INTELI – Instituto de Tecnologia e Liderança

Public Report – Project AtendeAI

Module 1: Platform Architecture and Navigable Prototype

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, April 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: User Understanding

In this chapter, we delve into understanding the current state of the user experience within citizen service systems. By mapping and analyzing the existing processes, we aim to identify pain points and uncover opportunities for improvement. Through this diagnostic approach, we gather both qualitative and quantitative data, allowing us to build a clearer picture of the challenges users face. The goal is to create a foundation for proposing more efficient and intuitive solutions that will improve the overall experience.

The analysis presented in this chapter serves as the starting point for the development of our solution, AtendeAI, which seeks to address the identified shortcomings and streamline the user journey.

2.1 Mapping and Analyzing Current Experience

To carry out the **Mapping and Analysis of the Current Experience**, we began with a diagnosis of the citizen service process currently in use, such as SP156. For this, we created a BPMN diagram representing the existing workflow. During this stage, we identified bottlenecks that hinder the user experience — one example is the mandatory selection of a service category, which many citizens struggle with, as they often don't know where their request fits.

In addition, we conducted a qualitative data analysis by collecting user reviews from the App Store, which highlighted important usability limitations. Many users reported difficulties in submitting their requests, mentioning confusing interfaces and an excessive number of steps.

All of these points were compared with the journey we are proposing in AtendeAI, which is much more simplified and intuitive. In our solution, citizens can submit a request simply by exchanging a few messages with the chatbot, without needing to navigate complex menus. This comparison clearly demonstrates how our approach brings real improvements in both efficiency and accessibility.

2.2 Personas and User Story Mapping

Personas are fundamental to product development and user understanding, as they allow us to map all individuals involved in the process — from the end user who will use the app on a daily basis (the citizen), to those responsible for maintaining and operating the application, such as the subprefecture analyst and future managers/developers of AtendeAI.

During Module 1, four potential personas were identified, each with different stories and characteristics addressed by the application — from the citizen who requires accessibility features to use the app, to the manager who seeks to extract insights from the app through a dashboard for business analysis and feature improvement.

Below, the personas are presented:

System Administrator:

- **Name and Age**: Rodrigo Pereira; 34 years old.
- **About Me**: Continuous developer of AtendeAI. Keeps app maintenance.
- **Needs**: Functional app with privileges to maintain service and gather data.
- **Pain Points**: Lack of detailed documentation leads to time wasted learning the app's code.
- **Goals**: Guarantee operational service with updates and collect data.
- **KPIs**: User satisfaction. App response time.

Subprefecture Analyst:

- **Name and Age**: Roberta Souza; 32 years old.
- **About Me**: Subprefecture worker. Focuses on processing requests.
- **Needs**: Well-defined requests with priorities to manage and process them.
- **Pain Points:** Current system is too complex and lacks transparency.
- **Goals**: Process requests and notify current status and ask for more data.
- **KPIs**: Amount of requests that can be processed per day.

Citizen 1:

- **Name and Age**: João Silva; 30 years old.
- **About Me**: Construction worker. Lives in the West Zone in a modest neighborhood.
- **Needs**: Raining where he lives led to falling trees near his house.
- **Pain Points**: Doesn't have time to visit a government office or call to.
- **Goals**: Submit requests via mobile app that is fast and easy to use.
- **KPIs**: Time of request processing. Quick and easy-to-understand notifications.

Citizen 2:

- **Name and Age**: Maria Oliveira. 68 years old.
- **About Me**: Retired teacher. Lives in the South Zone and uses WhatsApp to talk to family.
- **Needs**: Struggles with technology and doesn't want to deal with bureaucracy.
- **Pain Points**: Doesn't have time to visit a government office or call to.
- **Goals**: Submit requests via app that accepts audio, photos, and videos.
- **KPIs:** Amount of steps to open a request. Overall users' opinions about the app.

The User Journey allows us to understand over time how the app's users and those responsible for them will respond to the platform in terms of its usability. This allows us to create systems that are more faithful and integrated with everyone's needs and better reflect the emotions of a given user.

Three user journeys have been mapped, those being the System Administrator, Subprefecture Analyst and Citizen 1:

1. System Administrator

- **Goal**: Monitor agent performance and identify inefficiencies.
- **Stages**:
    - **Accessing the Platform**: Logs in securely to access dashboards.

  - **Viewing the Dashboard**: Views metrics like efficiency, response time, and resolved complaints.
  - **Identifying Issues**: Filters data to identify underperforming agents and takes action.
- **Pain Points**: Login issues, confusing UI, outdated or incomplete data may affect decisions.
- **User Sentiment**: Generally positive, appreciating dashboard clarity, though facing technical concerns.

2. Subprefecture Analyst

- **Goal**: Manage citizen requests through the system.
- **Stages**:
  - **Access Requests**: Logs in to view and handle incoming requests.
  - **Categorize/Forward**: Redirects or reprioritizes based on request type.
  - **Update Status**: Notifies citizens and updates request statuses.
  - **Collect Feedback**: Gathers feedback or reprocesses unresolved cases.
- **Pain Points**: Learning curve for new system, unclear request priorities, system overloads.
- **User Sentiment**: Positive overall, with some frustration over rework and system performance.

3. Citizen

- **Goal**: Request a public service (e.g., tree removal).
- **Stages**:
  - **Request**: Seeks help, installs AtendeAI, starts chatbot conversation.
  - **Send Documentation**: Provides address and data through the app.
  - **Wait for Action**: Tracks request progress via app and WhatsApp.
  - **Review Service**: Leaves feedback after service is delivered.
- **Pain Points**: Confusion about processes, past bad experiences with apps, form fatigue.
- **User Sentiment**: Initially anxious, later reassured by transparency and ease of use.

There is also a supplementary User Story Mapping for better understanding how all users interact between themselves how they key roles are must be integrated within the system.

User Story Map:

- **Personas**: Citizens, Analysts, and Managers.
- **Key Activities**:
  - **Citizens**: Create and track requests, send media, receive updates, stay anonymous if needed.
  - **Analysts**: Categorize, prioritize, forward, update and manage requests.
  - **Managers**: Manage users and permissions, view administrative dashboards.

- **Key System Tasks**:
    - Must support chatbot interaction, file uploads (images/audio), notifications, data privacy.
    - Allow user and organization management, trackable request flow, analytics and reports.
    - Public dashboards should be accessible and regularly updated.

Capítulo 3: Arquitetura da Solução

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 C4 Model

The Atende Aí solution diagrams were created based on the conversations and definitions made throughout the module, clearly representing how the main actors interact with the system.

- **Context Diagram**

The citizen is the entry point for the solution, using the chatbot to send requests and monitor the progress of their requests. These interactions are processed by the Atende Aí system, which connects to other support systems, such as the Secretariat System, responsible for storing data and history, and the Request System, which organizes and distributes demands among the responsible departments.

Public managers and departmental managers access this information to prioritize and process requests, while the Notification System ensures that the parties involved are informed about the status of requests via automatic emails.

This view provides a solid foundation for understanding the solution ecosystem and guides the detailing of the next architectural layers.

- **Container Diagram**

The Container Diagram details the main modules that make up the Atende Aí solution, as well as their responsibilities and technical interactions. The model was built considering both the citizen input channels and the internal services that support the platform's operation.

The solution is made up of different containers that are organized as follows:

- **Web Application**: Responsible for delivering the web interface to the citizen's browser. It is through this interface that users access the platform and interact with the chatbot, sending requests and monitoring their progress.

- **Single-Page Application (SPA)**: Dynamic interface to be developed in Next.js, which provides a continuous user experience. Integrates with IBM Watson to offer intelligent interactions through conversations with the chatbot.

- **Mobile App**: It offers an experience similar to that of SPA, but optimized for mobile devices. It also makes API calls to log requests and check the progress of demands.

- **API Application**: Developed in TypeScript, this layer acts as the solution's backend. It centralizes business logic, integrating with frontend containers, the database, and external systems. It exposes JSON/HTTPS endpoints for communication with applications and integration with external services such as the geolocation system.

- **Database**: Responsible for storing user data, requests, secretarial information, authenticated credentials, access logs, among others. Communication with the backend is done through an ORM (Object-Relational Mapping) layer.

In addition to internal containers, the solution interacts with external services, such as:

- **E-mail System**: SMTP service used to send email notifications to users and managers.

- **Geolocation System**: External service that provides data on Brazilian addresses and locations, used to enrich information during registration or routing of requests.

🔹 Component Diagram

The Component Diagram details the internal structure of the Atende Aí application's API container, revealing how the main code blocks interact to process user requests and orchestrate the operation of the chatbot and other functionalities.

The modeling was built based on the module discussions and divides the system logic into specialized components, promoting separation of responsibilities and greater cohesion.

The main components of the Application API are:

- **Sign-In Controller**: Controller responsible for handling the user authentication process, allowing them to connect to the system.

- **Security Component**: Implements security and authentication mechanisms with OAuth 2.0 support, including anonymous authentication where applicable.

- **Requests Controller**: Manages the sending of messages by users, maintaining the context history necessary for the continuity of conversations.

- **Requests Component**: Service responsible for interpreting messages based on the user's context, using an NLP (Natural Language Processing) engine. This component is integrated with Watson Assistant, which performs semantic understanding of messages with the help of the IBM SDK.

- **Requests History Controller**: Allows access to previous user requests through specific identifiers, functioning as a query layer.

- **Request History Component**: Performs searches on stored requests, enabling filters by fields such as unique identifiers or status.

The database is also segmented according to function:

- **PostgreSQL Database**: Stores structured data, such as user information, credentials, logs and secretarial data.

- **MongoDB Database**: Stores documents of requests made, including status and responsible parties, allowing flexibility in consultation and detailed history.

This modular arrangement facilitates the scalability and maintenance of the solution, promoting component reuse and separating the business logic from the interface and external infrastructure.

3.2 Technical Implementation

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 container orchestration, secure authentication, messaging and integration with natural language models (NLP).

The main technical aspects of architecture are highlighted below:

- **Cloud Services:**
  The solution is hosted on **IBM Cloud**, using services such as **IBM Cloud Kubernetes Service** for container orchestration (Docker/Kubernetes) and **IBM Cloud Load Balancing** to distribute requests efficiently. Horizontal scalability is guaranteed as user load increases.

- **Gateway and integrations**:
  All requests pass through the **API Gateway**, which acts as a central entry point for services including authentication, chatbot, database, and external APIs.

- **AI and Chatbot Integration:**
  Communication with the user is mediated by **IBM Watson Assistant** (Watson), with support for **Speech-to-Text (SST)** and **Text-to-Speech (TTS)** via **Watson Speech Services**, capabilities via Watson Speech Services, making it possible to extend use to voice accessibility.

- **Authentication and Security:**
  The system features authentication **OAuth 2.0/OpenID Connect**, integrated with the **IBM App ID**, ensuring secure access control that is compatible with modern best practices.

- **Data Persistence:**
  User and request data is stored in **IBM Cloud Db2**, enabling real-time queries and support for the RAG (Retrieval-Augmented Generation) model, with dynamic access to the latest information.

- **Automation with Terraform:**
  All infrastructure is managed as code using Terraform, enabling versioning, traceability and reproducibility of cloud configuration.

- **Observability and Monitoring:**
  IBM LogDNA and **IBM Sysdig** services are used to monitor logs and metrics, allowing real-time visibility into application behavior and proactive alerts.

This modern architecture provides a robust foundation for application growth, with a focus on modularity, high availability, and continuous integration.

3.3 Architectural and Technical Requirements

In this section, we present the requirements that guided the architectural decisions, both from a functional and non-functional perspective. These requirements were identified during the scope definition and refined throughout the solution structuring module. We will focus only on the essential requirements; in total, more than 40 requirements were defined to be developed throughout the 3 design modules.

🔹 Functional Requirements

Requirements	Code	Priority	Release
The system must allow the creation of requests.	FR01	1 - Essential	MVP
The system must allow the categorization of a request.	FR02	1 - Essential	MVP
The system should allow the user to view previous tickets.	FR05	1 - Essential	MVP
The system must allow the sending of audio files.	FR18	1 - Essential	MVP
The system must allow the sending and receiving of messages.	FR19	1 - Essential	MVP
The system must properly authenticate users, ensuring that the identity of the person making the request is confirmed.	FR23	1 - Essential	MVP
The system must ensure that the request was made by an authorized user, based on the permissions and privileges assigned.	FR24	1 - Essential	MVP

Non-Functional Requirements

Requirements	Code	Relation	Priority	Release
The system must integrate IBM technologies for natural language processing and artificial intelligence.	NFR06	-	1 - Essential	´MVP
The system must provide a dark mode option to enhance visual comfort, especially in low-light environments, and to help conserve battery life on mobile devices.	NFR01	-	1 - Essential	MVP
The system must be responsive for desktop, tablet and smartphone.	NFR03	-	1 - Essential	MVP
The system should use Natural Language Processing (NLP) to understand colloquial terms and synonyms.	NFR04	-	1 - Essential	MVP
The system must use IBM Watson for conversational chatbot and data analysis.	NFR05	-	1 - Essential	MVP
The system must be based on a microservices architecture to ensure scalability and modularity.	NFR07	-	1 - Essential	MVP
The system must allow users to adjust font size according to their preferences or visual needs,	NFR29	-	1 - Essential	MVP

promoting a more accessible and personalized reading experience.				
The system must allow users to send voice messages as an additional fast and effective communication option between citizens and the local government.	NFR31	-	1 - Essential	MVP
The system must support screen readers for audio accessibility.	NFR02	-	1 - Essential	R2
The system must be replicable across multiple clouds.	NFR08	-	1 - Essential	R2
The system must be fully dockerized.	NFR10	-	1 - Essential	R2
Access to the system must be restricted according to user profiles and the organization they belong to, ensuring that only authorized users can view and modify sensitive data.	NFR19	-	1 - Essential	R2

Chapter 4: Navigable Prototype

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 Features

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 Accessibility and Usability

Highlights of accessibility and usability include:

- **Adapted Interface for Desktop and Mobile**

The platform has a responsive design, allowing an optimized user experience on both computers and smartphones, ensuring practicality and efficiency on any device.

- **Dark Mode**

Provides the dark mode option, offering visual comfort, especially in low-light environments, in addition to contributing to battery saving on mobile devices.

- **Font Size Adjustment**

The user can adjust the font size according to their preferences or visual needs, promoting more accessible and personalized reading.

- **Reading Messages Aloud**

The voice reading functionality allows users to listen to the content of messages exchanged in the chat, which makes it easier to use for people with visual impairments or low vision.

- **Text Interpretation in Libras**

We include support for translating messages into Libras (Brazilian Sign Language), promoting inclusion and ensuring accessibility for people who are deaf or have hearing impairments.

- **Sending Audio**

In addition to text, the app allows the sending of voice messages, offering another alternative for quick and effective communication between citizens and the sub-prefecture.

- **Fluid and Intuitive Navigation**

The user experience was carefully thought out in a high-fidelity prototype (Miro or Figma), tested with a total focus on usability, simplicity of flow and clarity in interactions.

- **Responsiveness and Layout Adaptation**

The interface adapts seamlessly to different screen sizes and orientations, maintaining visual and functional consistency across all devices.

Chapter 5: Conclusion

In summary, the primary objective of the first module was to map out the key users, system architecture, and initial wireframes. These foundational components play a crucial role in guiding the development of an interactive AI-powered chatbot that effectively meets the functional and user requirements outlined in this initial phase of the project. Looking ahead, the next stages will focus on exploring interactive AI solutions leveraging Large Language Models (LLMs), Natural Language Processing (NLP) techniques, and cloud integration tools to deliver a robust and scalable system.