IOSR Journal of Business and Management (IOSR-JBM)
e-ISSN:2278-487X, p-ISSN: 2319-7668. Volume 27, Issue 6. Ser. 7 (June 2025), PP 07-23
www.iosrjournals.org

Chatbot For Optimization In The Registration Of Complaints In Public Security

Ashley Rhayssa Pinheiro Bezerra[1], Keven Matheus Coimbra Mendes[2], Renata de Oliveira Marinho[3], Jean Mark Lobo de Oliveira, David Barbosa de Alencar[4]
[1,2,3] *FAMETRO University Center, Metropolitan Institute of Education (IME), AV. CONSTANTINO NERY, 3204, CHAPADA, CEP: 69050-001, MANAUS/AM. Brasil.*
[4,5] *Professor, of the Postgraduate Program in Engineering, Process Management, Systems and Environmental (PGP.EPMSE) - Institute of Technology and Education Galileo of the Amazon –ITEGAM - Amazonas – Brazil.*

Abstract:
With technological advancement boosting several areas, it is crucial that public security shares these innovations. Among these advances, chatbots stand out as efficient tools in optimizing processes. This article presents GuardiAM, a prototype that uses chatbot technology, developed on Telegram to highlight how this tool can optimize the process of registering complaints by automating the service with the collection of information. For the development of this prototype, the Python programming language was used with API integration and, when tested, it demonstrated agile responses, stability and real-time information storage. Its results were able to confirm that chatbots have the competence to enhance the efficiency of public security authorities, in addition to demonstrating possible improvements in operation.
KeyWords*: Chatbot; Public Security; Denunciation; Prototype.*

Date of Submission: 04-06-2025 Date of Acceptance: 15-06-2025

I. Introduction

The accelerated development and spread of various technologies on the Internet has boosted and transformed several sectors, such as the areas of public security, health, finance and others, generating innovative and efficient solutions for tasks that previously required more time, constantly revolutionizing the way processes are carried out.

With regard to the scope of public security, constant technological updating is essential to ensure effective government management and, for the population, to ensure greater reliability in the services provided by the authorities. Among the various forms of communication on the internet, chatbots stand out, which enable an optimized conversation flow, promoting an effective connection of citizens with the services provided by security authorities. Among the services that a chatbot can improve, the complaint registration system stands out, which, with its use, can make the way citizens make their complaints and the way public security agents deal with the management of these occurrences more convenient, modern and effective.

It should be noted that after the Covid-19 pandemic, the advancement of technology in Brazil has led to a significant increase in the number of digital interactions in the public sector. Many people are increasingly used to solving everyday issues through apps and virtual assistants. With the use of chatbots, convenient and automatic interaction not only offers a means to optimize the entire reporting process, but also helps to reduce costs for the government and the population (JUSBRASIL, 2024).

By making use of a chatbot in the registration of complaints to complement public security services, a modern and promising solution is created in favor of optimizing the process, increasing operational effectiveness, because by automating a service with information collection, the chatbot aims to drastically reduce the waiting time and occupancy of attendants, allowing them to focus on other tasks. In this sense, the chatbot can also provide full availability (24 hours a day and 7 days a week), ensuring that citizens can carry out their occurrences at any time, in a practical and accessible way. The automated process of collecting information also cooperates for assertiveness and quality, favoring analysis and redirection of complaints.

From this perspective, the constant need to modernize technologies for public security is understood, leading this work to develop a prototype of a chatbot, called GuardiAM, in an instant messaging application, using Telegram, to attest that this technology can help public security authorities.

II. Bibliographic Reference

This section covers the key concepts that relate to the use of chatbots in public safety. The concepts of Chatbots, chatbot classifications, Public Safety, the tools and, finally, the libraries and APIs used in the construction of the prototype will be explained.

CHATBOTS

In a simpler understanding, chatbots are computer programs responsible for intermediating communication between machine and human, simulating a natural conversation through the processing of the conversation, whether it is a written or voice conversation. That is, with a chatbot, people can communicate with digital devices as if they were communicating with another person (ORACLE, 2025).

CHATBOT CLASSIFICATIONS

Rule-Based

In this first classification, the behavior of chatbots works as a state machine, that is, the rules determine that user inputs define the transition from one state to another, which means that rule-based chatbots follow a pre-defined structure of questions and answers made to direct the entire flow of interaction with the user. Therefore, chatbots guide the user to the solution of a specific objective, such as clarifying frequently asked questions about an establishment or checking the status of orders from an e-commerce. However, they have limitations when they need autonomy to understand user requests that have not been previously defined, being able to respond generically or repeat automatic messages when they receive an unexpected input (CORREA, VIANA and TELES, 2020).

AI-based

According to IBM (2025), AI-based chatbots are advanced tools that allow for more natural interactions with users, understanding users' questions regardless of how they have expressed them. Through Artificial Intelligence (AI) and Natural Language Understanding (NLU) techniques, these chatbots are able to identify the context of conversations and give relevant feedback in a more dynamic way. If it is not possible to understand the user's request or find multiple response possibilities, the chatbot can request clarification or offer alternatives so that the user can choose the desired action.

In a complementary way, AI chatbots are also able to learn from interactions over time, always improving their responses and becoming more efficient in providing accurate information. Using machine learning and deep learning algorithms, the more the chatbot interacts, the more it learns, always adapting to the user's needs. They can also remember past interactions, which makes the user experience personalized, such as in the case of a customer who returns to place an order and is automatically recognized. And if necessary, the transfer to a human agent is made so that he can take control without interruptions, with him having access to the history of previous interactions (IBM, 2025).

PUBLIC SAFETY

According to Vedosa (2018), public security can be understood as a set of measures and devices aimed at protecting citizens, ensuring their freedom from dangers, damages, and risks to life and property. In a complementary way, it involves political and legal processes that ensure the maintenance of public order and promote peaceful coexistence among individuals in society. It is not limited only to repressive and surveillance actions, but also encompasses an integrated and optimized system, which includes instruments of coercion, justice, defense of rights, health and social assistance. This process begins with prevention and culminates with the repair of damages, treating the causes and stimulating the reintegration of the offender into society.
Based on this context, the Federal Constitution of 1988 dictates the importance of public security as a duty of the State towards the citizen, as well as recognizing the responsibility of society in its implementation.

Public security, a duty of the State, a right and responsibility of all, is exercised for the preservation of public order and the safety of people and property, through the following bodies:

I. Federal police;

II. IFederal highway police;

III. Federal railway police;

IV. Civil police;

V. Military police and military fire brigades.

VI. Federal, state and district criminal police (BRASIL, 1988).

Evidently, public security in Brazil is structured in several bodies with each one having its attributions, whose performance is essential for the maintenance of order and the protection of citizens' rights. With this in mind, it is understood that the insertion of technological solutions, such as chatbots, is offered as a strategic and valuable resource to invigorate this structure, providing greater efficiency in the services provided, and also to expand the population's access to reporting and service mechanisms.

HOTLINE

The 181 hotline is a service intended to receive anonymous complaints that the complainant is aware of, providing efficient assistance to police work in its execution. As it is an anonymous complaint, the whistleblower does not need to identify himself and its occurrence remains under absolute confidentiality, as well as the identity of the accused. The hotline receives all kinds of criminal offenses, such as drug trafficking, robberies and thefts, corruption, violence, fugitives, etc. After the complaint is registered, it is forwarded to the internal affairs departments of the police and fire brigades. It should be noted that the main objective of this service is to register anonymous complaints for investigation, and not critical emergency situations, because in these cases the numbers to be activated are from the services of the Military Police, Fire Department and Civil Police, which are 190, 193 and 197, respectively (BRASIL, 2025).

TOOLS

Telegram

Telegram is an instant messaging application created in 2013 in Russia and, unlike Whatsapp, Telegram is characterized as an application that enables anonymity in communication. This characteristic comes from the hiding of the phone number linked to the user, in addition to the accentuated encryption and the possibility of deleting messages without leaving any trace (SILVA, SOUZA, 2023).
From this, because it promotes levels of anonymity, which is crucial for reporting and, according to Venâncio et al. (2024), because it is the 4th most used instant messaging application in Brazil, it is understood that Telegram is the ideal tool for implementing a chatbot prototype.

Python

Python is an open-source, object-oriented programming language that uses simple syntax, making it easier to understand writing and reading and simplifying its execution, making it an ideal language for prototyping and other tasks. This language is equipped with an extensive standard library that allows for various activities, such as file manipulation and connections to web servers (PYTHON SOFTWARE FOUNDATION, 2025).

Telegram Bot API

Telegram Bot API is an HTTP-based Telegram interface that allows you to create bots in the app for free. These bots are mini applications that run completely within the application itself, allowing users to interact with them through flexible interfaces, and can support all kinds of needs (TELEGRAM, 2025).

Google Sheets API

The Google Sheets API is a RESTful interface of Google Cloud Platform. This tool makes it possible to create spreadsheets, read and write values in spreadsheet cells, update spreadsheet formatting, and connected managements (GOOGLE, 2025).

Python-telegram-bot

The free python-telegram-bot and open-source library is a collection of tools developed for the Python language in order to facilitate the development of chatbots and integration with the Telegram Bot API. This tool provides an asynchronous interface in pure Python and is compatible with versions 3.9 and later of the language. In a pure complement to the API, this library provides several methods and shortcuts aimed at the convenience and simplicity of development, as well as several high-level classes contained in the telegram.ext submodule, making the task of developing chatbots simple and objective. (PYTHON-TELEGRAM-BOT, 2025).

Gspread

Gspread is a library of features for accessing and manipulating Google Sheets APIs, but unlike the API itself, this library abstracts away the complexities of using the Google Sheets API, which allows for more simplified spreadsheet manipulation. Like the API itself, it is characterized by.

- Opening spreadsheets by title, key, or URL;
- Reading, writing and formatting spreadsheet cells;
- Spreadsheet sharing and access control.

Google-auth

In order for a Python application to use Google's APIs, it needs to perform authentication. To do this, the google-ath library, which replaced the deprecated oauth2client library, allows this authorization through methods that abstract away from standard complexity. In addition, this library also provides integration with several HTTP libraries. (GOOGLE-AUTH, 2025).

III. Methodology

This research focused on the development process of the prototype of the GuardiAM chatbot, which used APIs to connect the Telegram *bot* to the *backend* in Python and to record the complaints in a spreadsheet in Google Sheets.

Regarding the approach, the research is qualitative simplified, as it seeks to focus on understanding the practical application of the prototype in the context of public security. About its nature, it is characterized as applied, as it aims to obtain knowledge focused on practical execution. As for the objectives, the research is exploratory and applied, as it focuses on the development of a prototype and on identifying the determining factors for the successful implementation of the *chatbot*. As for the procedures, they are of an experimental development nature. The research was also supported by a review of constant theoretical references.

STAGES OF DEVELOPMENT

The development of the prototype was based on six main steps:

- Elaboration of the GuariAM Operation Flowchart;
- Bot creation in BotFather;
- Creation of the Complaints spreadsheet in Google Sheets
- Configuration of the GuardiAM Project on Google Cloud Platform;
- Activation of Google Drive and Google Sheets APIs;
- Service account creation - activation of service account key;
- Development of backend code in Python;

Flowchart

First, a flowchart was developed with the online tool draw.io to visually represent the paths of the user's conversation with the *bot*. This flowchart details the states of the conversation, making the flow of questions and answers that GuardiAM processes clear and objective, ensuring that the user can reach the last stage of the service.

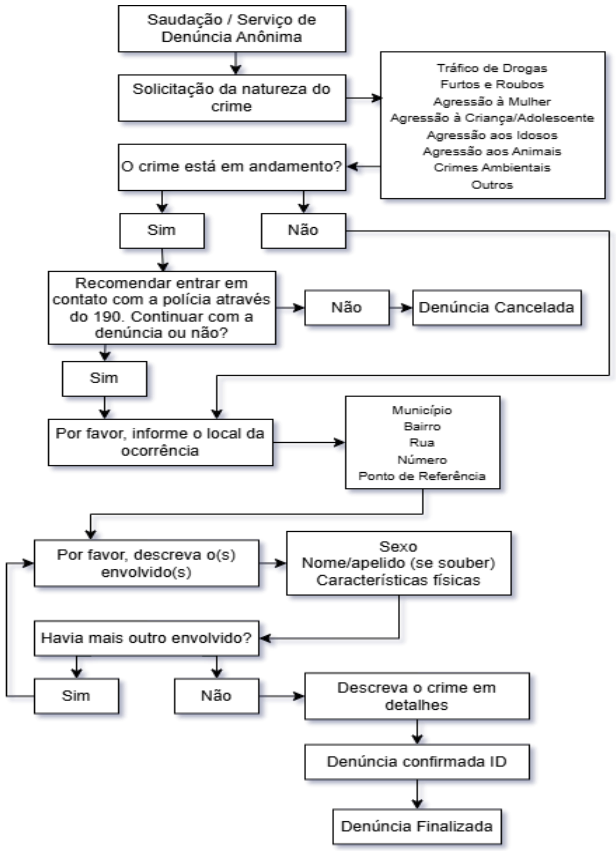


Figure 1 – Chatbot flowchart
Source: Authors, 2025

Bot Creation

The creation of the *bot* takes place through Telegram's official tool, BotFather. This tool makes it possible to create, configure, and manage *bots* in a simple way within Telegram itself.

To start, BotFather was accessed through Telegram Web, and using the /start command, he starts the conversation with a list of commands, as shown in figure 2.

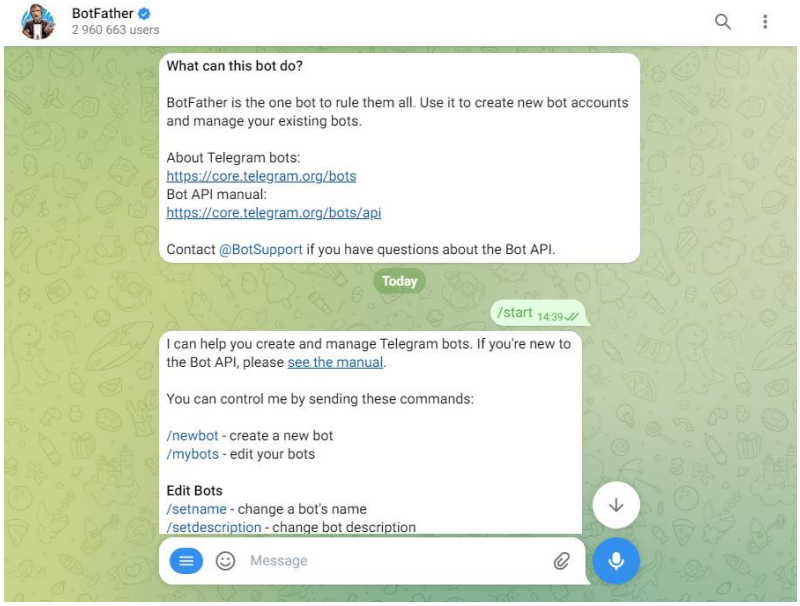


Figure 2 – BotFather
Source: Authors, 2025

After that, the /newbot command was used to create the bot. The BotFather requests the bot name and *bot username* , and the *username* must end with "*bot*". And after these steps, BotFather provided the *API token*, as shown in figure 3, which is the unique code needed to integrate the *bot* into the *prototype backend*, so the *token* in the figure is censored. Later, the token was stored securely to be used in the bot's connection to the *backend server*.

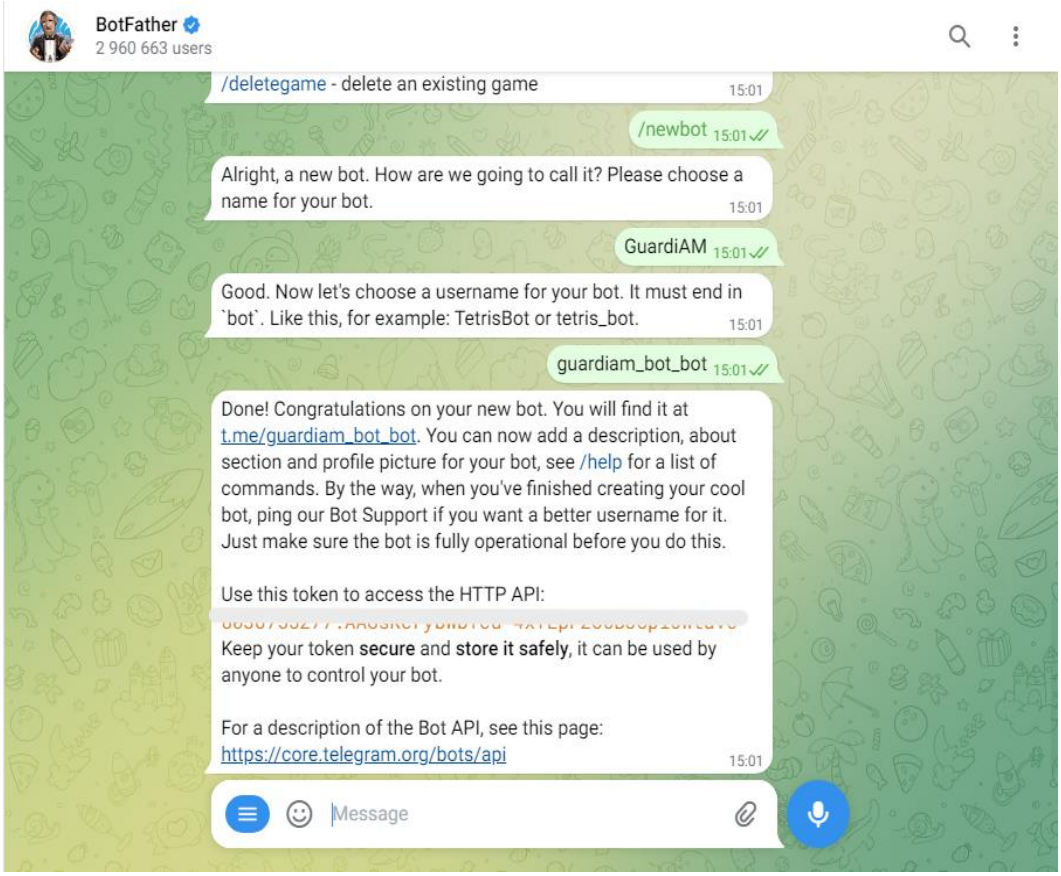


Figure 3 – Bot Creation
Source: Authors, 2025

Spreadsheet Creation

To store the complaints registered by the *chatbot*, a spreadsheet was created in Google Sheets to act as a database, called Complaints. This spreadsheet was structured with columns that refer to the questions that the *bot* asks to collect the information of the complaint, as shown in figure 4.

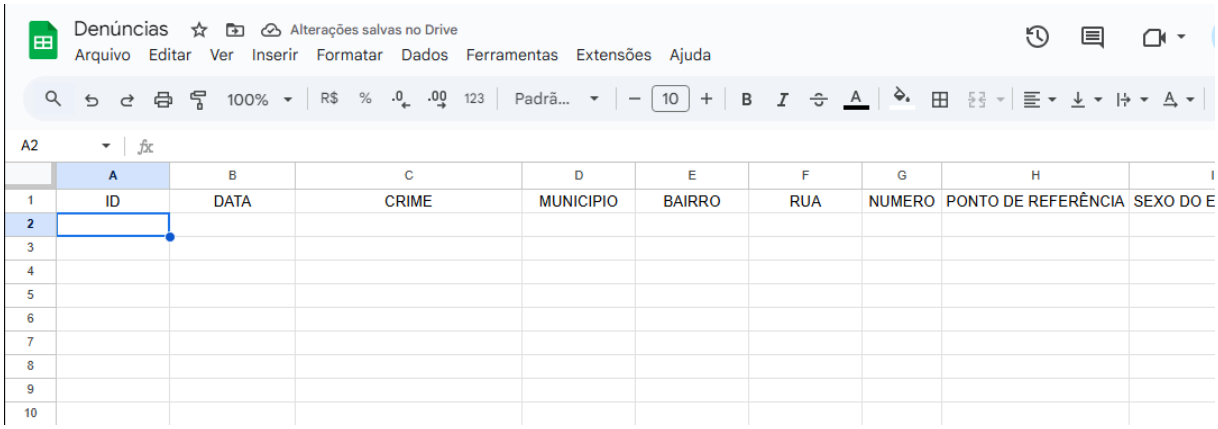


Figure 4 – Creation of the Complaints spreadsheet
Source: Authors, 2025.

Cloud Platform

With the *Telegram* bot access *token* generated, the next step was to create a project in the Google Cloud Platform console. This Google tool provides a variety of services for integrating APIs. The project called GuardiAM was then created, as shown in figures 5.

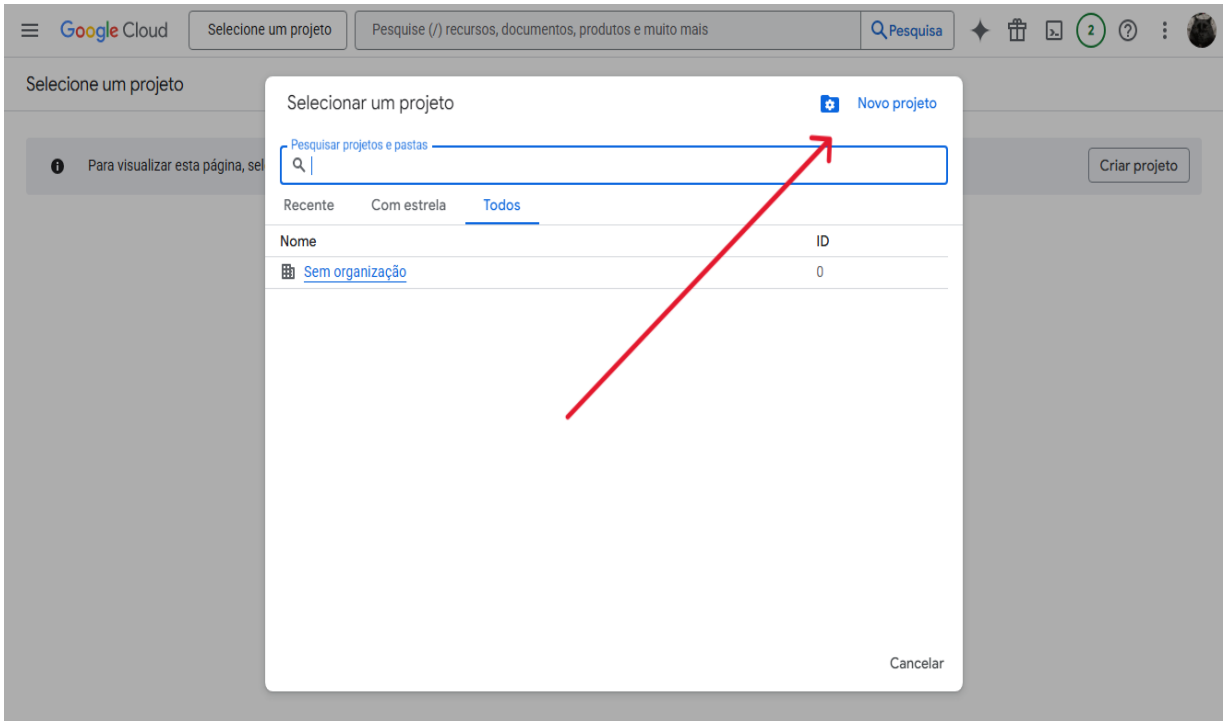


Figure 5 – Creating the Project
Source: Authors, 2025.

And for naming the project, the same one from GuardiAM was used, as shown in figure 6.

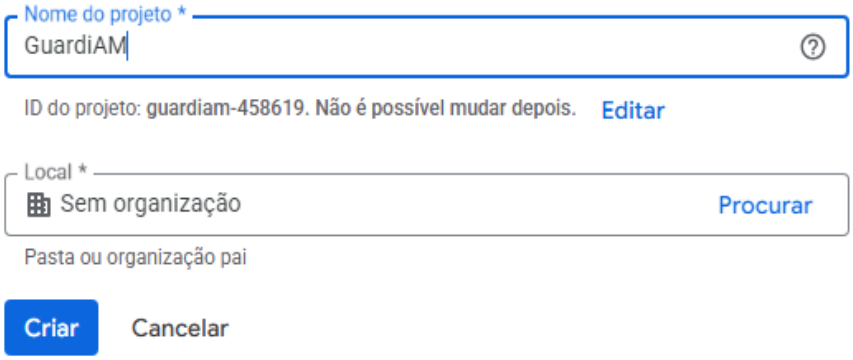


Figure 6 – Naming the Project
Source: Authors, 2025.

Enabling APIs

In order to make communication between the *backend* and the Complaints spreadsheet in Google Sheets possible, it was necessary to activate the necessary APIs, which are the Google Drive API, to grant access to the spreadsheet and the Google Sheets API, necessary for manipulation such as reading and writing data in the spreadsheet cells. To do this, within the Google Cloud Platform console and with the GuardiAM project selected, it was necessary to navigate APIs and services enabled in APIs and services. In this interface, the option to enable the APIs is shown, as shown in figure 7.

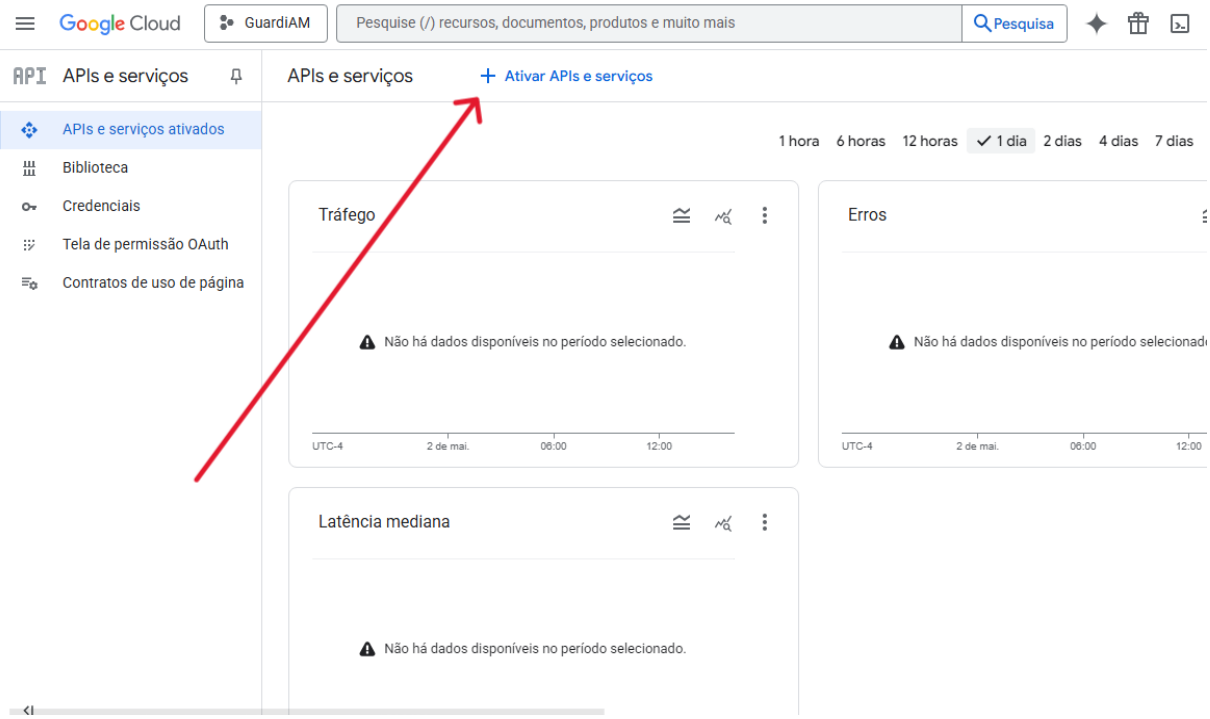


Figure 7 – API enablement interface
Source: Authors, 2025.

By clicking on the option indicated in Figure 7, a search screen for the necessary APIs opens, where the Google Drive and Google Sheets APIs were searched, as shown in Figure 8.

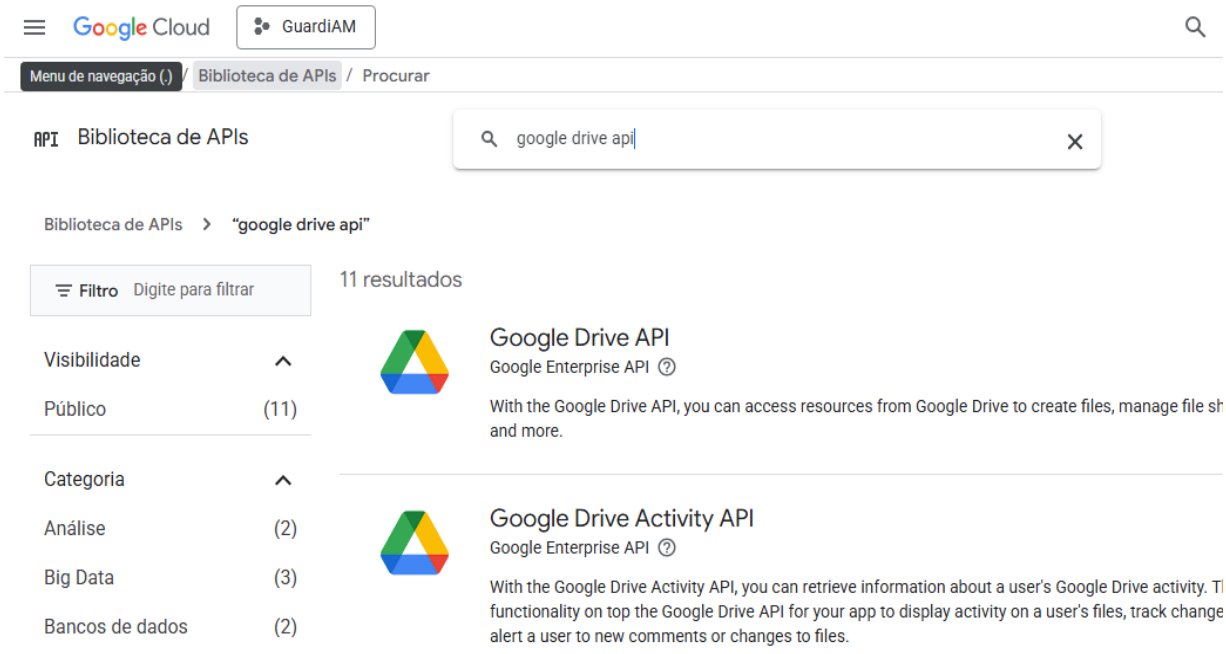


Figure 8 – Searching Google Drive API
Source: Authors, 2025.

With the Google Drive API selected, you were given the option to enable it, which was also selected. After that, the Google Drive API was enabled in the GuardiAM project, and the same process was repeated in the Google Sheets API. With this, the two required APIs were already enabled.

Service Account and Key Activation

In order for the *chatbot to* be able to connect securely to the Complaints spreadsheet, it is necessary to create a service account, which represents the access credentials. The creation takes place in the Google Cloud Platform console, navigating between APIs and services and in Credentials, you will find the credential creation interface. When you click Create Credentials, credential type options are shown, and the service account is selected, as shown in Figure 9.

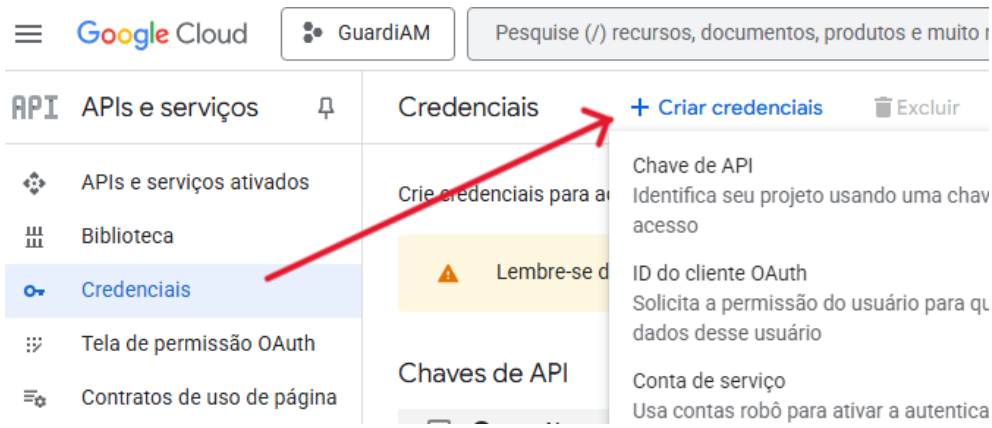


Figure 9 – Credential Creation Interface
Source: Authors, 2025

After selection, the service account creation screen is displayed, where fields such as name and description are requested, as shown in figure 10.

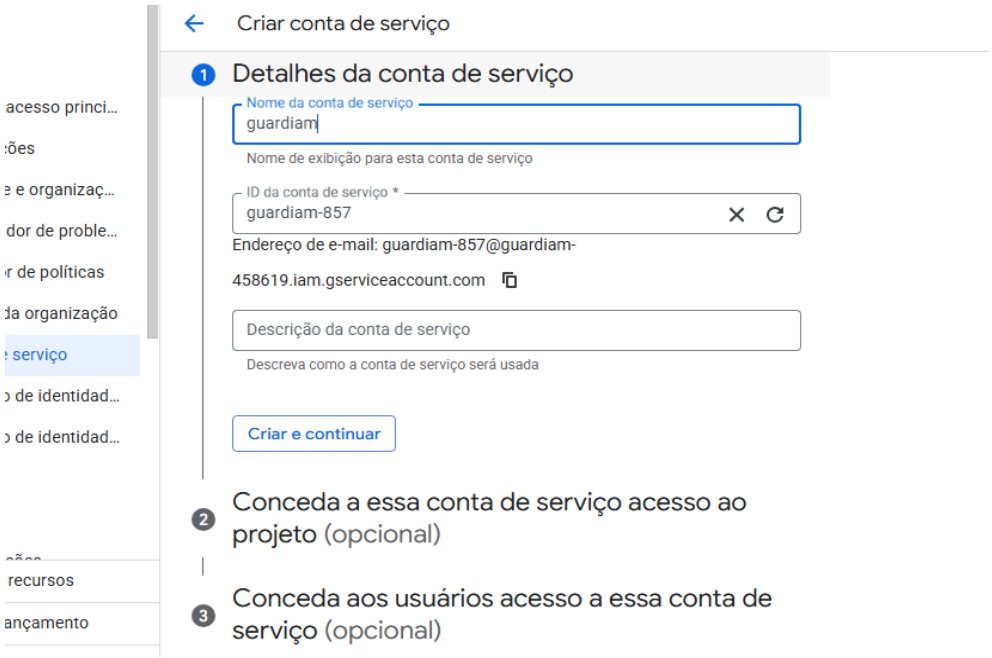


Figure 10 – Creating Service Account
Source: Authors, 2025

Naming and completing the creation of the service account, it was necessary to generate a service account key. To do this, the email of the service account was selected, navigating through Keys and selecting the options Add key and Create new key, as shown in figure 11.

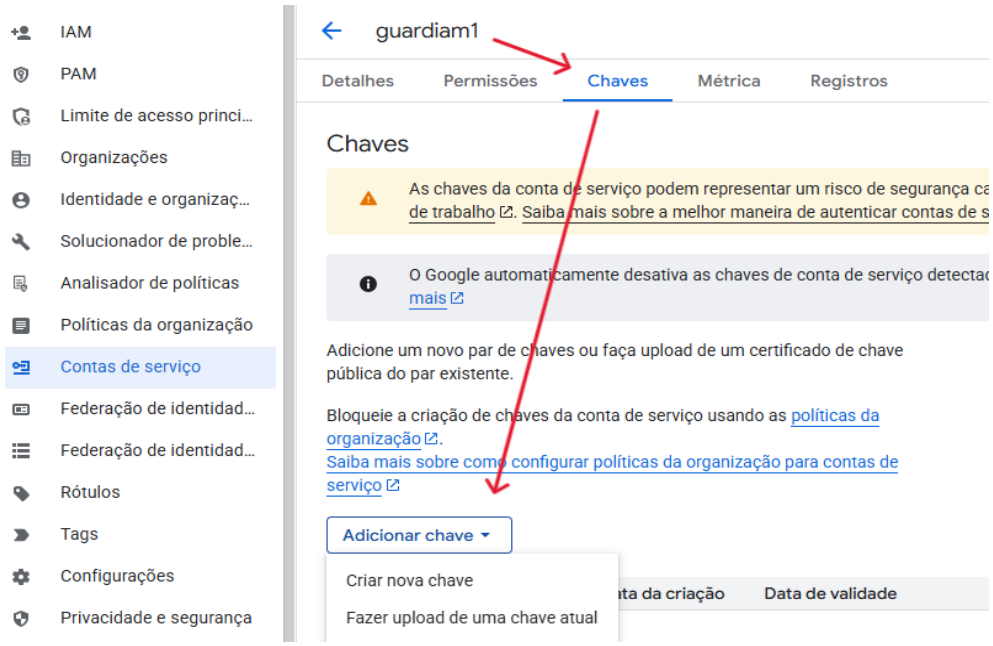


Figure 11 - Creating Service Account Key
Source: Authors, 2025.

The type of key chosen was in JSON format, as shown in figure 12 and, from the creation, the JSON file containing the credentials of the GuardiAM project service account is downloaded.

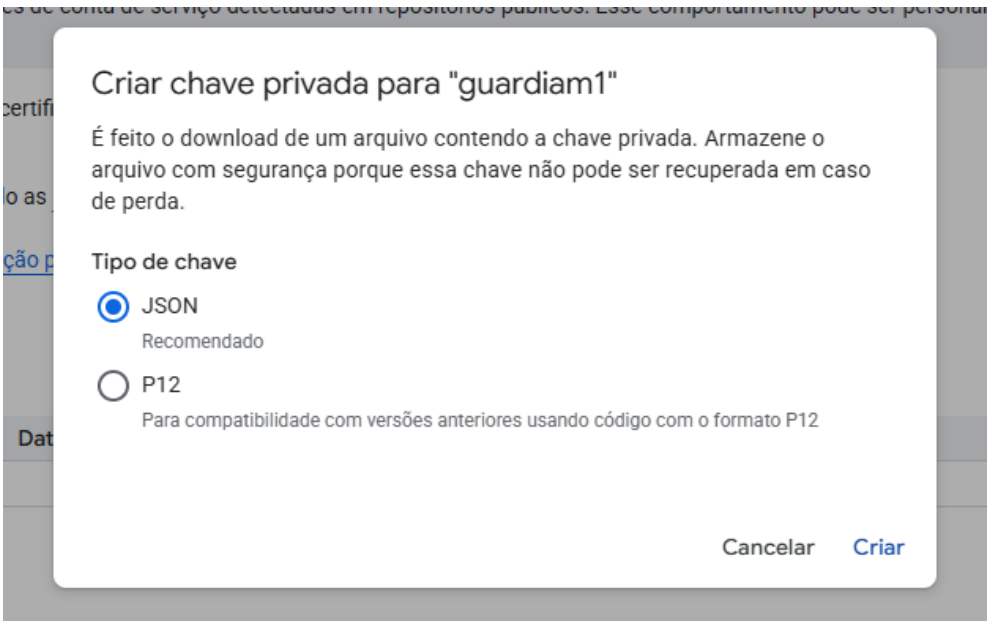


Figure 12 – Key in JSON format
Source: Authors, 2025.

Backend Python

Completing all the necessary API activations and collecting all the credentials for the development of the *chatbot* on Telegram, a *Python script* was written for the *prototype backend*, abstracting the essentials of the code's operation.

Imports

First, the imports of Python libraries and modules necessary for the initial configurations of the prototype were carried out, as seen in figure 13.

```python
import logging
import datetime
import gspread
from google.oauth2.service_account import Credentials
from telegram import Update
from telegram.ext import (
    ApplicationBuilder, CommandHandler, MessageHandler, ConversationHandler, ContextTypes, filters
)
```

Figure 13 – Imports of libraries and modules
Source: Authors, 2025.

Google Sheets Setup

Subsequently, so that the code could authenticate, access and manipulate data in the Complaints spreadsheet, the Google Sheets settings logic was written. According to the structure in the code, the scope of access to the spreadsheet and Google Drive was defined to be passed along with the JSON file containing the credentials of the GuardiAM project for authorization and opening of the Complaints spreadsheet, as can be seen in Figure 14.

```python
scope = ["https://spreadsheets.google.com/feeds", "https://www.googleapis.com/auth/drive"]
creds = Credentials.from_service_account_file(
    r"C:/Users/keven/Desktop/chatbot/guardiam-457500-94cb50988cff.json", scopes=scope
    )
client = gspread.authorize(creds)
sheet = client.open("Denúncias").sheet1
```

Figure 14 – Google Sheets Settings
Source: Authors, 2025.

Conversation States

States were defined for the conversation, with each one representing a part of the flow of the conversation with the user, from the user's response to the type of crime to the description of the fact, as shown in figure 15.

```python
(
    CRIME, EM_ANDAMENTO, CONTINUAR, MUNICIPIO, BAIRRO, RUA, NUMERO, PONTO_REFERENCIA,
    SEXO, NOME_APELIDO, COR_PELE, COR_CABELO, MARCAS, OUTRO_ENVOLVIDO,
    DESCRICAO, FINAL
) = range(16)
```

Figure 15 – Conversation States
Source: Authors, 2025

State functions

Next, asynchronous state functions were implemented, where each one of them refers to the state where the user is in the flow of the conversation. These functions are sequential and conditional, where each one validates the user's input so that the report data is collected in a structured way and, at the end of the execution of one, the return points to the next state, until the flow is finished, as can be seen in Figure 16.

```
27  async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
45
46  async def crime(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
67
68  async def em_andamento(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
76
77  async def continuar(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
84
85  async def municipio(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
89
90  async def bairro(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
94
95  async def rua(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
99
100 async def numero(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
104
105 async def ponto_referencia(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
109
110 async def sexo(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
114
115 async def nome_apelido(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
119
120 async def cor_pele(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
124
125 async def cor_cabelo(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
129
130 async def marcas(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
141
142 async def outro_envolvido(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
150
151 async def descricao(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
156
157 def salvar_denuncia(context: ContextTypes.DEFAULT_TYPE): ···
182
183 async def cancelar(update: Update, context: ContextTypes.DEFAULT_TYPE) -> int: ···
```

Figure 16 – State functions
Source: Authors, 2025.

Core Configuration

Finally, the main function was implemented, representing the main flow of the system, being responsible for starting the *bot* on Telegram by providing the *connection token* and managing the conversation flow. This function dictates the interaction states and associates each step of the conversation with its processing functions, in addition to making the *bot* always active and receptive to any new message after the end of a service. The function is illustrated in figure 17.

```
191 def main():
192     app = ApplicationBuilder().token("8036755277:AAGsKcrybWbYcu-4xYLpF2C6B3Cpi6Wtavc").build()
193
194     conv_handler = ConversationHandler(
195         entry_points=[
196             CommandHandler("start", start),
197             MessageHandler(filters.TEXT & ~filters.COMMAND, start)  # Inicia com qualquer texto
198         ],
199         states={
200             CRIME: [MessageHandler(filters.TEXT & ~filters.COMMAND, crime)],
201             EM_ANDAMENTO: [MessageHandler(filters.TEXT & ~filters.COMMAND, em_andamento)],
202             CONTINUAR: [MessageHandler(filters.TEXT & ~filters.COMMAND, continuar)],
203             MUNICIPIO: [MessageHandler(filters.TEXT & ~filters.COMMAND, municipio)],
204             BAIRRO: [MessageHandler(filters.TEXT & ~filters.COMMAND, bairro)],
205             RUA: [MessageHandler(filters.TEXT & ~filters.COMMAND, rua)],
206             NUMERO: [MessageHandler(filters.TEXT & ~filters.COMMAND, numero)],
207             PONTO_REFERENCIA: [MessageHandler(filters.TEXT & ~filters.COMMAND, ponto_referencia)],
208             SEXO: [MessageHandler(filters.TEXT & ~filters.COMMAND, sexo)],
209             NOME_APELIDO: [MessageHandler(filters.TEXT & ~filters.COMMAND, nome_apelido)],
210             COR_PELE: [MessageHandler(filters.TEXT & ~filters.COMMAND, cor_pele)],
211             COR_CABELO: [MessageHandler(filters.TEXT & ~filters.COMMAND, cor_cabelo)],
212             MARCAS: [MessageHandler(filters.TEXT & ~filters.COMMAND, marcas)],
213             OUTRO_ENVOLVIDO: [MessageHandler(filters.TEXT & ~filters.COMMAND, outro_envolvido)],
214             DESCRICAO: [MessageHandler(filters.TEXT & ~filters.COMMAND, descricao)],
215         },
216         fallbacks=[CommandHandler("cancelar", cancelar)],
217     )
218
219     app.add_handler(conv_handler)
220
221     app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, start))
222
223     app.run_polling()
224
225 if __name__ == "__main__":
226     main()
227
```

Figure 17 – Main Configuration
Source: Authors, 2025.

IV. Results

The prototype of GuardiAM developed served its main purpose by being able to record users' criminal complaints automatically in a Google Sheets spreadsheet through an instant messaging application. Tests were carried out in a simulated environment and with 10 real users, which showed the chatbot's operation with stability, without failures during user interactions or storage of the complaint in the spreadsheet.The backend integration with the Telegram APIs and Google Drive and Google Sheets services proved to be stable, with an average response time of less than 2 seconds for each bot return. In addition, the information collected could be recorded and viewed in real time in the spreadsheet, confirming the instant communication between the platforms. In the following subsection, a complete simulation of service using GuardiAM on Telegram is presented, which involves a user starting the conversation with a greeting and going through the entire bot flow until the end of the service.

SIMULATION

This simulation is illustrated with a sequence of figures that demonstrate the operation of the *chatbot*, starting with the user's input message and the greeting responses and list of crimes from GuardiAM, as shown in figure 18.

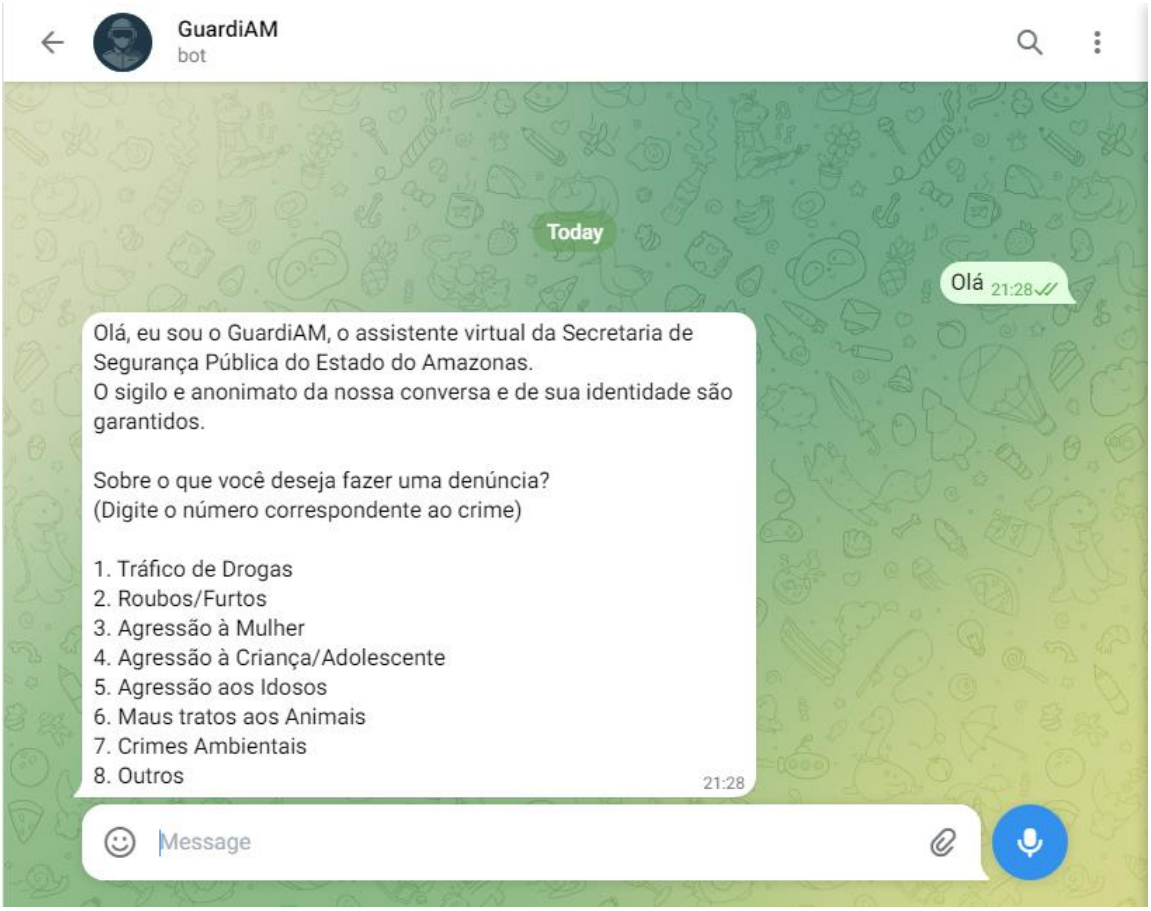


Figure 18 – Beginning of care
Source: Authors, 2025.

Continuing the flow, the user sent the digit 1, corresponding to the crime of Drug Trafficking. After that, GuardiAM questioned whether the crime was in progress. In the simulation, the fact was confirmed as in progress, which led the *bot* to recommend contacting 190 and offering to continue the complaint or close. The continuation was confirmed, as shown in figure 19.

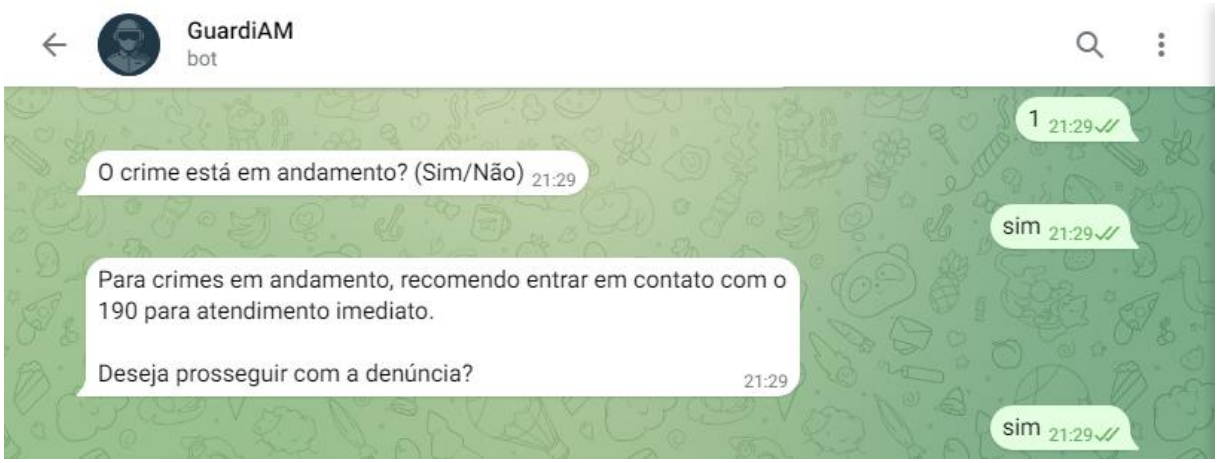


Figure 19 – Continuation of the crime
Source: Authors, 2025.

Subsequently, GuardiAM continued the flow by collecting information about the location of where the crime occurred, as shown in figure 20.

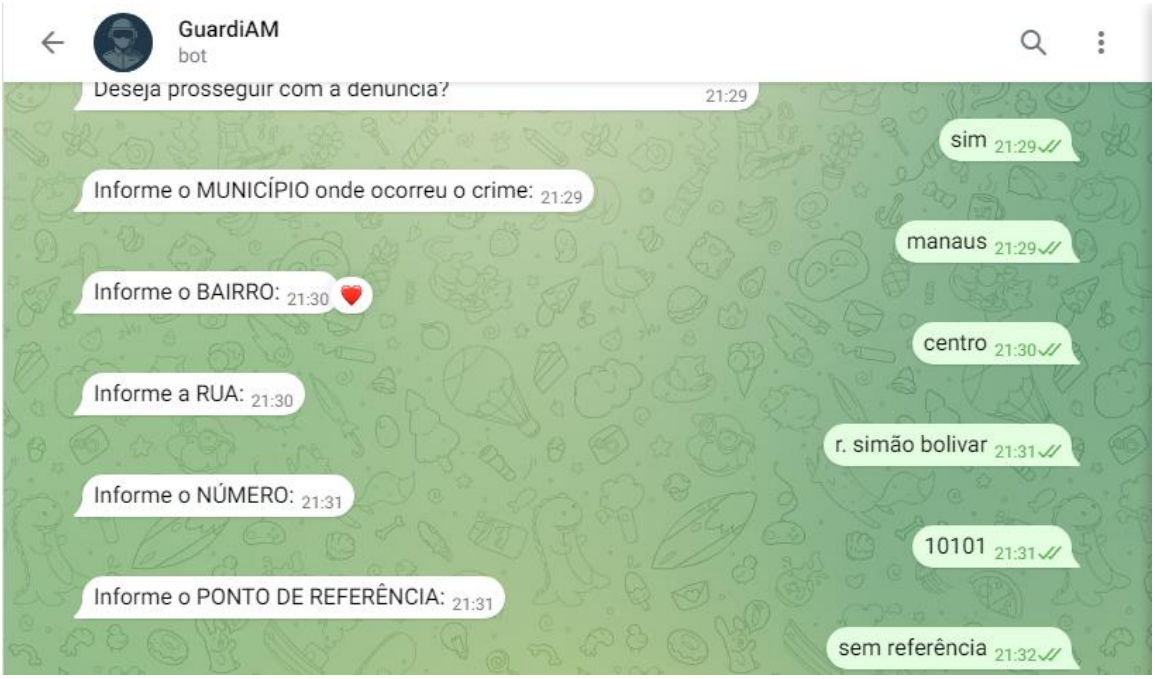


Figure 20 – Location Information Collection
Source: Authors, 2025.

Continuing to collect information, GuardiAM asked questions about the accused and whether there was another involved in the crime. If so, the same questions to collect information from the accused would be asked. In the simulation, it was decided to continue with only one, as shown in figure 21.

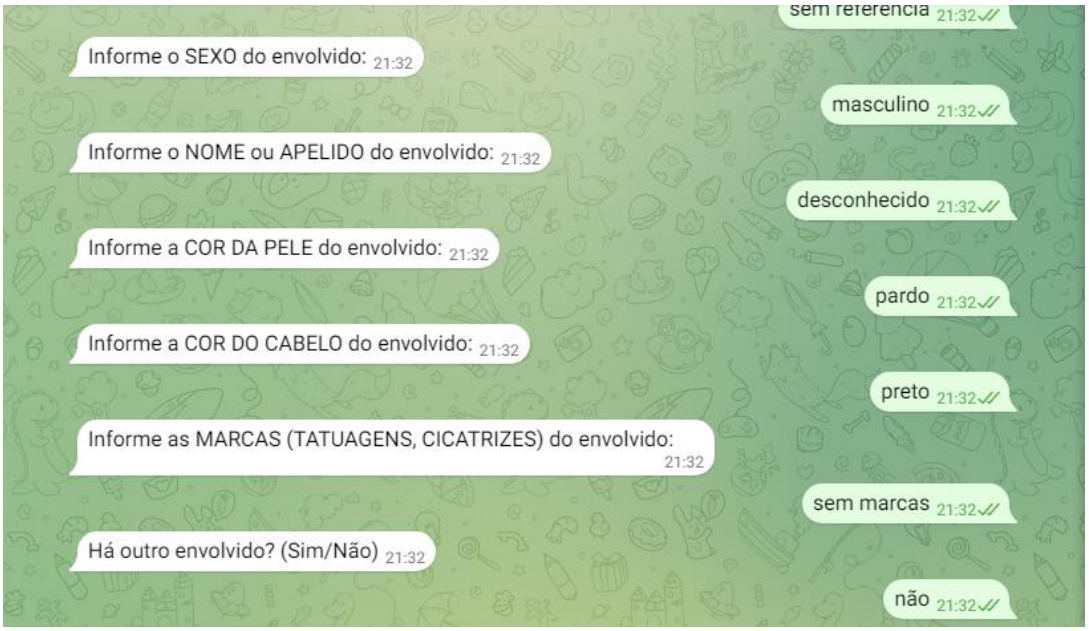


Figure 21 – Collection of information from the accused
Source: Authors, 2025

Concluding the collection of location information and the characteristics of the person involved, GuardiAM finally requested a description of what happened. After the user sent the description about the complaint, the *bot* ended the service, as shown in figure 22.

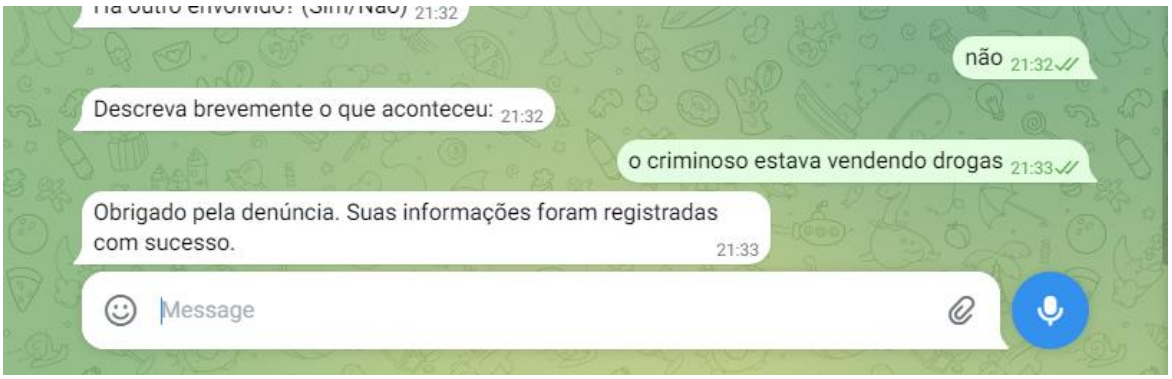
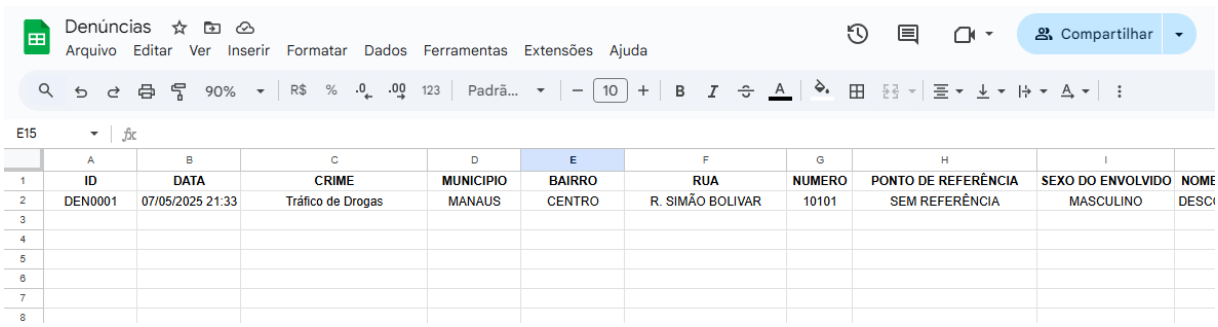


Figure 22 – Completion of the service
Source: Authors, 2025.

To conclude, all information was sent to the Complaints spreadsheet in Google Sheets immediately, as evidenced in figures 23.



	A	B	C	D	E	F	G	H	I	
1	ID	DATA	CRIME	MUNICIPIO	BAIRRO	RUA	NUMERO	PONTO DE REFERÊNCIA	SEXO DO ENVOLVIDO	NOME
2	DEN0001	07/05/2025 21:33	Tráfico de Drogas	MANAUS	CENTRO	R. SIMÃO BOLIVAR	10101	SEM REFERÊNCIA	MASCULINO	DESC
3										
4										
5										
6										
7										
8										

Figure 23 – Registered complaint
Source: Authors, 2025.

These results indicate that the GuardiAM prototype is ready for future phases of enhancement or actual implementation in the context of public safety.

OBSERVATIONS

From the simulation and with tests carried out with 10 users, everyone observed that the prototype efficiently met these 3 essential positive points for the user experience:

- accessibility, as the interactions were simple and direct;
- automation, as all the information collected was entirely made from the *bot,* without the need for human intervention;
- privacy, as it did not require any collection of the user's personal data.

In addition, they added that for future improvements, GuardiAM could offer features so that the user can send images, videos or audios to enrich information about the complaint, enhance the processing of the flow through the implementation of Natural Language Processing (NLP), replace the storage of complaints in a spreadsheet for a more secure and robust database, and allow the user to receive a protocol to consult the progress of the complaint.

V. Conclusion

The development of the GuardiAM prototype demonstrated the feasibility and efficiency of a chatbot for automated registration of criminal reports through an instant messaging interface, integrating Telegram APIs and Google Sheets services. The results obtained show that the system operates in a stable way, with adequate response time and structured collection of information, enabling the immediate registration of complaints in a safe and accessible environment.

The simulation of the service and the tests carried out with real users confirmed that GuardiAM has fundamental characteristics for user acceptance, such as accessibility, simplicity in interaction, complete automation of the process and respect for privacy, as it does not require personal data from whistleblowers. These aspects are essential to encourage the use of the tool in real contexts, contributing to the expansion of access to reporting and, consequently, to public safety.

However, this work also identified opportunities for future improvements, such as the implementation of multimedia resources for sending images, videos and audios, the adoption of advanced Natural Language Processing techniques for more natural and efficient communication, and the migration of the storage of complaints to more robust and secure databases. In addition, the functionality of issuing protocols for monitoring complaints can increase transparency and user confidence in the system. Therefore, GuardiAM is a promising basis for the development of technological solutions aimed at public safety, and can be expanded and adapted according to demands and technological advances. The present study contributes to the discussion on the use of chatbots and the integration of digital services in support of citizenship, highlighting the importance of automation and accessibility in the modernization of communication channels between the population and the authorities.

Acknowledgements

The authors would like to express their sincere gratitude to the FAMETRO University Center and the Metropolitan Institute of Education (IME) for their institutional support, which was essential to the development and completion of this research.

References

[1]. BRAZIL. Constitution (1988). Constitution of the Federative Republic of Brazil of 1988. Article 144. Available at: https://www.jusbrasil.com.br/topicos/10773132/artigo-144-da-constituicao-federal-de-1988. Accessed on: 8 Apr. 2025.
[2]. BRAZIL. Federal government. Dial-Denunciation 181. Available at: https://www.gov.br/pt-br/servicos-estaduais/servico_estadual?id=18834. Accessed on: 28 Apr. 2025.
[3]. CORREA, Joeckson; VIANA, Davi; TELES, Ariel. Developing ChatBots with Dialogflow. 2021. Available at: https://www.researchgate.net/profile/Davi-Viana-2/publication/357471808_Desenvolvendo_ChatBots_com_o_Dialogflow/links/71fa8b9a007fb504472fcd77/Desenvolvendo-ChatBots-com-o-Dialogflow.pdf. Accessed on: 31 mar. 2025.
[4]. FERREIRA, Douglas da Silva; SANTOS, Regina Ávila. Exploring the structure of Brazilian far-right networks on Telegram: a classification proposal to understand the division and links between their communities. Public Administration and Social Management, v. 17, n. 1, p. 1–25, mar. 2025. Available at: https://periodicos.ufv.br/apgs/article/view/17358. Accessed on: 28 Apr. 2025.

[5]. GOOGLE. Google Sheets API. Available at: https://developers.google.com/workspace/sheets/api/guides/concepts. Accessed on: 28 Apr. 2025.
[6]. GOOGLE-AUTH. google-auth. Available at: https://google-auth.readthedocs.io. Accessed on: 28 Apr. 2025.
[7]. GSPREAD. Available at: https://docs.gspread.org. Accessed on: 28 Apr. 2025.
[8]. IBM. Chatbot types. Available at: https://www.ibm.com/br-pt/think/topics/chatbot-types. Accessed on: 31 mar. 2025.
[9]. JUSBRASIL. Technology in the public sector: benefits, challenges and trends. Jusbrasil, 2024. Available at: https://www.jusbrasil.com.br/artigos/tecnologia-no-setor-publico-beneficios-desafios-e-tendencias/1275088901. Accessed on: 30 mar. 2025.
[10]. ORACLE. What is a chatbot? Available at: https://www.oracle.com/br/chatbots/what-is-a-chatbot/. Accessed on: 21 mar. 2025.
[11]. PYTHON SOFTWARE FOUNDATION. BeginnersGuide/Overview. Available at: https://wiki.python.org/moin/BeginnersGuide/Overview. Accessed on: 23 Apr. 2025.
[12]. PYTHON-TELEGRAM-BOT. python-telegram-bot. Available at: https://docs.python-telegram-bot.org/en/stable/. Accessed on: 28 Apr. 2025.
[13]. TELEGRAM. Telegram Bot API. Available at: https://core.telegram.org/bots. Accessed on: 28 Apr. 2025.
[14]. VEDOSA, Daiana. What is public security. JusBrasil, 2018. Available at: https://www.jusbrasil.com.br/artigos/o-que-e-seguranca-publica/587735277. Accessed on: 31 mar. 2025.
[15]. VENÂNCIO, Otávio R.; GONÇALVES, Gabriel H. S.; FERREIRA, Carlos H. G.; SILVA, Ana Paula C. da. Evidence of content dissemination on Telegram during the attack on Brazilian public agencies in 2023. In: BRAZILIAN SYMPOSIUM ON MULTIMEDIA AND THE WEB (WEBMEDIA), 30., 2024, Juiz de Fora/MG. Annals [...]. Porto Alegre: Brazilian Computer Society, 2024. p. 385-389. Available at: https://doi.org/10.5753/webmedia.2024.241972. Accessed on: May 12, 2025.