BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

SPECIALIZATION COMPUTER SCIENCE ENGLISH

DIPLOMA THESIS

TASKS MANAGEMENT USING CONVERSATIONAL AGENTS

Supervisor	**Author**
Lect. Dr. Sterca Adrian	**Timofte Alexandru Nicuşor**

2018

Contents

Abstract

This paper presents a concept platform that allows its users to easily manage their daily tasks using a text based software, namely a chatbot. The users can interact with their cloud based task management tool, for example Gitlab, talking to the chatbot platform like to a normal human through a chat application like Slack. The platform also allows users to be announced real-time when other teammates are updating or creating new tasks. The application consists of a web API which communicates with Slack API for interacting real-time with the user on the chat, Dialogflow API for understanding and processing the text from the user, and Gitlab API for interacting with the user's tasks. The web API was created using Javascript's server interpreter, NodeJS and MongoDB for data persistence.

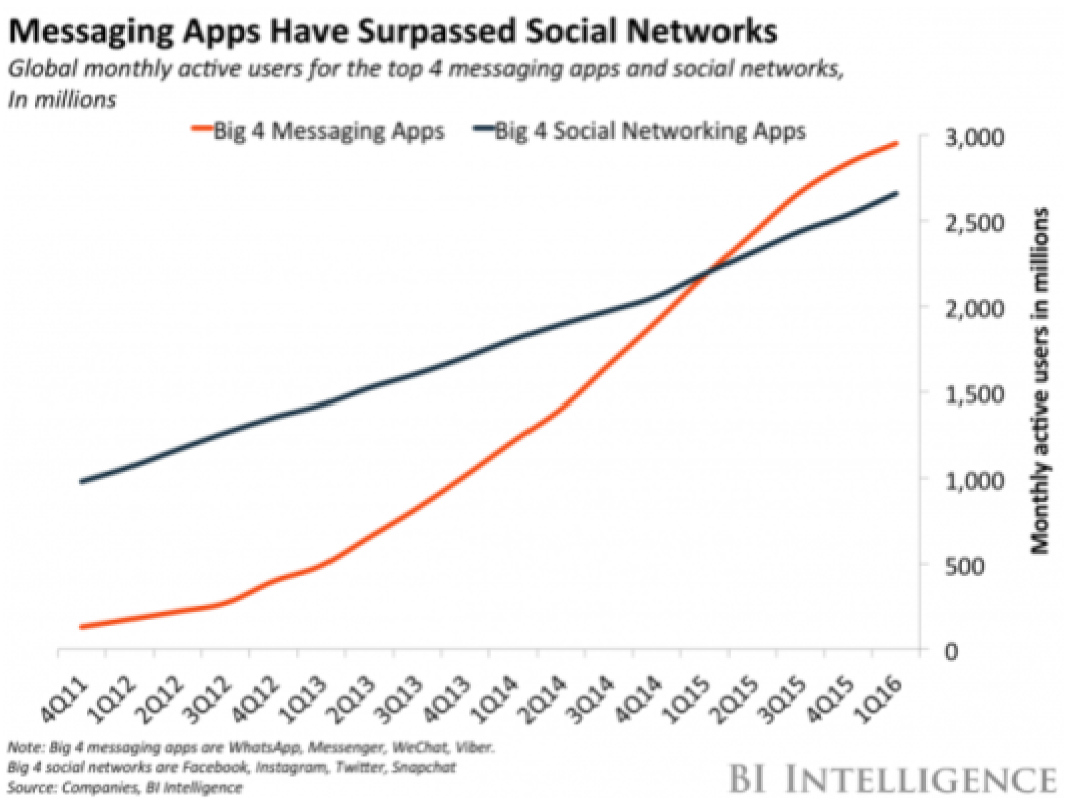
This paper is structured as follows. Chapter 1 provides an introduction, the reason behind creating this application and why this solution is needed. Chapter 2 presents some current possible solutions to the problem described in Chapter 1, and analyses their issues and faults. Chapter 3 goes through the main solutions used nowadays and why the application is superior to current solutions. Chapter 4 presents the application in details, both from a theoretical aspect and from a practical aspect, also providing samples of code throughout. Chapter 5 brings a conclusion and a short summary of the solution presented, while Chapter 6 contains the bibliography and inspiration used for the paper.

This work is the result of my own activity. I have neither given nor received unauthorised assistance on this work.

1. Introduction

In 2015, the top four social networks had been surpassed the user base of the top 4 messaging apps worldwide regarding the size of the combined user base.[1] Messaging apps have turned into huge platforms for which developers have a big incentive to create software. This kind of software is commonly called “chatbots” and is predicted by companies such as Microsoft and Facebook to take over much of the interaction currently happening in separate apps (mobile/desktop applications).

Chatbots, additionally called machine conversation systems, chetterbots, virtual agents and dialogue systems, are computer software that clients essentially interact with utilizing natural language. They offer entertainment or facilitate with anything from weather forecasts, shoe searching and booking conferences, to money recommendation or a virtual friend one can discuss with.



[1]

Figure 1: Monthly active users Messaging apps vs. Social networks

[1] "Messaging apps are now bigger than social networks - Business Insider." 20 sept. 2016, https://amp.businessinsider.com/images/5970ca60552be5ad008b56ce-1920-1527.png

1.1 Context

Effective task management is crucial to successful team collaboration. Whereas the previous decade has seen extended innovation in systems that manage and track clusters tasks, these innovations have generally been outside of the principal communication channels; email, chat, messenger. Groups formulate, refine, track, assign, and discuss the progress of their cooperative tasks over different of channels, nevertheless they have to leave these channels to update their task-tracking tools, making a supply of inefficiency and friction.

Companies of all sizes and area of expertise are always looking for ways to optimize time and ensure better work productivity of their teams. To achieve all this goals, there is a plenty of project management tools that teams are using. Some relevant examples are Jira from Atlassian, Gitlab and CodebaseHQ. These software tools enable users to easily do a variety of tasks regarding project management. Some of the most in demand features are described as follows[2]:

- **Create tasks and subtasks** - Break down work into separate tasks, split bigger ones into smaller subtasks and keep the work organized.
- **Set time estimation** - Set approved hours on tasks or time estimates to track the progress of the tasks and plan every phase of the project.
- **Track time spent on tasks** - View and track time to see how much time you and your staff spent working on tasks and subtasks.
- **Assign multiple people on taks** - Do a better project planning by assigning tasks to people based on their expertise.
- **Mark labels** - Use labels to categorize similar tasks and to define their nature, status, priority and severity.
- **Comment on tasks** - Directly add comments on tasks for a quick response. Ask questions, offer insights & provide all the additional information.

- **Complete tasks** - Whenever you have finished working on a task assigned to you, you can simply mark the tasks as complete with a single click.

1.2. Problem definition

Most of the software tools used for project management are doing a great job, but most of the people are finding it hard to use because of the complexity it implies, and many times they have a bad user interface and a non trivial user experience. To use such a platform it is mandatory to have an onboarding process, where the user takes tutorials and learns how it properly works before starting to use it.

Another problem with these platforms it that they are almost always web based, because the complexity of the business and the number for actions a user has to make are so high that it is almost impossible to implement it on a mobile application. It happens many times that users want to manage their tasks when they are not at a desk, or not having access to their PC or laptop, for example when being in a bus while going to work, or waiting at a queue. All these platforms make it impossible to be used from a mobile device, thus the user being constrained to work only from a desktop device.

Combining the two, task management was not ever done without a bit of challenge. But this application aims to make it easy to do task management and it integrates seamlessly in the day-to-day life of its users, while also providing enough tools for someone to manage to be productive using the app.

1.3. Proposed solution

Employees are expected to handle different tasks according to the demands of a project. While juggling different tasks isn’t easy, things becomes especially difficult if everything is done manually. Chatbots are effective tools for automating tasks because, unlike Excel, these tools can trigger commands helping you easily create and assign tasks[3].

The chatbot application provides a simpler and easier way to manage the tasks and automates the process by cutting the struggle of accommodating with the environment of a user interface based platform. The proposed solution is a simple chatbot, which can be used as a Slack plugin and needs to be installed only once per organisation in order for everyone to benefit from its capabilities. In the organization's chat, the chatbot can be found in the list of users and everyone can interact with it. The initiator user cand ask it different questions regarding his project tasks from Gitlab, and after the user authenticate to give the bot access to his Gitlab account, it(the bot) can respond with relevant information. The user can ask the bot to do different things like:

- showing the issues based on different criterias like scope or state
- assigning issues on other teammates
- editing issues
- changing the state of the issue, thus closing it when it's finished
- receiving real-time updates regarding the projects's tasks

Thus, by introducing the chatbot solution, users are now able to be device independent, because the chatbot can be easily used from a mobile device, besides desktop and web. Now, they can use their favourite chat platform in order to manage their daily tasks. Moreover, once they open the chat application, there's no need to open any other tool in order to track the daily job. As intelligent programs, chatbots can quickly sift through large amounts of data, locate the right information and execute a set of tasks based on it, thus dealing with buttons, lists, filters, endless and embarassing actions like login, and finding the right place to do the right thing will become obsolete.

2. Theory about chatbots

This chapter describes the theory of the chatbots and the current most used approaches to the problem. It also goes a little bit into each approach and its downsides for our goal of achieving an easy to use task manager chatbot platform .

2.1. Conversational Agents

Conversational agents are software programs that support conversational interaction between humans and machines in natural language. They assist, help, manage, organize or perform user's day-to-day tasks like knowing about the weather conditions, booking airline/movie tickets, checking for the best restaurant/shops around, getting updates about the traffic in the city, talking to customer care service and many more. Conversational agent is a collection of behavioral components that can interpret, sense, trigger and respond or reply to user.

2.2. Chatbots history

Eliza was the first chatbot ever created. It was developed in 1966 by a professor from MIT named Joseph Weizenbaum.

Around year 2009, a more advanced bot was created by a chinese company called WeChat. This platform made it easier to create simple chatbots. It was considered the best way for employers and marketers to reduce their work as they interact with customers online.

WeChat has many implications and it's less performant than the other messaging apps competitors Slack, Telegram, Facebook Messenger, but still can build a very smart bot on it. In 2012, a former Google employee founded Chumen Wenwen Company which has built a most sophisticated bot running on WeChat.

In 2016 we started to see the first wave of artificial technology in the chatbots' creation and design. Developers were enabled by apps like Facebook to build their own chatbots in order for their clients to carry out some of their daily tasks from the messaging platform.

We are brought to the time of conversational interface by the coming of the chatbots. This type of interface won’t need a screen or a mouse. The interface will be soon conversational and all the conversations will be the same as we have with our friends.

We have to go some steps back, in the day when the computer appeared and artificial intelligence and conversational interface begun, in order to understand what it’s all about[21].

ELIZA

The chatbot uses substitution methodology and pattern matching to simulate conversation. Eliza was designed to mimics human interaction and it worked by pairing the words received to a list of possible scripted responses.The script was considered a big impact on the natural language processing.

While Weizenbaum intended the bot to be more like a caricature of human conversation, the users were telling to the bot the most profound thoughts. Back then, the experts were declaring the withing a couple of years the bots would be the same as humans.

Weizenbaum disagreed the fact that machine could replace human capability of the mind. He argued that computers’ understanding of the language was dependent on the circumstances they were used, and they were just tools which could be considered as extensions of the human mind. He added that a more computer understanding of human language was not possible.

PARRY

In 1972 the American psychiatrist built Parry, a program that simulate and imitate a patient with the disease of schizophrenia. Basically, it resembles the thinking of an individual.

The bot works using a complicated system of attributions and assumptions which were triggered by the changed weights assigned to verbal inputs. It was tested using the Turing test in order to validate the work.

JABBERWACKY

The bot was built in 1988 by developer Rollo Carpented. It's main purpose was to simulate human conversation in an entertaining way.

The bot led to some other technological growth, as it was used for academic research. The bot was smarter than others before, as it was considered to use the AI technique “contextual pattern matching”.

Dr. SBAITSO

The chatbot was build for MS-Dos in 1992 by Creative Labs. It would converse with it's users from a psychologist perspective and it was one of the earliest work of introducing AI into a chatbot.

SmarterChild

The chatbot was created in 2001 and it was available on MSN Messenger and AOL IM. It could carry out fun conversation, with data access to other services.

Siri

Siri was created in 2010 by Apple and it is a personal assistant which which uses natural language as user interface and learns were the user navigates.

The bot was not limited to interact only in text, but it could also reply to text, audio, video images when the user transferred to it.

Google Now

The bot was launched by Google in 2012 and it was a Siri competitor. It could perform actions through requests and could also answered questions.

Google Now could initially get appropriate informations based on context such as time of the day and location. It’s currently build for use in smartphones, and can be seen as a predictive search.

The bot was part of some user interface modifications for Google products regarding mobile search. From the beginning it was a female-voiced portable assistant. Google Now is considered one of the more aggressive growth method of Google’s search.

Cortana

Cortana is bot created by Microsoft and launched in 2014 at Microsoft’s Build conference. It was integrated bot on Windows 10 PCs and Windows phone.

Cortana uses voice recognition to get and respond commands. It can perform a variety of tasks like setting reminders, playing games, find locations, files etc.

Alexa

Alexa was introduced in 2014 by Amazon and it’s integrated in to Amazon’s devices Echo, Show, Dot and others.

The personal assistant uses only the sound of the voice to search on the web, create shopping list, play music, get news or weather reports, control smart home devices and many others.

A valuable feature of Amazon Alexa is that developers can create skills for it using Alexa Skills Kit(ASK). In this way you can add extra capabilities to Alexa enabled devices.

2.3. Clasiffication of Conversational Agents

In this section we divide conversational agents into different categories. The purpose of this division is to understand the mediums of communications with the users, how are they used on different domains and on different devices.

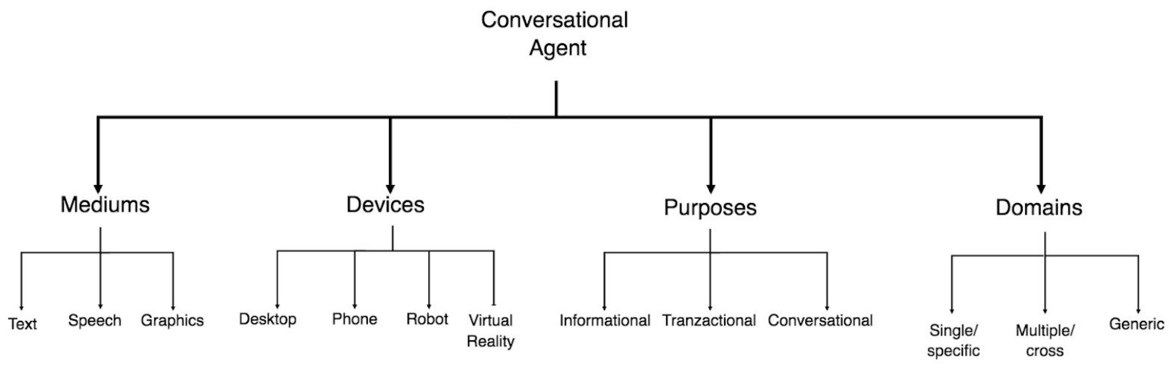


Figure 2: Classification of Conversational Agents

Mediums

The interaction between a human and a conversation agent text or speech based or it can be a graphical user interface system. In text based interaction the conversation is done using text, in spoken dialogue system it based on speech/audio format and for graphical user interface the interaction is based on graphical elements. A system is not limited for being developed on a single medium.

Devices

In the past the primary devices for using a conversation chatbot were desktops/PCs and laptops. But as the growing of the mobile phones usage, they can be used on phones based applications also.

Purposes

There are 3 different purposes which a conversational agent might serve. One of them is to provide information to user, thus being a informational agent. They can help user with things like travel timetable or weather information, so they are used in question answering systems to provide informational answers. Another purpose might be purchasing a bus or a movie ticket, which can be made by transactional conversation agents and the third one is conversational which is planed just for the sake of conversation with their users.

Domains

The conversational agents might be used in a variety of domains. The ones which serve only one domain are called single domain agents. Multiple domain agents are serving more than one domain, and the last is the generic one, which serve for the general conversation like small talk.

2.4 Chatbot as Text-based Conversational Agent

A chatbot is using a text as the only medium of communication and is used on desktops/laptops, majorly on mobile phones on different messaging applications. Such examples are Telegram, WeChat, Facebook Messenger, Slack which are used by millions of people to communicate with their colleagues, friends and companies[1]. The fact that messaging platforms are growing in popularity is driving a new era of interaction with chatbots. Chatbots development and discovery is now a functionality provided by messaging applications. There are currently two ways of using chatbots. One of them is represented by those integrated into a messaging application and the other one are those using existing platforms, such as WeChat, Telegram, Slack, Viber, etc. One main difference is that for the second type you can use already existing messaging application on your phone, while for the first one you have to install and use a dedicated application to communicate with a chatbot(e.g. Revolut). Some applications such Facebook Messenger, Tweeter, Telegram are widely used because they provide their own API or SDK for programmers to develop chatbots and integrate them easily into their chat applications.

I perform a short research on four widely used messaging platforms in order to understand the methodology they provide while developing a chatbot, their features, interfaces and functionalities. This survey can be useful to understand which messaging platform would be a suitable platform for implementing a chatbot in my thesis work. Table 1 shows messaging platforms and their different attributes.

	Slack	Facebook Messenger	Telegram	Twitter
Requirements/ Set up	Get Api Token	Set Up and integrate webhook	Get API token from BotFather	Getting keys and acces Token, Account verification by phone number
Input Format	Text, graphics, files	Text, speech, graphics	Text, graphics	Tweets
Output Format	Text, graphics, files	Text, speech, graphics	Text, graphics	Automatic following, Bot automatically responses to the user
Starting Point	By the user on mobile phone/Web/ Desktop	By the user on mobile phone, by searching in messenger/ scanning code/ Linked ads	By the user on mobile phone/ Telegram Web, User should search the Bot	Automatic following, Bot automatically responses to the user
Communication management	Slack SDK / Webhook	Webhook, The Web Service API, web	Communicatio n with intermediary telegram	REST API queries over a long-lived HTTP connection. Receives updates

		plugin to embed Messenger bots right on your website	server via simple HTTPS interface	on the latest Tweets matching a search query
Development support	Slack API	Own API named as Wit.ai Bot Engine	Telegram API	Twitter API
Message Updates	Messages updates in JSON, posted in Webhook	POST logs in JSON, Entries in Webhook	Messages updates in JSON format with Message information and logs	With your own tweet repository for your bot and logs
Special Features	Custom loaded messages, Reminders, Feeds subscription	Prediction for next actions, Natural Language Understandin g Assistance, AI features	Inline mode, bot can be added in group chat	Game and toys development, domain expertise, IoT interfaces, virtual assistance

Table 1: Survey of existing Messaging Platforms

While providing a nice user interface, these messaging platform frameworks also make the integration and development of a chatbot into different applications easy. They are used in wide variety of applications such as Customer support, where chatbot can quickly provide answers and address customer complaints, E-commerce where the chatbot can let you build your customized product and track your order. When it comes about news, a chatbots can get the latest headlines from mainstream media like CNN, Fox News, the Guardian, or you can get the latest tech headlines from TechCrunch or Engadget. Chatbots can also help you searching and tracking flights, by getting some vacation inspiration or comparing flights based on price and location, or even schedule a meeting by requesting in a Slack channel in order for the bot to look at everyone's calendars to find times when everyone is available[4].

2.5 Architecture of a Chatbot

In this subsection we aim to give an overview of a typical architecture to build a conversational chatbot. We will review the architecture and the components in detail.

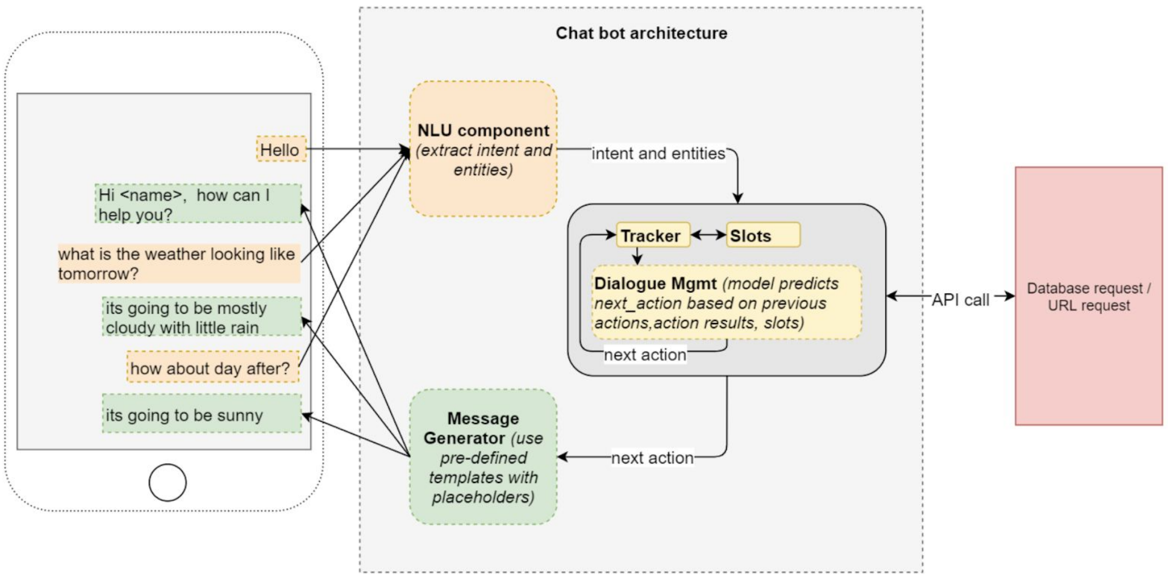

Chat bot architecture
Hello
Hi <name>, how can I help you?
what is the weather looking like tomorrow?
its going to be mostly cloudy with little rain
how about day after?
its going to be sunny
NLU component (extract intent and entities)
intent and entities
Tracker
Slots
Dialogue Mgmt (model predicts next_action based on previous actions,action results, slots)
next action
API call
Database request / URL request
Message Generator (use pre-defined templates with placeholders)
next action

Figure 19. Chatbot architecture[2]

[2] "Conversational AI chat-bot — Architecture overview – Towards Data ...." 9 feb.. 2018, https://cdn-images-1.medium.com/max/2000/1*0JqMq7W9ndcOmL3UQ5lPhQ.png

There are three main aspects regarding the architecture of a conversational agent:

1. A message processing begins from understanding what the user is talking about. Intent classification identifies the intent of user message. Typically it is selection of one out of a number of predefined intents, though more sophisticated bots can identify multiple intents from one message. Intent classification can use context information, such as intents of previous messages. Entity recognition extracts structured information from the message. For example the weather bot can extract location and date.

2. To process the user request the candidate response generator is doing all the domain-specific calculations. It can use different algorithms, call a few external APIs, or even ask a human to help with response generation. The result of these calculations is a list of response candidates. All these responses can't be just tons of random responses, it should be correct according to domain-specific logic. The response generator must use the context of the conversation as well as intent and entities extracted from the last user message, otherwise, it can't support multi-message conversations[5].

Referring to the above figure, the NLU(Natural Language Understanding) component helps extracting the intent and entities the user request. The NLU implies the following components:

- entity extraction model
- supervised intent classification model that is trained on a wide range of sentences as input and intents as target

3. When the chatbot is conversational, the required response is retrieved from the data sources. But when the chatbot is designed for transactional purposes, then the task manager executes the further actions. Task manager retrieves the required information from the data sources to provide to the user or executes the action which user asked to perform. After this, a response message is generated by response generation component and sent back to the user.

This is the high-level description of different components of a chatbot. The way each component works is discussed in detail in subsections below.

Natural Language Understanding

Natural language understanding component is first processing every incoming user request. There are different strategies used for Natural language understanding in a chatbot. The user request is parsed by the natural language component. Then, the NLU tries to get a meaning out of it by understanding user's intention and information associated with that intent. Entity extraction is identifying and extracting entities (people, places, companies, etc.) from the user request. After this pre-processing, structured data can be fed to natural language understanding strategies which actually take out the meaning of the structured data. Chatbots sometimes directly involve natural language understanding strategies by skipping the pre-processing of user requests and feeding user request directly to NLU unit as it is. Natural language understanding strategies used in chatbots are:

- **Pattern matching**: can classify text in order to produce a suitable response for customers. Artificial intelligence markup language(AIML) is a standard structure of these patterns. A simple pattern matching example[5]:

  ```
  <category>
          <pattern>What is his name</pattern>
          <template>His name is Michael N.S Evanious.</template>
  </category>
  ```

  The machine then gives an output[6]:

  Human: What is his name?

  Robot: His name is Michael N.S Evanious.

  Because the name is in the associated pattern the chatbot can infer the answer. Chatbots respond to anything related to the associated patterns. The drawback it that they can not go beyond the associated pattern. But here algorithms can help, taking it to a more advanced level[7].

- **Machine learning**: the problem of pattern based heuristics is that patterns should be programmed manually, and it is not an easy task, especially if the chatbot has to correctly distinguish hundreds of intents. Humans are not good at writing patterns and rules for natural language understanding, computers are much better at this task. Machine learning lets us train an intent classification algorithm. You just need a training set of a few hundred or thousands of examples, and it will pick up patterns in the data[5].

Nowadays various automatic tools, frameworks and APIs are available for natural language understanding and processing for the different methods mentioned above. Some of these tools and APIs are: Wit.ai, Dialogflow, Microsoft LUIS, Google Natural Language API, Init AI, Amazon Lex, IBM Watson etc. These tools simplify the natural language understanding process of a chatbot.

Dialogue Manager

The primary function of a dialog manager is to find a response when the user says something. It can be a question to the user for more information, a direct answer to a question or stating that is does not understand what the user said. It helps chatbot to actually understand that every incoming statement is in a connection with previous statement or answer of a bot’s follow-up question or all together a new statement. Dialogue manager is the coordinator of al the other components of a chatbot. It is involved in four main tasks such as providing a context for interpretation, coordinating other components, updating context of the conversation and deciding the information to convey and when to do it. The following are the dialogue management strategies that are used for handling conversations in a chatbot[8]:

- **Switch statement**: the user always has initiative in this type of dialog. Moreover, this kind does not keep conversation context and won’t remember anything previously said, thus the conversation won’t feel engaging since the input will always come from the user. The advantage in this case is that it’s easy to implement and maintain. For example it can be used for a frequently asked question task.

- **Finite state machine(FSM)**: this type is more exciting, since the bot can navigate to the next step based on what user said, and the user can select the direction the conversation should take. Many tools that can build conversational agents also provide tools for making decision diagram, thus having a FSM underneath, which is good when the number of things a user can say are limited. But there are some drawbacks which consist in limiting the flexibility. For example the conversation will go each time through the same steps even the conversation does need to skip some steps, like providing the home address when ordering something.

- **Goal based**: Thinking in terms of goals is a popular and better way you can think about dialogs, different from thinking as set of states, because in complex conversations the number of states cand quickly become unmanageable.
  Let’s suppose that your user ask for the location of a restaurant without giving its name. First of all, the system will receive a “looking_for_restaurant” intent and start a new goal “finding_restaurant”. After that, it will notice that to finish this goal it needs to know the name of the restaurant. Then it will ask the user for the name. When the user answers it will first analyze this response to see if it contains the name of the restaurant. If it does, it will save the name in its context. Finally the system will see if it now can finish the “finding_restaurant” goal. Since the name of the restaurant is now known, it can lookup the restaurant’s location and tell it to the user. This type of dialog makes it easier to manage different ways of asking the same questions or making decision based on what you know about the user, however the setup is certainly more difficult.

- **Belief based**: the dialog manager can only assume what user said and actually can't work with discrete rules because most NLU will classify intent and entities with a certain degree of uncertainty, so the bot needs to work with beliefs. The more complex the system is, the more likely the NLU makes the wrong decision. Dialog managers that work with uncertainty are usually based on Markov Decision Processes(MDP) and Partially observable Markov Decision Processes(POMDP).

Data Sources

Backend database, third party solutions, repositories containing information and content that will be delivered through the bot are just a some parts to consider planning a chatbots data source. Almost any useful chatbot will need to connect to data. During the design process, there are many details to sketch[10]:

- The data source - where the data is coming from: 3rd party APIs, bot specific databases, existing databases)
- The format of the data
- The queries underlying each dynamic component
- The relationship between the datasets

Chatbots use different data sources according to its requirements and domain. They sometimes use available AIML templates or available rules structure for understanding user requests and provide answers accordingly. For example, chatbots use weather APIs like 'openweathermap' to provide the current weather information and weather forecast. It can also rely on external applications like regional transportation applications, weather applications etc. to provide the information to users when chatbot is informational. In the case of the transactional chatbot, task manager coordinates with these external services or applications to perform some tasks, e.g. performing a task of booking hotel with booking.com

Response Generation

After information is retrieved or an action is performed, the chatbot needs to prepare a response message. This task is done by response generation component. There are two types of the response generation models as follow[11]:

- **Retrieval-based models(easier)** use a repository of predefined responses and some kind of heuristic to pick an appropriate response based on the input and context. The heuristic could be as simple as a rule-based expression match, or as complex as an ensemble of Machine Learning classifiers. These systems don't generate any new text, they just pick a response from a fixed set
- **Generative models(harder)** don't rely on predefined responses. They generate new responses from scratch. Generative models are typically based on Machine Translation techniques, but instead of translating from one language to another, we "translate" from an input to an output (response).

There are pros and cons on both approaches. Retrieval-based methods don't make grammatical mistakes, due to the repository of handcrafted responses. But may often be unable to handle unseen cases for which no appropriate predefined response exists. Generative models can refer back to entities in the input and give the impression that you're talking to a human. They are "smarter". But these models are likely to make grammatical mistakes, typically require huge amounts of training data and are hard to train.

2.6 Chatbot Related Challenges

Despite popularity and large growth in chatbots, they still face a lot of challenges and far from being perfect. In this section, we try to understand the challenges those current chatbots face[12, 13, 14, 15].

- **The NLP component**
  This component understands a free-form text or voice utterance and dissects it into intents and parameters. Note that NLP can only help your application dissect a sentence to a set of intents that you can take action on programmatically. While NLP discovers the intents from the conversation, software developers are on their own to figure out how to respond or take action on these intents.

- **Meaningful responses**

  Achieving logical responses is another challenge to overcome while building a chatbot. The chat bot must be well-equipped to answer consistently to inputs that are similar in linguistic terms.

  An intelligent chat bot must provide the same answer to queries which are similar in meaning. Though it looks straight forward but incorporating logic and sense into the model is a challenging task. The remedy to this challenge is to train the chatbot to produce consistent logical answers.

- **Context integration**

  Sensible responses are the major requisite of the chatbots. Integrating meaningful context into the chat bot is the first challenge while building a chat bot.

  There must be integration of both the physical as well as linguistic context to produce sensible responses. In order to incorporate the linguistic context, conversations are submerged in a force that becomes a challenging objective. While integrating contextual data, location, time, date or details about users and other such data must be integrated with the chat bot.

- **Reading Intention**

  In some cases, reading intention becomes a challenge. Take generative systems, for instance. They provide generic responses for several user inputs. The ability to produce relevant responses depends on how the chatbot is trained. Without being trained to meet specific intentions, generative systems fail to provide the diversity required to handle specific inputs.

- **Limited attention span**

  User attention span is limited and often users are very distracted, so it is not only that we understand them. Here is where conversational UI is at play. It is more about how can we hook them. So how you respond to a user message is where you grab user’s attention. The more effectively you do the more chances are to be used again. So writing responses to user queries should be taken very seriously.

- **The context or memory of a bot**
  To enable human-like interaction via a chatbot, the developer must maintain the context or memory of the conversation from beginning to end. Some chatbots need to maintain that context per user to be able to offer a personalized experience and history for a customer. If I book an appointment via a chatbot with a doctor for example, I would like that conversation to remember that context and remind me later that I have an upcoming appointment. More advanced chatbots might know my name, email, address, and so on.

2.7 The use of Chatbots in Business

Online chatbots save effort and time by automating customer support. The open doors gave by chatbot frameworks go a long way past offering reaction to clients' request. They are additionally utilized for some, different business undertakings, such as diminishing overhead costs, gathering data about clients and arranging gatherings. There is no big surprise that size of the chatbot advertise is developing exponentially.

There are different reasons why a business needs a chatbot, as concurrent handling of numerous requests from clients and disposing of routine undertakings. In addition, an enormous speed of handling customers' requests with chatbots helps getting customers' loyalty.

Customers also benefit from chatbots and they are getting progressively intrigued by this innovation. An investigation shown at the fourth International Conference on Internet Science in November, 2017 recognized reasons why individuals cooperate with chatbots. As indicated by this examination, the primary factors that motivate individuals to utilize chatbots are[16]:

- **Curiosity.** The oddity of chatbots sparkles interest. Individuals need to investigate their capacities and to have a go at something new.

- **Excitement.** Chatbots entertain individuals by giving them interesting tips, they likewise help killing time when clients have nothing to do.

- **Profitability.** Chatbots give the help or access to data rapidly and productively.

- **Social and relational variables.** Chatbots fuel changes and improve social encounters. Visiting with bots likewise stays away from loneliness, allows to talk without being judged and enhances conversational abilities.

So what can chatbots really do?
"I think chatbots are the future of engagement between a fan and a brand or celebrity." ~ Christina Milian

Startups and companies are currently joining intuitive agents into their everyday activities, sales forms and communication with clients. Chatbots can help to[17]:

- **Enhance client benefit.** It is the best choice for the individuals who don't need their clients to:
  - ➢ Sit tight for administrator's answer - "Stay on the line, your call is very important to us" is always annoying, isn't it?
  - ➢ Look for an answer in the FAQ—usually clients don't have time for looking over many pages with guidelines.

- **Automate repetitive tasks.** Most clients need to find solutions on the same questions—When do you work? What is your area? Do you make deliveries? It reduces the employees' workload if you make a chatbot in order not to write the same answer everytime.

- **Customize communication.** A chatbot answers the particular inquiries of guests instead of showing a considerable list of data. The client is more prominent of purchasing something if he gets more attention.

- **Enhance a reaction rate.** Around 90% of inquiries sent from Facebook business pages stay unanswered. Chatbot reacts to 100% of messages and changes over more guests into purchasers.

- **Streamline the shopping procedure.** It just takes to write what you need to the chatbot and the bot will send the data to the business office. You don't have to repeat a few times "I require the same, however with metal button". Plus, the chatbot recalls your inclinations and utilizations this data when you return.

3. Related Work

This chapter surveys previous work in conversational chatbots used for project and task management.

Conversational agents subtract the long tasks and organize them during a method that’s more economical, cost efficient and productive. Team members and project managers are using bots to accomplish shared goals. They are now enabled to communicate easier and to make reports regarding project progress. Various aspects of a task, such as the status, timelines, priority, update notifications and resource allocation, are handled by the chatbot. It automates tasks and enables team members to plan and schedule multiple tasks according to the set deadlines.[18].

The chatbots solutions allow cutting costs at a monumental scale. Chatbots act as employees, but the only difference is that a chatbot have to be paid only once. It can be seen as good substitute of keeping a variety of freelancers and contractors or paying new employees a monthly salary. For a chatbot you pay once, you train it, and it’ll will be useful for a long period time.

Moreover, a bot will provide the ability to train the employees when it’s about the new products or services of the company while at the same time it’s helping with project management. All of these skills of a bot will increase the productivity of the team/company.

Conversational agents have a big focus these days and there is a lot of work under the hood which feeds the evolution and productivity of chatbots and their role in automating task management in companies. In a small business or a team setting, there are bots such as PMbot, Howdy, KnightSpear’s Issabela, Fireflies Slackbot that can automate different aspects of task management.

Howdy is a Slack based bot that can run a team’s meetings. Howdy makes available multiple skills out of the box like daily team check in, collecting information from each teammate and collating it into a clean summary. The big goal of Howdy’s creator was to make a bot that anyone can customize to be their bot, in order for all teams to enjoy the benefits of automation and digital assistance, without having to build a bot.

Howdy has a dashboard on the website to make any edits for the bot. You can make a script for Howdy to know what to do, in which you set the time, days of the week, where you want Howdy to post the report, the waiting time for the response, and the participants for the recurring meeting. Now every person involved in the meeting can interact with the bot in order the respond him to the specific questions. The defacto use case for Howdy would be the daily meeting of the team. In this case each teammate would tell the bot what are they doing, what are their needs or what are they going to do. After each colleague finishes his conversation with the bot, the admin of the team can review each conversation and help or discuss with someone in case of a bottleneck.

The bot can also show a summary of the reports after each conversation is done. The report is grouped based on questions, and it might be shared on a design channel, in order to be seen and reviewed by each person. If a person is responding too late to the question, the bot cannot longer collect his data[19].

Issabela is a chatbot work coach, which has the ability to process data, send suggestions and reports to project managers in order to manage tasks efficiently, to strategically set up projects and even to keep their team driven and motivated.

Issabela can gather and collate information regarding trend within the team’s temperament and productivity, and use it to produce standing reports and build suggestion on how a project manager will efficiently manage tasks and interact her team[20].

The bot can make it easy for the users to onboard and help teams have effortless while using KnightSpear product. When augmenting user doing their work, it can display the number of tasks are overdue and can emphasize user whenever there’s a loose ends that needs to be tied up. Issabela can also bring tips daily, which proves that she is also an effective work coach besides a chatbot.

Fireflies is a conversation agent which runs on Slack, that can automate Project Management. It can be seen like a personal assistant which can handle all the routine tasks that came up in the Slack workflow. Moreover, it can track accountability and tell the user what each person was working on. This a valuable feature needed by a team, because to doing well are team collaboration and transparent organizations, but knowing what each person is supposed to be doing takes a lot of unnecessary oversight.

When chatting with Fireflies bot, it can automatically extract key tasks. We all know that a lot of thing are discussed in a chat, and many ideas and tasks which originate in Slack, would get lost soon, due to the big amount of messages. Many tasks are forgotten or lost deep down in the channel, and someone spends enough time to get the important actions which have to be completed.

It always happens that discussion get hidden in a chat conversation, and it needs a meeting with the entire team to reformulate the setup done in Slack. The team needs to spend enough time to decide who could to do what task. In order to reduce the amount of time spent discussing and figuring out which person what to do, the Fireflies bot can help auto-assign task to teammates while discussing on a Slack channel[21].

4. Study case

This chapter goes in depth into describing the applications and all the core components that make the whole platform work as a conversational agent.

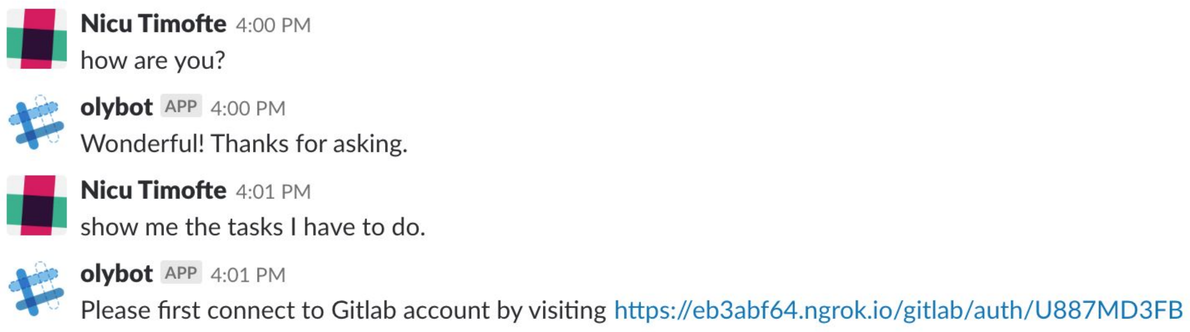
4.1 Specification

In order to illustrate the usability of a conversational chatbot I developed an application that enables the users to easily manage their tasks when building a software product.

The application was designed to use Gitlab platform for persisting and interacting with the tasks of a project and Slack for the user interface where the client could converse with the bot and give different commands to it when he wants to manage his tasks. The chatbot was developed as a Slack extension, thus it can be used only inside Slack chat for now. In order to interact with the bot, the admin of the Slack organization needs to install the application named Olybot from the Slack's apps marketplace, because the name I gave to the chatbot is Oly. After the app is integrated into the organization, every user can send a direct message to it in order to start chatting. The bot can be found in the left panel of the Slack application with the name 'olybot', where the channels and apps are. It can also be added as a normal user into a channel where usually teammates discuss specific topics(for example '#general').

The user is the one who initiates the conversation with the bot. Oly has enabled small talk, which consists in a variety of trivial chat interactions. He can reply to some basic questions like "How are you?", "Who are you?" or commands like "Tell me a joke". Oly has some fallback intents that he can responds when it doesn't understand what the user meant. Some responses could be "I didn't get that.", "Sorry, what was that?" or "I missed what you said. Say it again?".

When the user wants to request Oly specific commands regarding his tasks, the bot will ask the user to sign in with his Gitlab account at a given URL address in order for the application to make a mapping between a Slack account and a Gitlab account. After the user signs in, the application knows which is the Gitlab account of the user, it will have access to that specific account and it could manage the user's requests on the tasks. The received link will send the user to the Gitlab platform where he will give read and write access to the chatbot application on his account. When the user is chatting with the bot in a open channel where are more teammates, the application will send a message in private in order for others to sign in incorrectly.



Nicu Timofte 4:00 PM
how are you?

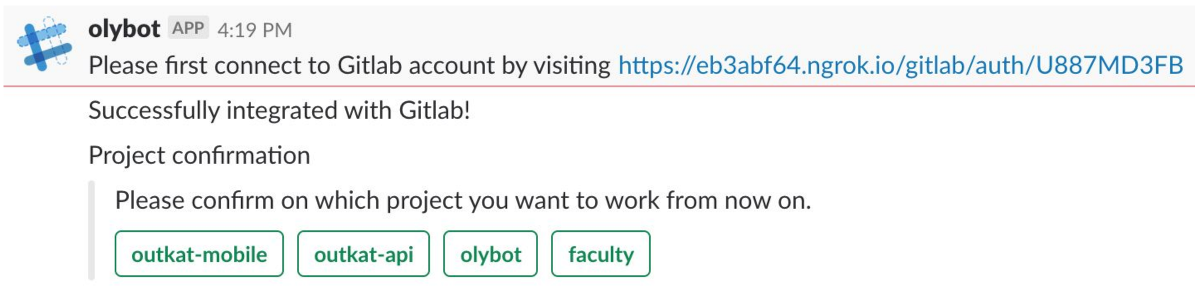
olybot APP 4:00 PM
Wonderful! Thanks for asking.

Nicu Timofte 4:01 PM
show me the tasks I have to do.

olybot APP 4:01 PM
Please first connect to Gitlab account by visiting https://eb3abf64.ngrok.io/gitlab/auth/U887MD3FB

Fig. 4 Gitlab sign in

Next, in order to continue, the agent will show the user a list with the projects that he has on the Gitlab platform in order to select on which one he wants to work further. After the selection is done, the bot will manage only the tasks regarding the selected project and the user can start giving commands to the bot.



olybot APP 4:19 PM
Please first connect to Gitlab account by visiting https://eb3abf64.ngrok.io/gitlab/auth/U887MD3FB

Successfully integrated with Gitlab!

Project confirmation

Please confirm on which project you want to work from now on.

Fig. 5 Project confirmation

Olybot can understand in natural language what is requested to do. It can do almost every daily used operation regarding issue management. Some “skills” that the chatbot is able to provide are as follows:

- create and edit an issue
- close and open an existing issue
- assign and unassign a person on/from a task (teammate or self assign)
- add and remove label to/from issue (e.g. “in progress”, “done”)
- add and remove issue scope (e.g. “closed”, “open”)
- show the list of the issues based on different criterias like issue scope or label
- subscribe user to project notifications (e.g. when a task is updated, he can receive message)
- send real-time notification as a chat message when a given task is updated

When requested to do a specific operation the chatbot is able to understand multiple versions the same meaning, because he was trained to understand like that. For example the request “show me the opened tasks” will make the bot to give the same response as the request “what I have to do today?”, because both have the same meaning and refers to the opened tasks which can be done.

Oly has the ability to understand the meaning from the context. For example when requesting “show me the tasks” it will get the list with all issues from Gitlab, but after asking “and closed” or “and in progress” he will understand that the user is referring to the list of the tasks.

More details regarding the functionality of the application and how it can be used will be described later in subchapter 4.4

4.2 Conversational Chatbots using Dialogflow

This subchapter describes the theoretical aspects and the tools used for creating the application.

4.2.1 Technologies

There are several technologies and tools used in the implementation of the conversation agent, each having different roles and responsibilities, dependending on the part of the component it is integrated in. The following diagram presents the technology stack used in the application and the protocols used for the communication between the services and tools.

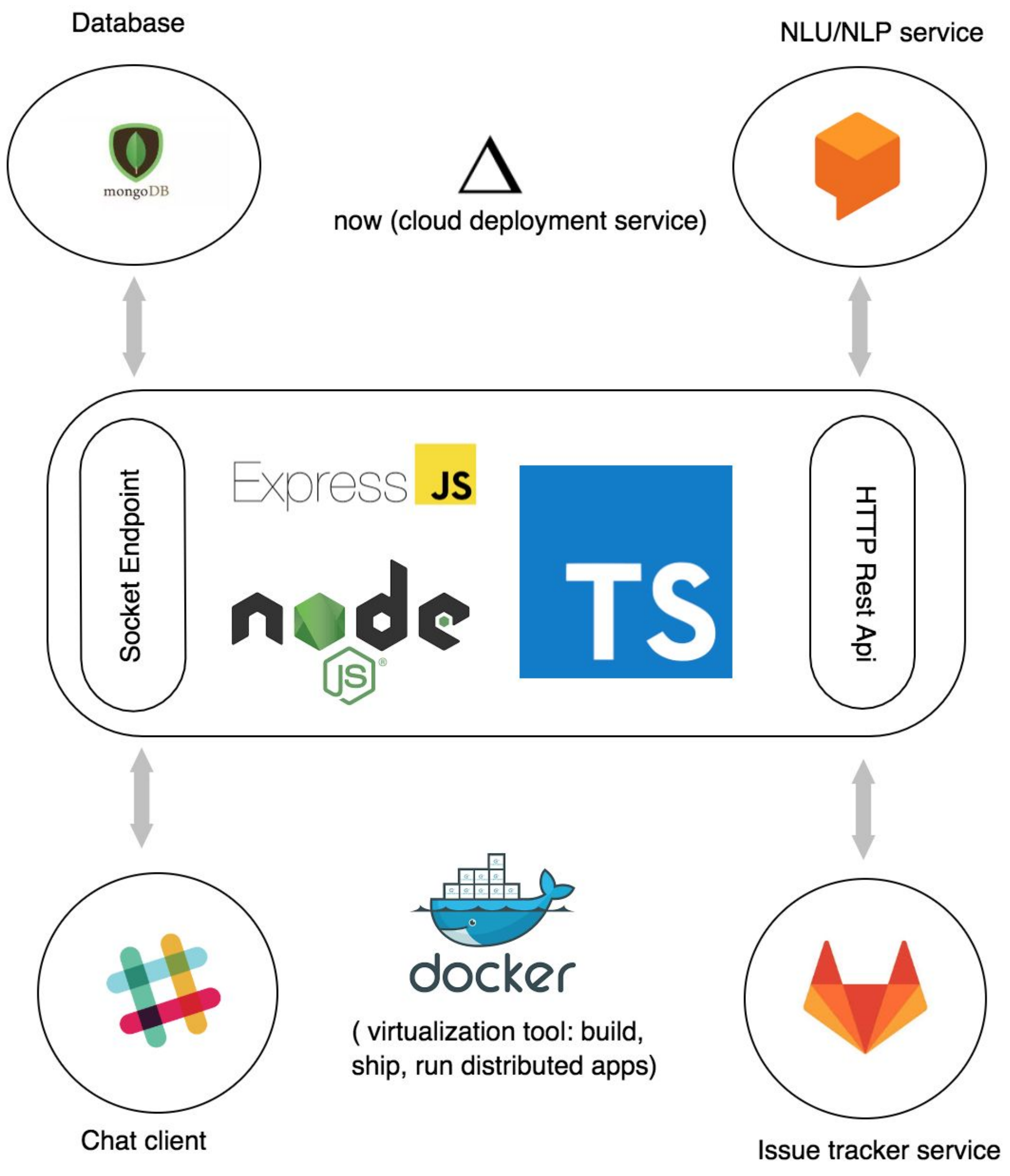


Fig. 6 Component diagram presenting the used technologies

The language used for handling all the back-end logic is NodeJS, in conjunction with the Express framework. NodeJS is a lean, fast, cross-platform JavaScript runtime engineered on Chrome’s V8 JavaScript engine. Shortly, it's JavaScript on the server. Node allows you to execute Javascript code outside the browser, in a §computing setting (such as a server or native development environment) instead of a browser environment). It uses an event-driven non-blocking I/O model that creates it efficient and lightweight environment[23].

NodeJS is meant to run on a dedicated HTTP server and to use one thread with one process at a time. Node applications are event-based and run asynchronously. Code engineered on the Node platform doesn't follow the normal model send, wait, receive, process. Instead, incoming requests are processed by Node in a constant event stack and so it sends tiny requests one after the other while not expecting the response[24].

The framework used for the online API is ExpressJs. It's a “fast, unopinionated minimalist server-side Web framework for Node.js” as they describe themselves. Written in JavaScript, specific acts as a skinny layer of core web application options. in contrast to an enormous, extremely narrow framework like Ruby on Rails, Express has no out-of-the-box object relative mapping or templating engine. Express framework it’s not engineered around specific elements, having “no opinion” concerning what technologies you plug into it[26, 25].

TS stands for Typescript, shortly “JavaScript that scales” as they wish to describe. It's a superset of JavaScript that primarily provides nonmandatory static typewriting, interfaces and classes. One of the large advantages is to enable IDEs to produce richer environment for recognizing common errors as you type the code. For an oversized Javascript project, adopting typescript may lead to a additional robust code, whereas still being deployable wherever regular JavaScript application would run.

The application currently uses MongoDb as database. MongoDB is an open source database that uses a document-oriented data model. It is one amongst many databases which arose within the mid-2000s beneath the NoSQL banner. Rather than using tables and rows as in relative databases, MongoDB is designed to work with collections and documents. Documents comprise set of key-value pairs and are the essential unit of data in Mongodb. Like alternative NoSQL databases, MongoDB support dynamic schema style, permitting the documents in a collection to have completely different field and structures. The information uses a document storage and data interchange format referred to as BSON, that provides a binary illustration of JSON-like documents. Automatic sharding permits data in a collection to be distributed across multiple systems for horizontal scalability as data volumes increase[27].

Along with Mongo as database, the application uses MongooseJS. Mongoose is an Object Document Mapper (ODM) that make using MongoDB easier by translating documents in a MongoDB database to object in the program. Besides MongooseJS there are many different ODM's that are developed for MongoDB including Mandango, Doctrine, MongoLink[28].

For easier development and and deploying the application to now, I used Docker. Docker is a tool designed to make it easier to create, deploy and run applications by using containers. Containers enable a developer to package an application with all of the elements it wants, like libraries and alternative dependencies, and ship it all out as a single package. By doing so, because of the containers, the developer will rest assured that the application can run on every Linux system machine regardless of any customized setting that machine might have that could differ from the machine used for writing and testing the code[29].

Dialog is a cloud service provided by Google used in the application for natural language processing and natural language understanding. Dialogflow (previously called API.AI) is wherever the magic happens. It works on linguistic communication process and backed by Machine Learning. At Dialogflow the complete 'conversation' take place. Dialoflow is backed by Google and runs on Google infrastructure, which means you'll be able to scale millions users[30].

Gitlab is a web platform where users used to put their code. It’s both a Git-repository and a issues-tracker, which uses a open-source licence, developed by Gitlab Inc. The application uses Gitlab for the feature of tracking issues. That’s the place where you can manage all your tasks of the software product you develop. It is a great tool mainly regarding the fact that has a well documented API, which you can interact with, in order to automate the interaction with both your project and your issues. You can also set webhooks which makes it a great tool for real-time notification system, when it takes about tasks.

Slack is a collaboration hub that connect a organization. At heart it is an instant messaging and collaboration system. Slack’s channels help you focus by enabling you to separate messages, discussions and notifications by purpose, department or topic. Integration is what takes Slack from a normal online instant messaging and collaboration system to a solution that enables you to centralize all your notifications, into one searchable place where your team can discuss and take action on each. Slack is a brilliant tool regarding its API, because it enables developers to build automated chatbots which can receive and respond to messes, let users complete and create simple tasks or talk to users to automate tasks.

4.2.2 Implementation

This subchapter describes the implementation of all the technologies described above. With the theoretical part in mind, this chapter's goal is to explain how these technologies are used to create one cohesive application.

The application connects to the interface, in this case the Slack instant messaging API, both through a websocket and a web http embedded in a sdk provided by the Slack team. The library provides two main classes that expose different methods which enables developers to use almost all APIs implemented by the Slack API. The two classes are RTMClient which is used in RTMClientController to receive real-time message from Slack channels and WebClient which enhances the WebClientController which respond back to a Slack channel. First one is implemented using websockets, thus the application being able to implement real-time functionalities, like instant response to the chat users. The second exposed class(WebClient) uses HTTP requests to connect to the Slack API, this one being used to send messages in a channel, or to request information to Slack about a given user, channel and many more. In order to use the Slack sdk, one must create Slack application, which can later become available in the Slack apps marketplace for being used inside other organizations.

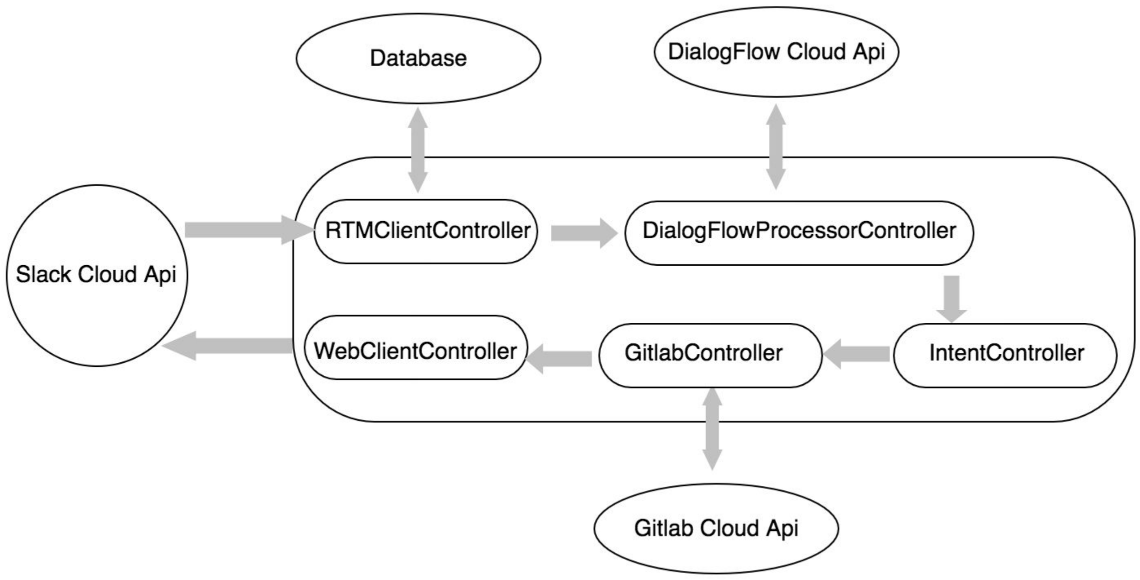


Fig.7 Components diagram

When creating a Slack app, an Oauth Access Token and a Bot Access Token become available for the developer to use them when instantiating the Slack SDK, in order to have access when connecting to the Slack's APIs. Another component which enables Slack to send POST requests is the InterractionController which consists of Rest API that can receive POST requests. Whenever a user has to confirm something in a message sent by the application, Slack send a request which is handled by the Interaction Controller.

The DialogflowProcessorController is responsible to connect with the Dialogflow through a Rest API using HTTP requests. The application sends a text(called query) in natural language to the Dialogflow API and the service will extract the meaning of the text, providing the corresponding intent to the meaning. An intent represents a mapping between what a user says and what action should be taken by the software. Dialogflow has a system which use machine learning to better take decision regarding the intent it takes, but before being able to take a decision one or more intents must be created. The intent need several information such as training phrases, action, response and context when created. This task needs to be done inside Dialogflow service which consists of a web interface.

The response from the Dialogflow API will contain the intent matched(if any), the parameters extracted from the text, the context, the score and many other information that I din not use personally. To filter out false positive results and still get variety in matched natural language inputs for the agent, the user can tune the machine learning classification threshold. If the returned "score" value in the JSON response to a query is less than the threshold value, then a fallback intent will be triggered or, if there are no fallback intents defined, no intent will be triggered. Parameters are elements generally used to connect words in a user's response, to entities. In JSON responses to a query, parameters are returned in the following format:

```
    {“parameter_name”:”parameter_value”}
```

When the JSON response contains a triggered intent, the DialogProcessorController takes the meaning of the user message together with the relevant parameters extracted from the query and sends it to the IntentController.

The IntentController is the component which handles the response back to the user. It's main purpose is to process the received data from Gitlab and to create an appropriate answer for the user. When the normal flow is not met, for example the one of the services - Gitlab - is not working, the component can intercept the problem and notifies the user with a suited response.

The GitlabController is the place where the relevant information extracted from the intention of the user will be processed in order to have the same meaning as the internal form of the Gitlab API. Shortly, the data is prepared for the corresponding request to the Gitlab API. In order to connect to the API and have the appropriate access to it, the controller must receive information regarded to the user, such as the identifier(id) of the project he is working on and his access token received when the user signed in. The response of the request is sent back to the IntentController which handles the response processing.

The processed message arrives in the WebClientController which is responsible with sending data to the Slack API using the HTTP protocol. The component needs to receive the identifier of the channel together with the processed text in order to send a message.

These are the main components which handles the biggest part of the conversation between the user and the chatbot.

4.2.3 Integration

This subchapter describes the integration of the implemented technologies described above. It’s main purpose is to explain how the described components interact with each other.

The RTMClientController is the component which connects to the Slack service and receives the messages typed by the user to the chatbot. When first interacting with the chatbot, the user it not known by the system from the Gitlab perspective, thus he is requested to sign in into his Gitlab account.

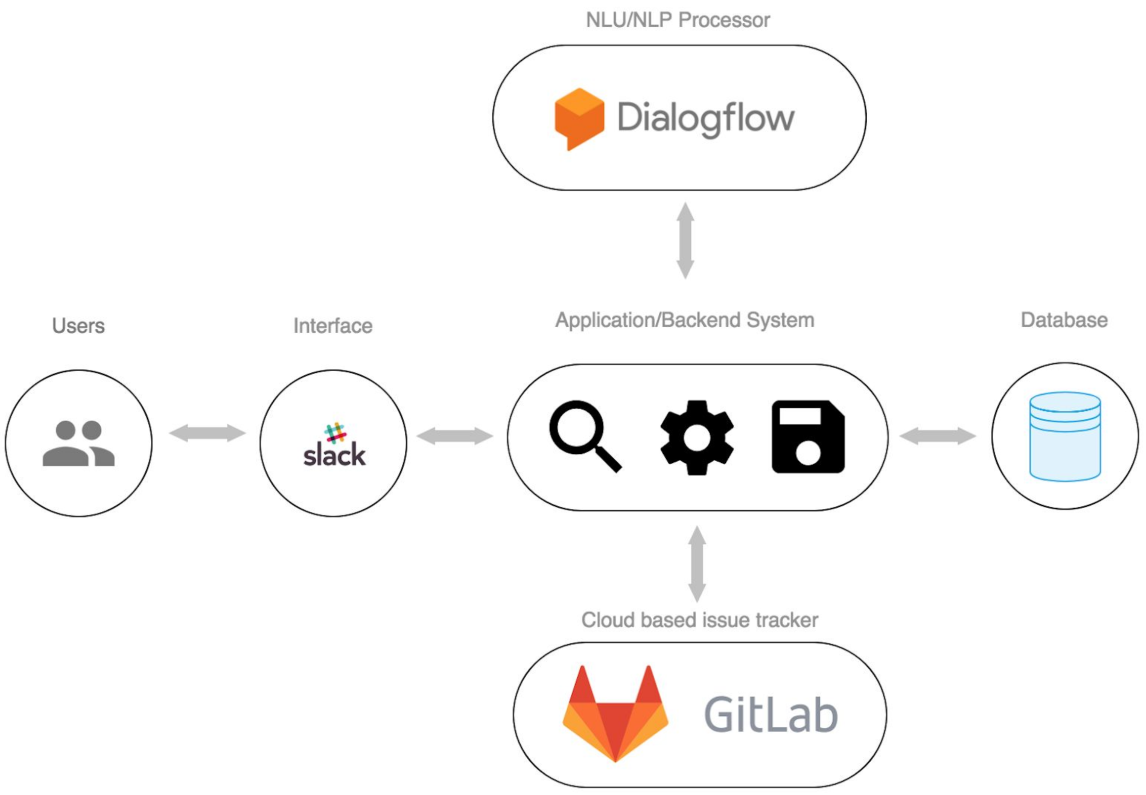


Fig. 8 Components integration

The system uses Ouath2 standard protocol for authorization. OAuth2 is an authorization framework that enables applications to obtain limited access to user accounts on an HTTP service, such as Facebook, Gitlab, and Google. It works by delegating user authentication to the service that hosts the user account, and authorizing third-party applications to access the user account. OAuth2 provides authorization flows for web, desktop applications and mobile devices. A short explanation of the steps can be seen in the diagram below.

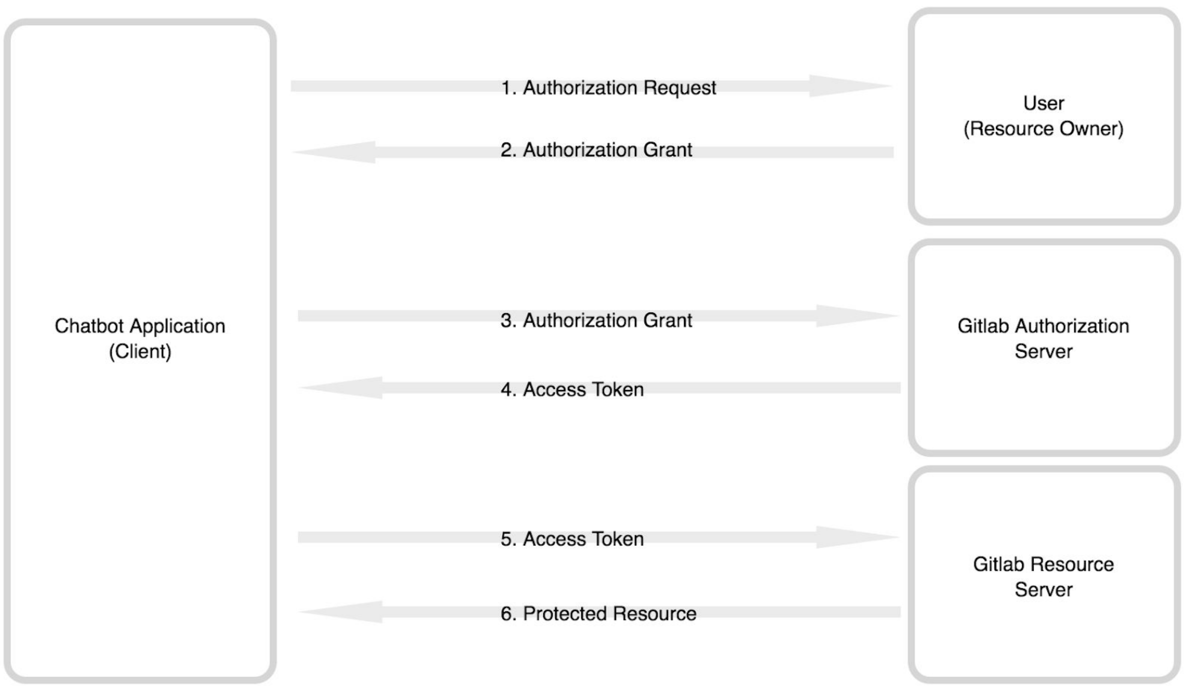
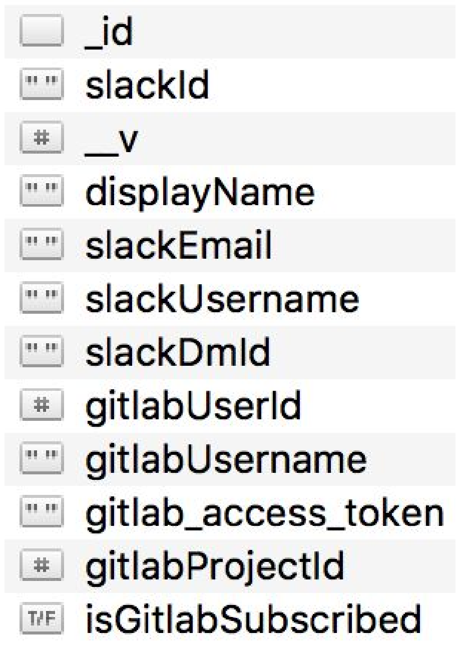


Fig. 9 Oauth2 Protocol Flow

After the system has access and receives the access token, it memorize it along with the user information in order to access resources in the future. The access token is saved in the MongoDB database, which only consists of a single Document, where the Users information is stored. The Document(Table equivalent from SQL databases) structure is described in the Fig. 17.



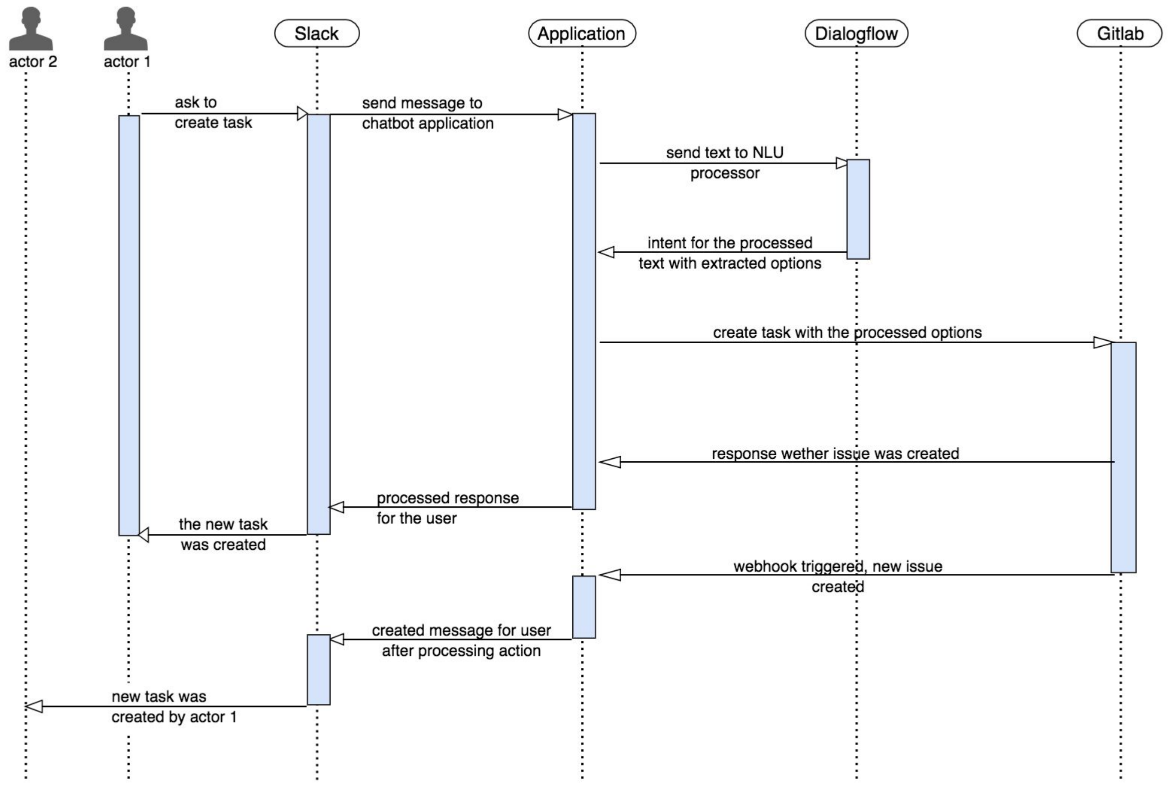
_id
slackId
__v
displayName
slackEmail
slackUsername
slackDmId
gitlabUserId
gitlabUsername
gitlab_access_token
gitlabProjectId
isGitlabSubscribed

Fig. 18 Users Document

Having access to the Gitlab account of the user, the chatbot retrieves his projects names and asks the user to confirm on which project he wants to work further. The confirmation request is sent by the Slack system to the Rest API component of the application which is handled by the SlackInteractController. Here the project identifier is saved into the database in order for the application to know for which project to request the data to Gitlab later on. The component also subscribes the current user to the Gitlab webhook which is responsible to notify each subscribed user when a task is created or updated.

After the sign in process and project confirmation is done, each interaction with the bot will have the same flow. The process starts when the RTMClientController receives a message through the websocket connection, then it will be passed to the DialogflowController component along with the information of the user which sent the message. After querying Dialogflow to get the intention of the user, having the access token of the user, the system can now query the Gitlab API and update or retrieve the intended tasks.

In the diagram below is shown a sequence diagram where the process of a task creation is described. There are 2 people implied in the process, let’s say they are both teammates. When the actor 1 creates a new task, after all processes, the chatbot comes with a reply to the user when the task was created successfully, and the real-time system permits the other users working on the project to receive the notification that the a new task was created. This sequence describes how the technologies are integrated together and how they work in a real use case.

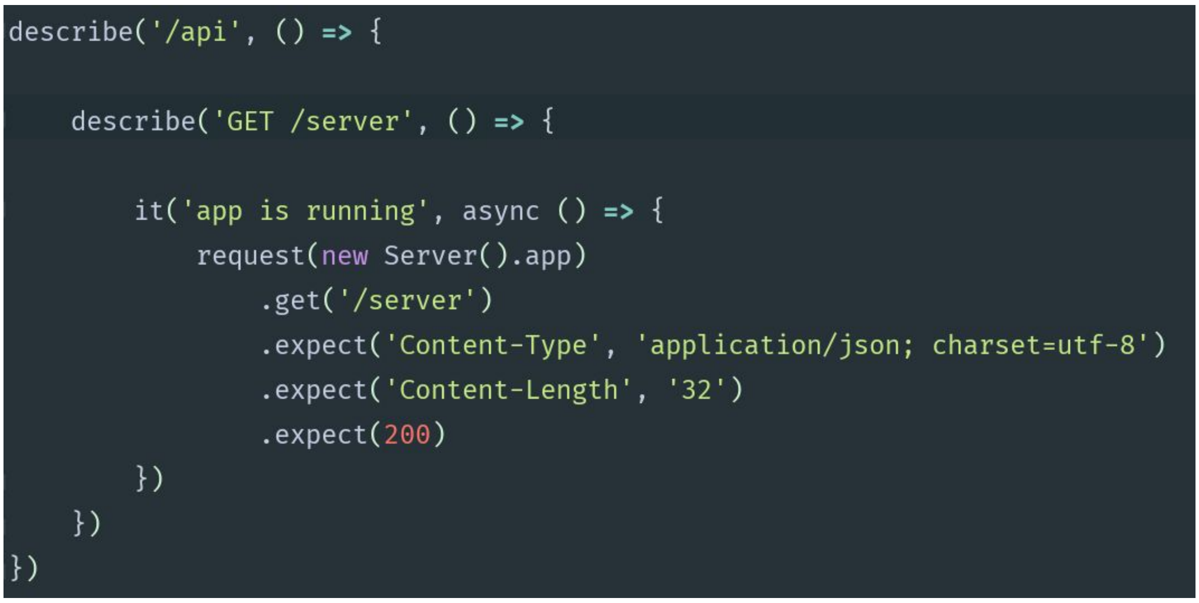


actor 2
actor 1
Slack
Application
Dialogflow
Gitlab
ask to create task
send message to chatbot application
send text to NLU processor
intent for the processed text with extracted options
create task with the processed options
response wether issue was created
processed response for the user
the new task was created
webhook triggered, new issue created
created message for user after processing action
new task was created by actor 1

Fig. 17 Sequnce diagram - task creation

4.2.4 Validity and testing

Testing is a crucial part of developing an application since it helps write more bug-free code from the beginning. In order to do this there are some tests plugins in the npm registry which run smoothly with ExpressJS. First of all there is Mocha, a test framework for Node.js which also works with Typescript. Chai is the BDD(Behaviour-driven development)/TDD(Test-driven development) assertion library which works with any javascript testing framework, in this application being used with Mocha. Another library used was 'supertest' which provides a higher-lever of abstraction for testing HTTP. Its main purpose in this application was to provide integration tests, by verifying bigger components from the functionality and integrity perspective.



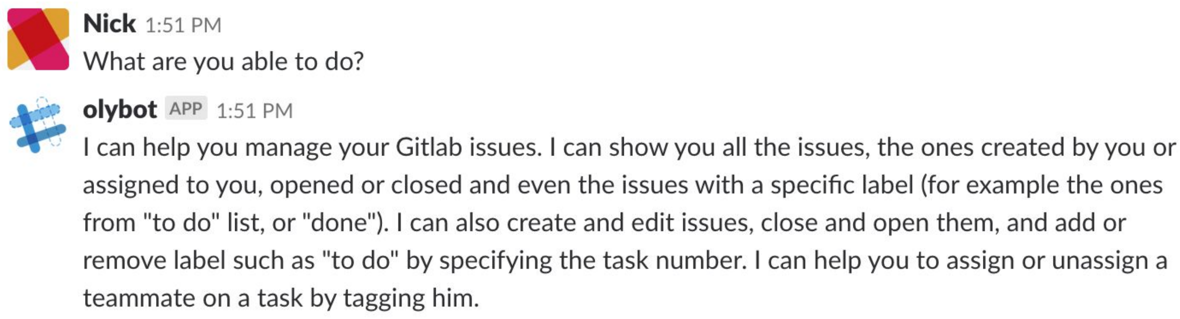
```
describe('/api', () => {

    describe('GET /server', () => {

        it('app is running', async () => {
            request(new Server().app)
                .get('/server')
                .expect('Content-Type', 'application/json; charset=utf-8')
                .expect('Content-Length', '32')
                .expect(200)
        })
    })
})
```

Fig. 16 Application tests

This suite helps a developer test anything from methods to controller actions and request handling. You can automatically simulate HTTP requests towards the API and assess the results of that specific call. Since they don't come out of the box with the Express framework it is a great integration with the entire ecosystem because of the pluggable nature of each one.

4.3 Usage

When starting to chat with the agent, after registering with the Gitlab account and choosing the working project, what pops to the mind of the user is what the chatbot can actually do. Being that there is no such thing as Help button, or a user interface which has a menu with options like in the classical applications, all it remains to do is to put the agent questions such as "How can you help me?", "What do you know to do?" or even "What are you capable of?".

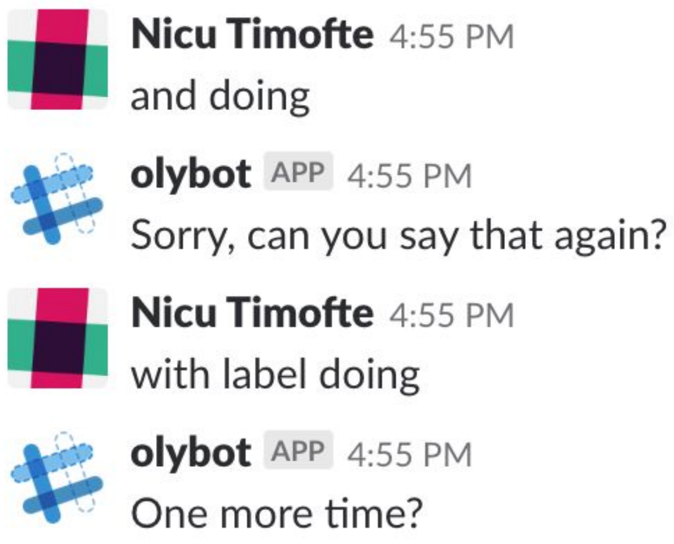


Nick 1:51 PM
What are you able to do?

olybot APP 1:51 PM
I can help you manage your Gitlab issues. I can show you all the issues, the ones created by you or assigned to you, opened or closed and even the issues with a specific label (for example the ones from "to do" list, or "done"). I can also create and edit issues, close and open them, and add or remove label such as "to do" by specifying the task number. I can help you to assign or unassign a teammate on a task by tagging him.

Fig. 10 Chatbot’s capabilities

The proposed conversational agent is designed to process and understand natural language, instead of using plain old buttons. The chatbot has a finite number of tasks it can provide, thus when it doesn’t understand what the user means, it will get back with corresponding replies such as “Sorry, can you say that again?” or “I missed what you said. Say it again?”. That means it was not designed to respond such a command or maybe it does not understand the current format of the command and the user may rephrase it to a simpler form. Below, in the Figure 11, it can be seen a relevant example of the fallback response when the chatbot doesn’t understand the command.



Nicu Timofte 4:55 PM
and doing

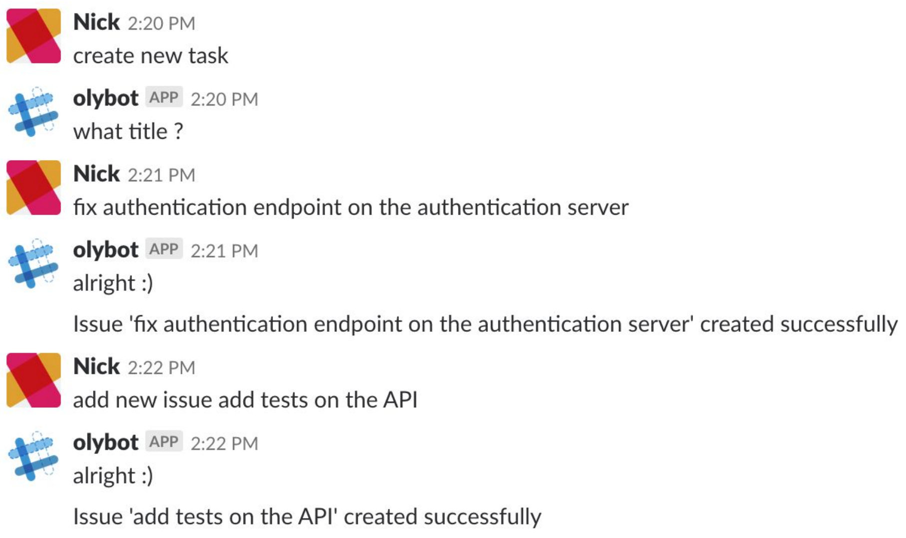
olybot APP 4:55 PM
Sorry, can you say that again?

Nicu Timofte 4:55 PM
with label doing

olybot APP 4:55 PM
One more time?

Fig. 11 Chatbot fallback responses

After asking the agent makes it clear what it’s capable of, the user can form appropriate text message in natural language in order for the bot to understand better. As previously mentioned, the user needs to reformulate his command in case the bot doesn't get if from the first try. Given that main purpose of the agent is to deal with managing tasks, in the first place it has the ability to create a new issue by specifying a title and to edit it when needed. The bot is capable to understand many forms of the query, even when the title of the issue is not quoted. It may happen sometimes that it get the title in a wrong manner, but taking into consideration the fact that it has a training process, it will get better over time. When the user specifies that he wants to create a new issue but does not provide the title of it, the agent will come back with a prompt question, asking for the required field of the command which in this case is represented by the title. Some examples of querying the bot to create a new issue could be: "Create new task firebase login" or "Add new task github authentication" where "firebase login" and "github authentication" could represent the titles of the issues. In the Figure 12 below can be found examples of creating issues in both forms when specifying the title in the same command and when forgetting the title.


Nick 2:20 PM
create new task

olybot APP 2:20 PM
what title ?

Nick 2:21 PM
fix authentication endpoint on the authentication server

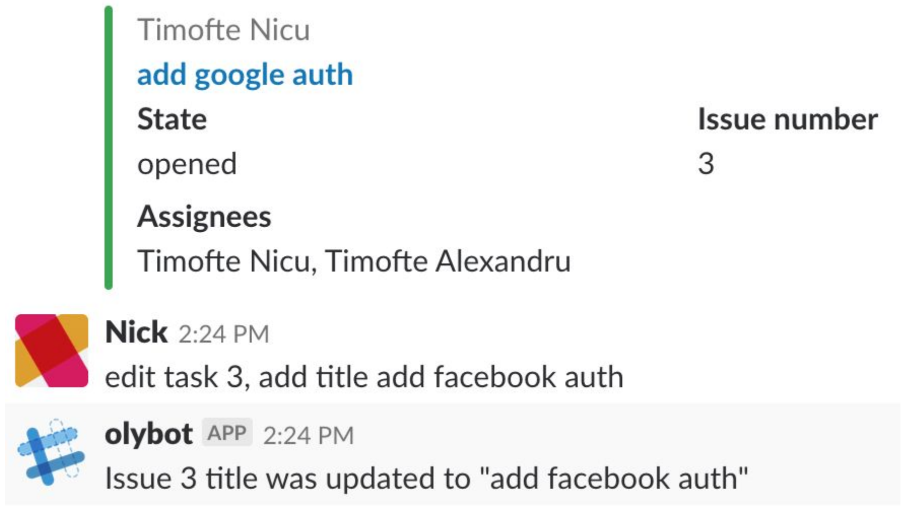
olybot APP 2:21 PM
alright :)
Issue 'fix authentication endpoint on the authentication server' created successfully

Nick 2:22 PM
add new issue add tests on the API

olybot APP 2:22 PM
alright :)
Issue 'add tests on the API' created successfully

Fig. 12 Issue creation

Once there is at least one task in the repository, the chatbot is able to provide editing functionality. Before asking the agent to update a specific issue, one must know it's identifying number which is shown in the list of issues. Some very relevant commands which can be send to the bot when updating an issue would be "Modify task 10, modify title to ‘new title’ ” or “Edit issue 1, add title ‘new title’ ”. Here is an appropriate example of how it goes:


Timofte Nicu
add google auth
State
opened
Issue number
3
Assignees
Timofte Nicu, Timofte Alexandru

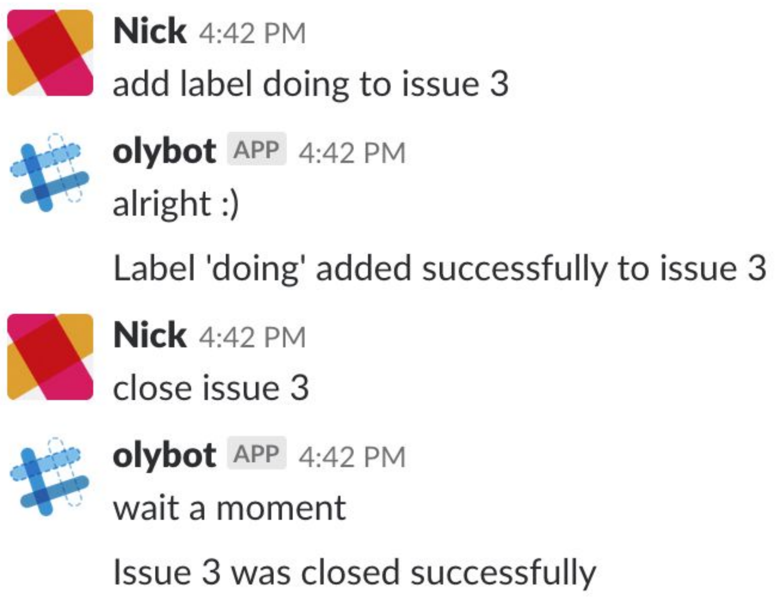
Nick 2:24 PM
edit task 3, add title add facebook auth

olybot APP 2:24 PM
Issue 3 title was updated to "add facebook auth"

Fig. 13 Issue editing

When it comes to the situation of a task, there can be multiple categories in which one can be dropped. Such examples could be ‘bug’, ‘doing’, ‘dev ready,’ to do’, etc. The agents knows this terms as labels. One task has no label when is created, but later on it can be added a specific label, depending on the situation. For example when a developer starts implemententing the ‘add facebook auth’ task, it will ask the bot to put it the label ‘doing’. There are 2 ways to make the bot understand what to do: one of them would be “add label doing to issue 3” and the other “move issue 3 in doing list”, where both forms of the command refers to the fact that someone started to make task number 3.

There exist another mode of a task, which is known as ‘state’. The are only two states which a task could have: ‘opened’ and ‘closed’. When a task has the ‘opened’ state, all the other labels are relevant, give a meaning to the current situation of it, and also refers to the fact that the issue is currently in the backlog and needs attention. The state of a specific issue becomes ‘closed’ all the labels become obsolete due to the fact that nothing has to be done further on with that issue. The agent has also the capability of closing an issue or even opening it again. For instance when the implementation of a feature was closed before being tested it needs to be opened again(Fig.14).



Nick 4:42 PM
add label doing to issue 3

olybot APP 4:42 PM
alright :)

Label 'doing' added successfully to issue 3

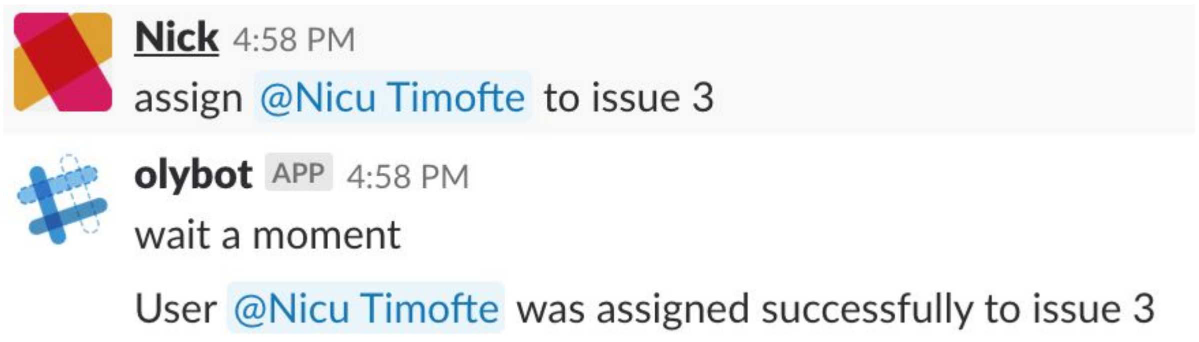
Nick 4:42 PM
close issue 3

olybot APP 4:42 PM
wait a moment

Issue 3 was closed successfully

Fig. 14 Issue's state update

When it comes to the 'doing' part of an issue, there must be always someone responsible who makes it. That persons is called the assignee or the assigned person. The agent is also able to assign a teammate to a specific issue, and even unassign when needed. In order to assign another user beside the one who makes the assignation, the other teammate involved must have the Gitlab registration done in order for the agent to know who is the new assignee, from the perspective of a Gitlab account. The newly assigned user must be specified by tagging him, namely adding his username in the command. In this way the agent can make a connection between the tagged user and a Gitlab account, together found in the database. The user can also self assign to an issue in the same manner, namely tagging himself in the command(Fig. 15).


Nick 4:58 PM
assign @Nicu Timofte to issue 3

olybot APP 4:58 PM
wait a moment

User @Nicu Timofte was assigned successfully to issue 3

Fig. 15 User assignation

5. Conclusion

In this thesis an exploration, discussion and implementation about *how a chatbot can designed to be used as a collaboration tool for managing tasks on software projects* was performed. For this, a minimalistic application which provides easier interaction and information retrieval was created and shown to be successful solving the problems mentioned at the beginning because (I) it helps users manage their tasks easier, in a fewer steps and (II) the job can be done using the same tool for communicating with the team without learning and using another platform.

First of all, the problem of using a user interface where many operations are needed in order to do a specific action is solved entirely. Everything is done through a chat application interface that does not require any extra platform or extension beside the one used in communicating with the team. One of the main advantage of the platform is that it requires no learning process regarding to the fact that it’s designed to understand natural language, thus every person makes it easier to use in comparison with a web based user interface. For better user experience the application also makes it possible for the users to have a small talk, making it more entertaining.

Furthermore, since every interaction with the agent is based on the Slack chat application, any device and operating system could make use of the chatbot, given that Slack has many dedicated chat applications such as a Web platform, mobile applications for both IOS and Android, and a Desktop application built with ElectronJS which makes it runnable cross-platform on Windows, Linux and Mac OS. That being said, the second problem of the current solutions being implemented only on web technologies with impossibility to be used from a mobile device is solved. Now the users can manage their daily tasks through their mobile device interface when being in impossibility of a working from a PC or a laptop.

There are a few guidelines for future additions in the design of the chatbot that can be taken into consideration. One of them consists of the ability to present the users with alternatives to use either a quick replies or free text at all time, because not all them desire to discuss with the agent in a natural language manner, but having the possibility to choose from a bunch of options.

Another idea for further development of the application would be to have the possibility of selecting which chat platform to use or even not being agnostic regarding the issue tracker service. It would be considered a great opportunity for every team to have the possibility to use the agent in their favourite chat application such as Skype, Facebook Messenger or even Whatsapp.

6. Bibliography

1. "Chatbot Market 2017: Stats, Trends, Size ... - Business Insider." 20 oct.. 2017, http://www.businessinsider.com/chatbot-market-stats-trends-size-ecosystem-research-2017-10.
2. "5 Essential Features for Task Management Software - ProjectManager ...." 23 nov. 2016, https://www.projectmanager.com/blog/5-essential-features-task-management-software.
3. "Small Business Use Cases of AI Chatbots in Project Management." https://www.softwareadvice.com/resources/ai-chatbots-project-management-smb/.
4. "11 Best Uses of Chatbots Right Now – The Mission – Medium." 3 iul.. 2017, https://medium.com/the-mission/11-best-uses-of-chatbots-right-now-1c27764b7e62.
5. "Chatbot Architecture – Pavel Surmenok – Medium." 11 sept.. 2016, https://medium.com/@surmenok/chatbot-architecture-496f5bf820ed.
6. "How do Chatbots work? A Guide to the Chatbot Architecture - Maruti ...." https://www.marutitech.com/chatbots-work-guide-chatbot-architecture/.
7. "A Brief Guide to Chatbot Architecture - DZone AI." 8 sept.. 2017, https://dzone.com/articles/a-brief-guide-to-chatbot-architecture.
8. "Dialog management – Bot Tutorials." 20 iul.. 2017, https://tutorials.botsfloor.com/dialog-management-799c20a39aad.
9. "How to build smarter chatbots | VentureBeat." 11 oct.. 2016, https://venturebeat.com/2016/10/11/how-to-build-smarter-chatbots/.
10. "10 Simple Tips on Bot Strategy and Design – Chatbots Magazine." 16 sept.. 2016, https://chatbotsmagazine.com/10-simple-tips-on-bot-strategy-and-design-a4b48116ee76.
11. "Deep Learning for Chatbots, Part 2 – Implementing a Retrieval-Based ...." 4 iul.. 2016, http://www.wildml.com/2016/07/deep-learning-for-chatbots-2-retrieval-based-model-tensorflow/.

12. "Challenges of Building an Intelligent Chatbot - DZone AI." 14 iul.. 2017, https://dzone.com/articles/challenges-in-building-an-intelligent-chatbot.

13. "Beyond NLP: 8 challenges to building a chatbot | InfoWorld." 6 dec.. 2017, https://www.infoworld.com/article/3239926/application-development/beyond-nlp-8-challenges-to-building-a-chatbot.html.

14. "Challenges of building intelligent chat bots - House of Bots." 21 feb.. 2018, http://houseofbots.com/news-detail/2189-1-challenges-of-building-intelligent-chat-bots.

15. "Challenges with Chatbots — not just technical – Chatbots Life." 19 mar.. 2017, https://chatbotslife.com/challenges-with-chatbots-not-just-technical-ecb39612422f.

16. "What is a Chatbot and How to Use It for Your Business - Medium." https://medium.com/swlh/what-is-a-chatbot-and-how-to-use-it-for-your-business-976ec2e0a99f.

17. "Top 5 Benefits Of Using Chatbots For Your Business – Chatbots ...." 7 nov.. 2017, https://chatbotsmagazine.com/top-5-benefits-with-using-chatbots-for-your-business-159a0cee7d8a.

18. "How Enterprise Chatbot Platforms And AI Are Fundamentally ... - Forbes." 29 iun.. 2017, https://www.forbes.com/sites/mnewlands/2017/06/29/how-enterprise-chatbot-platforms-and-ai-are-fundamentally-changing-the-way-we-do-business/.

19. "How We Automated In-House Content Sharing Using Slack Chatbots." 1 sept.. 2016, https://chatbotsmagazine.com/how-we-automated-in-house-content-sharing-using-slack-chatbots-da438c0bda1e.

20. "KnightSpear's AI Work Coach Isabella Will Help Project Managers ...." 6 iul.. 2017, https://chatbotslife.com/knightspears-ai-work-coach-isabella-will-soon-help-project-managers-make-better-decisions-83eb8ebe7b58.

21. "6 reasons why we need to automate Project Management on Slack." 29 nov.. 2016,

https://chatbotslife.com/6-reasons-why-we-need-to-automate-project-management-on-slack-9355fd11aeb5.

22. "The History of Chatbots – Onlim – Medium." 12 oct.. 2017, https://medium.com/@onlim_com/the-history-of-chatbots-2530dd3cdac5.

23. "What is NodeJS? What can you do with it? Why should you use it?." 17 iul.. 2016, https://medium.com/@paynoattn/what-is-nodejs-what-can-you-do-with-it-why-should-you-use-it-8c8d6df32d6d.

24. "About | Node.js." https://nodejs.org/en/about/.

25. "What is Express.js and Why Does It Matter? | ProgrammableWeb." 5 mai. 2017, https://www.programmableweb.com/news/what-expressjs-and-why-does-it-matter/analysis/2017/05/05.

26. "Node.js Express FrameWork Tutorial - Learn in 10 Minutes - Guru99." 30 mai. 2018, https://www.guru99.com/node-js-express.html.

27. "What is MongoDB? - SearchDataManagement - TechTarget." https://searchdatamanagement.techtarget.com/definition/MongoDB.

28. "An Overview of MongoDB & Mongoose – Chingu – Medium." 30 mar.. 2017, https://medium.com/chingu/an-overview-of-mongodb-mongoose-b980858a8994.

29. "What is Docker?." https://www.docker.com/what-docker.

30. "Dialogflow." https://dialogflow.com/.