Chatbot to ease the effort of handling convocations for groups.

Lorenzo Hidalgo-Gadea

Resum– En el cas que tingui instal·lada qualsevol aplicació de missatgeria instantània i formi part d'alguna conversa en grup, sabrà la gran quantitat de missatges irrellevants que es reben i el que suposa posar-se d'acord o gestionar una convocatòria. És per això, que en aquest article, exposem com hem desenvolupat un xatbot per a Telegram per a combatre els missatges de contingut brossa a l'hora de gestionar convocatòries. Hem aprofitat tambe aquest desenvolupament per a fer una pàgina web en la qual l'usuari pugui analitzar el comportament dels usuaris per a poder maximitzar l'eficàcia de les convocatòries.

Paraules clau– Aplicació Web, Automatització, Chatbot, Convocatories, Python, Raspberry, Servidor, Telegram.

Abstract– If you have any kind of instant chat application and are part of a chat group, it is very likely that you repeatedly receive unwanted messages and know how hard it is to handle a convocation. This is why, in this article, we showcase how we developed a Chatbot for Telegram to combat the high amount of irrelevant messages sent in chat groups and to ease the handling of convocations. We also take advantage of the project data collection to develop a website, which enables the analysis of the users' behaviour to maximize the convocations efficiency.

Keywords– Web Application, Automatization, Chatbot, Convocations, Python, Raspberry, Server, Telegram.

1 Introduction

SOCIALIZING is one of the most important aspects of human behaviour and with the appearance of new technologies, like social networks and instant chat applications, the urge of sharing information with other people has exponentially grown. With this growth new problems have appeared, for instance most people are unaware of the amount of messages they send nor if the used channel is the most appropriate one. This behaviour makes handling convocations in Chat Groups very difficult, since the irrelevant messages are mixed in with the important ones making it difficult to follow the conversation.

For example, the *Catalan Red Cross* has different emergency response teams formed by large number of volunteers. This organization uses different Chat Groups to rapidly handle convocations to cover the different situations that may appear. Normally those Chat Groups are only used when an emergency response team has to be activated, but often people tend to forget the purpose of the group and end up sending irrelevant messages or spam.

Another example would be a training team, like for instance the rowing team *Llagut Squad*. Since a minimum number of people are needed to carry out a training, the team captain uses a Chat Group to send convocations to all the team members. This groups are mainly used to handle the convocations and share valuable information about the training sessions or the team itself, but it is also often used to share irrelevant messages or funny pictures.

In both cases there is the need to contact a large group of people, get their responses and treat them accordingly. As we have seen, the most popular solution is to use a Chat Group formed by all the members who need to be contacted. This solution has the benefits of enabling the convener to contact instantly a large group of people with a single message and to get all their responses via the same channel. But some inconveniences are present too. The convener has to read and treat each and every message of the Chat Group, organize them and respond with the selected members and the proposed positions. But most users usually don't use this Chat Groups according with their purpose, sending unwanted messages that in most cases induce other members to silence this group's notifications.

• E-mail de contacte: LorenzoHidalgoGadea@gmail.com
• Menció realitzada: Enginyeria de Computadors
• Treball tutoritzat per: Dolores Isabel Rexachs del Rosario (Departament d'Arquitectura de Computadors i Sistemes Operatius)
• Curs 2018/19

To try to improve this situation, we propose a Chatbot based solution to handle the convocations.

This article is divided in eleven different sections. We will start by analysing the starting point and explaining our objective, in sections 2 and 3. Following the objective we will make a brief introduction to what a Chatbot is and comment the state of the art of the different chat applications, sections 4 and 5. The section 6 is the main section of this article, where we describe how we designed and developed the Chatbot. In the following two sections, 7 and 8, we showcase the resulting Chatbot and web application. Finally we expose our results, conclusions and future work in the last three sections, 9, 10 and 11.

2 STARTING POINT

To be able to evaluate the situation we have decided to analyze the behaviour of the members of four different task-oriented Chat Groups, which I am a member of. Following a brief explanation of the activity that each group carries out:

- **Group 1:** A group of volunteers from the Catalan Red Cross that prepares activities for the elderly.
- **Group 2:** A sub-group of volunteers of the Catalan Red Cross sea search and rescue team.
- **Group 3:** An emergency response team of the Catalan Red Cross specialised in emergency shelters.
- **Group 4:** A group of 12 recreational athletes in a local rowing team.

Once the teams were selected, we exported the chats to a text file and analysed all messages sent during a specific time period, Table 1.

Table 1: GROUP DETAILS

Group	G-1	G-2	G-3	G-4
Days	144	82	43	102
Members	31	10	72	12
Messages	1126	280	94	1051
Important ones	171	155	33	431
Spam	84.8%	45.6%	64.9%	60%

After analysing these results, it is obvious that none of the groups are being used as intended and with this high percentage of spam messages the important information is more likely to pass unnoticed.

We have also used the following formula to get the number of minimum messages everyone would get if we handled the convocation through a common Chat Group.

$$TotalMessages = 2 + minimumMembers \quad (1)$$

In this first case, we would have an opening and a closing message, therefore the two as a constant. To this variable we would add all the needed minimum positive responses. For G-3, there is a minimum of 5 people for an emergency response team, therefore the minimum of $TotalMessages = 2 + 5 = 7$ messages sent to the group if the convocation is accepted.

Whereas if everyone denies its participation in the convocation, the following formula would apply:

$$TotalMessages = 2 + (totalMembers - (min. - 1)) \quad (2)$$

In this case, to be forced to cancel a convocation we would need to be unable to create a minimum team, therefore we substract one to the minimum Members needed to form a group. We would get $TotalMessages = 2 + (72 - (5 - 1)) = 70$ in the case that every user would respond in a single message.

With a brief look at the equation 2 we can see that the number of messages will grow linearly with the growth of the users in the group.

3 OBJECTIVE

Our main objective is to develop a method to instantly send convocations privately to a group of users and automatically handling the responses, by developing a Chatbot. Thereby reducing the number of messages received by each user to a constant number of two messages, independently of the number of participants.

This development would enable the organizers to have a reliable communication method to contact the team members without the chance of irrelevant messages interfering.

To add additional value to the Chatbot and by gathering all the usage data, the most important statistics and configurations will be available through a web application.

The developed solution should be as versatile as possible to ease future modifications and implementations for future clients.

4 CHATBOTS

For those unfamiliar with the term, a Chatbot is a program that simulates interactive human conversations with a predefined behaviour based on specific words' or sentences' recognition.[1]

Although its use is just now gaining popularity, the concept falls back to 1966 when the MIT developed a Chatbot simulating a psychotherapist called ELIZA. Since then, the development of Chatbots has increased its functionality to the point of being able to create personal assistants like SIRI, Google Now or Alexa.

To the date, Chatbots are mostly implemented to carry out simple or repetitive tasks like Customer Service, Frequently Asked Questions and basic Personal Assistance. Most of them are implemented independently or as an added functionality for a website or application.

In our use case, the Chatbot will be implemented to handle the convocation responses.

5 State of the art

As of today there is a large number of applications that enable their users to chat with each other and to implement Chatbots to automatize conversations. All of them with different characteristics, benefits and inconveniences. Following, an analysis of the 3 most downloaded communication apps in Spain as of April 2018 [2] that allow developers to implement Chatbots, Fig. 1:

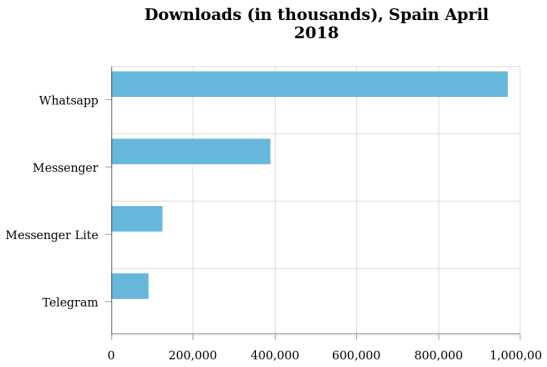

Fig. 1: Downloads in Spain April 2018

- **WhatsApp Messenger:** The most popular Chat application. A user needs a telephone number to register. Users have the ability to chat privately with another user, to create Chat Groups and broadcasting channels. WhatsApp announced the launch of its Business API, but as of march 2019 it's a closed beta program and they only grant access to enterprises. (967.410 downloads)

- **Facebook Messenger:** Facebook is one of the most known social media applications and one of their features is the capability to send messages between members. As Facebook's popularity grew, they separated their messaging feature to a stand alone application for mobile devices. To use the app users need to have a Facebook profile and, in some cases, be befriended with the user they want to chat with. Messenger gives their users the capability to create Chatbots and in-chat games. (386.920 downloads + 123.540 downloads for the lite version)

- **Telegram:** Telegram is one of the most used chat applications for the most tech-savvy users. The project is open source, with a great developer community. It's more privacy oriented than the above mentioned applications and it has the most in-chat functionalities. Telegram can run on a mobile device or on a desktop application, which makes it very versatile. (90.050 downloads)

Table 2: Chat Applications Compared

	Downloads	Open Source	Bot support
WhatsApp	967.410		
Messenger	386.920		X
Telegram	90.050	X	X

Having the Table 2 and the previous information in mind, the most attractive chat application for our purposes is Telegram, due to the fact that it's open source, it has support for Chatbot implementations and its well established developer community.

6 Designing and Developing The Chatbot

With the four analysed groups in mind, Table 1, we will be designing, developing and testing the Chatbot for the fourth group, the sports team. Doing a pilot with the volunteers organization won't be possible for ethical reasons, since the lack of testing and the reliability compromises could affect the organizations outcome.

The development will be done in different modules, so that the final result is personalized for the sports team but the main modules of the code may be utilized for future developments for other teams with different purposes.

6.1 Project risks

The main risks and handicaps that the project has are the platform and the ease of use of its functionalities. Following, a brief explanation of the detected risks.

- **Chat platform**: Since the Chatbot will be accessed throughout a chat application, the election of the same will impact the user's adoption. If the chat application isn't easy to install, slightly popular or free, the chances of the user's rejection will be very high.

- **Usability**: The procedures and options have to be as intuitive and easy to use as possible. The lack of intuition or desired functionalities will drive users to reject the Chatbot and continue using the previous methods.

- **Usage Benefits**: Since the user will, most likely, have to install a new application and adapt to the new methodology, there have to be user oriented improvements and benefits. If users don't recognize any improvement, they will most likely reject the new methodology.

6.2 Modules - Functionalities

Every Chatbot will have the following basic functionalities. A more in depth chart of the specific modules and functionalities, that will be developed for the proof of concept with the training team, has been added to the appendixes, A.1 and A.2.

- **Privileges**: The following user types have been designed to control the access to the different functionalities.

  1. **User**: Restricted access.
  2. **Captain**: Restricted access, full access to its team/group.
  3. **Manager**: Full access.
  4. **Administrator**: Full access.

- **ChatBot**:
    1. **User information**: Each user will be able to view and edit their stored information.
        (a) **View**: Name and Telegram ID.
        (b) **Edit**: Name.
    2. **Team information**: Each user will be able to view the stored team information.
    3. **Convocations**: Only the administrator, captain and manager will be able to send messages. The users will be able to respond to the convocations with their availability.
        (a) **Send**: New convocations, team alignments and important related information.

- **Web interface**: Only available for the administrator, team captain and section manager.
    1. **User information**: It will be possible to view the basic user information.
    2. **Team information**: It will be able to edit relevant team information, such as: Team name, members and captain.
    3. **Training Sessions**: It will be possible to edit the predefined text for the convocations, training sessions ratings, team members ratings and team members statistics will be shown.
    4. **Statistics**: The most important statistics will be shown.

A part from the above mentioned basic functionalities, additional functionalities and GUI improvements were developed.

6.3 Development methodology

For the development of the project, and with the purpose of starting to record as much usage data as possible, we followed the *DevOps* methodology [3] starting the development with *Convocations*, Fig. 2. Once this module had been tested and deployed, the pilot began to gather usage data while the development of the rest of functionalities were being worked on.

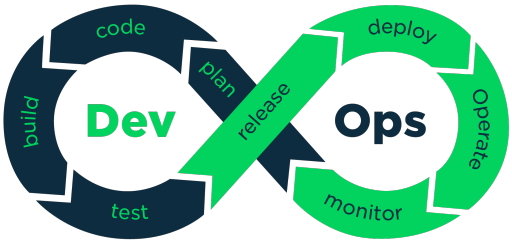

Fig. 2: DevOps Methodology.

Source: https://medium.com/@neonrocket/devops-is-a-culture-not-a-role-be1bed149b0

One of the main arguments to choose this model is the quick generation of a working software in the early stages of the software life cycle. Since the development was planned by different Modules or functionalities, it eased the use of the DevOps model, by separating each module development in different iterations.

6.4 Planning the development

We initially planned the development to be done in eight iterations, each of them consisting of two weeks. A part from those iterations we also planned to have some free time to be able to write the different reports that we needed to hand over.

- **Iteration 1:** Legal Viability Study, Platform and frameworks selection Data Base design and implementation.
- **Iteration 2:** Starting the development of the modules 'User Information' and 'Convocations'.
- **Iteration 3:** Starting the pilot with the functionalities developed in the second iteration and developing the modules 'Team Information' and 'Training Sessions'.
- **Iteration 4:** Adding the newly developed functionalities to the pilot, solving possible occurred problems and applying the received feedback to make the first upgrades.
- **Iteration 5:** Deploying a web server and developing the first two modules of the Web Site.
- **Iteration 6:** Finishing the web site by developing the two remaining modules and letting it ready for deployment.
- **Iteration 7:** Deploying the web site, solving possible occurred problems and upgrading the Chatbot with the received feedback from the pilot.
- **Iteration 8:** Final touches to the Chatbot and the Website and writing all the needed user documentation.

Although this planning might seem linear, it was intended that after a functionality was developed and tested it would be deployed starting the *Ops* cycle. During all Iterations there was a constant monitoring of the deployed Chatbot to ensure a correct execution and to find possible bugs and improvements.

6.5 Considerations

During the development of the project some crucial decisions were taken. Following a brief argumentation of each consideration taken.

6.5.1 Legal Viability

In order to develop a viable project we had to comply with the *General Data Protection Regulation (RGDP)* [4] which came into force in May 25, 2018 and the *Real Decreto 1720/2007 (RLOPD)*[5].

The data that we had planned to store, and could be seen as personal information, was: Names, Telegram ID codes

and usage data linked to every user. This information was for organizational and statistical use only.

For the proof of concept we asked all our users for a written authorization to store and handle the above mentioned information. For future implementations of the Chatbot, all responsibility will fall on the client.

6.5.2 Platform and frameworks selection

Chatbot:
Following the statistics shown in the state of the art, Fig. 1, the best option would be to develop the solution for WhatsApp, since it is the most popular and used chat application. The main problem is that to the date there is no public API from WhatsApp itself and the only available options are third party API's that are neither open source nor accepted by the terms of use of WhatsApp.

Having this in mind, the best solution was to implement it for Telegram. Telegram is an open source project with publicly available source code and a specific BOT API, which is highly documented [6] and used by a large developer community. To ease the development we developed the Chatbot using one of their recommended third party frameworks *python-telegram-bot* [7] which under *GNU Lesser General Public License v3.0* [8] we were able to use freely for our purpose.

The *python-telegram-bot* framework is written in python and allows the developer to code Telegram Bots [9] easily by taking care of all the Webhook calls needed. It is also a very good documented [10] and up to date project, allowing us to use the latest deployed features.

Data Base:
For our Data Base server we used *MariaDB*, an open-source fork of *MySQL*, developed by the original *MySQL* developers, and also distributed under the *GNU General Public License*. Our Main arguments to use this RDBMS were the developer community and the vast amount of documentation available.

Website:
For The Website development we decided to use the *Laravel PHP Framework* [11], because of the short learning period needed to start developing, the use of the MVC model, the fact that it is a lightweight framework and that it has a vast documentation and developer community.

To ease the development we used *Bootstrap* [12] as the CSS library, *Laravel Charts* and *Frappe* to facilitate the visualization of the statistics and *DataTables* to beautify and empower the HTML tables with different functionalities.

All of those libraries and frameworks are available under the MIT [13] software license and therefore free to use for our project and are working together as shown in Fig. 3.

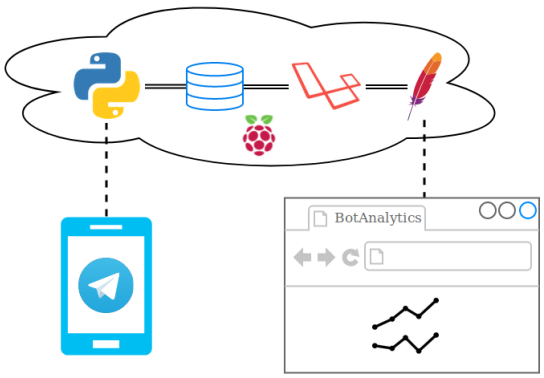

Fig. 3: System Components

6.5.3 Server selection

Once we selected the software and decided the different modules and functionalities our Chatbot and Website should have, the only thing left was to choose the Hardware we would use.

Having seen other Telegram Chatbot scripts before, we now that the HW requirements to run them are very limited, there are even projects developed on ESP8266 microcontrollers [14].

From previous projects we also know that a *Raspberry Pi 2* is very capable for hosting basic Websites and Data Bases without putting to much stress on the system.

Therefore we used the *Raspberry Pi 2*, connected directly via Ethernet cable to the router, as our production server and a laptop as the local development server.

6.6 Development

During the development we encountered several problems, most of them during the implementation of the different functionalities to the Chatbot with the chosen library.

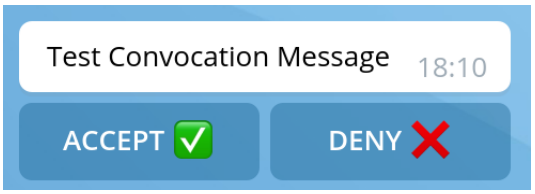

Fig. 4: Received Convocation Message.

For Example, when a user received a convocation message, Fig. 4, he was presented with two buttons, one to accept and one to deny. By clicking the button the message would automatically update, Fig. 5, and the response was saved in the data base.

Fig. 5: Accepted Convocation Message.

Since it is possible to have multiple convocations active at the same time, we needed a method to be able to link the answer, that was send by clicking a button, and the original convocation identifier to save the response.

The class used to create the buttons was *class telegram.InlineKeyboardButton(text, url=None, callback_data=None, switch_inline_query=None, switch_inline_query_current_chat=None, callback_game=None, pay=None, **kwargs)* and the input arguments important to us were *text and callback_data*.

Once a button was clicked, it would send a message with the *callback_data* to the original sender. This data is normally used to control the behaviour of the Chatbot.

To solve our problem, and seeing that the returned data was *callback_data*, we concatenated the action name that the bot needed and the convocation identifier as so *InlineKeyboardButton("ACCEPT, callback_data='ACCEPTED '+str(Convocation_ID))*.

With this method the bot was able to parse the received data and to extract the action, *ACCEPTED*, and the convocation identifier. Being now able to save the selected answer related to an user identifier and a convocation identifier in the data base.

The development of the website was pretty straight forward, without any encountered problems related to the frameworks or libraries used to develop it.

7 The Chatbot

In the developing stage we were able to implement and test all the functionalities proposed in A.1 and the different user roles.

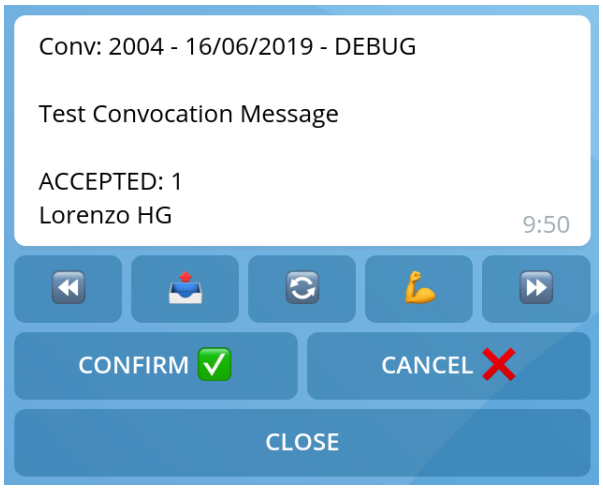

Fig. 6: Convocation Manager.

For example, Fig. 6, is the message viewed to manage all the convocations. Depending on the User role, you are able to see the convocations that you have send or all the convocations from all Teams. A regular user would not be able to access this view.

Following a brief explanation of the different buttons.

- ⏪: To view the previous convocation.
- **Output post**: To broadcast the current status to the whole team.
- **Reload**: To reload the current convocation.
- **Arm**: To link a training type to the convocation.
- ⏩: To view the next convocation.
- **Confirm**: To confirm the current convocation, it prompts you to add an alignment.
- **Cancel**: To cancel the current convocation, it prompts you to add a cancel type.
- **Close**: To close the current view, all buttons disappear.

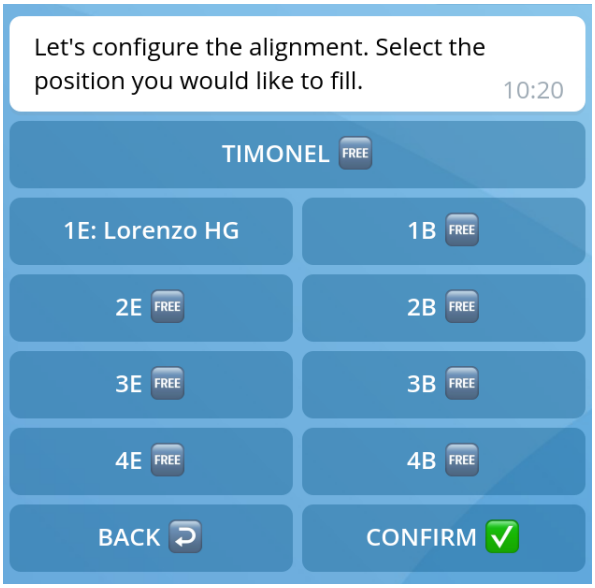

Fig. 7: Alignment view.

The view to select an alignment, Fig. 7, is prompted once you click the confirmation button on the Convocation Manager view, Fig. 6.

All free positions appear with the *FREE* symbol, and once you add a user, its name appears on the button as in Fig. 7.

To add a user to a position click the desired position and you will be prompted with a list of the users left to assign. By selecting a position that is already assigned, you are able to reassign it to a new user. By doing this, the user who had already been assigned is added back to the list of users pending to be assigned.

It's also important to note that once you press a button, the current message updates itself with the new view.

8 The Website

Besides the Chatbot, and to add even more value to the Chatbot solution, we successfully developed all proposed functionalities for the Website, A.2.

The web application enables the registered users to modify its stored data, to view all the convocations sent to

their teams and to view their teams' information.

Users with privileges, such as Section Managers and Administrators, are able to create and modify users and teams, to view all teams' convocations and to view global and team specific statistics. Team Captains are only able to check global statistics and the statistics of their teams.

8.1 Statistics

The most important part of the website is the statistics section. This is the one which adds more value to the project, by empowering its users with an analysis from the stored data and enabling them to make decisions based on users' behaviours.

In this section we have three types of statistics: basic information, based on the convocations and based on the responses. Some statistics have a table icon near the name to expand a dynamic table with the shown information, Fig. 9.

8.1.1 Basic information

In this brief section the user is presented with a table, Fig. 8, with the most basic information about the analysed data set.

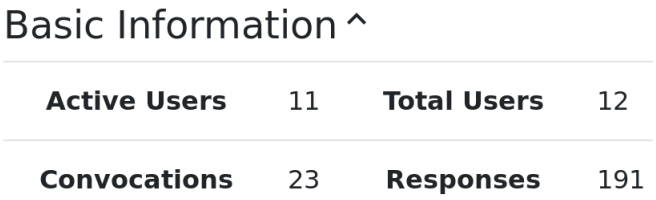

Fig. 8: Basic Information section

8.1.2 Based on Convocations

These analysis are based on the final status of the convocations.

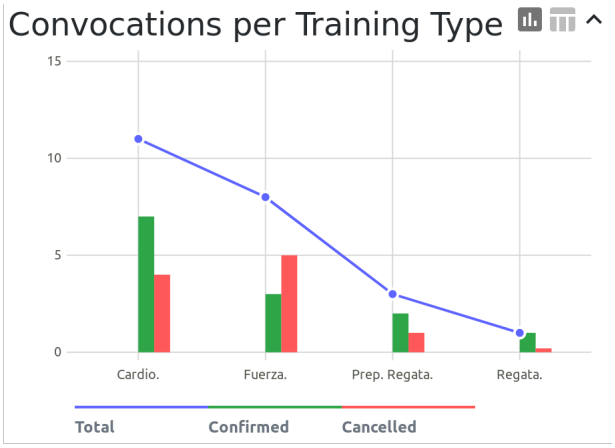

Fig. 9: Number of Convocations per Training Types

- **Convocations:** A progress chart showing the percentage of confirmed versus cancelled convocations.
- **Convocations per Cancel Types:** A progress chart showing the percentage of cancelled convocations depending on the cancel type.
- **Convocations per Training Types:** A bar chart showing the number of confirmed and cancelled convocations depending on the training type and a line chart showing the total amount of convocations per training type. Fig. 9.
- **Users per Convocations:** A line chart showing the number of users that attended each convocation.

8.1.3 Based on Responses

This analysis are based on the responses received from the users.

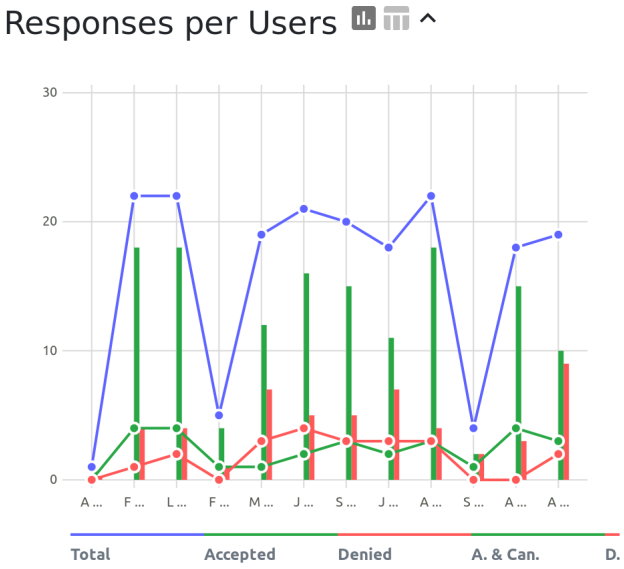

Fig. 10: Number of Responses per Users

- **Responses:** A progress chart showing the percentage of accepted versus denied responses.
- **Responses per Training Type:** A bar chart showing the number of accepted and denied responses depending on the training type and a line chart showing the total amount of responses per training type.
- **Responses per Weekday:** A bar chart showing the number of accepted and denied responses depending on the weekday and a line chart showing the total amount of responses per weekday.
- **Responses per Training Day:** A bar chart showing the number of accepted and denied responses depending on the training day and time and a line chart showing the total amount of responses per training day and time.
- **Response Time per Daytime:** A line chart showing the minimum, maximum and average response time depending on the weekday and time the convocation was send.

- **Responses per Users:** A bar chart showing the number of accepted and denied responses depending on the user. A line chart showing the total number of responses from each user and the number of accepted and denied responses from each user when the convocation was cancelled, because of a lack of people, Fig. 10.

9 Results

As a result of our work, we have been able to reduce the number of messages sent and received by all the members of the group and eased the handling of the convocations for the captain. Using the previous convocation method, *WhatsApp*, we would have a minimum of $TotalMessages = 2 + 7 = 9$ to confirm a convocation and a minimum of $TotalMessages = 2+(12-(7-1)) = 8$ to cancel a convocation, as seen in the formulas that we used at the beginning, equations 1 and 2, and considering that there is a minimum of seven people needed to carry out a training. Furthermore, we must also take into account the high amount of irrelevant messages that were sent, Table 1.

Thanks to the Chatbot, we have reduced the number of received messages per user to the constant part of the equation. Independently of the case, if the convocation was confirmed or cancelled, every user would only receive two messages, the opening message and the closing message.

Thanks to this reduction and to the fact that through this channel the users were unable to send irrelevant information, we achieved a higher response ratio to the convocations without the need of reminding the users to respond.

Apart of reducing the total number of messages sent to a bare minimum, we have provided the team captain and the section manager with added functionalities and advantages.

With the web application they are able to have a better understanding of the team members' behaviours thanks to the statistical analysis that it performs.

For example, using the statistic *Response Time per Daytime* they would be able to determine the best moment to send the convocations to each team, reducing so the waiting time to confirm or cancel a convocation.

They could also use the statistics of *Convocation per Training Types* and *Responses per Training Type* to modify or boost the training types that are most welcomed by the team.

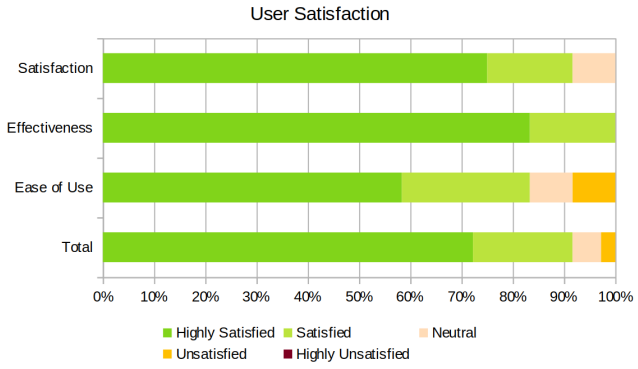

Fig. 11: User Satisfaction Poll

After three months of testing the Chatbot and applying new updates, based on users' feedback, we asked the users to answer a brief poll about their satisfaction, Fig. 11.

A global satisfaction of over 90% was reached, which is a big success taking into account, that they had to download and use a new application, *Telegram*, which they didn't previously know about.

10 Conclusions

As the conclusions, we may say that we have successfully developed a method to manage convocations for groups of people without the need of a common chat group.

This has the benefits of reducing the number of received messages to a bare minimum, in our case two messages. This amount may have varied, if the captain had chosen to broadcast the convocation status to encourage the responses from the team members. However, the amount of messages received would still be considerably reduced. Therefore avoiding, that some users mute the chat group or simply don't read the messages sent thinking it's all spam.

Also with the newly added statistic web site, the users in charge of sending the convocations and handling the teams are able to make decisions based on the behaviour of the team members. Therefore creating a better environment amongst the team members. For example, as a consequence of reducing the waiting time to confirm or cancel a convocation, team members would be able to plan other activities in case the convocations was canceled and to avoid that users end up canceling because of the long waiting time. Another example, team members would be more pleased and motivated with the training since their preferences could be taken into consideration.

Furthermore, the Chatbot reduces the work needed to handle the convocation by doing most of the work. Automatising most of the convocation process by: Presenting the response options, convocations status and the results in an intuitive way. It even notifies the sender when the minimum members needed have been reached in order to confirm a convocation, releasing the sender from the pressure of continuously checking the convocation status.

Another important aspect is, that the hardware resources needed to host our solution are very limited. In our Pilot we used a *Raspberry Pi 2* as our server, which was able to handle the Chatbot for a group of 12 people, the database and the web application at the same time flawlessly.

Since the message distribution is handled via webhooks to the Telegrams' servers and it is very unlikely that all users respond at the same time, we could use this same setup to host the chatbot for multiple or even bigger groups. The solution is also easily dockerizable or implementable as a cloud service, making it very scalable and cheap to deploy.

11 Future Work

This project has a vast variety of possible new development lines to improve different aspects of the project. But the main line of work would be adding new functionalities to the Chatbot and to the website to make them as versatile as possible.

For example, right now the users are not able to change their responses to the convocations or to add some motives if they deny it.

Another useful upgrade could be enabling the Chatbot to actively recommend actions to the captains and managers depending on the statistical analysis. For example sending reminders of when a convocation should be sent or recommending different time or training type for the convocation.

Since most functionalities are purpose dependent, for example the statistics or the alignment functionalities, we will post the developed Chatbot and web application in a GitHub Repository [15] under the MIT License. Allowing everyone interested in such a solution to fork the project and modify it to fit their needs.

Acknowledgements

I would like to thank Dolores Isabel Rexachs del Rosario for her excellent support and tutoring, Moise Otniel Romaniuc Demeter for his support while debugging different website related problems and the official *python-telegram-bot* library chat group for the help with problems related to the use of the library.

References

[1] Techopedia.com. What is a chatbot? - definition from techopedia. https://www.techopedia.com/definition/16366/chatterbot, 2019.

[2] Statista. Spain: top android messaging apps by downloads 2018 — statistic. https://www.statista.com/statistics/698478/leading-android-communication-apps-in-spain-by-downloads/, 2018.

[3] Medium. Devops is a culture, not a role! https://medium.com/@neonrocket/devops-is-a-culture-not-a-role-be1bed149b0, 2019.

[4] Eur-lex.europa.eu. Eur-lex - 32016r0679 - en - eur-lex. https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX:32016R0679.

[5] BOE. Real decreto 1720/2007 de protección de datos de carácter personal. https://www.boe.es/buscar/pdf/2008/BOE-A-2008-979-consolidado.pdf.

[6] Core.telegram.org. Telegram bot api. https://core.telegram.org/bots/api/.

[7] Python telegram bot.org. python-telegram-bot. https://python-telegram-bot.org/.

[8] Free Software Foundation. Nu lesser general public license v3.0- gnu project - free software foundation. https://www.gnu.org/licenses/lgpl-3.0.html.

[9] Core.telegram.org. Bots: An introduction for developers. https://core.telegram.org/bots/.

[10] Python telegram bot.org. Welcome to python telegram bot's documentation! — python telegram bot 11.1.0 documentation. https://python-telegram-bot.readthedocs.io/en/stable/.

[11] T. Otwell. Laravel - the php framework for web artisans. https://laravel.com/.

[12] Bootstrap. The most popular html, css and js library in the world. https://getbootstrap.com/.

[13] Open Source Initiative. The mit license. https://opensource.org/licenses/mit-license.html.

[14] Shebin Jose Jacob. Telegram bot with esp8266. https://www.hackster.io/ShebinJoseJacob/telegram-bot-with-esp8266-dbada8.

[15] Lorenzo Hidalgo Gadea. Convobot github repository. https://github.com/Lorenzohidalgo/ConvoBot.

APPENDIXES

A.1 Chatbot Modules

Module	ID	Description	Users
User Information	01	Module to view and edit the user information.	Variable
	01.01	View: The Bot will answer with all the information about the user.	ALL
	01.02	Edit: The user will be able to edit the stored information.	ALL
	01.03	Add: The user will be able to add new members.	Administrator, Team Captain and Section Manager.
	01.04	View Other: The Bot will answer with all the information about the asked user.	Administrator, Team Captain and Section Manager.
	01.05	Edit Other: The user will be able to edit the stored information.	Administrator, Team Captain and Section Manager.
	01.06	Delete: The user will be able to delete members.	Administrator, Team Captain and Section Manager.
Team Information	02	Module to view and edit the Team information.	Variable
	02.01	View: The Bot will answer with all the information about the Team.	ALL
	02.02	Edit: The user will be able to edit the stored information and members.	Administrator, Team Captain and Section Manager.
	02.03	Add: The user will be able to add new Teams.	Administrator and Section Manager.
	02.04	View Other: The Bot will answer with all the information about the asked Team.	Administrator and Section Manager.
	02.05	Edit Other: The user will be able to edit the stored information.	Administrator and Section Manager.
	02.06	Delete: The user will be able to delete Teams.	Administrator and Section Manager.
Convocations	03	Module to send training convocations and see the response.	Variable
	03.01	Send: The user will be able to send a convocation.	Administrator, Team Captain and Section Manager.
	03.02	Respond: The user will be able to respond to the convocations by clicking in one of the proposed options.	ALL
	03.03	View Responses: The user will be able view all responses to convocation.	Administrator, Team Captain and Section Manager.
	03.04	Manage: The user will be able to edit and see all active convocations.	Administrator, Team Captain and Section Manager.
	03.05	Confirm: The user will be able to confirm the convocation, send the team alignment and training to the assistants.	Administrator, Team Captain and Section Manager.
	03.06	Abort: The user will be able to abort the ongoing convocation and mark the specific reason.	Administrator, Team Captain and Section Manager.
	03.07	Alignments: Once the convocation is confirmed an alignment can be selected and send to the participants.	Administrator, Team Captain and Section Manager.
	03.08	Training Type: It will be possible to assign a specific training type to a convocation.	Administrator, Team Captain and Section Manager.

A.2 Web-Site Modules

Module	ID	Description	Users
User Information	05	Module to view and edit the user information.	Administrator, Team Captain and Section Manager.
	05.01	View: The user will see all the information stored .	Administrator, Team Captain and Section Manager.
	05.02	Edit: The user will be able to edit the stored information.	Administrator, Team Captain and Section Manager.
	05.03	Delete: The user will be able to delete members.	Administrator, Team Captain and Section Manager.
Team Information	06	Module to view and edit the Team information.	Variable.
	06.01	View: The user will be able to see all the information about the Team.	Administrator, Team Captain and Section Manager.
	06.02	Edit: The user will be able to edit the stored information and members.	Administrator, Team Captain and Section Manager.
	06.03	Add: The user will be able to add new Teams.	Administrator and Section Manager.
	06.04	Delete: The user will be able to delete Teams.	Administrator and Section Manager.
Convocations	07	Module to send training convocations and see the response.	Administrator, Team Captain and Section Manager.
	07.01	View: The user will be able view all convocations and the information related to them.	Administrator, Team Captain and Section Manager.
Statistics Tool	08	Module to analyze the stored data.	Administrator, Team Captain and Section Manager.
	08.01	General: The user will be able to view statistical data about all teams.	Administrator, Team Captain and Section Manager.
	08.02	Particular: The user will be able to view statistical data about a specific team.	Administrator, Team Captain and Section Manager.