

Bot Integration to Website Application

Integrating an application with Microsoft Teams to have a chatbot collect and send useful information to the system.

Jonas Carlsson

Main field of study: Computer Engineering
Credits: 15 Hp
Semester/Year: VT 2021
Supervisor: Jan-Erik Jonsson
Examiner: Patrik Österberg
Course code: DT099G
Degree program: Bachelor's degree

Abstract

As technology evolves, more and more companies strive to automate as much work as possible. The use of chatbots in messaging apps is becoming more and more common as replacements in the operation of customer service because of the efficiency and reliability it poses. Replacing customer service with chatbots also saves companies money. In this paper an IT company have asked for the implementation and proof of concept of integration between their server-side application, to Microsoft Teams. They want it to be possible to use a chatbot in Teams to send tickets containing data to their system in order to be handled by users of their application. In this paper a solution to this problem is described as well as discussions about the chosen implementation. The discussion is focused on the final solution, user tests, related work, ethical issues as well as future work. User tests of the implemented solution are included in the results. The conclusion of the work is that this implementation is useful and saves time and effort both for the customer and the employees. There is much potential for future work to be expanded upon, including specialized Ticket Forms and implemented AI.

Keywords: Integration, C#, Chatbot, API, Adaptive Cards, Teams.

Abstrakt

Medan teknologin ständigt utvecklas strävar mer och mer företag efter att automatisera arbetsuppgifter. Användningen av chatbotar sociala media blir allt mer vanligt som en ersättning av kundtjänster då de både är effektivt och pålitligt, samt att företagen sparar pengar. I denna projektrapport har ett IT-företag frågat efter en implementation och konceptbevis på en integration mellan deras applikation och meddelande tjänsten Microsoft Teams. De vill att det ska vara möjligt att använda en chatbot i Teams för att kunna fylla i information i ett formulär som sedan skickas som ticket till deras system. De som arbetar i systemet ska då kunna se ticketen och direkt kunna börja arbeta på uppgiften. I denna rapport beskrivs en lösning på detta problem samt, hur det fungerar, hur det implementeras samt en diskussion på lösningen. Diskussionen fokuserar på den slutliga lösningen, användartesterna, relaterat arbete, etiska problem samt framtida arbete.

Keywords: Integration, C#, Chatbot, API, Adaptive Cards, Teams.

Table of Contents

Abstract	1
Abstrakt	2
1 Introduction	1
1.1 Background and problem motivation.....	1
1.2 High-level problem statement	1
1.3 Concrete and verifiable goals.....	1
1.4 Scope	2
1.5 Outline	2
1.6 Contributions.....	2
2 Theory	3
2.1 API.....	3
2.1.1 REST	3
2.1.2 API Endpoint.....	3
2.2 Bot Software.....	3
2.3 Integration.....	4
2.4 JSON.....	4
2.5 Ticket System.....	4
2.6 Easit-Go Application	4
2.6.1 Restservice.....	6
2.6.2 Import Handler.....	7
2.7 Postman	7
2.8 Microsoft Development	7
2.8.1 Microsoft Teams.....	7
2.8.2 Adaptive Cards	8
2.8.3 Microsoft Application Development	9
2.8.4 Azure Bot Service.....	9
2.8.5 Bot Framework Emulator	9
2.9 IDE	9
2.10 Apache Maven.....	10
2.10.1 Jetty plugin.....	10
2.11 ZK8	10
2.12 LESS	10
2.13 Node.js.....	10
2.14 GIT	10
2.14.1 Bitbucket.....	10
2.15 MySQL	10
2.16 Ngrok.....	11

2.17 Waterfall Dialog	11
3 Method	12
3.1 Workflow	12
3.2 Project Goals	12
4 Model	14
4.1 Setup of Environment	15
4.2 Bot Setup	17
4.3 Bot Development	18
4.4 Integration.....	19
4.5 Research Question.....	19
5 Results	21
5.1 Analyzing two question forms in Appendix A and B	25
6 Discussion	28
6.1 Integration.....	28
6.2 Bot.....	28
6.3 Question Form.....	28
6.4 Ethical and Societal Discussion.....	31
6.5 Future Work.....	32
References	34
Appendix A: Question Form 1.....	37
Appendix B: Question Form 2	40

1 Introduction

API integration has the possibility to allow different applications of exchanging data. An example of this is when information needs to be sent to a database from outside the system. Maybe a company wants a customer to be able to send work requests into a database without having to contact an employee or being inside the system. The idea behind API integration is allowing communication between applications to exchange usability.

1.1 Background and problem motivation

Within the company Easit, customers have expressed a growing interest in ease of access for customers to create work requests via Microsoft Teams. The company works with creating and selling a work application called Easit-GO where employees can, for example, see work tasks and access a database. To date there is a REST-API implementation for the database but no implementation of integration to other applications. In this study we have taken on the task of implementing an integration from the Easit-GO application to Microsoft Teams. A solution to this problem is sought after by customers to improve the work environment and make it easier for their customers to make requests to their company without having to establish contact.

1.2 High-level problem statement

The project's aim is to develop a technical solution to allow integration between a backend Rest API to a .NET bot. This report is proof of concept to show that such integration is possible.

1.3 Concrete and verifiable goals

The project has an objective to answer the following problems:

- Use the API to connect to the database from a Bot service.
- Create a Bot that will answer commands and share a Ticket Form.
- Have the Ticket Form send the data to the database for the Easit-GO application to read.
- Have the Bot work in Microsoft Teams.

Explain the following Questions:

- How does the solution solve an existing problem?
- Justify the efficiency of the solution.

1.4 Scope

This project has its focus on developing a technical solution but will also investigate the importance of integration between applications.

1.5 Outline

Chapter 2 contains all the necessary information needed about the software and methods used during the project.

Chapter 3 describes the project workflow and how the goals will be answered.

Chapter 4 describes the process of performing the work needed to complete the project goal.

Chapter 5 contains all the results gathered during the project.

Chapter 6 is where all the results are discussed and conclusions about the project are made.

1.6 Contributions

Most of the work in this project has been solved by own hand. Parts that needed help to be completed included: Installation of the Easit-GO application on the PC, explanation of the already existing API with documentation, and setup of the dynamic database to allow tickets to be inserted. The testers filling out the question forms were also a great help.

2 Theory

This chapter is going to cover all the theory, software and background information needed for the reader to better understand the work done during this project.

2.1 API

API stands for application programming interface and is a way for applications to use and communicate with the software that has an API set up. It can be seen as a contract between an information provider and an information user, but sometimes the user is the one giving the provider information. The interface of the API is usually a set of functions that provide different uses for the data. Normal functions include retrieving, sending and updating data.

These functions are based on API requests which allow you to send data to, or retrieve data from, a data source. APIs run on web servers and by exposing endpoints they allow other applications to use their functions. Each API request uses the HTTP method. These methods include GET and POST which are used to retrieve, send, and update data.[1]

2.1.1 REST

A REST API is an API that follows the constraints of REST architectural style. REST architecture describes how machine-to-machine communication can be achieved by webservices. It often uses JSON for data transfer but can also use other text formats.[2]

2.1.2 API Endpoint

An endpoint is the outer end of the communication channel to the API. The endpoint can include an address of a server or service and is the place where APIs send requests. It is how an application can contact the API.[3]

2.2 Bot Software

A bot service is a software application programmed to automatically perform operations. In this project the bot software that will be used is a chatbot.

Chatbots are bots programmed to respond to messages sent by users in a chatroom. These bots work by running on a server constantly waiting for messages. When a message arrives, the bot will react according to how it has been programmed, followed by continued waiting for the next message.[4]

2.3 Integration

Software integration is a process to make different systems work together as one. It allows a system to use functionality of another and work as one system. One goal of integration can be to increase value to the customer. Other times it is machine-to-machine communication between systems.

An example of how software integration can be used is a normal website with its own database that retrieves data from other sources or websites. In this case integration is the step for the database to fetch data from various sources.[5]

2.4 JSON

JSON stands for JavaScript Object Notation and is a file format that stores data in objects consisting of attribute-name pairs and arrays. JSON is a popular method to store and transmit human readable data since it is easy for software to parse and generate. [6]

2.5 Ticket System

A “Ticket System” is a management tool used to handle customer service requests. The requests can be called tickets or issues and are added to the system either by ticketing services or by the employees handling the system.

A ticket usually contains the information needed by the employee to handle the request. Such information can include personal information about the customer and information about the specific order or request. [7]

2.6 Easit-Go Application

The Easit-GO application is a product the company Easit has developed and is selling to other organizations. The application is dynamically built, in order to easily be adjusted to what their customer needs. It is a web application connected to a dynamic database with many different uses.

In this project we are specifically looking to understand more about the ticket system that exists in the application.

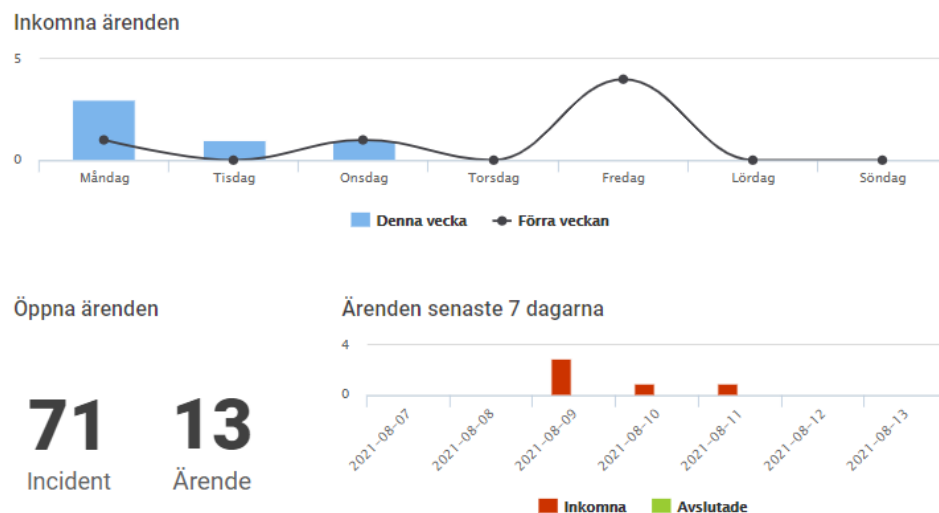


Figure 1: Incoming Tasks to the system

As seen in figure 1, there exists a function showing the incoming tasks to the database. These tasks are added from inside the application by employees handling the workflow. This means that each task in the system could represent a customer with a problem, calling or emailing the company for an employee to answer and create the task in the system.

Ärenden - Nya ärenden Filtrera ^ x

<input type="checkbox"/>	ID	RUBRIK	ANMÄLARE	ORGANISATION	SKAPAD	OBJEKTTYPE
<input type="checkbox"/>	1229	Mail går inte iväg	Karl-Erik Josefsson	Gröna skolan	2015-08-26 07:59	Incident
<input type="checkbox"/>	1228	Fel på rubrik	Jocke Uppfinnare	Kalles RÖR och VVS	2015-08-26 07:59	Incident
<input type="checkbox"/>	1224	Fel på färgband	Jocke Uppfinnare	Kalles RÖR och VVS	2015-08-25 09:29	Incident
<input type="checkbox"/>	1221	Ny toner i skrivare	Kalle Rörsson	Kalles RÖR och VVS	2015-08-21 11:06	Ärende
<input type="checkbox"/>	1219	Fixa rapport över projekt	Karolin Tjäder	Exempelkund intern	2015-08-21 11:03	Incident
<input type="checkbox"/>	1218	Fel på transaktioner	Paul Anka	Easit AB	2015-08-21 11:02	Incident

« < 7 / 9 > » [151 - 175 / 207]

Figure 2: Existing tasks in the database with possibility to add, delete and update tasks.

An employee can create tasks in the system for other employees to see them and fix the issues as they pop up. In figure 2 we see the task view showing many different tasks and clicking on an item shows even more information about the task stored in the database. The customer either contacts the company via email or by calling, to give the company a work request. [8]

2.6.1 Restservice

To start working on integration to other applications an API has been developed for communication with the database. In the Easit-GO application an employee can thus add an API key connected to their own user. These keys are what will later be used to establish connections to the database.

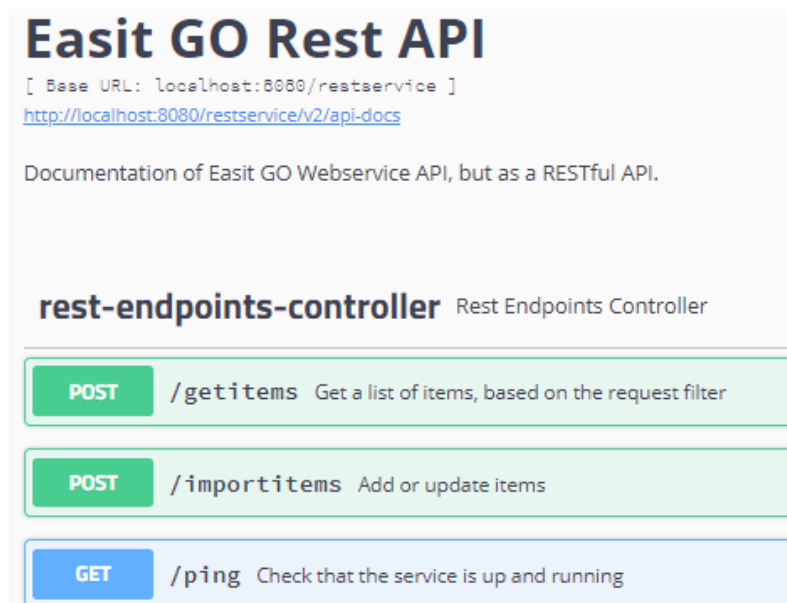


Figure 3: Swagger-UI Documentation on how to use the API

API documentation was created to make understanding and using the functions easier.

As seen in figure 3, there are three functions that will be used to communicate with the database: getitems, importitems and ping. The first one is used to retrieve information from the database, the second one is to create or update information and the last one exists just to check if connection to the database is possible.

For the API to be successfully used, the data sent must be set up in the correct format. This part is done from within the Easit-GO application. Since the database is dynamically built the object type that is to be imported with the API must be created and specified in the application. Once an object type is set and the data to be included is known, the JSON data can be written.

2.6.2 Import Handler

The Import-Handler is an operation used when the ImportItems endpoint is called. An Import-Handler is needed to be able to update existing Items in the database. The Import-Handler is also what defines what Items are included in the object.

2.7 Postman

Postman is an API client used for running and testing APIs. Postman allows users to create and save different HTTP requests. Postman also reads the responses of the API when testing them out.

In this project Postman is mainly used as a tool to make sure connections are working and making sure the database retrieves the correct data.

When a HTTP request is being set up in Postman the correct authorization must be set up and the body need to contain the correct data format. In this case JSON data format is used and so the correct set up of the JSON data needs to be written for the request to be successful.

If the HTTP request is not set up in the correct way, postman will respond with the corresponding error message. This is a very efficient way of trying to find a correct solution to set up APIs, since it often tells you what is wrong.[9]

2.8 Microsoft Development

In this section all the software and tools created by Microsoft that were used in this project will be explained.

2.8.1 Microsoft Teams

Microsoft teams is a business communication platform offering workspace chats and videoconferencing. Because of the significant

increase in users in the recent past, the platform has gotten much more attention in demands for integration.

Teams provide many features and possibilities and include tight connectivity with other Microsoft apps. Examples of this include Microsoft Adaptive Cards and Microsoft Azure Bot Service, which are two very important features in creating working bots with a good graphical interface.

Teams allow bots to be added to conversations which is a great way of letting business partners and customers receive automated answers to questions or commands.[10]

2.8.2 Adaptive Cards

“Adaptive Cards are platform-agnostic snippets of UI, authored in JSON, that apps and services can openly exchange. When delivered to a specific app, the JSON is transformed into native UI that automatically adapts to its surroundings. It helps design and integrate light-weight UI for all major platforms and frameworks.” (Adaptivecards.io, 2021-08-24), is how it is described on their own website and honestly, it’s the best explanation.

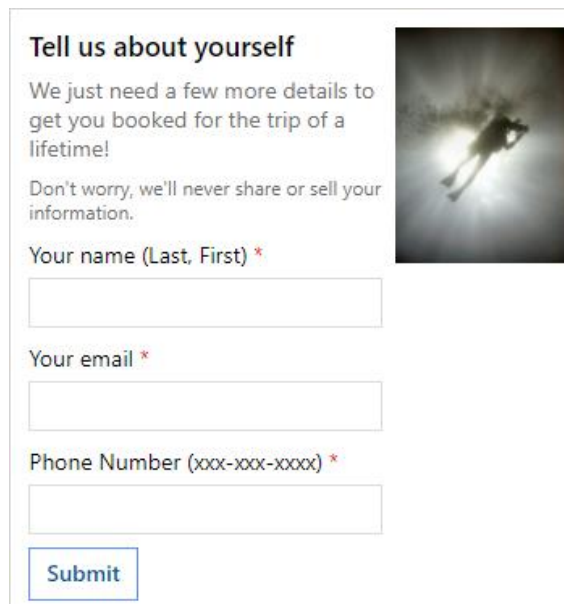
The image shows a screenshot of an Adaptive Card, which is a UI component that can adapt its appearance based on the platform it is viewed on. The card has a title "Tell us about yourself" and a subtitle "We just need a few more details to get you booked for the trip of a lifetime!". Below the subtitle, there is a reassuring message: "Don't worry, we'll never share or sell your information." To the right of the text, there is a small image of a person skydiving. The form contains three input fields: "Your name (Last, First) *", "Your email *", and "Phone Number (xxx-xxx-xxxx) *". Each field has a corresponding input box. At the bottom of the form, there is a "Submit" button.

Figure 4: Adaptive Card Example given by Official Website

Adaptive cards are honestly the biggest reason to why integration with Teams is such an attractive idea. With adaptive cards it is possible to create forms the user can fill out or gain information from, which a normal text bot wouldn't be close to achieving. Not only is it possible to create forms that a bot can send, but the developer can even design the adaptive cards to their liking. Figure 4 is an example of an adaptive card given by Microsoft's own samples.[11]

2.8.3 Microsoft Application Development

C# is a programming language created by Microsoft and is the most common language to use for creating Microsoft desktop applications.[12]

The .NET Framework is a software development framework used to develop Microsoft software. [13]

ASP.NET is the main tool that is present in the .NET Framework and aimed at simplifying the creation of dynamic webpages.[14]

2.8.4 Azure Bot Service

Azure is a bot deployment service created by Microsoft. Azure Bots can be added to services like Microsoft Teams but is hosted and paid for in the Azure Portal. [15]

2.8.5 Bot Framework Emulator

The Bot Framework Emulator is a useful desktop application that allows testing and debugging bots locally. The main use of the emulator is to test the chat responses of the bot without having to set up the whole bot server. [16]

2.9 IDE

IDE stands for Integrated development environment and is a software application used for software development. An IDE normally consist of a source code editor, build automation tools and a debugger.[17]

Eclipse is an IDE mainly used for building Java applications.[18]

Visual Studio is an IDE developed by Microsoft. With both .NET and Visual Studio being Microsoft products, it's an ideal platform for developing .NET software. [19]

2.10 Apache Maven

Maven is a software project management and comprehension tool. Maven is a standard way to build projects and is a definition of what the project consists of. Its main goal is to make the build easy whilst also providing project information and a uniform build system.[20]

2.10.1 Jetty plugin

Jetty is a Maven plugin used for rapid development and testing. Jetty periodically scans the project for changes and automatically redeploys if any are found. This means that any changes made in the IDE are picked up, allowing testing of the recent changes straight away.[21]

2.11 ZK8

The ZK framework 8 is a Java framework for building web and mobile applications.[22]

2.12 LESS

Less, short for Leaner Style Sheets is a backwards-compatible language extension for CSS. Style sheets languages can be used for styling webpages.[23]

2.13 Node.js

Node.js is a JavaScript runtime environment that executes JavaScript code outside a web browser. Node.js LTS is the long-term support version of Node.js. [24]

2.14 GIT

Git is a distributed version control system designed to handle projects with speed and efficiency. [25]

2.14.1 Bitbucket

Bitbucket is a Git-based source code repository hosting service. A git repository is a place to store code with ease of access. Project codes are usually stored in repositories. [26]

2.15 MySQL

MySQL is a database service used to store data in an organized way. It is a relational database where data can have relations with other data. [27]

2.16 Ngrok

Ngrok is a tunneling service that allows communication from a webserver to your locally specified address. Ngrok creates its own address that can replace the local address when establishing contact. [28]

2.17 Waterfall Dialog

A waterfall dialog is a way for a bot to guide the user through a series of steps by sequentially leading the user through questions or other methods. Stepping out of the dialog would lead the user back to the beginning, meaning that actions can be locked by previous conditions. This is good to implement in a chatbot, such that users do not access options that they are in no need of. [29]

3 Method

3.1 Workflow

The workflow of this project consists of the implementation of the product combined with the research required. In order to start the project, research of the area was required. During setup and implementation even more research was required in order to understand and prepare the necessary steps needed.

During implementation Agile workflow was followed in order to always have a working implementation after every new part of the code was finished.

Continuous research and implementation were followed by even more research into the research questions once the solution was completed.

3.2 Project Goals

The goals of this project are defined in Chapter 1.3 and here we will discuss the method used to answer these goals.

- Use the API to connect to the database from a Bot service.

The first goal requires the setup of a Bot service as well as the setup of the Easit-GO application. This goal will be achieved through implementation of the existing API with the created bot. The Easit-GO application requires a lot of research and help from the company in order to understand and set up. This goal is completed with implementation and testing of the system.

- Create a Bot that will answer commands and share a Ticket Form.

This goal has its focus on the bot service. It is achieved through extensive research of Bot implementations and the possibilities to send form cards. The goal is completed once the implementation can be run and tested on an emulator for testing bots.

- Have the Ticket Form send the data to the database for the Easit-GO application to read.

This goal focuses on making sure that the two previous goals work together, and that the data created can be seen on the Easit-Go application. This goal requires even more research into how the database and workflow functions work in the Easit-GO application. Once the necessary setup is prepared, the goal is achieved by combining the previous steps and testing if the data arrives at the website.

- Have the Bot work in Microsoft Teams.

This goal relies on the fact that the previous steps were successful and requires the bot to be set up for Microsoft Teams. This step requires research on how bots can be hosted and published in different software. After implementation has been achieved, this goal is completed when the bot service developed can be used in a chatroom in Teams, and the data arrives to the database.

- How does the solution solve an existing problem?

This question will be answered after researching and discussing the importance of integration and bot services. Research papers will be read in order to compare and draw conclusions from the question.

- Justify the efficiency of the solution.

This question will be answered by having testers try the bot service and analyzing their behavior and experience. The area will be discussed to even further theorize the pros and cons of the solution.

4 Model

In this project, communication between a Chatbot in Microsoft Teams, and the database for the EasitGo application is what is being implemented.

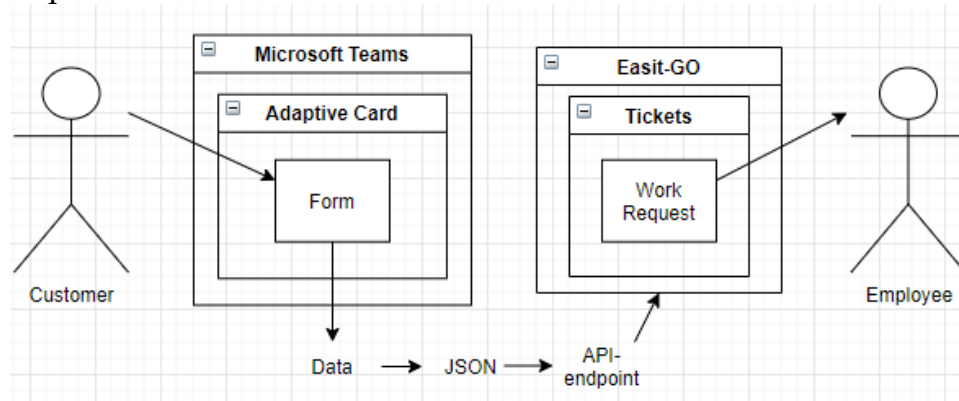


Figure 5: Overview of the system

As shown in figure 5, the customer should be able to have a Teams conversation with a chatbot. The Chatbot should then show the user Adaptive cards which contain useful information and provide forms that the user can fill out and send. The filled-out forms should be translated into JSON data which then is sent to the API-endpoint. The Form is then posted to the database and readable to the employee using the Easit-GO application.

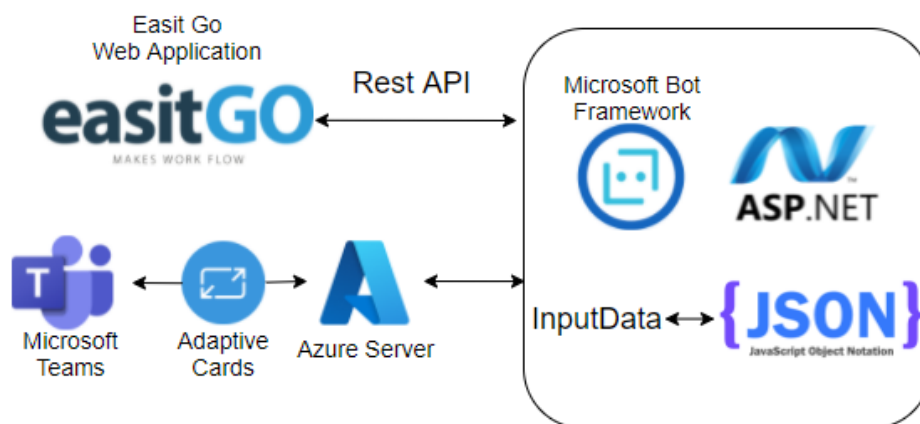


Figure 6: Overview of the system

In figure 6, we see the different steps of the implementation from the perspective of the different tools and methods used. We see that

Microsoft Teams is shown Adaptive Cards through the bot provided and hosted by an Azure Server. The implementation of the bot and the adaptive cards are given to the azure server by the code implementation. The program is converting the input data to JSON format to handle and use the API-endpoint. The program also uses C# code with the help of ASP.NET and the Microsoft Bot Framework to implement a functional chatbot.

This project implementation is thus what is contained within the box to the right of figure 6. The things to the left are the tools used in order to integrate between systems.

4.1 Setup of Environment

To begin with, the Easit-GO environment had to be installed, in order to use the API-endpoint and connect to the database. This was a long and tedious process since many installations were needed before the application was correctly installed.

To begin with Java JDK Version 11 or later is needed, since Easit-GO is developed in Java.

Maven is a tool used to handle builds, dependencies, project structures, and other things. Easit-GO requires Maven in order to run.

ZK Framework 8 used by the Easit-GO application requires LESS for theme handling and styling. This requires Node.js Version LTS to be installed

Using Eclipse as an IDE was recommended since most developers of the company use that environment.

To be able to download the application code, Git was used with Bit Bucket. The Bit Bucket code was only accessible after the company created a company email that could access the Git code and download it. The Git was also used for version handling to make sure the correct code version containing the API was used.

The application also requires a database set up. There were a few choices as to what database to run, but MySQL was chosen. Important here is to make Character Set UTF-8.

Once all the necessary installations and preparations were ready, it was time to get the Easit-GO application working. First off, Maven was used to compile the code downloaded from Git and install all the modules in the local repository. This was easily done from the command prompt, with the command “mvn install”, whilst standing in the correct repository.

In the IDE a workspace was created. Eclipse made this part easy since it had a function for importing existing Maven projects. Additional settings that needed adjustment in Eclipse were to enable editing of ZUL-files, change settings to use Character Set UTF-8 and to create a local version of properties.xml such that the code finds the correct path to the local database.

The last step to get the Easit-GO application up and running is to start debugging. Jetty is used as the choice of debugging tool since it is small, fast and easy to use. Jetty is run as a Maven plugin.

Once the server is up and running and the application can be accessed through <http://localhost:8080/>. After installation and setup was completed, it was now time to start learning how to use the application and API-endpoint.

Even though the application was working at this step, the database is still empty. This means that all the necessary data needed to log in and all the example data that fills out the application is missing. To work around this, a dataset was provided to fill the database with.

Another necessary step was to disable the function that requires a password for logging in. This was done to be able to get inside the system and change password on the administrator login. Once the password was changed, the administrator login was accessible, and the code could be changed back.

Postman is used to make sure importing and getting from the database is working. An API key was created for the administrator user, by using the API-key generator in the Easit-GO application. This key was then used inside Postman for authorization. Basic authorization was chosen with the key as username and password left blank. Testing a GET with the ping-endpoint through <http://localhost:8080/restservice/ping> was

successful and responded with correct data. Testing a POST with either /getitems or /importitems did not work at this point, since the POST must be sent with a corresponding body, and the body content is not yet known.

To be able to make POST work in Postman, an Import Handler must be created in the Easit-GO application. The setup of the Import Handler also specifies what Items exist in the object. This information is what is used when writing the body in Postman. The objects formed by the Import Handler is easily converted to JSON format and is thus what is sent through the Postman body.

Once the Import Handler has been created and the JSON format for the import is known, the /getitems and /importitems will finally work. Testing to POST with these API-endpoints with the correct body format, now returns a success-message along with the corresponding return data. The return data is defined in the Import Handler. We can also see in the Easit-GO application that there are new incidents/tasks in the system, and each task inserted has all the information available.

4.2 Bot Setup

With the server side of the project up and running, the next step is to figure out how to develop a working bot.

After researching bot implementation for Teams and advice from employees at Easit, it was decided that the code would be written in C# with the use of ASP.NET and the bot running on an Azure Server for integration to Teams.

To start off with this research, sample bots were downloaded and executed on Visual Studio to try and get a bot running. During this step a Bot Framework Emulator provided by Microsoft was downloaded in order to check if the bot was working locally first.

Once a working bot was running on the emulator, connection was to be established to Azure. Azure is a paid service where you pay for the messages the bots send, but since this bot testing would not send many messages, I could get away with using their free trial.

One problem that appeared whilst using Azure is that the code is run locally whilst the Azure server requires connection to the bot through a

Http address. This could be bypassed by using a network tunneling software called Ngrok. With Ngrok tunneling the localhost address for the bot could be tunneled through a usable Http address, such that the Azure Server could then access the bot. One last step to get Azure to work was to use a secret ID and Value generated by Azure and change the properties file in the bot to match the secret data. After doing this step, the bot could be accessed through the chat testing in the Azure Portal.

Now that the bot is working on the Azure Server, it is time to add it to Microsoft Teams. This can easily be done in the Azure Portal by going to the Channels tab and adding Teams from the featured channels. Now that Teams has been added, a bot embed code can be retrieved which can then be used as an address by pasting in the browser search bar. This bot embed code will add the bot as a contact in Teams. The bot is now fully implemented and working in Teams.

4.3 Bot Development

Now that we know the bot service will work with Teams and we have seen the API-endpoints work, it is time to figure out how the bot will get the data needed for the import.

Now we are getting into the reason Teams is such a suitable application to do integrations towards. That reason is Adaptive Cards. Adaptive Cards is a way of implementing designed cards that can be sent instead of messages in supported environments. Teams is one of these environments.

With the help of all the different sample bots showing the diverse ways you can use Adaptive Cards; Figuring out how to write code suited for the type of form card that was needed, was possible. To make development easier, the import data was adjusted to only take a string for subject and a string for description.

Since the bot is supposed to share a Ticket-form used to fill out necessary information and then send it to the database, the card would need a field for every different data item, and a send button.

The bot is also supposed to be able to do other things, which is why more cards were also created. When the bot/user first joins the conversation the bot also sends a welcome message explaining how to use the bot. A

Help command was created for the users to get a list of commands to use. This help card is designed to appear anytime the user writes a faulty command to the bot.

The Bot will follow the Waterfall dialog design. This is to make the bot easier to use and create consistency in how it works. Waterfall dialog is also useful since it prevents the user from accessing bot commands that have prerequisites.

Now that the Ticket Form is created, the data inserted into the form must be handled. This is possible since every time the buttons are pressed, all other information inside that card is saved in the corresponding Activity. All data that was inserted can thus be accessed and used in other functions.

4.4 Integration

Now that all other steps have been taken care off, the last step is to implement the Integration. To separate the API-endpoint from the rest of the code, a class was created. This class contains the necessary steps of defining a HttpClient, setting all the header data according to standards and adding Authentication. This is all like how Postman is set up.

The big difficulty here comes from setting up the body. First, a class object must be created in the same way as the JSON body will be set up. All the attributes and arrays must be correctly set up. When this is the case, a simple conversion can be made to transform the object into JSON data.

When calling upon the class, the data that is set to be imported is handled and the Import function is awaited. This makes it so that the bot will wait for the data to reach the database until it sends a confirmation message. It is then possible to check the Easit-GO application for confirmation that the data reached the database successfully.

4.5 Research Question

To be able to answer the research question, actual comparable data is needed. To acquire some relevant data for analyzing the method, we let testers test the different methods and answer a question form.

Before this integration there were mainly two different methods for customers to send work requests to the company. These two methods require the customer to either call or email the customer service of the company. The employee is then required to add the work request to the system using the Easit-GO application.

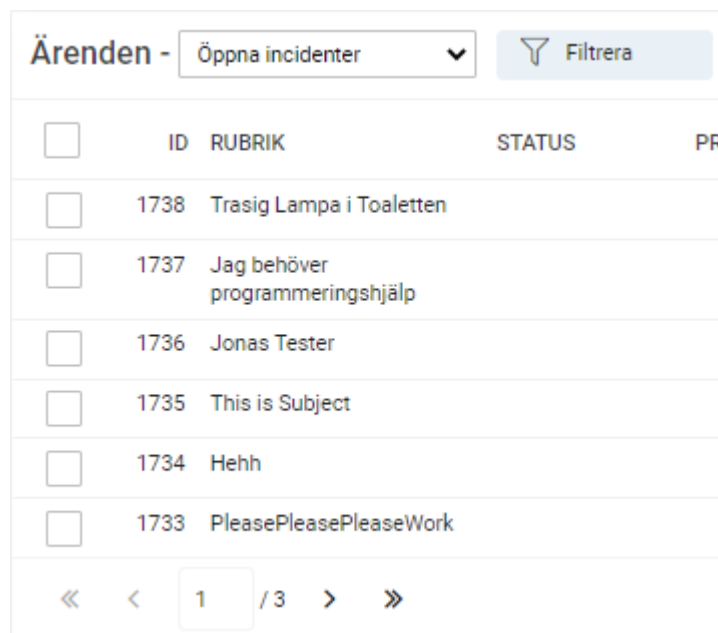
To get some tangible data that can be analyzed we need to let testers use the two preexisting methods and then also use the bot to send a work request. The testers will then answer the form to rate the different methods used and write down how much time it took to perform the small task. Since this process also requires an employee handling the requests, the testers will also play the role of employee to record the same data but from a unique perspective.

The two different question forms for sending and receiving a work request will then be compared and analyzed. Hopefully the data will show that the project has solved an existing problem and the efficiency of the software.

5 Results

The integration to a Teams bot from the Easit-GO application was successful. With the existing software developed during the project, it is possible to open a conversation with the bot and ask for a ticket form. With the ticket form it is possible to write all the necessary information asked for by the form, followed by sending it to the Easit-GO database with the “Send” button. Sending the information will also have the bot send an acknowledgement message to the user confirming that the message was sent successfully.

Following we will see the process as seen from the web application.



<input type="checkbox"/>	ID	RUBRIK	STATUS	PRI
<input type="checkbox"/>	1738	Trasig Lampa i Toaletten		
<input type="checkbox"/>	1737	Jag behöver programmeringshjälp		
<input type="checkbox"/>	1736	Jonas Tester		
<input type="checkbox"/>	1735	This is Subject		
<input type="checkbox"/>	1734	Hehh		
<input type="checkbox"/>	1733	PleasePleasePleaseWork		

« < 1 / 3 > »

Figure 7: Recent Tasks in the system before sending a ticket.

In figure 7 we see the different work tasks already existing in the database, before sending the ticket from the bot.

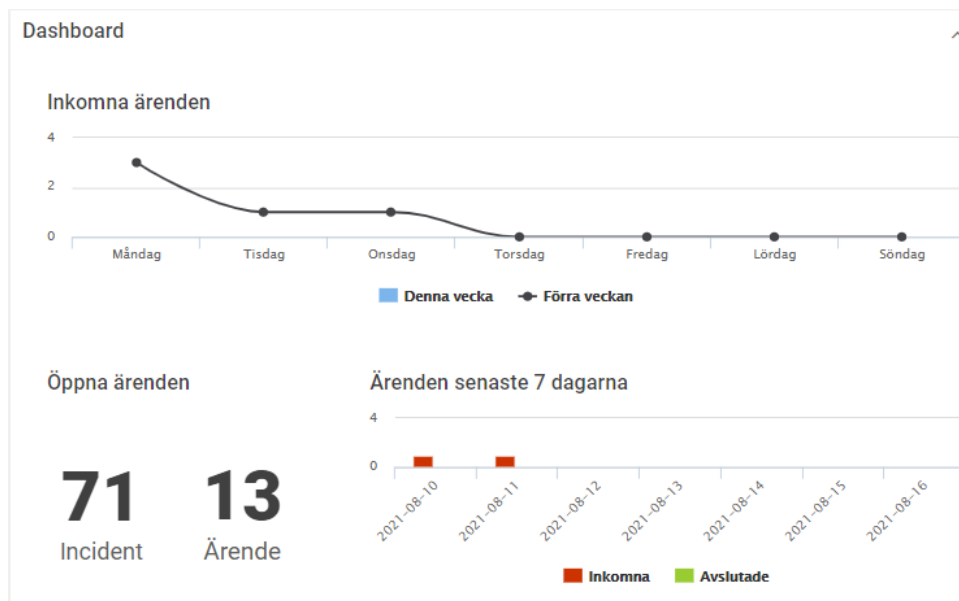


Figure 8: Recently incoming tasks in the system before sending a ticket

In figure 8 we also see that there were no incoming tasks in the current week and that there are 71 tasks in total in the system.

Hello User. Welcome to the Easit-GO Ticket Bot! Type Ticket or any other Command given by typing Help.

2 minutes ago

Ticket

Just now

Present a form and submit it back to the originator

What is your ticket about? *

What is the description of the problem? *

Cancel

Submit

Figure 9: Chatbot after joining chat and using command

In figure 9 we can see the process of connecting to the bot. First off, the bot welcomes the user and tells them about the commands. After the user writes the “Ticket” command, the bot answers with the ticket form. The form requires the user to fill out all questions in order to send the work request.

The image shows a chat interface with a bot. The first message is a form titled "Present a form and submit it back to the originator". It contains two required questions: "What is your ticket about? *" and "What is the description of the problem? *". Both questions have been answered with "Ticket Nr 1 - 08/16". Below the questions are two buttons: a "Cancel" button in a light blue box and a "Submit" button in a dark blue box. The second message is a text box that says "For other commands write: Help", with a timestamp of "A minute ago". The third message is a confirmation box that says "Ticket Was Sent!", with a timestamp of "Just now".

Present a form and submit it back to the originator

What is your ticket about? *

Ticket Nr 1 - 08/16

What is the description of the problem? *

Ticket Nr 1 - 08/16

Cancel

Submit

For other commands write: Help

A minute ago

Ticket Was Sent!

Just now

Figure 10: Using bot to send a ticket.

In figure 10 we can see that the data was filled out and sent successfully to the database. During this step, I made sure to send three requests with different data just to show that they all reached the database. I did send four other requests with other data that I decided to delete in order to have better named subjects in the results.

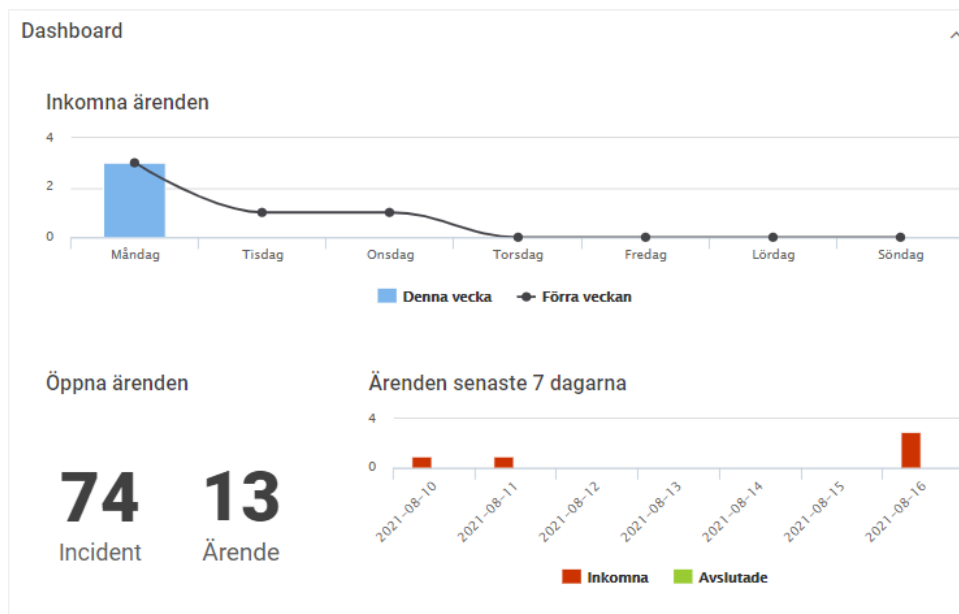


Figure 11: Recent Tasks in the system after sending three tickets.

Here in figure 11, we see that the data did reach the database and we have three new work tasks sent in on a Monday.

Ärenden - Öppna incidenter				
Filtrera				
<input type="checkbox"/>	ID	RUBRIK	STATUS	PRI
<input type="checkbox"/>	1744	Ticket Nr 3 08/16		
<input type="checkbox"/>	1743	Ticket Nr 2 08/16		
<input type="checkbox"/>	1742	Ticket Nr 1 08/16		
<input type="checkbox"/>	1738	Trasig Lampa i Toaletten		
<input type="checkbox"/>	1737	Jag behöver programmeringshjälp		
<input type="checkbox"/>	1736	Jonas Tester		
<< < 1 / 3 > >>				

Figure 12: Recent Tasks in the system after sending three tickets.

The three tickets can be seen in figure 12 but have higher Ids since I deleted four tasks before.

The data sent from the bot can now be accessed through the Easit-GO application and changed/deleted at will. This means that all employees now can see the new tasks and start working straight away.

5.1 Analyzing two question forms in Appendix A and B

When the testers acted as the customer sending the work request, 80% of them preferred using the bot. The tester that preferred Calling customer service said that it was because they thought it was faster and that the information would be more accurately conveyed. The consensus of the testers preferring the bot is that it is simple to use and that all the information they would need to be filled out is given to them as a form. Two of them also liked the fact that you got confirmation that the form was sent.

When the testers acted as the employee handling the work request, all of them preferred using the bot. The consensus was that they preferred the bot since there was no work needed to put the request in the system. They could thus start working on the problem straight away instead of any additional work. One tester said that they liked using the bot since they did not have to deal with the customer. One of the testers also pointed out that they might need to check that all the information was correctly filled out.

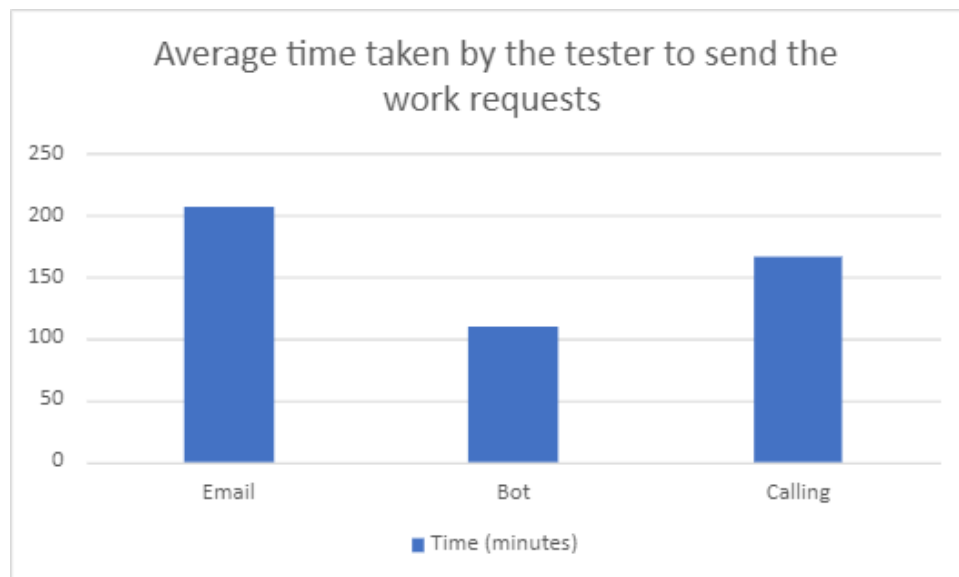


Figure 13: Average time taken to send ticket

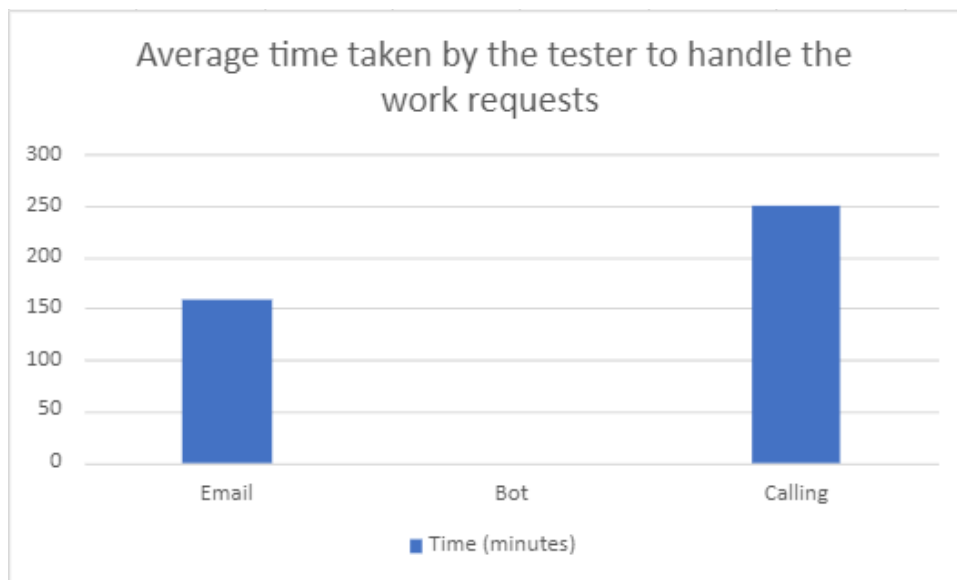


Figure 14: Average time taken to handle ticket

Most testers said that handling the work request sent from the bot takes no time for the employee as seen in figure 14, but one tester pointed out that there might be a need to check that the information is correct. Looking at figures 13 and 14, we see that using the bot takes less time than calling and email for both the user and the employee. Email takes less time than calling as the employee, but the opposite is true for the user. A user could take less time calling in if we assume that no queue takes place whilst the employee takes the most time to answer the phone.

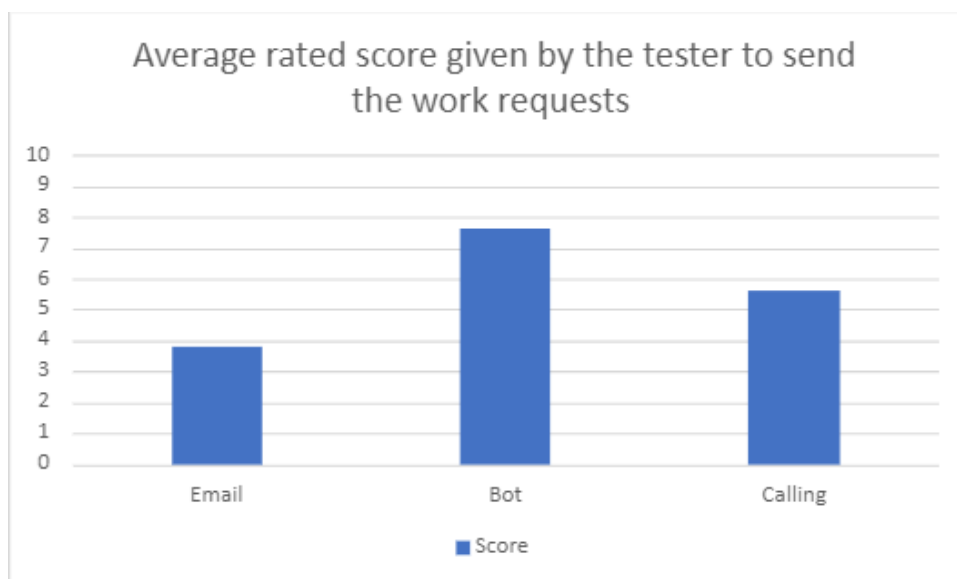


Figure 15: Average score for the methods to send tickets

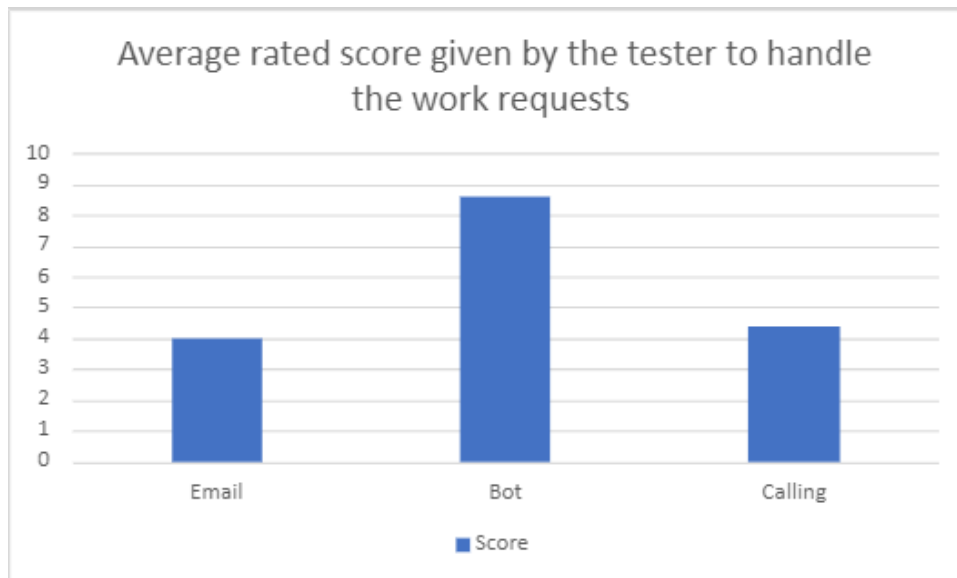


Figure 16: Average score for the methods to handle tickets

In figures 15 and 16 we see that using the bot was highest rated. In figure 15, the customers have rated emailing the lowest with calling in second place. The users acting as employees had no real preference between handling an email or a call.

Additional input given by the testers:

The tester that preferred calling customer service gave the input that they would be using the bot once it provided the same level of service as human customer support.

One tester included the input that they would like an email confirmation with the work request after they sent it in via the bot.

One tester gave the input that they thought having the customer write and send in the work request via the bot was more efficient and less time consuming for the employees.

6 Discussion

6.1 Integration

As we can see from the results, the integration was successful. It is now possible to post tasks from the bot into the database without having anything to do with the Easit-GO application.

Since using the Bot Service via Teams does not require any type of connection to the Easit-GO application, it is now entirely possible for customers to create Tickets from outside the system.

This means that it is possible to allow users to post to the database without posing any security threat of being inside the system. What might be a problem though, is access to the bot. At this moment, anyone having access to the bot would be able to post data to the database. It could be an issue if someone maliciously spammed the database with data. This issue could be solved by limiting access to the chatbot.

The integration of the two systems was successful, which means that integration between other systems would also be entirely possible. This successful integration thus works as proof of concept that this kind of implementation is possible.

6.2 Bot

The chatbot implementation was done with simple input data for the forms. This does not have to be the case when implementing a future bot. Any kind of input data can be added as long as the same data is prepared for the database in the Easit-GO application.

Many different types of adaptive cards are available for development and can possibly be designed to include information from the Easit-GO application given by data obtained through the API. The Bot could even be implemented to give ticket forms tailored to the specific customers.

This chatbot is a simple design that gets the work done to show the proof of concept.

6.3 Question Form

We can see from the results of the question form that the bot was the most preferred way of sending a work request. One of the testers liked calling

the company more though. He said that he would use the bot if it provided the same level of service as a human would. This concern that important information would be missed or misunderstood by text is not something to be ignored.

In this topic using email as the method should also be discussed. As we see in the results, no one seems to like email compared to the other two options. The reason could be that the customer might not know what information they need to include when sending the work request. When calling, the employee would ask you for the necessary information and a well programmed bot would require you to include the necessary information before sending. Another thing about emails is that you usually include unnecessary information such as greetings and such, which would not be needed in a bot.

The issue of giving the same information during a call and to a bot could be solved with a ticket form card sent by the bot that is adjusted for the type of work request that is to be sent. Another thing calling the company is advantageous is that you get the chance to ask questions whilst also giving you confirmation that the request has reached the company. The confirmation part can be implemented in the bot though. After sending the work request the bot can answer if the information was successfully sent, and additional code can be added to make sure the customer receives an email with the order information.

Calling the company also has its flaws. Some companies have automated response bot directing you to the correct employee, which can take time. Other times you may sit in a queue for a long time waiting to get through. There is also the problem of no one picking up and only being able to call during work hours. All these problems are easily bypassed by using the bot, which is accessible at any time of the day.

Looking at the average time it took the testers to hand out the work request to the company, we see that the bot took the least amount of time followed by calling. The real-world situations could differ more though, but the user testers time gives a sense of how it could work. For example, writing an email could take way less time whilst calling would have you wait 20 minutes to get through. The results make sense for a minimum time taken to send the request. Using the bot would require you to fill out a form and press send, which is a short task, whilst sending an email

requires you to think about what to include in the message. Calling could be either fast or slow depending on the situation.

The ratings the testers gave the different methods show a base line of how the different methods are liked. Email, being the most disliked method and the bot service the most liked, shows that having to write the work request is not the problem. Knowing that the correct information is given is one of the most crucial factors in the decision. That and how easy it is to send work requests to the company.

When we get to the results of the testers acting as employees there is suddenly a significant difference in the ratings of the methods. Calling and emailing are suddenly both very low whilst the bot is highly rated. Looking at the average time taken for the employee to handle the task shows us why. Suddenly using the bot takes no time for the employees since all the work requests are automatically sent to the database and shown on the application. One of the testers even wrote the input that they liked not having to deal with the customer.

Having the customers use the bot has now saved even more time for employees to work on solving the tasks instead of being customer service. As noted by a tester, there might be more work though. There might be a need to double check that the work requests were correctly filled in or there might be a need to contact the customer for additional information. Despite this, I would still argue that the ease of access for the customers and the time saved for the employees is a big reason for using the bot.

- How does the solution solve an existing problem?

As we can see from the results, this solution releases a lot of time needed to handle work requests. Instead of spending that time on handling requests the employees can now continue doing more important things. It is also easier for customers to fill out the needed information in the form given by the bot compared to the customer having to write an email. The customer can also create a work request at any time of the day which would not be possible when calling.

- Justify the efficiency of the solution.

The efficiency of the solution is shown in the time it takes for a customer to fill out the form and send the request but is even more visible for the

employees. Instantaneous delivery of the work request to the database without any needed work of an employee is an improvement to the previous methods.

Both the previous questions can also be answered with the word automation. The automation that is created by implementing integration between applications is a good reason to try to use integration more as a solution to problems.

Related Work

In the paper “An Overview of Chatbot Technology” [31], it is discussed how Chatbots can reach out to broad audiences on messaging apps whilst being more efficient than humans are. They conclude that chatbots provide significant savings in the operation of customer service departments which strengthens the conclusions reached in this project. [31]

6.4 Ethical and Societal Discussion

When discussing software ethics, we should look at the challenges that may arise. These include security concerns, unexpected behaviors from what we create, technological deficiencies and negative social impacts.[30]

If there are any security concerns in this project it probably has to do with directing business related information over the Teams application. This is a concern that arises no matter if you use the ticket bot, email or calling as the method to contact the company. The choice the company must make is if they want to use Teams or not. The weak point of security is the information stored in chat messages and possibly the use of the API endpoint.

Unexpected behavior could include an outsider getting access to the bot and thus being able to spam unwanted tickets to the database of the company. Such an issue would require the company to be careful about who they give access to the bot. Other unexpected behavior could arise but, would need to be fixed as they appear since there is no way of knowing they will happen since they are unexpected.

Technological deficiencies could be difficult to predict if the software is seemingly working to begin with. This is something that time and use of the software will show and thus require a fix for afterwards.

Social impact is in this case on the positive side since it releases the workload from customer service and allows more efficient work to be completed. Changing the way customers interact with the company from using email or calling to instead be using a bot seems to be a positive improvement looking at the results. How this affects society in the long run is hard to say, but the way technology is moving is more contact with automated bots and less contact with employees.

Replacing customer service with bots leads to less human contact. This might be a problem since in some cases the problem might be very complicated. In these cases, having direct contact with the company might be a better option. This type of bot service does not exclude customer service though. It is still entirely possible for a company to offer both customer service by calling and with a bot service.

6.5 Future Work

Since this project focuses on making the integration work, there are a bunch of aspects of this integration that can be improved upon.

The company this project was done for will of course have to adjust the code so that it is usable for their own customers. The code developed in this project is more of a guide for them to use when setting up and preparing for whatever use their customers will ask for.

More functionality for the bot. The bot doesn't only have to be a Ticket bot but could also give other information. The welcome message and commands/help message can be improved upon with Adaptive Cards. Adaptive Cards could also be used to send information about prices, products and services the company offers. The functionality can be adjusted to whatever the specific company desires.

Waterfall dialog for the bot. To make the bot easy to use after more functionalities have been added, there needs to be a good structural guideline for how the bot will work. All the additional code should then be written with Waterfall Dialog in mind, such that a user can easily use

the bot. With that method the user should not be able to get lost in the functions and always easily get back to the beginning.

Possibly implement the chatbot to use AI for answering questions. This is something we often see with automated answering systems and could possibly be useful, depending on what the bot is meant to be used for.

Send an email confirmation of the order when it reaches the database. Not only should the bot tell the user that it is currently sending a request, that the request arrived successfully, but it could also be advantageous to send the user confirmation via email with the information sent with the ticket. Such an email could be used as proof of order for the customer.

Disable old form cards in such a way that it's only possible to send one request per card called upon. This is an important feature to work on since the customer could accidentally send a duplicate of the same ticket without knowing. Disabling the card after use would make it more difficult for the customer to send tickets that aren't intended to be sent.

Add more required form questions depending on the database object. The database objects are decided on the web application and thus must be adjusted for in the code. Depending on what type of object is to be created and what information is needed, the Ticket and JSON format needs to be changed accordingly.

Add more requestable Ticket Cards depending on the problem type. The companies running the application might want more types of Tickets available for the customers and thus must be created and adjusted for the type of information needed.

References

- [1] IBM, "Application Programming Interface (API)"
<https://www.ibm.com/cloud/learn/api> Retrieved 2021-08-23.
Retrieved 2021-08-23.
- [2] Roy Thomas Fielding "Architectural Styles and the Design of Network-based Software Architectures", Chapter 5
https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm Retrieved 2021-08-23.
- [3] Stackoverflow, "What is an api-endpoint?"
<https://stackoverflow.com/questions/2122604/what-is-an-endpoint>
Retrieved 2021-08-23.
- [4] Oracle, "What is a Chatbot?"
<https://www.oracle.com/chatbots/what-is-a-chatbot/> Retrieved 2022-01-07.
- [5] Generation Digital, "What is an API Integration?"
<https://www.gend.co/blog/what-is-api-integration-a-guide-for-non-technical-people> Retrieved 2022-01-07.
- [6] Official JSON website, "Introducing JSON"
<https://www.json.org/json-en.html> Retrieved 2021-08-23.
- [7] Helpshift, "What is a Ticket System"
<https://www.helpshift.com/glossary/ticketing-system/> Retrieved 2021-08-24.
- [8] Easit, "Lär känna plattformen" <https://easit.se/produkter/easit-go/>
Retrieved 2021-08-23.
- [9] Postman Learning Center, "Introduction"
<https://learning.postman.com/docs/getting-started/introduction/>
Retrieved 2021-08-23.
- [10] Microsoft Teams, "Frontpage" <https://teams.microsoft.com/>
Retrieved 2021-08-23.
- [11] Microsoft Adaptive Cards, "Adaptive Cards"
<https://adaptivecards.io/> Retrieved 2021-08-23.

- [12] Microsoft Docs, "Csharp" <https://docs.microsoft.com/en-us/dotnet/csharp/> Retrieved 2021-08-23.
- [13] Microsoft .NET, ".NET" <https://dotnet.microsoft.com/> Retrieved 2021-08-23.
- [14] Microsoft .NET, "ASP.NET" <https://dotnet.microsoft.com/apps/aspnet> Retrieved 2021-08-23.
- [15] Microsoft Azure, "Azure" <https://azure.microsoft.com/services/bot-services/#overview> Retrieved 2021-08-23.
- [16] Microsoft Docs, "Bot Framework Emulator" <https://docs.microsoft.com/en-us/azure/bot-service/bot-service-debug-emulator?view=azure-bot-service-4.0&tabs=csharp> Retrieved 2021-08-23.
- [17] Codecademy, "What is an IDE?" <https://www.codecademy.com/article/what-is-an-ide> Retrieved 2022-01-07.
- [18] Eclipse Official website, "Eclipse" <https://www.eclipse.org/> Retrieved 2021-08-23.
- [19] Visual Studio, "Visual Studio 2019" <https://visualstudio.microsoft.com/vs/> Retrieved 2021-08-23.
- [20] Apache Maven Project, "Introduction" <https://maven.apache.org/what-is-maven.html> Retrieved 2021-08-23.
- [21] Eclipse Foundation, "Jetty Maven Plugin" https://wiki.eclipse.org/Jetty/Feature/Jetty_Maven_Plugin Retrieved 2021-08-23.
- [22] ZK official website, "ZK Framework" <https://www.zkoss.org/product/zk/zk8> Retrieved 2021-08-23.
- [23] LESS official website, "Overview" <https://lesscss.org/> Retrieved 2021-08-23.
- [24] NodeJS official website, "About Node.js" <https://nodejs.org/en/about/> Retrieved 2021-08-23.

- [25] Git official website, "Git" <https://git-scm.com/> Retrieved 2021-08-23.
- [26] Bitbucket official website, "Bitbucket" <https://bitbucket.org/> Retrieved 2022-01-07.
- [27] Mysql, "MySQL" <https://www.mysql.com/> Retrieved 2021-08-23.
- [28] Ngrok, "What is Ngrok?" <https://ngrok.com/product> Retrieved 2021-08-23.
- [29] Microsoft Docs, "About component and waterfall dialogs" <https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-concept-waterfall-dialogs?view=azure-bot-service-4.0> Retrieved 2021-08-23.
- [30] TheNewStack, "The Five Principles of Software Ethics" <https://thenewstack.io/five-principles-software-ethics/> Retrieved 2021-08-23.
- [31] Adamopoulou E., Moussiades L. (2020) An Overview of Chatbot Technology. In: Maglogiannis I., Iliadis L., Pimenidis E. (eds) Artificial Intelligence Applications and Innovations. AIAI 2020. IFIP Advances in Information and Communication Technology, vol 584. Springer, Cham. https://doi.org/10.1007/978-3-030-49186-4_31. Retrieved 2021-08-23

Appendix A: Question Form 1

Bot Integration: Sending Work Request

1. What way of contacting the company did you prefer?

[More Details](#)

● Emailing	0
● Using The Bot	4
● Calling	1



2. What did you prefer about that way of contacting the company?

5 Responses

ID ↑	Name	<i>Method</i>	Responses
1	anonymous	<i>Bot</i>	It was a guided form that made it easy and simple to follow. Plus I didn't need to do any guess work , in case I forgot to include some information that the company might need to know later.
2	anonymous	<i>Calling</i>	Calling is fastest and most accurate
3	anonymous	<i>Bot</i>	Easy to use. It was straight forward what I would do when I were filling out my problem.
4	anonymous	<i>Bot</i>	Jag vet alldrig hur länge jag får sitta i kö för att få iväg det jag behöver beställa. Email beställning vet jag inte om de svara på.
5	anonymous	<i>Bot</i>	smidigt och man får bekräftelse direkt

3. Using Email: How much time did it take on your end to send the message to the company?

5 Responses

ID ↑	Name	Responses
1	anonymous	2 min 33 sec
2	anonymous	5 min
3	anonymous	4 min 37 sek
4	anonymous	2 min 27 sekunder
5	anonymous	2 minuter, 34 sekunder

4. What would you rate using Email?

5 Responses

ID ↑	Name	Responses
1	anonymous	5
2	anonymous	2
3	anonymous	5
4	anonymous	2
5	anonymous	5

5. Using the Bot: How much time did it take on your end to send the message to the company?

5 Responses

ID ↑	Name	Responses
1	anonymous	1 min 32 sec
2	anonymous	1 min
3	anonymous	2 min 22 sek
4	anonymous	2 min och åtta sekunder
5	anonymous	2 minuter 10 sekunder

6. What would you rate using the Bot?

5 Responses

ID ↑	Name	Responses
1	anonymous	7
2	anonymous	6
3	anonymous	9
4	anonymous	8
5	anonymous	8

7. Calling: How much time did it take on your end to send the message to the company?

5 Responses

ID ↑	Name	Responses
1	anonymous	2 min 47 sec
2	anonymous	3 min with no queue time
3	anonymous	3 min 11 sek
4	anonymous	2 minuter
5	anonymous	2 minuter, 34 sekunder

8. What would you rate Calling?

5 Responses

ID ↑	Name	Responses
1	anonymous	6
2	anonymous	8
3	anonymous	3
4	anonymous	5
5	anonymous	6

8. What would you rate Calling?

5 Responses

ID ↑	Name	Responses
1	anonymous	6
2	anonymous	8
3	anonymous	3
4	anonymous	5
5	anonymous	6

9. Thanks for answering! Any other Input?

5 Responses

ID ↑	Name	Responses
1	anonymous	No thanks'1
2	anonymous	I'd use help bots if they provided the same level of service as human customer support
3	anonymous	No
4	anonymous	Bra med ett email när beställningen är påbörjad och klar
5	anonymous	nej

Appendix B: Question Form 2

Bot Integration: Receiving Work Request

1. What way of handling the work request did you prefer?

[More Details](#)

● Emailing	0
● Using the Bot	5
● Calling	0



2. What did you prefer about that way of receiving a work request?

5 Responses

ID ↑	<i>Method</i>	Responses
1	<i>Bot</i>	I didn't need to manually input the problem. And I received an instant note about the problem which I can get to working on straight away.
2	<i>Bot</i>	The work of recieving and interpreting the request is already done by the bot, and all that is left is to check that everything is correct rather than inputing the data on your own
3	<i>Bot</i>	No extra time for unnecessary stuff. I could start right away with what was requested by the customer
4	<i>Bot</i>	det var redan inlagt och klart i systemet
5	<i>Bot</i>	SLippa prat med den som ufärdar ordern

3. Using Email: How much time did it take on your end to handle the request?

5 Responses

ID ↑	Name	Responses
1	Alabbas Saifuldeen Hayder Hayder	3 min 24 sec
2	Adil Aboulkacim	2:05
3	Linus Blomqvist	3 min 47 sec
4	Jonas Carlsson	2 minuter, 3 sekunder
5	anonymous	1 min 57 sek

4. What would you rate using Email?

5 Responses

ID ↑	Name	Responses
1	Alabbas Saifuldeen Hayder Hayder	5
2	Adil Aboulkacim	2
3	Linus Blomqvist	5
4	Jonas Carlsson	6
5	anonymous	2

5. Using the Bot: How much time did it take on your end to handle the request?

5 Responses

ID ↑	Name	Responses
1	Alabbas Saifuldeen Hayder Hayder	Instant
2	Adil Aboulkacim	20 seconds to check that the request information is correct from the customer
3	Linus Blomqvist	I didn't have to do anything with the customer request of their problem
4	Jonas Carlsson	0 sekunder
5	anonymous	Ingen tid

6. What would you rate using the Bot?

5 Responses

ID ↑	Name	Responses
1	Alabbas Saifuldeen Hayder Hayder	8
2	Adil Aboulkacim	8
3	Linus Blomqvist	8
4	Jonas Carlsson	10
5	anonymous	9

7. Calling: How much time did it take on your end to handle the work request?

5 Responses

ID ↑	Name	Responses
1	Alabbas Saifuldeen Hayder Hayder	5 min 21 sec
2	Adil Aboulkacim	3:30
3	Linus Blomqvist	4 min 20 sec
4	Jonas Carlsson	3 minuter, 34 sekunder
5	anonymous	4 min

8. What would you rate Calling?

5 Responses

ID ↑	Name	Responses
1	Alabbas Saifuldeen Hayder Hayder	6
2	Adil Aboulkacim	4
3	Linus Blomqvist	4
4	Jonas Carlsson	4
5	anonymous	4

9. Thanks for answering! Any other Input?

5 Responses

ID ↑	Name	Responses
1	Alabbas Saifuldeen Hayder Hayder	No
2	Adil Aboulkacim	As an customer service employee handling a filled out ticket seems easier and less time consuming than receiving an email or call
3	Linus Blomqvist	Nope
4	Jonas Carlsson	nej
5	anonymous	nej