INTELI – Instituto de Tecnologia e Liderança

Public Report – Project AtendeAI

Module 2: Intelligent Agent and Integration

Students:
Felipe Saadi Siegert – A2022.1A.0145 – Software Engineering
Jonas Viana Sales – A2022.1A.0192 – Computer Science
Raissa da Silva Sabino – A2022.1A.0252 – Software Engineering

Supervising Teacher: Ana Cristina dos Santos

São Paulo, June 2025

Summary

Chapter 1: Module Overview

1.1 Introduction

This report presents the results obtained in Module 1 of the AtendeAI project, developed in partnership with PRODAM and IBM. The main objective was to deliver a navigable prototype of the citizen experience and define the initial architecture of the solution. Over the course of 5 sprints, we sought to refine the understanding of the problem, define the architecture, build the prototype, and subsequently refine it to include accessibility and support for multiple platforms.

1.2 Work Methodology

The work methodology adopted throughout this first flow of the AtendeAI project was based on an iterative and incremental development cycle, structured in five fortnightly sprints. The work was organized through distinct tracks, which allowed us to focus on different aspects of the project: technology, business, leadership and user experience. This approach enabled frequent validations with the partner, PRODAM, ensuring that deliveries were aligned with the expectations and needs identified. Deliveries were made iteratively, presenting technical and functional artifacts in each cycle. For effective management of deliverables and team communication, we used the tools Trello, Slack and GitHub. The work methodology that guided the development was Scrum, providing flexibility and adaptability throughout the project.

Chapter 2: AI Agent

This chapter details the construction and core components of the AI Agent, which is designed to transform how users interact with and request government services. Instead of analyzing past user journeys, our approach focused on building an intelligent system from the ground up. The development was centered on three fundamental pillars: **Vectorized Knowledge Base, Advanced Prompt Engineering** and **Standardized Service Request Model.**

2.1 Evolution and Training of the AI Model

The evolution of our AI model began with a diagnostic analysis of existing citizen service platforms, such as SP156. By creating a BPMN diagram of the current workflow and analyzing user reviews from the App Store, we identified critical bottlenecks. A primary pain point was the mandatory selection of a service category, a step that often confused citizens who did not know how to classify their requests. Additionally, qualitative data revealed user frustration with confusing interfaces and an excessive number of steps to submit a formal request.

To directly address these shortcomings, the training and architecture of the AtendeAI agent were centered on three technical pillars:

1. **Creation of a Vectorized Knowledge Base:** Instead of forcing users to choose a category, we transformed all service descriptions and procedures into a vectorized knowledge base. This "training" allows the model to understand a user's needs in natural language and automatically identify the appropriate service, directly solving the category selection problem.
2. **Advanced Prompt Engineering:** We developed a set of sophisticated prompts to refine the model's behavior. These prompts act as a core part of its "training," guiding the agent to correctly interpret the retrieved service specifications and interact with the user in a clear and effective manner.
3. **Development of a Standardized Service Request Model:** To combat interface complexity, we designed a simple, conversational model for formalizing requests. After identifying the correct service, the agent uses this framework to guide the user through a few messages, streamlining the journey and eliminating the confusion of traditional forms.

This evolution from a problem-focused analysis to a solution-oriented architecture represents the core of our AI model's development, ensuring it is purpose-built to improve efficiency and accessibility.

2.2 Integration with IBM Watsonx and Assistant

The integration between Watson Assistant and the watsonx platform is a central component of the system's architecture, designed to be both direct and decoupled. This is achieved through a dual-API approach:

- One API is responsible for managing the primary backend logic and the Watson Assistant instance itself.
- A second, dedicated API serves as a specialized bridge for all communication with the watsonx services.

To ensure this integration is robust, efficient, and maintainable, the communication between these components is standardized by the OpenAPI specification. This allows the Watson Assistant to leverage watsonx's capabilities seamlessly while maintaining a clean and scalable architectural design.

Chapter 3: Backend of the Application

In this module, we focus on preparing the project structure, allowing the design of a solution in a more robust, modular and evolution-ready way. The architecture is presented at different levels of abstraction through the C4 model, which facilitates the visualization of the interactions between the components and actors involved. We also detail the technical aspects related to the implementation of the solution, in addition to the essential architectural requirements to ensure scalability, security, resilience and alignment with the functional and non-functional objectives defined throughout the module.

3.1 Backend Architecture and Components

The backend architecture is designed with a focus on modularity and a clear separation of concerns, ensuring the system is scalable and maintainable. It is logically structured into several key components, each with distinct responsibilities:

API Layer: An API Gateway serves as the single, secure entry point for all client requests. It is responsible for request routing, authentication, rate limiting, and forwarding traffic to the appropriate internal services.

Service Layer: This core layer contains the primary business logic of the platform. It is composed of multiple services that encapsulate specific business functionalities, process data, and execute the main operations of the system.

Data Persistence Layer: This layer is responsible for all interactions with the underlying databases and data stores. It abstracts the complexities of data storage and retrieval, providing a consistent interface for the service layer to manage data.

Supporting Services: A set of internal services runs alongside the core application to handle asynchronous tasks, such as background jobs, message queuing, notifications, and integrations with third-party systems.

These components interact through well-defined internal APIs, ensuring that the system remains decoupled and robust.

3.2 Message Handling and Persistence

The solution was designed to ensure scalability, security and resilience, using IBM Cloud infrastructure and modern DevOps practices. The technical environment is composed of multiple integrated services, with services persistence orchestration, secure authentication, messaging and integration with natural language models (NLP).

3.3 Communication with IBM Services (Watsonx, Assistant)

A key system requirement is to enable direct and decoupled communication between the Watson Assistant and the watsonx platform services. To meet this need in an efficient and sustainable manner, the architecture was designed around two primary APIs: one for managing the backend and the assistant, and another dedicated to the integration with watsonx. The communication between these components is standardized by the OpenAPI specification, ensuring a robust and maintainable integration capability.

Chapter 4: Frontend of the Application

The development of a high-fidelity navigable prototype, built in tools such as Miro or Figma, was an essential step in the process of creating our application. This prototype allowed us to simulate the real user experience, test navigation flows, validate functionalities and identify improvements even before coding.

With it, it was possible to ensure that the interface met the needs of users, promoting an intuitive, accessible and efficient journey. In addition, the prototype served as a valuable tool to align expectations between the design, development and stakeholder teams, accelerating decision-making and reducing rework costs.

4.1 User Interface and Functionalities

Among the main features, the following stand out:

🔹 **Ticket Opening**
Users can open tickets directly through the app to report issues or suggest improvements in their area. Each ticket is logged with details such as location, description, and category.

🔹 **View Request History and Status**
Monitor the progress of your requests in real time. The user has access to a timeline with the updated status of each ticket (e.g.: "Under analysis", "Approved", "Completed"), promoting transparency and trust in the process.

🔹 **Chat Interaction with Chatbot and Attendants**
The app has a chat feature that allows direct communication with sub-prefecture attendants, as well as a chatbot that automatically answers the most common questions, optimizing service and speeding up problem-solving.

🔹 **Real-Time Notifications and Updates**
Receive instant notifications about updates to your requests, new chat messages and important information from the subprefecture.

🔹 **Message Center**
Agents have access to all their conversations organized in a clear and easy-to-use interface, with a complete history and separation by priority.

4.2 Integration with Backend and APIs

The integration between the frontend and backend was implemented using asynchronous API REST calls, ensuring smooth communication between the user interface and the processing and data storage services. Requests made by citizens through the chatbot are sent to the backend, which processes them and interacts with the classification module and the database hosted in the cloud.

To ensure scalability and maintainability, a decoupled architecture pattern was adopted, allowing functionalities to be easily expanded. All routes and endpoints were tested using tools such as Postman and Insomnia, validating security, response time, and correct data transmission.

Additionally, integration with external APIs, such as location services or authentication, was essential to enrich the user experience and ensure the reliability of the data processed by the application.

4.3 Visual and Accessibility Enhancements

The application's frontend was developed with a focus on **usability, visual clarity, and accessibility**, using **React with Tailwind CSS** to ensure responsive components and consistent styling.

Several enhancements were applied to the interface based on accessibility best practices (WCAG), including:

- Proper contrast between text and background;
- Adjustable font sizes to support different user profiles;
- Full keyboard navigation support;
- Clear chatbot message reading by screen readers;
- Semantic HTML structure to ensure compatibility with assistive technologies.

In addition, navigation was organized logically, with simple and intuitive flows. The colors used in the request cards were chosen based on contrast and visual signaling principles, making it easier to read for users with low vision or color blindness.

These improvements ensure that **AtendeAI is accessible to all citizens**, regardless of age, digital literacy level, or visual condition, promoting **digital inclusion** and equity in access to public services.

Chapter 5: System Integration

5.1 End-to-End Flow (Frontend → Backend → IBM Services)

The complete flow of the AtendeAI system begins in the Frontend, where the citizen interacts with the chatbot through an intuitive and responsive web interface. The messages sent are handled by the Backend, which manages communication with IBM Cloud services such as:

- Watsonx Assistant, responsible for natural language conversations;
- Watsonx.ai, for automatic classification of service requests;
- Database and messaging services, which store and distribute information to the appropriate modules.

After processing and classification, the backend returns the response to the frontend with the request status and real-time updates.

5.2 Integration Testing and Logs

To validate the robustness of the integration between modules, **end-to-end integration tests** were carried out, simulating real usage scenarios. The tests verified:

- Sending and receiving messages between the chatbot and the user;
- Correct classification and prioritization of requests by AI;
- Registration of requests in the database;

These tests were crucial for validating the consistency of the integration and for fixing unexpected behaviors before releasing the MVP.

5.3 Identified Challenges and Fixes

During the integration process, several challenges were identified, such as:

- **Inconsistencies in input data formats** (e.g., requests without location or with very short descriptions);
- **Limitations in the Watson Assistant response structure**, which required backend adjustments to correctly interpret returned intents and entities;

All of these issues were documented and corrected throughout the testing process, ensuring a smooth and effective integration between the system components.

Chapter 6: Sustainability in AI

IBM Watsonx.ai offers a growing catalog of third-party large language models (LLMs), including both text-based and multimodal (vision + text) options. Among these, only a few models have publicly available data on energy usage and carbon footprint during inference.

6.1 Energy Consumption of AI Models

Unfortunately only LLaMas models have publicly available data about energy and co2 consumption. All other models (DeepSeek, Mistral, Codestral) have no readily available data, thus they were omitted from the following table:

Model	Size (B params)	Energy/query (Wh)	CO_2/query (g)	Notes & Sources
LLaMA-3.2-90B-Vision-Instruct	90B	1.077	3.447	No public energy data
LLaMA-3.2-11B-Vision-Instruct	11B	0.071	0.214	OpenLLM Leaderboard
LLaMA-3.2-3B-Instruct (text-only)	3B	0.115	0.377	OpenLLM Leaderboard
LLaMA-3.2-1B-Instruct (text-only)	1B	0.070	0.218	OpenLLM Leaderboard
LLaMA-2-70B-Chat	70B	~0.33	~4.5	Meta Paper - CodeCarbon Reference
LLaMA-3-70B-Instruct	34B	~0.33	~4.5	Same as above

Chapter 7: Final Considerations

The development of the AtendeAI system throughout Module 2 represented a significant step forward in integrating artificial intelligence into public service delivery. By focusing on the intelligent agent's design and integration, as well as on building a robust and modular system architecture, the team was able to deliver a functional MVP that simulates a simplified, accessible, and effective citizen experience.

Key technical achievements include the construction of a vectorized knowledge base, the application of advanced prompt engineering, and the implementation of a standardized request model. The backend was structured to support scalability and decoupled integrations, and the frontend prioritized usability and accessibility in accordance with international standards. Additionally, the integration between components—frontend, backend, and IBM services—was successfully validated through end-to-end tests, ensuring the reliability and coherence of the solution.

The sustainability dimension introduced in this module enabled a critical analysis of the energy and environmental impact of AI technologies. Although current public data is still limited, especially among proprietary and closed-source models, the information gathered establishes an important foundation for evaluating the environmental footprint of deploying LLMs in public-facing systems.

In summary, the AtendeAI project not only achieved its core objectives for Module 2 but also laid a solid groundwork for the system's future evolution. The architecture and implementation decisions made in this phase enable ongoing refinement, new feature development, and potential scalability to broader citizen service domains. Looking ahead, further exploration of responsible AI use, continuous performance evaluation, and improvements in user engagement mechanisms will be essential to ensure that AtendeAI delivers both technological excellence and meaningful public value.

References

LLaMa-2-70B-Chat and LLaMa-3-70B-Instruct: https://doi.org/10.48550/arXiv.2307.09288

OpenLLM Leaderboard (Archived): https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard