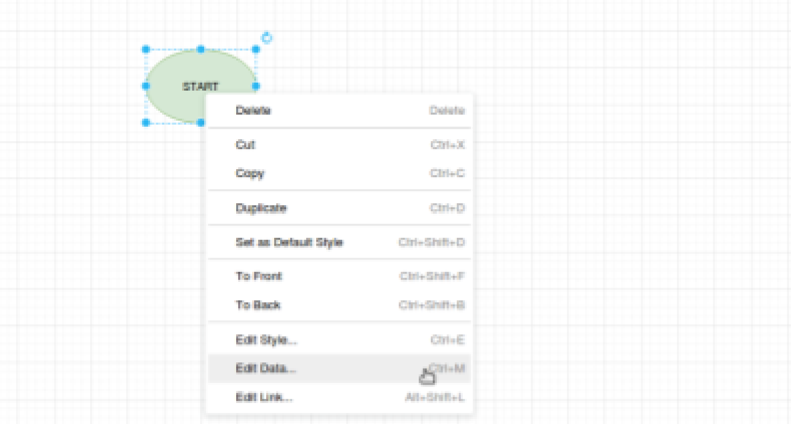
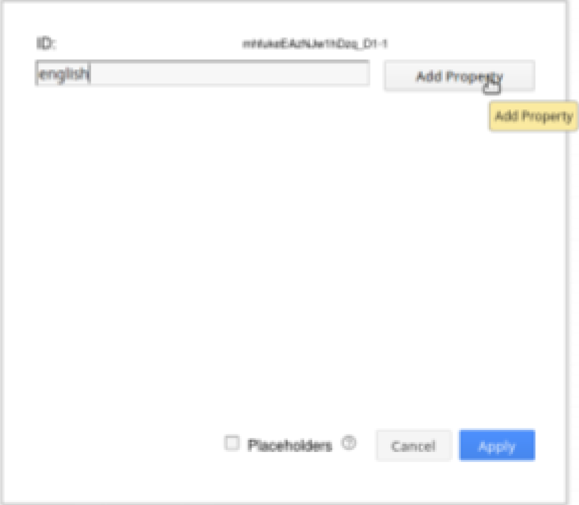


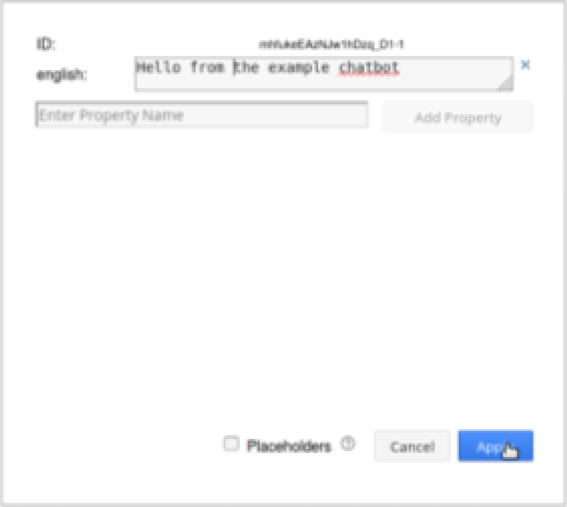
Figure 14. Add properties by clicking "edit data…"



ID:

english

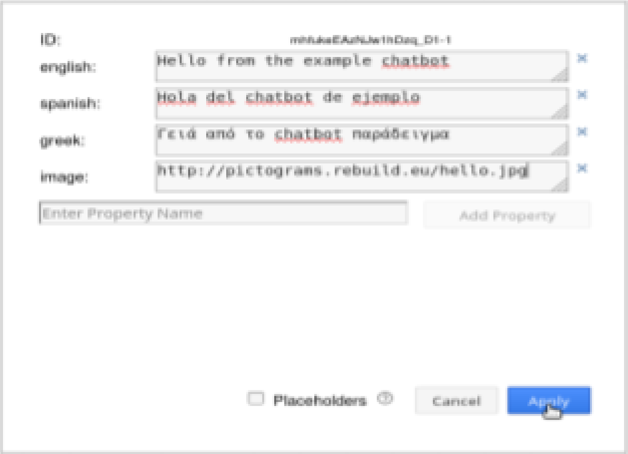
Figure 15. Add the property name.



ID:

english: Hello from the example chatbot

Figure 16. Add the caption to display for "english".



ID:

english: Hello from the example chatbot

spanish: Hola del chatbot de ejemplo

greek: Γειά από το chatbot παράδειγμα

image: http://pictograms.rebuild.eu/hello.jpg

Figure 17. Add captions for the other languages and an "image" for the pictogram.

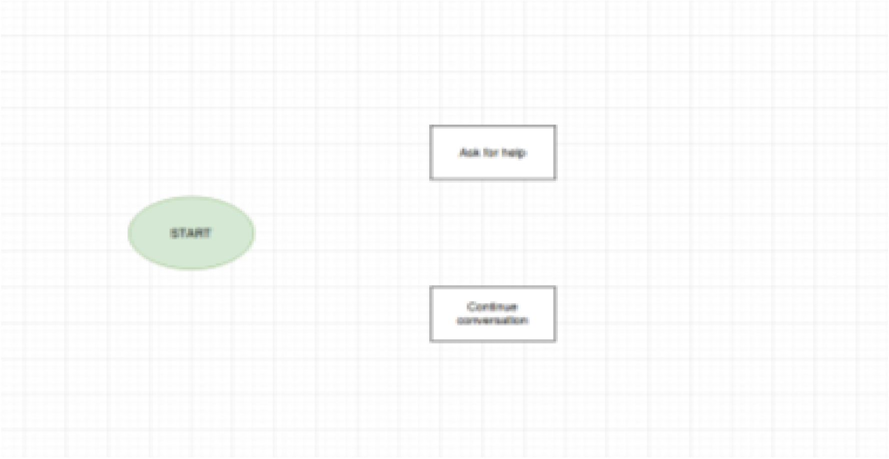
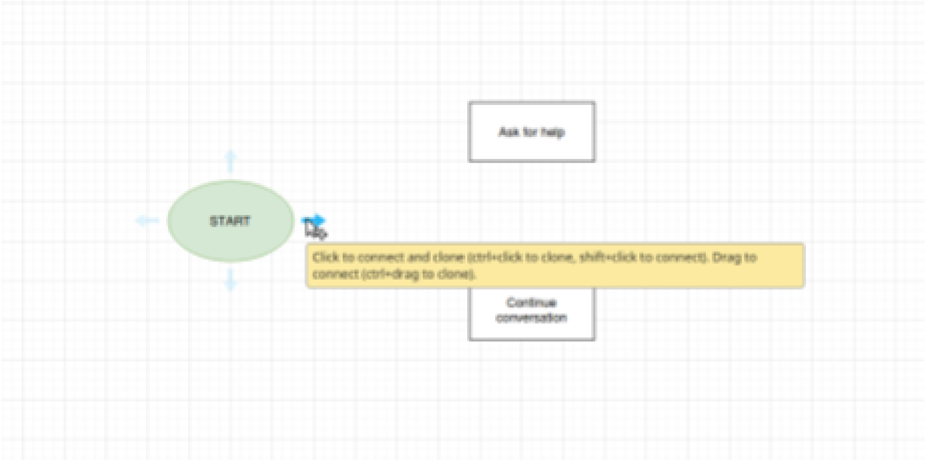


Figure 18. Add two possible outcomes for this state: the user wants help with something and the user continues the conversation.



Click to connect and clone (ctrl+click to clone, shift+click to connect). Drag to connect (ctrl+drag to clone).

Figure 19. By dragging from the arrow to a node, a connecting adge can be created.

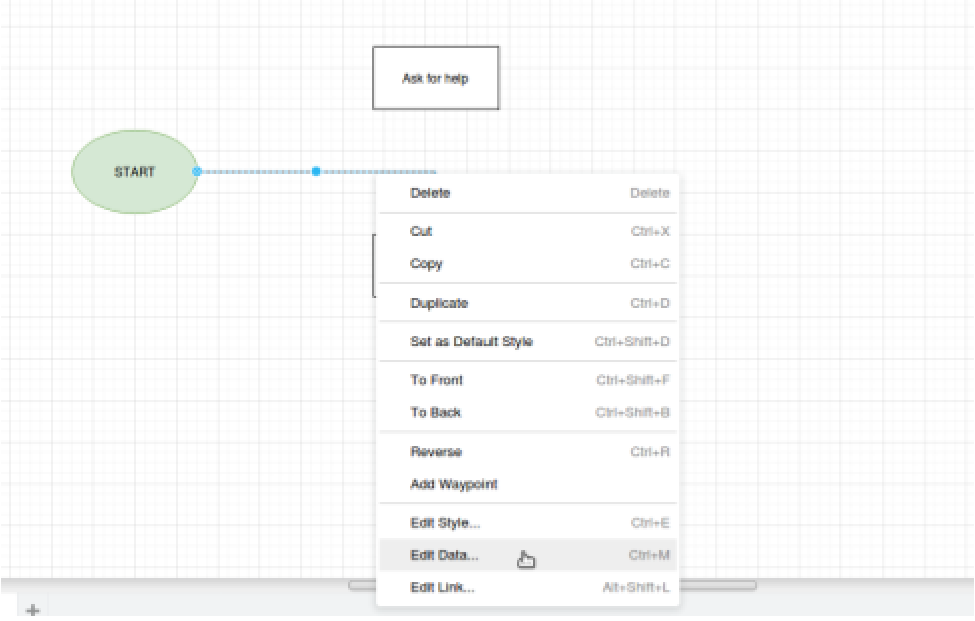
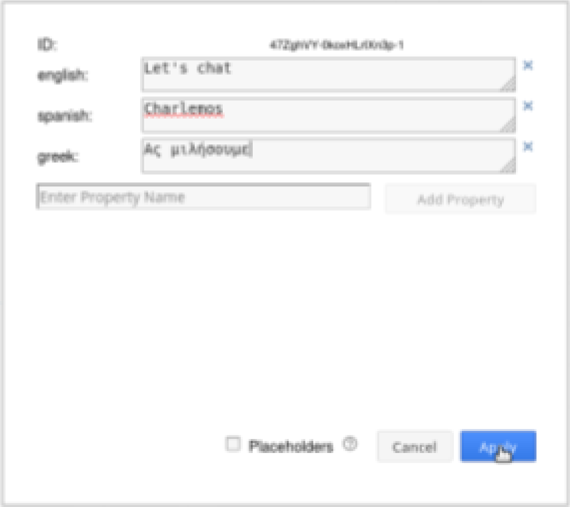


Figure 20. The same as with nodes, properties can be added using the "Edit Data…" option.



ID:
english: Let's chat
spanish: Charlemos
greek: Ας μιλήσουμε

Figure 21. Add responses for English, Spanish and Greek.

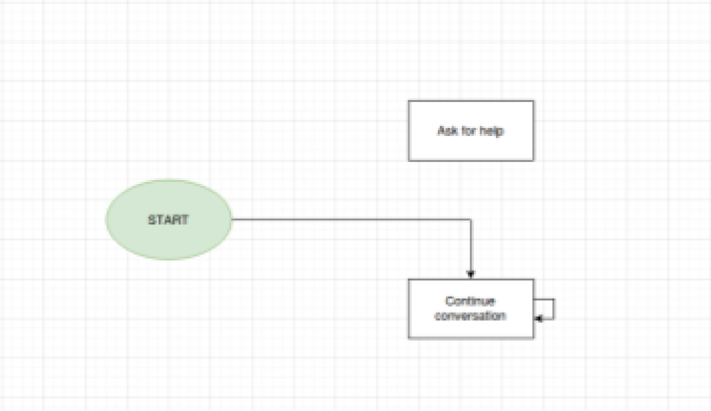


Figure 22. Iterative loops can be created by dragging the edge to the originating node.

3.3 RECOMMENDER CONVERSATIONAL AGENT

For the purposes of REBUILD, the Conversational Agent should be a tool that assists migrants in interacting with local societies and providing an improved quality of life. Aside from the creation of a Conversational Agent, which would assist migrants in learning a new language, a recommendation system in conjunction with the dialogue system would provide important guidance to migrants in meeting the local societies. This section describes the implementation of a recommendation system adapted to the Conversational Agent.

The Conversational Agent was built using the Python programming language. The Conversational Agent was developed following an Object Oriented Programming style in the sense that it is implemented as a single class with some static methods that can be easily deployed. Furthermore, this OOP approach incorporates flexibility and scalability in the sense that new functionalities and methods can be included in the class to be used by the chatbot. This class's static methods are designed to perform the following functions:

- Building the input of the model.
- Tokenizing the message and the "persona" sentence using the Tokenizers from Transformers library.
- Decoding the message using the top-k sampling method described in previous sections.
- Constructing a Recommendation message about available social events according to the city that the user selected and the user's profile.

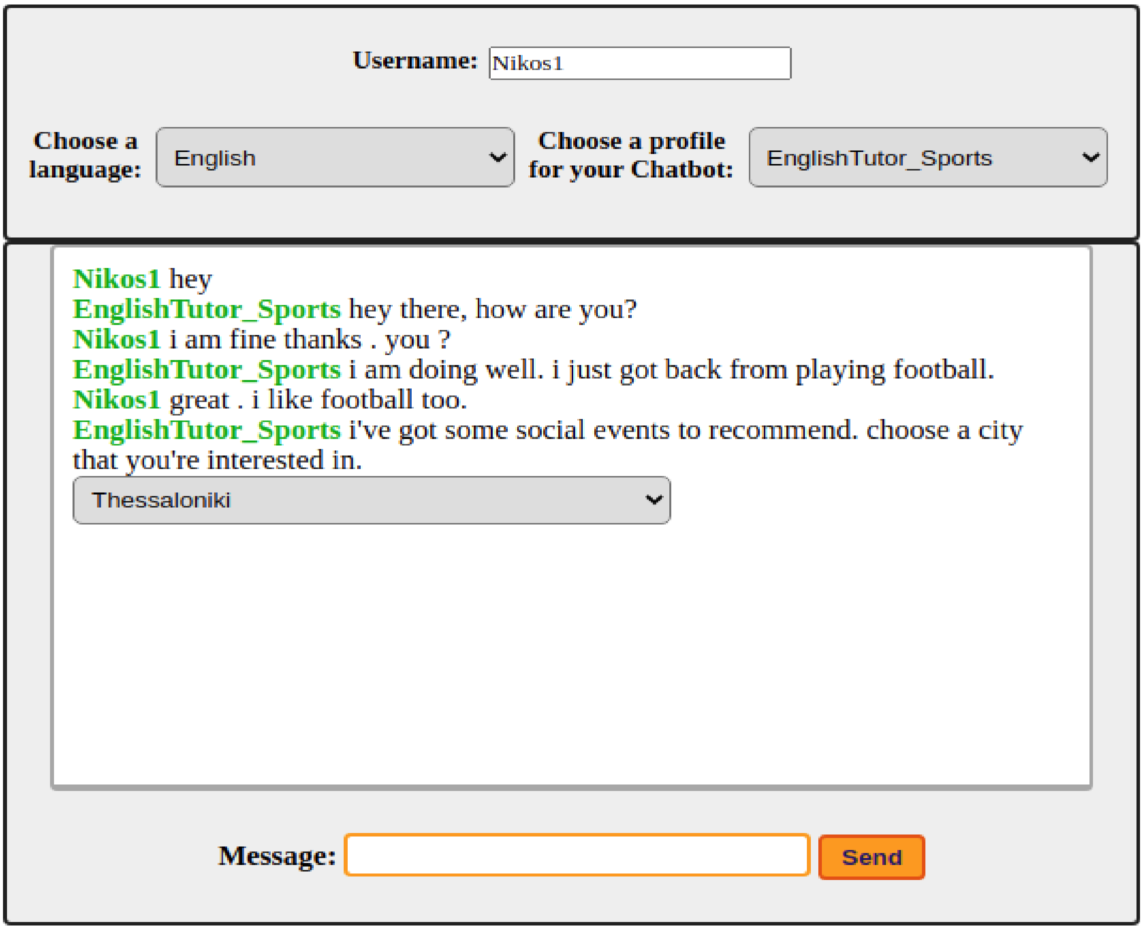
- Responding in the appropriate format (text).

In addition, this Conversational Agent class inherits another class that is developed in order to handle the messages. It receives and extracts the required information about each notification from the endpoint, such as the sender, the message, the recipient and the timestamp, and then calls the Agent to respond. Finally, it posts the response (string type) from the Conversational Agent along with its name (e.g. EnglishTutor_Sports). This class implementation is simple and is based on the Socket.IO library that enables real-time, bidirectional and event-based communication between the client and the server.

The dropdown form "Choose a profile for your chatbot" shown in Figure 23 is where the user can select the personality of the Conversational Agent. It is implemented with two chatbot profile options: Sports and Music. More specifically, these profile options (personas) are sentences that are concatenated with the history of the previous two turns of the dialogue in order to be fed to the model as input and generate an answer. The use of "persona" sentences makes the dialogue appear more engaging to the user.

The recommendation message is constructed from a method by calling the REBUILD API to retrieve all available social events. Lists of all available events are created based on the cities. More specifically, the Agent is designed to output a dropdown form in the third turn of the dialogue, asking from the user to select a city of interest, as it is shown in Figure 23. The early selection of the third turn of dialogue is chosen in order to keep the chat with the user more interesting and engaging. If there are no available events in the city that the user selected, it displays a message informing the user of the event's unavailability at the time and recommending events from another city.

Furthermore, in order to make the chat-flow appear more normal in conjunction with the recommendation messages, we designed the Agent in such a way that when it receives a question from the user, it postpones the recommendation mechanism for the next turn. So, in the third turn of the dialogue, if it receives a question from the user, it will respond as a Conversational Agent and will continue to act in this manner until the user has no more questions before displaying the recommendation message. After sending the recommendation message, the chatbot resumes its role as an open-domain Conversational Agent and continues the conversation with the user as it is demonstrated in Figure 24. This is a dynamic way of displaying the recommendation message in order to make the dialogue appear more human to the user.



Username: Nikos1

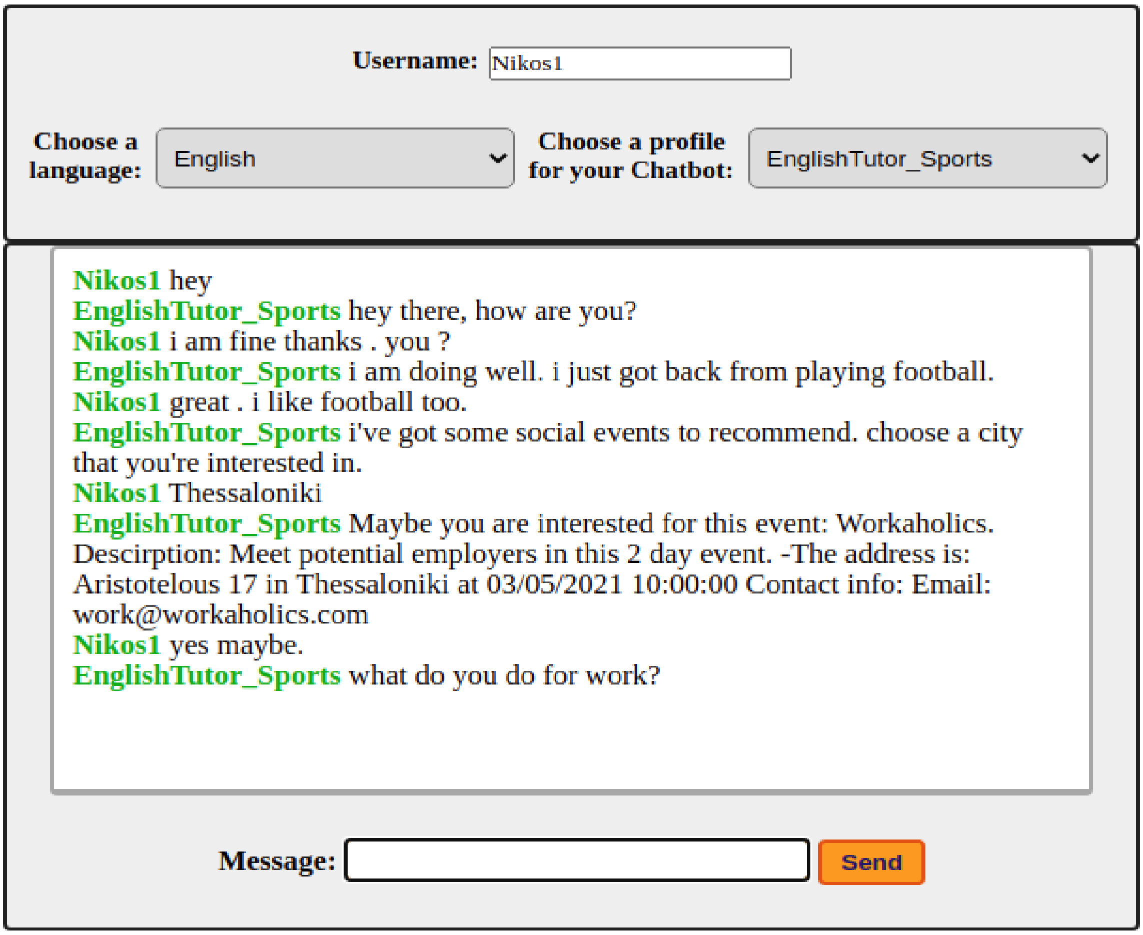
Choose a language: English

Choose a profile for your Chatbot: EnglishTutor_Sports

Nikos1 hey
EnglishTutor_Sports hey there, how are you?
Nikos1 i am fine thanks . you ?
EnglishTutor_Sports i am doing well. i just got back from playing football.
Nikos1 great . i like football too.
EnglishTutor_Sports i've got some social events to recommend. choose a city that you're interested in.

Thessaloniki

Figure 23. In the third turn of dialogue it is displayed a message and a dropdown form (with cities as options) that asks the user to select a city of interest.



Username: Nikos1

Choose a language: English

Choose a profile for your Chatbot: EnglishTutor_Sports

Nikos1 hey
EnglishTutor_Sports hey there, how are you?
Nikos1 i am fine thanks . you ?
EnglishTutor_Sports i am doing well. i just got back from playing football.
Nikos1 great . i like football too.
EnglishTutor_Sports i've got some social events to recommend. choose a city that you're interested in.
Nikos1 Thessaloniki
EnglishTutor_Sports Maybe you are interested for this event: Workaholics. Descirption: Meet potential employers in this 2 day event. -The address is: Aristotelous 17 in Thessaloniki at 03/05/2021 10:00:00 Contact info: Email: work@workaholics.com
Nikos1 yes maybe.
EnglishTutor_Sports what do you do for work?

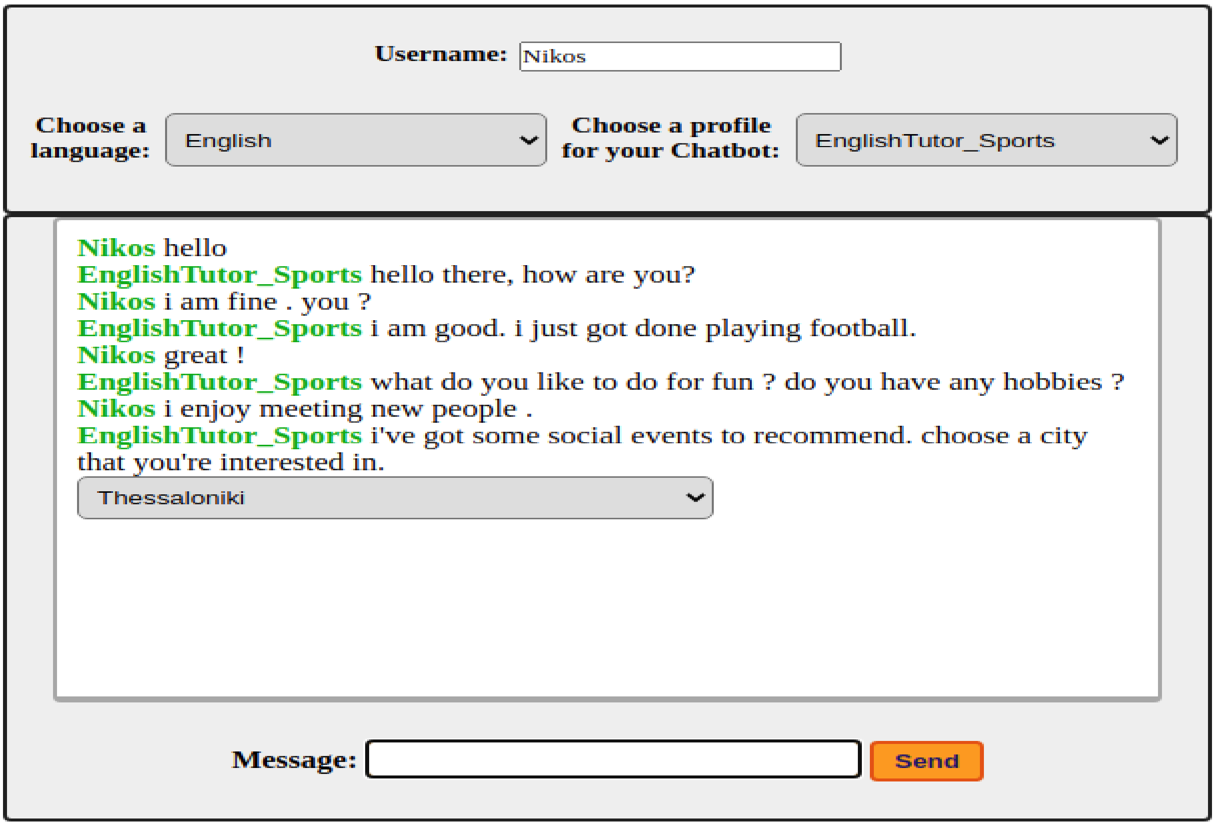
Message:

Figure 24. Following the selection of the city ("Thessaloniki") from the user, the recommendation message with all available information for the event is displayed. After the recommendation message, the Agent continues the open-domain conversation with the user.

<u>Personalization of the recommendations:</u> The recommendation system could be improved by providing more personalized recommendations based on the user profile or even by the dialogue itself. A tool for this purpose has been developed and is described further below. In order to personalize the recommendations of the open-domain chatbot according to the user's profile or even by information that the user will give through the dialogue such as a hobby, we used the *Sentence Transformers* library.

This library provides an easy method to compute dense vector representations for sentences. Text is embedded in vector space in such a way that similar text is close and can be found quickly using cosine similarity. The transformer-based model we used for this purpose is from the sentence-transformers-repository and it can be found open sourced on the *hugging face* platform *(https://huggingface.co/sentence-transformers/paraphrase-distilroberta-base-v1)*. This model was trained on millions of paraphrase sentences and more information can be found in the work of *(Nils Reimers and Iryna Gurevych, 2019)*.

As a result, for personalization, we built a method in the chatbot's class that takes all available social events (as a list of strings) in the city the user selected. This method generates the embeddings of these sentences-social-events using the model described above and compares them to the profile embeddings using the cosine similarity metric. The chatbot is used to obtain the user's profile or preferences, as shown in Figure 25. Finally, it keeps the social event with the highest cosine similarity score and returns it to the user as a personalized recommendation, as demonstrated in Figure 26.



Username: Nikos

Choose a language: English

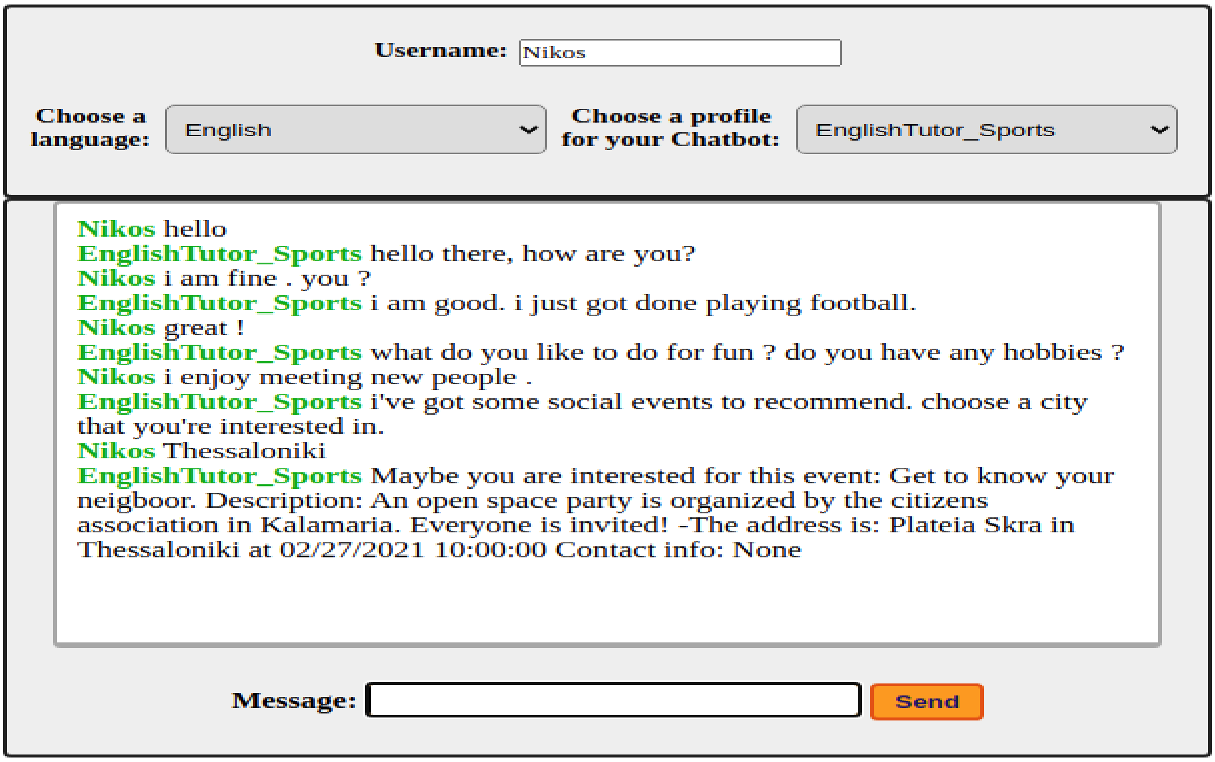
Choose a profile for your Chatbot: EnglishTutor_Sports

Nikos hello
EnglishTutor_Sports hello there, how are you?
Nikos i am fine . you ?
EnglishTutor_Sports i am good. i just got done playing football.
Nikos great !
EnglishTutor_Sports what do you like to do for fun ? do you have any hobbies ?
Nikos i enjoy meeting new people .
EnglishTutor_Sports i've got some social events to recommend. choose a city that you're interested in.

Thessaloniki

Message:

Figure 25. Through the dialogue, the conversational agent collects data for the user's preferences and the city of his/her choice.



Username: Nikos

Choose a language: English

Choose a profile for your Chatbot: EnglishTutor_Sports

Nikos hello
EnglishTutor_Sports hello there, how are you?
Nikos i am fine . you ?
EnglishTutor_Sports i am good. i just got done playing football.
Nikos great !
EnglishTutor_Sports what do you like to do for fun ? do you have any hobbies ?
Nikos i enjoy meeting new people .
EnglishTutor_Sports i've got some social events to recommend. choose a city that you're interested in.
Nikos Thessaloniki
EnglishTutor_Sports Maybe you are interested for this event: Get to know your neigboor. Description: An open space party is organized by the citizens association in Kalamaria. Everyone is invited! -The address is: Plateia Skra in Thessaloniki at 02/27/2021 10:00:00 Contact info: None

Message:

Figure 26. The conversational Agent suggests the social event with the highest cosine similarity score to the user as a personalized recommendation.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

4. INTEGRATION

The provided services are implemented in the REBUILD project to interact with end users via the chatbot interface for migrants and the dashboard for LSPs. The Job Seeking service is used to match migrants' skills and competencies to available jobs in the area where they live, while the Education Training service is used to match migrants' missing skills to relevant courses, and the Social Mentoring service is used to automatically match candidate mentors with candidate mentees instead of the LSP.

As a result, for the first two services, the chatbot is only integrated with the recommender system. Both services use the same basic procedures for integrating the chatbot with the recommender system:

- Because it is the primary interaction tool for the user, the chatbot can be thought of as a specific starting point. In this regard, the chatbot is in charge of gathering the user's information that the system requires as inputs for the recommendation (various elements such as work experience, location, whether the user has a work permit, and skills).
- The recommender is then called to perform matchings.
- Finally, the chatbot displays the matches to the end user as a typical response.

4.1 JOB SEEKING SCENARIO

The first step, information gathering, is defined in the relevant conversation tree that the chatbot follows for each scenario. The migrant's previous work experience, as well as the user's additional skills and competencies, are the most important pieces of information for both scenarios. Complete information on the conversation trees and the data collected for each scenario are defined and demonstrated in deliverable D3.4. Figure 27 depicts a portion of the conversation tree construction for the Job Seeking scenario.

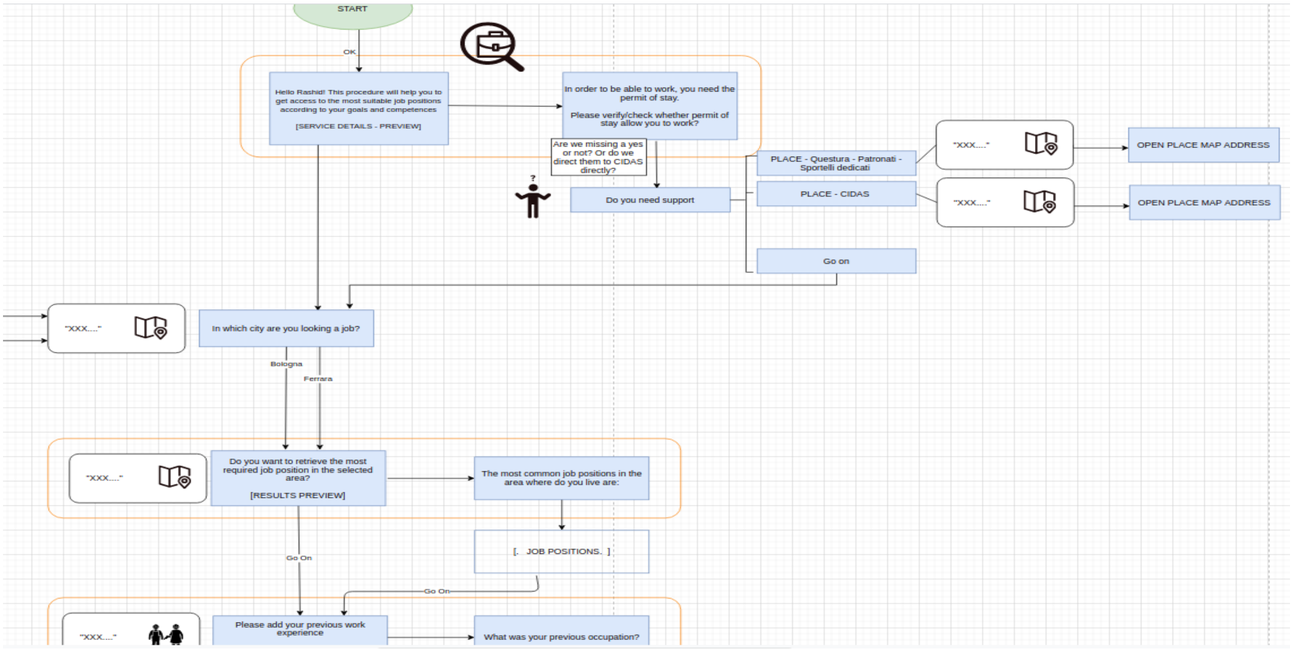
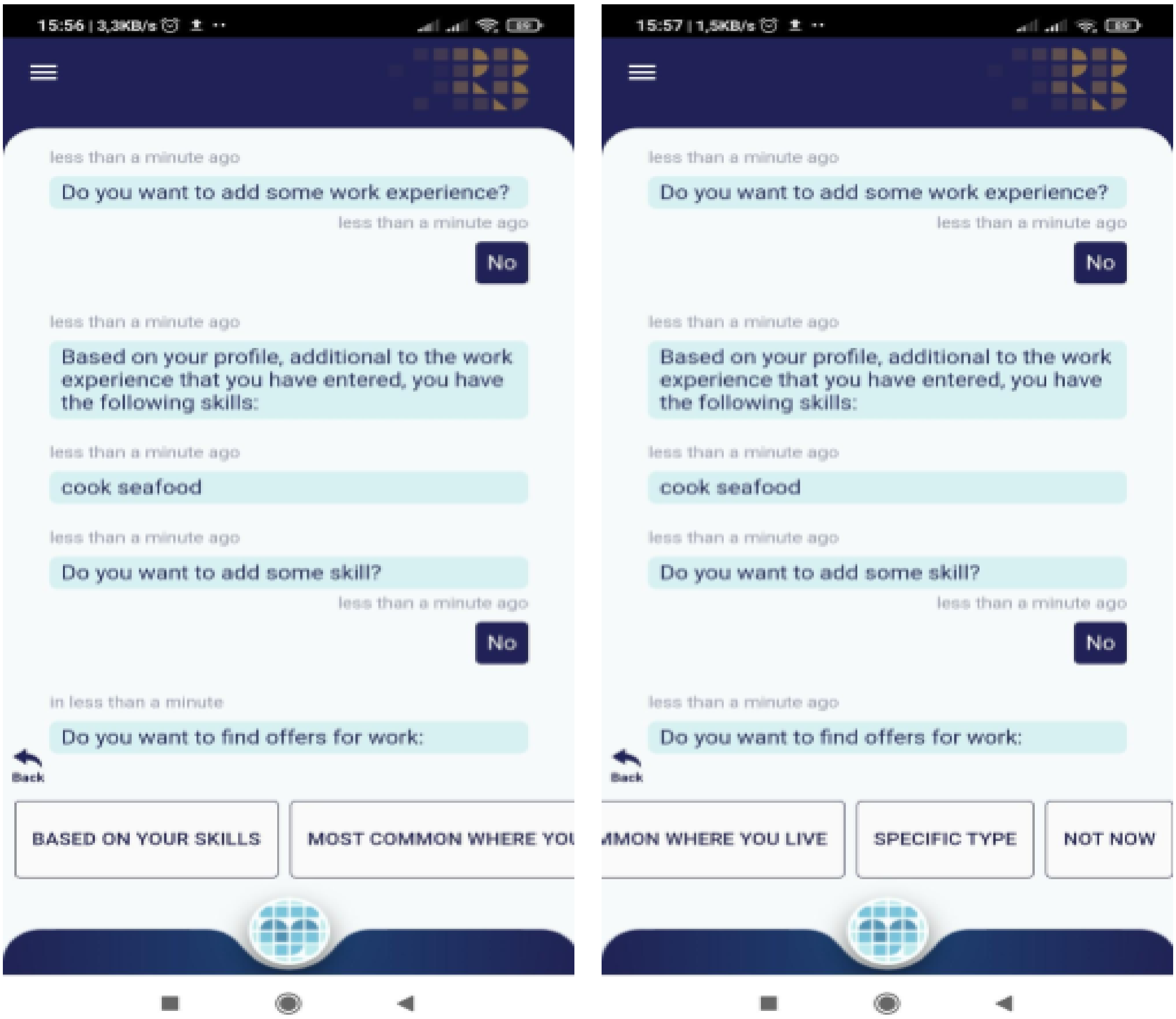

START
OK
Hello Rashid! This procedure will help you to get access to the most suitable job positions according to your goals and competences
[SERVICE DETAILS - PREVIEW]
In order to be able to work, you need the permit of stay.
Please verify/check whether permit of stay allow you to work?
Are we missing a yes or not? Or do we direct them to CIDAS directly?
?
Do you need support
PLACE - Questura - Patronati - Sportelli dedicati
PLACE - CIDAS
Go on
"XXX...."
OPEN PLACE MAP ADDRESS
"XXX...."
OPEN PLACE MAP ADDRESS
"XXX...."
In which city are you looking a job?
Bologna
Ferrara
"XXX...."
Do you want to retrieve the most required job position in the selected area?
[RESULTS PREVIEW]
The most common job positions in the area where do you live are:
[. JOB POSITIONS.]
Go On
Go On
"XXX...."
Please add your previous work experience
What was your previous occupation?

Figure 27. Conversation tree for the Job Seeking scenario.

The recommender system is called in the second step to return the best matching job offers based on the user's skills. The call is made by passing the user id and the maximum number of recommendations to the provided API endpoint. The buttons that allow the user to choose the type of search to be performed are shown in the example conversation of Figure 28. The recommender system can then be called through the chatbot. As shown in Figure 28, the user is given three options:

- to look for career prospects depending on his or her skills and competencies. The migrant's skills and competencies, based on previous work experience and soft skills, are assessed against all accessible job opportunities to offer a list of the most suitable opportunities for the migrant to pursue.
- to look for opportunities based on a title provided by the user. With this option, the user can find work prospects in his or her area of residence based on a specific ESCO title, regardless of his or her personal skills and competencies. This scenario can be used to evaluate whether there are opportunities in various fields before choosing whether certain skill sets are worth gaining, i.e. if there are enough opportunities for a given work type to decide whether you want to gain the skills to perform this job type.
- to look for the most common job openings. This option displays a list of the most prevalent job titles in the area where you live.



15:56 | 3,3KB/s

less than a minute ago
Do you want to add some work experience?
less than a minute ago
No

less than a minute ago
Based on your profile, additional to the work experience that you have entered, you have the following skills:

less than a minute ago
cook seafood

less than a minute ago
Do you want to add some skill?
less than a minute ago
No

in less than a minute
Do you want to find offers for work:

15:57 | 1,5KB/s

less than a minute ago
Do you want to add some work experience?
less than a minute ago
No

less than a minute ago
Based on your profile, additional to the work experience that you have entered, you have the following skills:

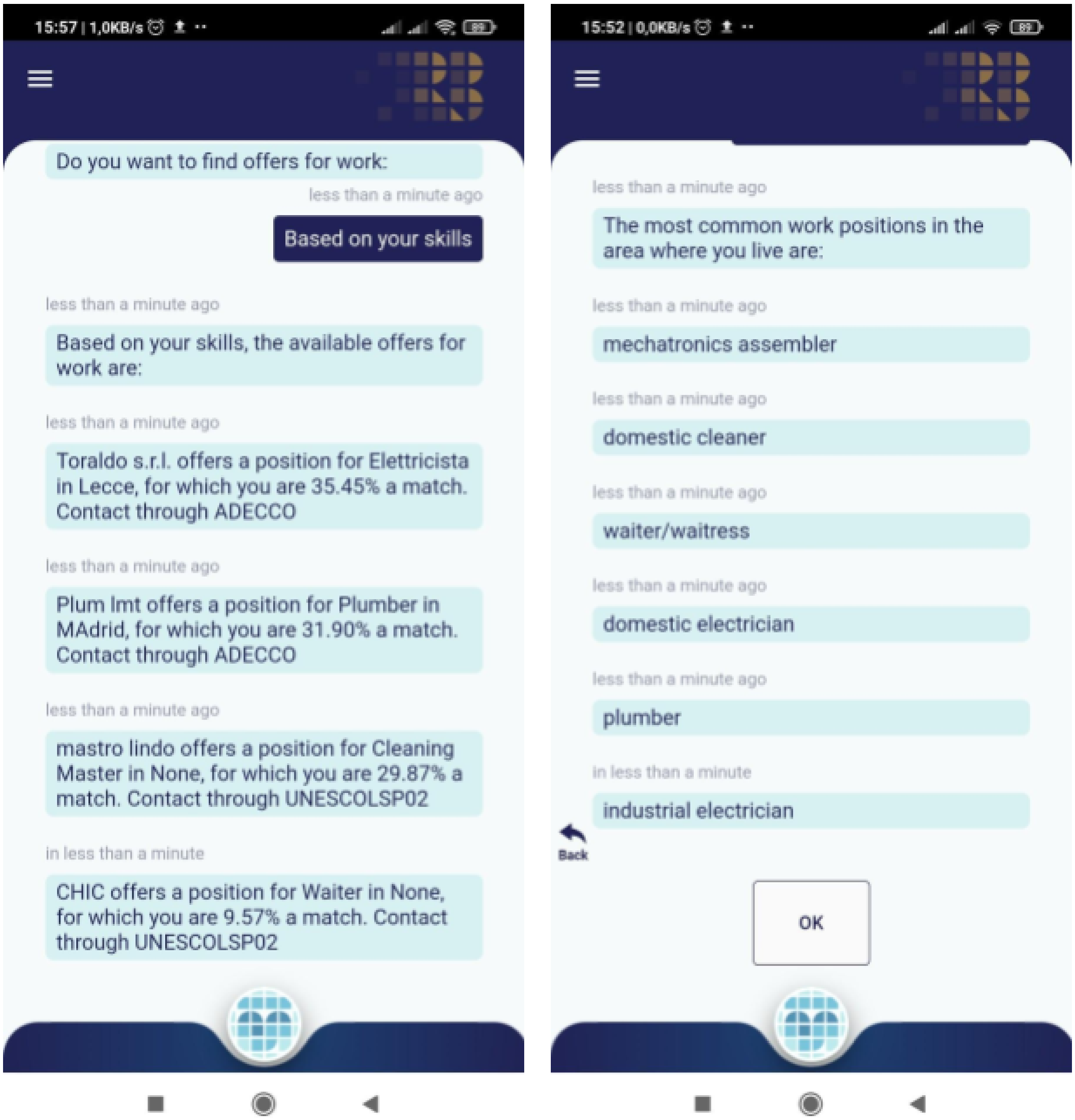
less than a minute ago
cook seafood

less than a minute ago
Do you want to add some skill?
less than a minute ago
No

less than a minute ago
Do you want to find offers for work:

Figure 28. Example conversation to call the job seeking scenario recommender through the chatbot.

The final step is to display the retrieved matches to the end user. The recommender returns a list of job offer ids as well as the percentage of matching to the criteria provided. Of course, a list of ids isn't very useful to the user. A call is made to the job offers' endpoint for each job offer id returned by the recommender in order to obtain more information about the employment offer. The message that is displayed to the user includes the name of the company that is offering the position, the actual job title, and the name of the LSP that should be contacted for further information about the position that is being offered. Figure 29 demonstrates an example of the chatbot's response.



Do you want to find offers for work:

less than a minute ago

Based on your skills

less than a minute ago

Based on your skills, the available offers for work are:

less than a minute ago

Toraldo s.r.l. offers a position for Elettricista in Lecce, for which you are 35.45% a match. Contact through ADECCO

less than a minute ago

Plum lmt offers a position for Plumber in MAdrid, for which you are 31.90% a match. Contact through ADECCO

less than a minute ago

mastro lindo offers a position for Cleaning Master in None, for which you are 29.87% a match. Contact through UNESCOLSP02

in less than a minute

CHIC offers a position for Waiter in None, for which you are 9.57% a match. Contact through UNESCOLSP02

less than a minute ago

The most common work positions in the area where you live are:

less than a minute ago

mechatronics assembler

less than a minute ago

domestic cleaner

less than a minute ago

waiter/waitress

less than a minute ago

domestic electrician

less than a minute ago

plumber

in less than a minute

industrial electrician

Figure 29. Example response of the chatbot with the most relevant job offers.

4.2 EDUCATION SCENARIO

The educational courses scenario can be used in combination with the job searching scenario to present the user with recommendations for specific courses that will boost the user's matching with career prospects. A form that correlates courses with specific abilities that will be acquired through the course is provided, similar to the employment opportunity skill definition dialogue.

The recommender system can then compare the required skills for a job description to the user's skill set to determine which skills are lacking for the user to be a perfect match. Then, a second comparison can be performed, comparing the list of missing skills to the skills provided by each course in order to identify the courses that will most benefit the user. Figure 30 demonstrates a part of the conversation tree for the educational courses scenario.

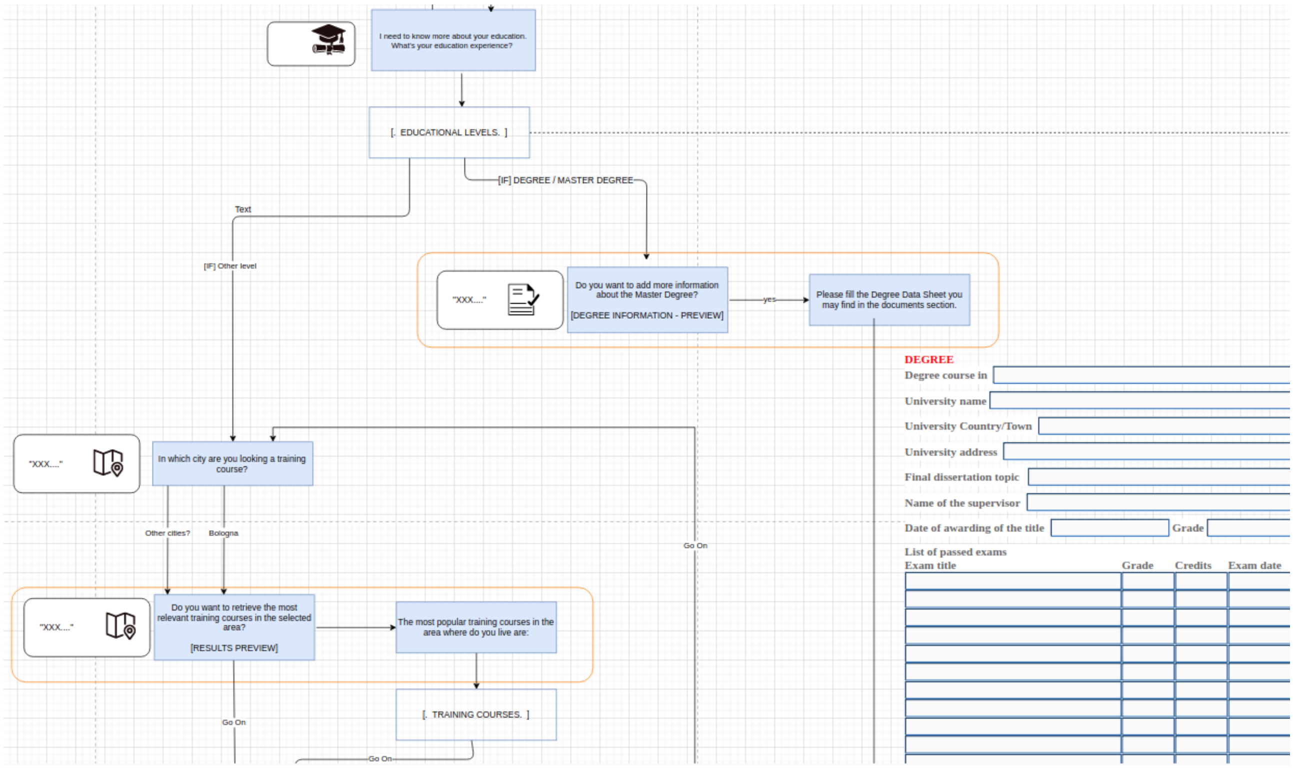

I need to know more about your education. What's your education experience?
[. EDUCATIONAL LEVELS.]
[IF] DEGREE / MASTER DEGREE
Text
[IF] Other level
"XXX...."
Do you want to add more information about the Master Degree?
[DEGREE INFORMATION - PREVIEW]
yes
Please fill the Degree Data Sheet you may find in the documents section.
DEGREE
Degree course in
University name
University Country/Town
University address
Final dissertation topic
Name of the supervisor
Date of awarding of the title
Grade
List of passed exams
Exam title
Grade
Credits
Exam date
"XXX...."
In which city are you looking a training course?
Other cities?
Bologna
Go On
"XXX...."
Do you want to retrieve the most relevant training courses in the selected area?
[RESULTS PREVIEW]
The most popular training courses in the area where do you live are:
[. TRAINING COURSES.]
Go On
Go On

Figure 30. A part of the Conversation tree for the educational courses scenario.

4.3 Healthcare Scenario

During the Health scenario, the immigrant uses the Digital Companion in the REBUILD mobile app to gain access to the REBUILD framework. After entering some personal information and selecting the Health domain, depending on the host country, the user is guided through a different scenario.

I. Healthcare Scenario in Greece

When the host country is Greece, the user is guided through a scenario targeting to book a medical appointment. The path that needs to be followed can be overviewed in Figure 31. Figure 32 shows an example of the Digital Companion using this conversation tree.

Initially, the questions posed to the user aim to gather the necessary information about the condition of the person in need. Questions such as "Is this an emergency?" and "What kind of doctor do you need?" can help discern both the criticality of the situation and the nature of the illness. If the user has trouble identifying the nature of the problem, the option to record a video or send a picture explaining the request can be sent to a human operator.

The second set of questions gathers information about the nature of the requested appointment. This includes the location of the user, i.e. in which city, and if the user prefers an appointment at a privately practicing doctor or at the public healthcare sector. Not all combinations are available because the public healthcare sector in Greece requires that a social security number (AMKA) be issued to the patient. Additionally not all doctor specialties are available in all of the facilities, i.e. there might be hospitals that do not have the requested specialty.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

Finally, in order to book an appointment, the patient's personal information is needed. This includes the name and surname as well as the social security number. Of course, before requesting the information, the user is notified that this information will be handled by a representative of the responsible LSP.

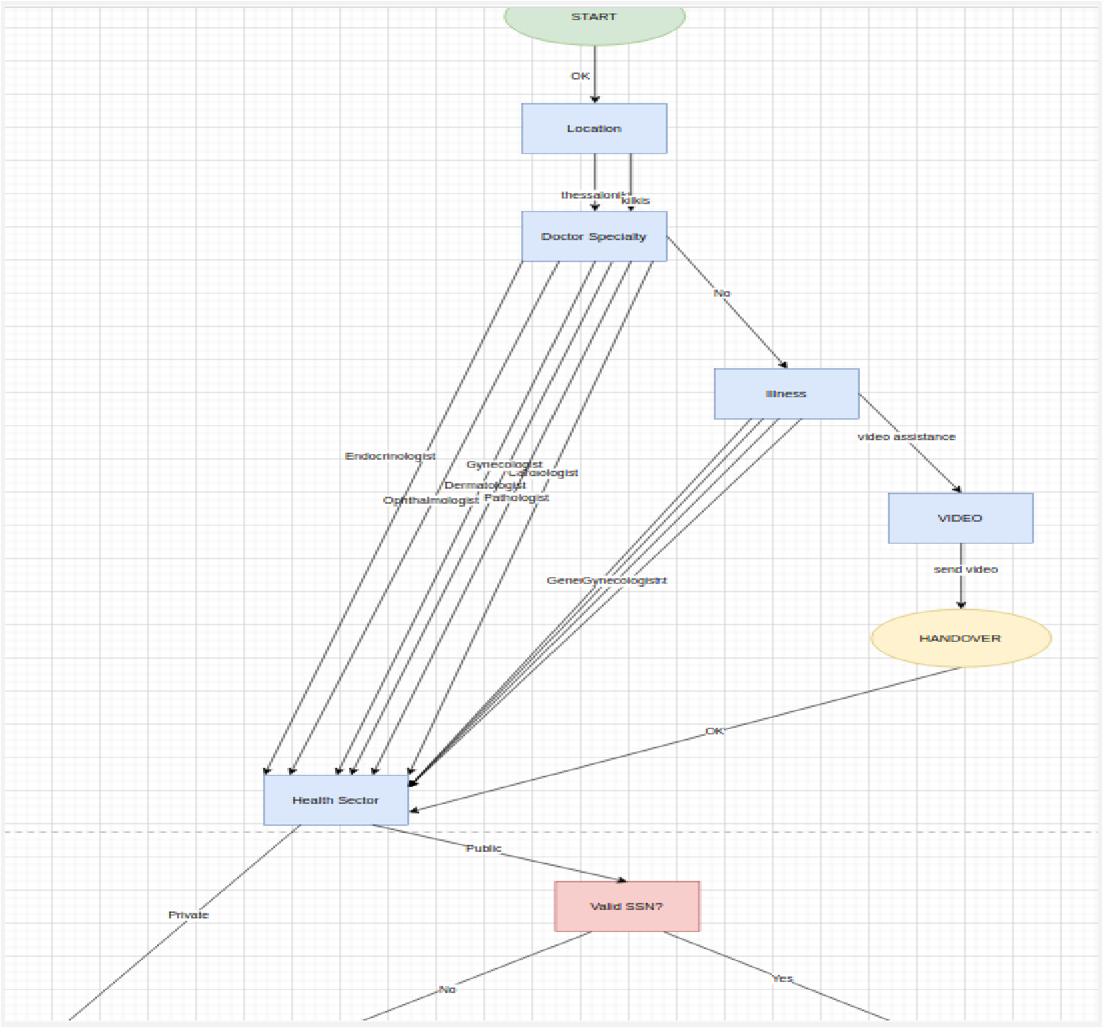
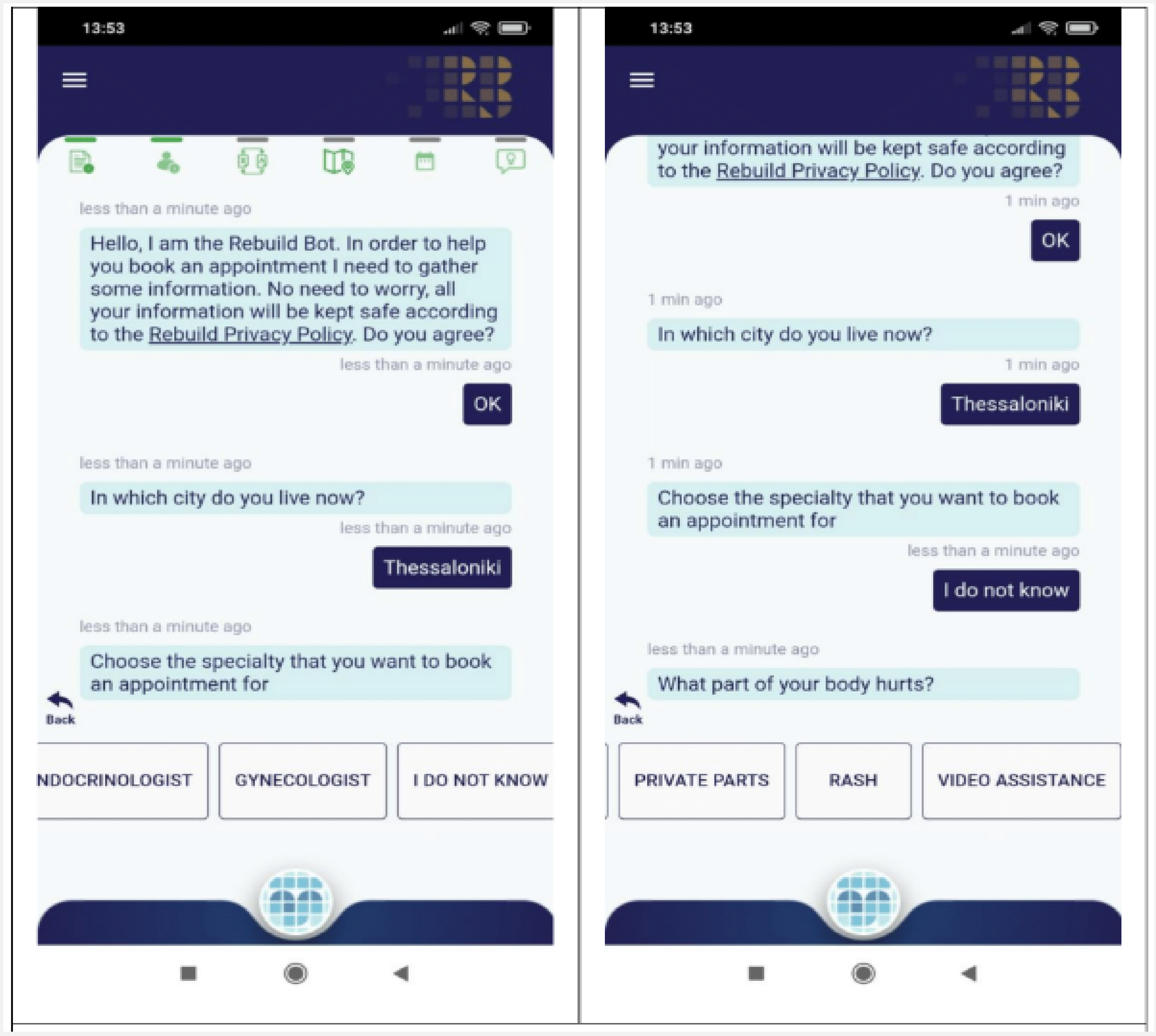


Figure 31. A part of the Conversation tree for the healthcare scenario.

At the time this deliverable is written, there is no way for an automated booking of a medical appointment in either the private or the public sector. Therefore, in the REBUILD framework, requests for appointments are transmitted to an LSP operator, who is seated in front of the REBUILD dashboard. Through the REBUILD dashboard, the operator can see the requests that are created by the chatbot as tasks and either respond himself/herself or delegate the requests by using the Task Solver application (deliverable D4.6).



less than a minute ago

Hello, I am the Rebuild Bot. In order to help you book an appointment I need to gather some information. No need to worry, all your information will be kept safe according to the Rebuild Privacy Policy. Do you agree?

less than a minute ago

OK

less than a minute ago

In which city do you live now?

less than a minute ago

Thessaloniki

less than a minute ago

Choose the specialty that you want to book an appointment for

your information will be kept safe according to the Rebuild Privacy Policy. Do you agree?

1 min ago

OK

1 min ago

In which city do you live now?

1 min ago

Thessaloniki

1 min ago

Choose the specialty that you want to book an appointment for

less than a minute ago

I do not know

less than a minute ago

What part of your body hurts?

Figure 32. The chatbot is using the conversation tree to gather information to assist the immigrant in completing an appointment.

II. Healthcare Scenario in Italy

On the other hand, the healthcare scenario in Italy is mostly informative. While there is no provision on booking a medical appointment, the migrant is guided through a series of questions to the correct information regarding his/her status.

The foremost situation that must be checked is if the migrant is in a health emergency. In this case, information about how first aid services can be accessed is presented to the user. Having ascertained that there is no immediate danger, the chatbot will then provide information about how a doctor can be selected in the Italian healthcare system and what the necessary documents are for visiting said doctor. Additionally, information on how medicines can be purchased is also provided.

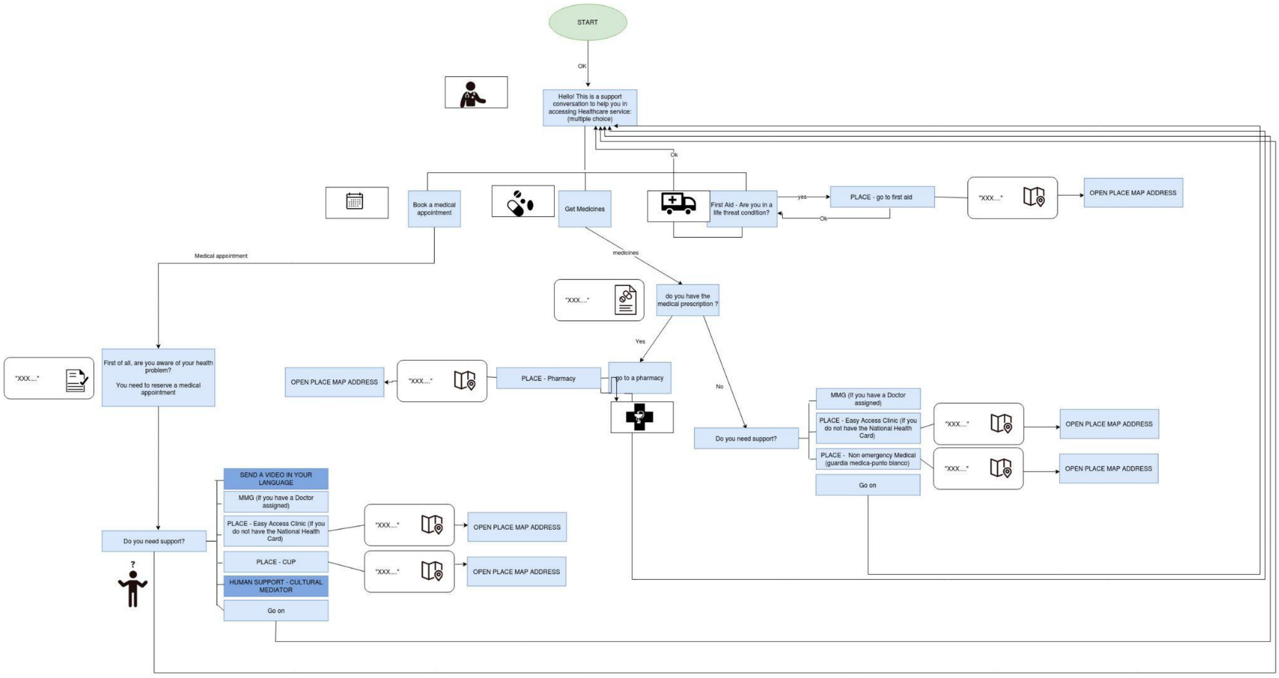

START
OK
Hello! This is a support conversation to help you in accessing Healthcare service: (multiple choice)
Ok
Book a medical appointment
Get Medicines
First Aid - Are you in a life threat condition?
yes
PLACE - go to first aid
Ok
"XXX...."
OPEN PLACE MAP ADDRESS
Medical appointment
medicines
"XXX...."
do you have the medical prescription ?
Yes
No
"XXX...."
First of all, are you aware of your health problem?
You need to reserve a medical appointment
OPEN PLACE MAP ADDRESS
"XXX...."
PLACE - Pharmacy
go to a pharmacy
Do you need support?
MMG (If you have a Doctor assigned)
PLACE - Easy Access Clinic (If you do not have the National Health Card)
PLACE - Non emergency Medical (guardia medica-punto bianco)
Go on
"XXX...."
OPEN PLACE MAP ADDRESS
"XXX...."
OPEN PLACE MAP ADDRESS
SEND A VIDEO IN YOUR LANGUAGE
MMG (If you have a Doctor assigned)
PLACE - Easy Access Clinic (If you do not have the National Health Card)
PLACE - CUP
HUMAN SUPPORT - CULTURAL MEDIATOR
Go on
Do you need support?
?
"XXX...."
OPEN PLACE MAP ADDRESS
"XXX...."
OPEN PLACE MAP ADDRESS

Figure 33. Depending on the needs of the migrant, the chatbot can guide him/her on how to visit a doctor or get access to medicine

4.4 SOCIAL MENTORING SCENARIO

The social mentoring scenario is another example of how the recommender can be used. In the social mentoring scenario, users who apply to be mentors and users who apply to be mentees are compared. In terms of matching inputs, they are requested via the chatbot via a common conversation, along with the user's designation as mentor or mentee.

Depending on the selected role, the user interacts with the chatbot to provide information that the recommender will use to produce possible matches. Table 1 contains a list of the data collected for each role. Some additional information is acquired during personal interviews, but it is only utilized to determine program eligibility and is not derived from the recommender system.

	Mentor	**Mentee**
	Link with the program (Student/Teacher/PAS)	Time availability for meetings
	Specific link (Field of study/Field of teaching/Activity)	Location
	Time availability for meetings	Languages spoken
	Location	Prefered language for interaction
	Languages spoken	Expectations from the mentor
	Prefered language for interaction	Hobbies
	Contributions for the mentee	Gender

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 822215

	Hobbies	Gender preference
	Gender	Age
	Gender preference	Age preference
	Age	
	Age preference	

Table 1. Information gathered and used for creating matches in the social mentoring scenario.

5. CONCLUSION

This document describes the work completed thus far in order to implement the Digital Companion for the purposes of REBUILD. The Digital Companion is made up of two kinds of chatbots: a rule-based and an open-domain chatbot. Both types of chatbots use a simple messaging exchange server (Chat Server) that was developed in order to facilitate all communications.

On the one hand, the rule-based chatbot developed to adhere to strict rules will be the one to address any potential critical issues that migrants may face with a first response. If the chatbot is unable to handle a situation, the contact will be forwarded to community members, volunteers, or successfully integrated migrants. On the other hand, the open-domain chatbot is developed for producing daily human-to-human conversations, which may be useful for migrants attempting to learn a new language. The development of this multi-language open-domain chatbot is based on the pre-trained DIALOGPT model fine tuned on PERSONA-CHAT dataset in combination with a translation framework based on Transformer models.

In addition, a social event recommendation system is implemented adapted to the Conversational Agent to provide important guidance to migrants in interacting with the local societies. To improve the performance of this recommendation system, a personalization tool was developed that is based on the *Sentence Transformers* library. The embeddings of these sentences-social-events are extracted using a transformer-based model that is available in *Sentence Transformers* repository*.* Finally, these embeddings are compared with the user-profile embeddings using the cosine similarity metric.

6 REFERENCES

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. 2019. BERT: Pre-training of deep bi-directional transformers for language understanding. *NAACL 2019.*

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, Jason Weston. 2018. Personalizing Dialogue Agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, Bill Dolan. 2020. DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation. *arXiv preprint arXiv:1911.00536* (https://github.com/microsoft/DialoGPT)

Thomas Wolf, Victor Sanh, Julien Chaumond, Clement Delangue. 2019.TransferTransfo: A Transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149* (https://github.com/huggingface/transfer-learning-conv-ai)

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. 2018. Improving language understanding by generative pre-training. *Technical report, OpenAI*

Lorenzo De Mattei, Michelle Cafagna, Felice Dell'Orletta, Malvina Nissim, Marco Guerini. 2020. GePpeTto Carves Italian into a Language Model. *arXiv preprint arXiv:2004.14253*

Marco Baroni, Silvia Bernardini, Adriano Ferraresi,and Eros Zanchetta. 2009. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation, 43(3):209–226.*

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv preprint arXiv:1908.10084*

REBUILD

ICT-enabled integration facilitator and life rebuilding guidance

Deliverable: D4.8 The digital companion prototype

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 822215.