MDSAA

Master's Degree Program in
Data Science and Advanced Analytics

Business Cases with Data Science

Case 4: Chatbot

André, Lourenço, number: 20240743

Emir, Kamiloglu, number: 20240945

Manuel, Andrade, number: 20240571

Rute, Teixeira, number: 20240667

Victor, Silva, number: 20240663

Group T

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

May, 2025

Table of Contents

1. EXECUTIVE SUMMARY 3

2. BUSINESS UNDERSTANDING, NEEDS AND REQUIRED OUTCOME 4

2.1 Business Objectives 4

2.2 Business Success Criteria 4

2.3 Determine Data Mining Goals 5

3. METHODOLOGY 5

3.1 Data Understanding 5

3.2 Data Preparation and Extraction 6

3.3 GPT-4o Mini Model Configuration 8

4. DEPLOYMENT 9

4.1 User Interface Design 9

4.2 Voice Recognition and Language Detection 10

4.3 Chat History and Session Rate 11

4.4 Follow-Up Questions Suggestions 12

5. PERFORMANCE EVALUATION 13

6. FUTURE WORK: 14

6.1 Internal Integration 14

6.2 Maintenance Plan 17

6.3 Cost-benefit analysis 18

6.3.1 Benefits 18

6.3.2. Cost Structure 19

7. CONCLUSION 20

1. Executive Summary

Fidelidade, Portugal's leading insurance provider with over two centuries of history, is addressing a critical strategic challenge: enhancing the commercial efficiency of its insurance agents in an environment where financial literacy among customers remains persistently low. Although the company offers a strong range of savings and investment products such as "My Savings" and "PPR Evoluir," adoption of these products has been limited due to customers' difficulty in understanding financial concepts. As a result, insurance agents are often required to spend excessive time explaining basic principles, which diminishes their ability to focus on high-value, consultative engagements with clients.

To overcome this barrier, the project centered on the development of a document-driven, AI-powered chatbot specifically designed to support insurance agents in real time. Built around OpenAI's GPT-4o mini model, the assistant delivers accurate and context-aware responses grounded in Fidelidade's official documentation and financial literacy materials. By automating the explanation of complex product features and regulatory nuances, the chatbot enables agents to improve client interactions, reduce response time, and elevate the quality of advice provided. It supports both text and voice input and maintains dialogue continuity across multi-turn interactions, ensuring agents receive responses that evolve naturally with the conversation.

The chatbot was deployed using Streamlit, a Python-based framework that enabled the team to rapidly develop and launch an interactive web interface accessible via browser. The final deployment includes a branded and accessible layout named "Assistente Fidélia," complete with dual-logo headers, session history tracking, and a user-friendly experience that accommodates both text typing and speech input. Voice recognition was implemented through the Google Speech Recognition API, while chat history is managed using Streamlit's session state functionality. Additionally, the assistant suggests follow-up questions generated by GPT-4o Mini using prompt-instructed JSON formatting, allowing agents to seamlessly explore related topics and maintain conversational momentum.

Evaluation of the chatbot's performance was conducted using a two-phase framework. The quantitative evaluation used cosine similarity to assess how closely the model's responses matched verified reference answers provided by the company. In the case of the "My Savings" product, similarity scores ranged from 0.83 to 0.94, with most responses exceeding 0.87. For "PPR Evoluir," the scores ranged from 0.84 to 0.94, demonstrating consistently high semantic fidelity. These results indicate that the chatbot reliably captures the intended meaning of official responses, even when rephrased. Complementing this, a qualitative evaluation was performed in which team members manually reviewed the chatbot's outputs to assess tone, completeness, and professional alignment. The results confirmed that the assistant maintained a high standard of clarity and factual accuracy, supporting its intended use in customer-facing advisory roles.

Looking ahead, the chatbot is ready to be integrated into Microsoft Teams, enabling insurance agents to access it directly within their daily communication environment without disrupting their workflow. This next step will further streamline its adoption and reinforce its role as a practical and scalable support tool.

2. Business Understanding, Needs and Required Outcome

Fidelidade operates in a competitive and tightly regulated insurance market, where maintaining leadership depends not only on offering quality financial products but also on delivering clear, consistent communication and personalized client support. In 2024, the company recorded a profit of 173 million euros, a 3.8% decline from the previous year, despite a strong 18.5% increase in gross written premiums, which reached 6.172 billion euros. This divergence highlights growing pressure on operational efficiency and the urgent need to modernize internal processes.

Insurance agents remain central to Fidelidade's distribution strategy, but their role is increasingly challenged by the complexity of financial products and low levels of financial literacy among clients. Agents are often required to act as both educators and salespeople, which slows down interactions and limits their ability to focus on building long-term relationships. To respond to these challenges, Fidelidade must provide its agents with intelligent tools that streamline the communication of complex information and support faster, more effective client engagement. The goal is to improve the adoption of key products, enhance agent productivity, and maintain the company's competitive edge through a more efficient and informed advisory process.

2.1 Business Objectives

To support these strategic priorities, the solution involves deploying an AI-driven assistant integrated within Fidelidade's internal systems. Rather than serving clients directly, the tool is designed to assist agents by providing instant access to consistent, well-structured information on financial products and related concepts. This backend support reduces the cognitive and time burden on agents, particularly when responding to frequently asked questions or clarifying complex product features. The system not only facilitates faster service but also enhances the precision of client guidance. In turn, this operational upgrade supports broader goals around product uptake and customer empowerment, while reinforcing Fidelidade's innovation credentials in the financial services sector.

2.2 Business Success Criteria

The effectiveness of the chatbot will be evaluated through a combination of operational analytics and user experience metrics. Internally, agent feedback and system usage data will help assess the impact on workflow efficiency, particularly in reducing the frequency and duration of routine product related inquiries. Externally, metrics such as engagement rates, interaction completion, and follow-up actions will offer insight into the tool's utility. Special emphasis will be placed on the chatbot's performance in maintaining dialogue continuity, understanding nuanced queries, and delivering coherent multi-turn interactions. These insights will feed into a continuous improvement loop, allowing the system to be refined based on real-world usage patterns, ensuring it remains aligned with both agent needs and evolving client expectations.

2.3 Determine Data Mining Goals

The primary technical goal of this project is to develop a reliable, AI-powered chatbot that enhances the efficiency and effectiveness of Fidelidade's insurance agents by delivering clear, accurate, and contextually appropriate responses. A key objective is to ensure that the chatbot operates strictly within the boundaries of approved content, providing guidance only on validated financial products and literacy materials, thereby upholding regulatory standards and brand integrity.

Another key goal was to replicate the natural flow of human dialogue to support agents during complex, real-time consultations. We aimed to incorporate multi-turn conversational capabilities so that the chatbot could maintain context and deliver more relevant, coherent answers across extended exchanges. Voice input was included to reduce friction for agents who may not have time to type during live client interactions. Additionally, we designed the system to suggest relevant follow-up questions, not simply as a convenience feature, but to anticipate agents' likely next inquiries and help them navigate complex or layered product topics more efficiently. These decisions were driven by the need to streamline interactions, reduce response times, and improve the overall speed and clarity of support during high-pressure advisory scenarios.

At the start of the project, the main data resources available consisted of approximately ten unstructured PDF documents provided by Fidelidade, including official product brochures, internal sales guides, and educational materials. These documents varied in formatting and structure, making extraction and semantic organization a key technical challenge. Their static nature also posed constraints, as updates to product information or compliance requirements would require periodic reprocessing and reintegration into the system.

Finally, the solution is built with scalability in mind. Another core objective is to allow easy integration of new documents into the knowledge base, ensuring the chatbot remains up to date with evolving product offerings and regulatory changes. Together, these goals aim to deliver an intelligent assistant that not only saves time and improves communication quality but also empowers agents to provide consistent, high-value client interactions across all service channels.

3. Methodology

3.1 Data Understanding

This project is built on a hybrid data strategy that combines curated internal documentation from Fidelidade with supplemental contextual input to support broader financial literacy goals. The primary knowledge base consists of unstructured PDF documents provided by Fidelidade (Figure 1), which are carefully selected to reflect key financial products, sales strategies, and educational material. However, the chatbot is also configured to answer general questions on financial concepts by leveraging context provided at runtime, ensuring both relevance and flexibility in user interactions. These documents can be divided into:

1. Product Documentation – Fidelidade Savings

These documents detail the features, terms, guarantees, and investment strategies of the *Fidelidade* "My Savings*"* product, including its various options:

2. Product Documentation – PPR Evoluir

These cover the structure, positioning, and operational management of “PPR Evoluir”, a retirement-oriented product that dynamically balances risk with age through “Proteção” and “Ativo” components.

3. Market Intelligence and Benchmarking Reports

These reports provide external comparisons of Fidelidade's offerings (especially PPR Evoluir and Fidelidade Savings) against market competitors based on risk, returns, and investment strategies.

4. Internal Commercial and Support Material

Files supporting internal usage and sales guidance—such as structured FAQs, pitch decks, or customer advisory material.

The system is carefully configured to restrict the model from answering queries outside the defined scope, ensuring that all responses are grounded in official documentation. Later in the project, we received an e-mail with additional context, which we extracted into a PDF file of the same name and manually added to the knowledge base.

Figure 1 - PDF Example

3.2 Data Preparation and Extraction

This project does not rely on conventional data preparation methods such as cleaning structured datasets or working with relational databases. Instead, our team designed and implemented a custom pipeline to extract and prepare information from unstructured PDF documents provided by Fidelidade.

These documents contained the full breadth of relevant content for the chatbot, including official product information for “My Savings” and “PPR Evoluir,” competitor’s products and public information.

All the PDFs were stored locally in a structured folder system that our group maintained to ensure organized document management throughout development. We chose a hybrid extraction approach, using the **pdfplumber** library for text and **fitz** library for images. Our script was programmed to automatically iterate through each file, reading the content page by page and capturing not just plain paragraphs but also bullet points and other formatting elements that signal key information. The focus was on capturing complete, readable content while preserving its logical structure.

Once the raw text was extracted, we developed a function to **split the content into smaller chunks**, around 700 characters, broken it into semantically meaningful segments. We also ensured that these chunks were formed logically, using paragraph boundaries where possible so that each chunk contained a coherent idea or topic.

Next, we used the **OpenAI text-embedding-ada-002 model** to do embedding, converting each chunk into a high-dimensional vector. These embeddings numerically represent the meaning of each chunk of text, allowing us to compare user queries to the document content based on semantic similarity, rather than just keyword matching.

After generating embeddings for all the document chunks, our group implemented **cosine similarity search** using the **FAISS library**. We first normalized the embeddings and then stored them in a **FAISS index**, which is optimized for efficient similarity search across large sets of vectors. This setup allows us to retrieve the most relevant content in milliseconds.

When a user submits a query, the system embeds the question using the same text-embedding-ada-002 model. It then compares this embedding with all stored document vectors in the FAISS index to identify the **top-matching chunks** of information. These results are dynamically inserted as context into a prompt for the **GPT-4o mini model**, enabling it to generate **accurate, context-aware responses** grounded in the original documents.

One of our main innovations was shifting from a static, prompt-based design to a **retrieval-augmented generation (RAG)** approach. This change significantly improved response accuracy and adaptability, as the chatbot now retrieves real, document-based context instead of relying solely on generic pre-trained knowledge. Our design ensures that all product-related answers are drawn from Fidelidade’s official documentation, minimizing the risk of hallucinated or non-compliant content.

In addition to handling product-specific queries, the chatbot is also designed to support **general financial literacy questions**. We addressed this through **system prompt Engineering**, where we specifically instruct the model that financial literacy is within scope, allowing it to explain broader concepts like investment principles or retirement planning even when those topics are not fully covered in the PDFs.

Finally, our system was built with **scalability and maintainability** in mind. Adding new documents is as simple as placing a new PDF into the source folder and rerunning the same pipeline. The new content is automatically extracted, chunked, embedded, and indexed with no re-engineering required. This ensures that the chatbot can evolve alongside Fidelidade’s product offerings and educational materials, making it a long-term, adaptable solution.

In summary, our group engineered a full-stack, document-driven chatbot pipeline that combines **custom PDF processing, semantic vector search, and advanced language modeling** to deliver fast, accurate, and explainable answers to both agents and clients. This setup balances technical

robustness with ease of use and aligns with Fidelidade's goal of enhancing both **sales efficiency and financial education**.

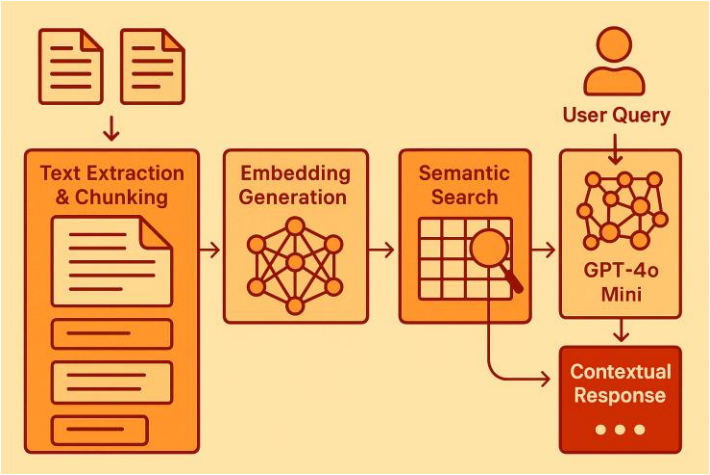

Figure 2 - PDF Information Extraction Process

3.3 GPT-4o Mini Model Configuration

As mentioned, to generate accurate and contextually relevant responses, we integrated the **GPT-4o mini model** into our chatbot pipeline. This model was selected for its strong balance between performance and efficiency. It delivers high-quality responses with significantly faster processing times and lower costs than the full GPT-4 model, making it ideal for scalable deployment in a support assistant setting.

For each user query, the chatbot retrieves the top k most relevant document chunks using semantic search with FAISS. These chunks are compiled into a single context string, which is then inserted into a custom prompt. The prompt is passed to the GPT-4o mini model using OpenAI's API. To avoid exceeding token limits, the context is truncated to approximately 300 tokens before being sent.

The prompt structure consists of two main components:

1. A **system message** that defines the model's persona and constraints through prompt engineering.
2. A **user message** that combines the retrieved document context with the user's actual prompt or question. This format ensures the model is grounded in validated content and can respond accurately to specific inquiries.

<u>System Message</u>

To define the model's persona, we explicitly established that the chatbot should act as a **sales assistant supporting insurance agents**, providing only **certified, document-based responses** from Fidelidade's official information and national financial education sources. It emphasizes clarity, professionalism and client-friendly language.

To handle uncertainty and limits, we define the scope for which the bot can answer, which is Fidelidade products, its competitors' and Financial Literacy topics. If the model detects that a question cannot be answered based on the retrieved context or the defined scope, it is programmed to respond with predefined, professional fallback messages. These messages are intentionally phrased to reflect transparency and accountability, avoiding any fabricated or speculative answers. For example, for out-of-scope prompts, the chatbot replies with: *This question is outside of my reach. I would love to help you with questions related to Fidelidade products, their competitors' products (comparison) or financial literacy. Can I assist you within those boundaries?'.* This approach not only safeguards against hallucination but also ensures the conversation remains within the model's validated knowledge scope. By doing so, the chatbot aligns with best practices for responsible AI use in regulated industries, especially in domains such as insurance and financial services, where misinformation can have serious consequences. It reinforces the assistant's role as a reliable digital support tool, capable of maintaining professional standards throughout the interaction.

The model is configured with a **temperature setting of 0.3**, which reduces response randomness and promotes factual, stable answers. This is a deliberate change from the default (which was previously 1.0) to improve the consistency and trustworthiness of the chatbot's outputs.

Parameter **seed** is set at 42, to ensure that the same input will reproduce the same answer throughout usage.

Additionally, we trained our model by tuning parameters such as *top_p, frequency_penalty* and *presence_penalty*, which regulate precision by probability, repetition and novelty of terms (respectively), but after testing, we concluded that these parameters were not increasing the performance of our model.

4. Deployment

The deployment was implemented through a standalone Python script (fidelia.py) using the **Streamlit** library to create an interactive web interface. While the technical details of the chunking process, embedding generation, FAISS index creation, and GPT-4o model configuration have been discussed in previous sections, this part of the project focuses on how those components are loaded and prepared for real-time use. At runtime, the application performs the deserialization of document chunks, FAISS index loading, and model initialization. This setup phase bridges the earlier modeling work with the real-time deployment environment, ensuring that the interface is seamlessly connected to both the document-based knowledge base and the language model that powers the assistant.

4.1 User Interface Design

The chatbot interface (figure 3) features a clean, branded layout designed to be both intuitive and visually aligned with Fidelidade's identity. A central white-framed container is positioned against a soft grey background, helping distinguish the interactive area and enhance focus for users.

The header includes dual logos, Fidelidade's corporate logo on the top-right and a secondary logo on the top-left designed using Fidelidade dog and a headset representing our chatbot, laid out using

proportional column spacing for visual balance. The title *"Assistente Fidélia"* appears prominently in bold, accompanied by a welcoming subtitle that guides user interaction. The assistant was named "Fidélia", and intentionally designed as a female persona, to humanize the tool and strengthen emotional connection with users. This decision aimed to make the interaction feel more natural, trustworthy, and engaging, especially in a support-driven context. Inputs are styled with a soft red background and rounded edges, while the main action buttons use Fidelidade's red palette and feature hover effects for interactivity. Responses appear in distinct chat bubbles, with a blue-highlighted sidebar for user inputs and green for assistant replies, enabling easy role differentiation within the conversation.

An expandable history panel allows users to revisit their full chat session, ensuring continuity and context across interactions. This feature is particularly useful for insurance agents who may need to refer back to previous answers during complex consultations. The interface also incorporates accessibility-friendly design elements such as large, easily clickable buttons, high-contrast text for readability, and flexible input methods that include both voice and text. These inclusive features reduce friction for users with varying levels of technical proficiency or accessibility needs, promoting a more comfortable experience for a diverse workforce.

Figure 3 - General Interface of the Chatbot

4.2 Voice Recognition and Language Detection

To enhance accessibility and usability, we incorporated voice-based input allowing users to submit queries using speech rather than typing, if preferred (figure 4). The implementation uses the **speech recognition** Python library, a widely used module for converting spoken audio into text via external APIs, in this case, Google Speech Recognition. In this case, the technical workflow can be divided into 4 sections:

1. **Input Method Selection**: Here users select their preferred input methods via a Streamlit radio buttons, like "texto" that represents text or "Fala (microfone)" that represents audio.
2. **Voice Capture**: If the user selects the voice option the application will activate the system microphone via the library **sr.Microphone()**, adjust the background noise using the library **recognizer.adjust_for_ambient_noise()**.

3. **Speech-to-Text Conversion:** After that, the audio is captured and converted into text input, which is then treated the same way as a manually typed question.
4. **Error Handling:** The system includes basic exception handling to manage failed recognition attempts. For instance, if the service cannot understand the input, the user is shown a warning message and prompted to try again.
5. **User Confirmation:** Lastly, once voice input is successfully transcribed, the application will display the interpreted question on the interface, for user transparency and confirmation and then answer it.

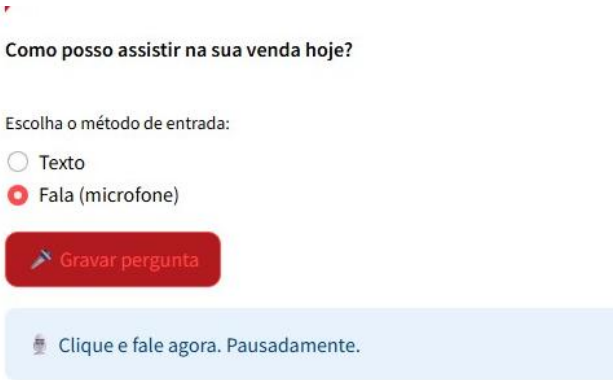

Figure 4 - Voice Input Oprion

Additionally, the chatbot interface includes language detection, so that it can dynamically adapt responses to the user's preferred language. This feature works for both text and speech input and was implemented using **langdetect** Python package. The pipeline consists in a customized system prompt that is then passed into the OpenAI model configuration. This approach reinforces a user-centered design philosophy, extending the chatbot's usability and promoting inclusiveness.

4.3 Chat History and Session Rate

To support multi-turn dialogue and maintain conversational coherence, the chatbot implements stateful memory using Streamlit's **st.session_state** object. At application startup, the session state is initialized with a single message, the system prompt, which defines the assistant's role, scope, and communication style.

As users interact with the interface, each query and corresponding model response is stored in **st.session_state.chat_history as** a list of dictionaries, each representing a message with its associated role **("user"** or **"assistant"**) and content. This dynamic session memory persists throughout the user's session, enabling the chatbot to retain full conversational context.

When a new input is received, the complete chat history is duplicated and used to construct the message payload for the GPT-4o-mini model. In addition to the accumulated user-assistant exchanges, a system-level context string containing the most relevant document chunks is injected at index 1 of the message list, immediately after the initial prompt. This structure ensures the model has access to both historical conversation and domain-specific information during inference.

The chat history is also rendered in the UI using a collapsible **st.expander**, allowing users to review past messages. A “Limpar conversa” button enables users to programmatically reset the session state to its original configuration, preserving only the system prompt and discarding all other messages. This architecture ensures consistent, context-aware interactions while giving users control over their session lifecycle.

4.4 Follow-Up Questions Suggestions

To enhance the conversational flow and assist users in exploring related topics, the chatbot implements a **follow-up question suggestion system** powered by the GPT-4o-mini model (figure 5). This feature allows the assistant not only to answer the current query but also to propose additional, context-relevant questions that the user may want to ask next. The mechanism is driven by prompt engineering. In the system prompt, GPT-4o-mini is explicitly instructed to include, after each answer, a **JSON-formatted list** of 4 follow-up questions.

After the model generates a response, the raw text is scanned using a regular expression to extract the JSON array or follow-up suggestions. Then, the list is parsed using **json.loads()** library, to convert it into a Python list of strings. In case of an error, for example the parsing fails because a malformed JSON, the system ignores it. Afterwards, the suggestions are displayed as clickable buttons in the Streamlit interface, and when a user clicks on one, the button value is saved, and the app is re-run to treat the selected suggestion as a new user query.

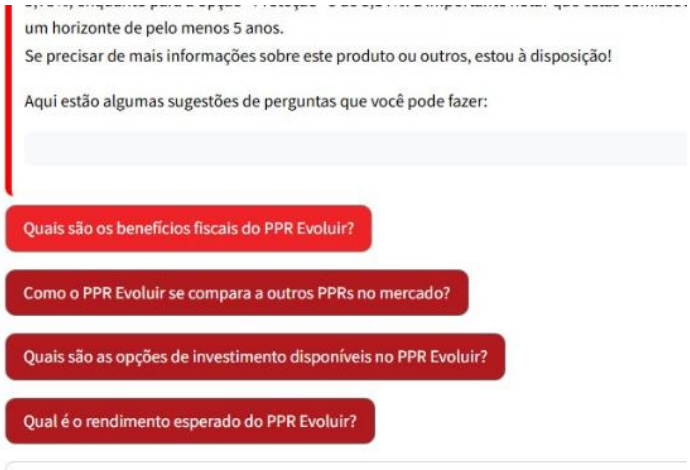

Figure 5 - Follow Up Questions Example

This feature is important because it improves user engagement by guiding the user toward deeper exploration of topics, making the conversation feel more interactive and helpful. It also reduces cognitive effort by offering structured next steps, which is especially beneficial to anticipate what the client may want the agent to answer next. Additionally, by suggesting relevant follow-up questions automatically, the chatbot mimics proactive, human-like assistance, resulting in a more natural and fluid conversational experience.

5. Performance Evaluation

To measure the performance of the chatbot, we implemented a quantitative evaluation framework designed to assess the semantic accuracy of the model's responses. The goal was to determine how closely the answers generated by the GPT-4o mini model aligned with a set of reference answers based on Fidelidade's official documentation.

To calculate the proximity between the two responses, we introduced **Cosine similarity,** a common metric in natural language processing that measures how semantically similar two pieces of text are by comparing the angle between their embedding vectors. We chose this metric because it captures meaning - even when wording varies, aligns well with our embedding-based system, and provides an objective way to evaluate and compare model performance.

Thus, the evaluation process was structured in 2 phases, as follows:

1. **Test Sets**: We created two dictionaries of question-answer pairs. One for the "Fidelidade My Savings" product and another for "PPR Evoluir". Each dictionary contained realistic, frequently asked questions along with verified, expert-written answers that reflect accurate product information.
2. **Automate Testing Functions**: We developed a Python function, **evaluate_similarity()**, that runs each test case through the entire chatbot pipeline. The function first calls the chatbot, performing semantic retrieval and generating a context-aware response using the GPT-4o Mini model. It then computes the cosine similarity between the chatbot's generated answer and the reference answer stored in the dictionaries. This is done by embedding both responses using OpenAI's **text-embedding-ada-002** model and comparing their vector representations.

The resulting similarity scores ranged from approximately 0.83 to 0.95, with most scores above 0.87. Scores above 0.85 generally indicate that the chatbot captured the core meaning of the answer, even if it rephrased it or structured it differently. Lower scores (e.g., around 0.83) may indicate either minor content variation or more general phrasing but still reflected factually correct and relevant content. While cosine similarity does not measure grammatical quality, user experience, or tone, it offers a solid and interpretable baseline for evaluating semantic fidelity, that is, whether the chatbot is conveying the correct information with respect to trusted documentation. The detailed results for each test case are presented in Table 1 and 2 below.

In addition to the quantitative analysis, a qualitative evaluation was also performed. Group members manually reviewed and compared the chatbot's responses to the expected answers to assess the clarity, completeness, and tone of each reply. This complementary review ensured that the model's output not only aligned semantically but also met the standards of professionalism and user-friendliness required for customer-facing communication.

Table 1 - My Savings Questions Results

Question	Cosine Similarity
O Fidelidade Savings é mais indicado para que perfil de risco?	0.9532
O Fidelidade Savings permite dedução no IRS?	0.8282
Tenho de pagar algum imposto sobre os lucros, ou o rendimento já é líquido?	0.8537
Qual é a comissão de gestão anual do Fidelidade Savings?	0.9183
Quais os riscos associados ao Fidelidade Savings para o cliente?	0.8710
Posso subscrever e resgatar em qualquer momento?	0.8812

Table 2 - PPR Questions Results

Question	Cosine Similarity
Que tipo de investimento é o PPR Evoluir — Capital e rendimento garantido ou tem risco?	0.9388
Posso escolher a componente que pretendo (Ativo ou Proteção)?	0.8300
Existe diferença fiscal entre fazer resgate/reembolso antes ou depois dos 5 anos?	0.9191
Se tiver feito uma entrega de 1.000€ há 10 anos e outra de 100.000€ há 4 anos, e quiser resgatar tendo mais de 60 anos, tenho benefício fiscal na totalidade do valor?	0.8461
Qual é a comissão de gestão anual do PPR Evoluir?	0.9012

6. Future Work:

6.1 Internal Integration

As a next phase of deployment, one promising direction is the integration of the chatbot into Microsoft Teams, which serves as a central communication hub for many of the company's internal teams, particularly insurance agents. Embedding the chatbot within Teams would streamline access, allowing agents to consult it directly from the platform they already use in their daily workflow.

One of the most accessible and low-effort approaches to this integration is through the use of Teams tabs. Microsoft Teams supports the embedding of external web applications via HTTPS links. This means that the current Streamlit-based chatbot, once it is hosted and made available through a usable, secure URL, can be integrated into Teams as an interactive tab within a channel or group conversation.

Procedure:

To achieve this, the chatbot application can be deployed on a company-owned internal server, where it runs as a persistent service behind the corporate firewall. This setup ensures the chatbot remains accessible only to authorized users within the organization, offering the necessary level of

security and compliance. The internal server is configured to run the Streamlit application continuously, exposing it via a private HTTPS URL (e.g., https://chatbot.fidelidade.int). This URL can then be added into Microsoft Teams by simply creating a new tab in a conversation or team channel, selecting the “Website” option, and inserting the chatbot link (figure 6,7 and 8). From that point onward, agents can access the assistant directly from Teams with no installation required—allowing them to consult product information and get support in real time during client interactions.

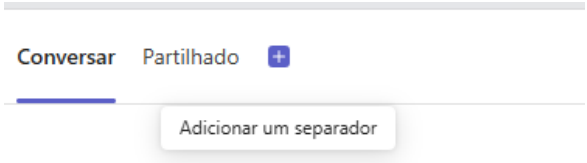

Figure 6 - Add a new APP

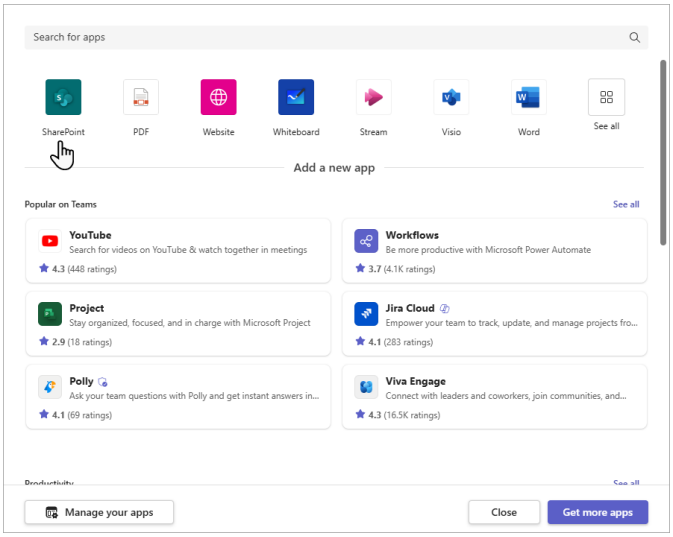

Figure 7 - Choose Website Option

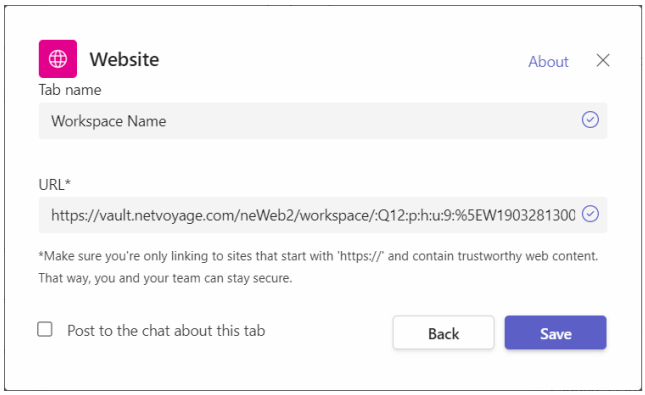

Figure 8 - Insert the ULR of the ChatBot

This method offers a lightweight and cost-effective integration strategy that does not require additional development in the form of bot packaging or Azure Bot Services. It leverages existing infrastructure and makes use of built-in Teams functionality, making it ideal for internal rollout and feedback collection. Should broader adoption or additional features be required in the future, this

foundational integration can later be extended into a more advanced Teams bot architecture using the Microsoft Bot Framework or Azure services. Figure 9 illustrates the full flow of this integration process.

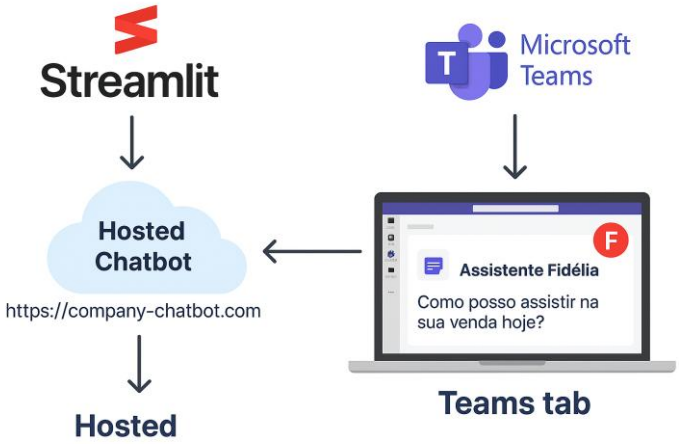

Figure 9 - Microsoft Teams implementation Process Overview

Besides that, the modular architecture of “Assistente Fidélia” enables seamless integration across multiple communication platforms through API-based connections. The core RAG pipeline and GPT-4o mini model can be connected to Microsoft Teams through Azure Bot Service for native workplace integration, WhatsApp Business API for customer-facing support, Telegram Bot API for informal team communications, or Slack API for cross-departmental collaboration. Each platform integration requires only lightweight message handling adapters that translate platform-specific message formats to the chatbot’s unified input structure, while the document retrieval and response generation remain identical across all channels. This approach allows Fidelidade to deploy consistent AI assistance across multiple touchpoints, from internal Teams conversations to external customer WhatsApp interactions, ensuring agents have access to the same reliable product knowledge regardless of their preferred communication platform, while maintaining enterprise-level security and governance policies.

Figure 10 - Whatsapp Integration Example

6.2 Maintenance Plan

To ensure that the chatbot remains functional, up to date and aligned with Fidelidade's products, while continuing to deliver value to agents and internal clients, it is necessary to implement a maintenance plan. This plan will include various technical operations, content updates, feedback analysis, and performance monitoring, and will be divided into a short-term and a medium-term maintenance strategy.

1. Short-Term Maintenance (0 to 3 Months)

Short-term maintenance aims to ensure the proper functioning of the chatbot following its initial implementation, guaranteeing technical stability and continuous content updates. This phase focuses on system monitoring, integration of new documents, and early detection of failures, ensuring that the assistant remains useful, accurate, and aligned with the needs of agents. It is divided into three main areas:

- **Technical maintenance**, to be carried out on a monthly or continuous basis, ensures that the application remains operational and stable. This includes verifying the status of the server hosting the chatbot, monitoring the performance of the GPT-4o Mini API (via Azure), and analyzing any failures in the assistant's responses. Additionally, it is necessary to monitor resource usage such as CPU, memory, and bandwidth to prevent unexpected interruptions.
- **Document and input updates** should be performed quarterly, or whenever relevant changes occur, such as the launch of new products, fiscal or contractual updates, or new marketing campaigns. In these cases, the new documents must be processed through the segmentation and embedding pipeline, with the FAISS index being regenerated to ensure the chatbot reflects the most recent information. It is also essential to ensure proper data versioning, allowing traceability of the knowledge base used by the model.
- **Session and log management**, recommended monthly, involves the periodic deletion of old sessions and the review of unresolved interactions, especially those handled with generic or out-of-scope responses. This analysis helps to identify recurring issues, knowledge gaps, and possible adjustments to the prompt or document base, contributing to the continuous improvement of the assistant's performance.

2. Medium-Term Maintenance (4 to 6 Months)

Medium-term maintenance aims to ensure the continuous evolution of the chatbot, maintaining its relevance, accuracy, and alignment with the organization's digital strategy. This phase goes beyond basic technical stability, focusing on the evaluation of response quality, the collection and analysis of user feedback, the updating of the knowledge base, and the progressive adaptation of the assistant to emerging business needs. As with short-term maintenance, this phase is divided into three main areas:

- **Feedback evaluation**, to be conducted on a quarterly basis, helps understand user perceptions regarding the usefulness, clarity, and accuracy of the responses provided. If a direct feedback mechanism is implemented (such as short surveys), the collected data should be systematically

analyzed to identify recurring dissatisfaction patterns, improvement opportunities, and common suggestions. Holding listening sessions with frequent users (such as sales agents) may also complement this process with qualitative insights.

- **Content validation**, recommended on a semi-annual basis, involves the manual review of a sample of chatbot responses to ensure they remain factually correct, up to date, and consistent with Fidelidade's institutional language. This process may require the revision or replacement of documents in the knowledge base, adjustments to the system prompt, or strategic reformulations in how information is presented to users.
- **System performance evaluation**, to be carried out quarterly or semi-annually, includes applying quantitative metrics such as embedding similarity between expected and generated responses, average response time, usage frequency, and the rate of out-of-scope replies. This monitoring process ensures the semantic quality of the model over time and helps assess the impact of the solution on agent productivity and overall user experience.

Thus, medium and long-term maintenance is essential to ensure the chatbot evolves alongside the organization, remaining a smart, reliable, and adaptable support tool in response to the constantly changing needs of its users.

6.3 Cost-benefit analysis

At this stage of the report, it is clear that our proposed solution performs as intended and aligns closely with the client's operational needs. However, assessing the financial implications of implementing this solution is essential to support strategic decision-making.

During our Q&A with the client, no specific budget or timeline was provided, as the project remains exploratory. Nevertheless, presenting a cost-benefit projection allows us to estimate the investment required and highlight the operational advantages that this solution could bring.

6.3.1 Benefits

<u>Improve quality of service</u>

- **Conveying credibility and professionalism,** reinforcing brand reputation
- **Enhance customer experience,** with faster answers

<u>Reduce Operational Costs</u>

- **Support onboarding** by acting as a training tool
- **24/7 support,** enabling one assistant to serve all agents simultaneously
- **Increase agent productivity**, focusing more on sales-conversion during interactions and saving time while

<u>Manage Risk</u>

- **Centralize and standardize information,** enabling quick access to reliable and up-to-date content
- **Knowledge retention,** reducing flow of paper and documents within organization reducing impact of agent's turnover
- **Normalize communication,** ensuring agents gives out the same consistent information
- **Minimize risk of misinformation** — instead of managing 3,900 different voices, you only need to train one assistant.

6.3.2. Cost Structure

Direct Costs (Financial)

These are the costs that directly impact the company's budget — usually related to external services or infrastructure.

Table 3 - Direct Costs (Financial) Summary

Item	Notes
Initial development	Depends on scope, integration, UI
GPT-4o Mini API (Azure)	Varies with usage volume (number of requests).
Cloud Infrastructure	Hosting, storage, analytics, depends on usage

Indirect Costs (Internal Work Hours)

These do not represent direct payments but rather the time spent by internal teams - Data, IT, Product, etc. They are measured in hours, not payroll, based on time allocation.

Table 4 - Indirect Costs (Internal Work Hours) Summary

Task	Frequency	Notes
Document Updates	Quarterly	For new products, campaigns, etc.
Session and Log Management	Monthly	Analysis + cleanup
Feedback Evaluation	Quarterly	Includes analysis and team meeting
Content Validation	Semi-Annual	Manual review of answers
Performance Evaluation	Quarterly	Includes quality metrics review

Productivity Savings (Financial)

Assuming each of the 3,900 agents saves **15 minutes/day,** with a conservative hourly cost of 8€/hour:

- **15 mins/day × 3,900 agents × 230 workdays/year = 224,250 hours/year = €1,794,000**

<u>Summary of Yearly estimated cost structure</u>

Table 5 - Yearly Estimated Costs Summary

(-)	Direct Costs	€48,000 – €96,000
(-)	Indirect Costs	€12,000 – €24,000
(=)	**Total Costs**	**€60,000 – €120,000**
(+)	Productivity Savings	€1,794,000
(=)	**Net Value**	**~ €1.5M**

In other words, **every €1 invested in the chatbot assistant could return €14 in a pessimistic scenario**, not including additional revenue from better conversion and customer satisfaction.

7. Conclusion

The development and deployment of the "Assistente Fidélia" chatbot marks a significant step forward in how Fidelidade empowers its insurance agents to navigate increasingly complex client interactions. By combining robust technical design with a clear understanding of business needs, the solution offers practical benefits: improved efficiency, more consistent communication, and stronger support for financial literacy.

What sets this project apart is not just its technical soundness, but its thoughtful integration into real-world workflows. From the decision to ground all responses in official documentation, to the seamless user interface and the inclusion of voice and multilingual support, every aspect was designed with the agent's daily experience in mind. Performance metrics, both quantitative and qualitative, confirm that the assistant delivers accurate, timely, and professional support, exactly the kind of tool that helps organizations stay agile in a fast-moving industry.

Looking ahead, the roadmap for integration into Microsoft Teams and WhatsApp, paired with a structured maintenance plan, ensures this isn't a one-off experiment, but a scalable, long-term asset. With strong return-on-investment projections and a flexible architecture ready for future updates, the chatbot has the potential to become a cornerstone of Fidelidade's digital transformation strategy.

Ultimately, this project reflects the power of blending data science with human-centered design. It demonstrates that, when done right, AI doesn't replace people, it helps them do their jobs better.