EurecaChat: A Multi-Agent LLM-Based Chatbot for Supporting Students, Faculty, and University Administration

Matheus Hensley de Figueiredo e Silva, Levi de Lima Pereira Júnior, David Eduardo Pereira, Beatriz de Souza Meneses, Kemilli Nicole dos Santos Lima, José Robson da Silva Araujo Junior, Leandro Balby Marinho, Eliane Cristina Araújo
Federal University of Campina Grande
{matheus.figueiredo.silva,levi.pereira.junior,david.pereira,beatriz.souza.meneses,kemilli.nicole.santos.lima}@ccc.ufcg.edu.br
joserobson@copin.ufcg.edu.br
{lbmarinho,eliane}@computacao.ufcg.edu.br

ABSTRACT

Academic data plays a crucial role in guiding students as they plan their curricular paths, supporting professors in refining their courses, and enabling administrators to make evidence-based decisions. Despite its importance, such data are often difficult to access. Even under open data policies, barriers to usability and accessibility frequently limit its impact. In this work, we present EurecaChat, a multi-agent chatbot system that allows users to retrieve academic data through natural language. Built on Large Language Models (LLMs), EurecaChat tailors responses to the needs of different user profiles: students, faculty, and staff, while ensuring compliance with data privacy constraints. Preliminary evaluation shows good performance in several use cases, underscoring the potential of the system to democratize access to academic data and promote more informed decision making in educational environments.

KEYWORDS

chatbot, university, multi-agent, LLM

1 INTRODUCTION

Despite increasing digitalization, public universities in Brazil still struggle to make institutional data accessible in usable formats. In most cases, critical academic and administrative data remains siloed in legacy systems or made available only through static documents, limiting transparency and hindering innovation. At the UFCG[1], the *Eureca* Project[2] was created to reverse this trend by exposing structured real-time academic data through a public Application Programming Interface (API). In doing so, *Eureca* not only improves institutional transparency, but also opens the door to the development of data-driven applications that serve the university community.

This paper presents EurecaChat, a multi-agent conversational system that uses the *Eureca* API to provide intelligent, role-sensitive access to UFCG's academic data. Powered by a **Multi-Agent System** (MAS) based on **Large Language Models** (LLMs) and enhanced with **Retrieval-Augmented Generation** (RAG), EurecaChat enables **students**, **staff**, and **faculty** to query institutional data in natural language and receive accurate, contextualized responses aligned with their access privileges. Rather than functioning as a generic chatbot, EurecaChat acts as an intelligent intermediary between users and complex institutional systems, facilitating access to up-to-date academic records, program structures, student performance data, and administrative procedures.

In educational contexts, chatbots have been widely adopted to support a variety of tasks, particularly in assisting students during the learning process. Most of these systems act as virtual teaching assistants, offering support in answering course-related questions or guiding students through learning activities [4, 6]. However, more recent developments, extend beyond the classroom, supporting institutional workflows such as administrative processes [5], enrollment procedures [1]. This paper aligns with this latter trend, but goes further. EurecaChat is not intended to support academic learning directly, instead, it serves as a conversational interface that enables students, faculty, and administrative staff to interact more effectively with a wide range of university data through natural language.

Prior work addressing administrative burdens in universities typically falls into two categories: (i) systems built on rule-based chatbots and structured knowledge bases, and (ii) systems that integrate machine learning (ML) or generative models to handle more complex interactions. Early implementations relying on static FAQ databases or SQL backends connected to rule-based engines [2, 11, 12] faced limitations in scalability and flexibility, particularly for ambiguous or multi-turn queries. To address these challenges, recent solutions have incorporated classical ML techniques such as BERT-based classifiers [9], generative models built on LLMs [3], or hybrid architectures combining generative models with decision trees or structured pipelines [5, 7]. MAS have emerged as promising alternatives [8] to improve the current landscape of chatbots in educational contexts. For example, ARGObot [13], which simulates an academic advisor using a tool-enhanced LLM, and EduMAS [10], which coordinates multiple agents with knowledge graphs to improve contextual understanding and response accuracy.

[0]Demo video: https://tinyurl.com/eureca-demo
[1]Federal University of Campina Grande
[2]https://eureca.sti.ufcg.edu.br

In: XXIV Workshop de Ferramentas e Aplicações (WFA 2025). Anais Estendidos do XXXI Simpósio Brasileiro de Sistemas Multimídia e Web (WFA'2025). Rio de Janeiro/RJ, Brasil. Porto Alegre: Brazilian Computer Society, 2025.

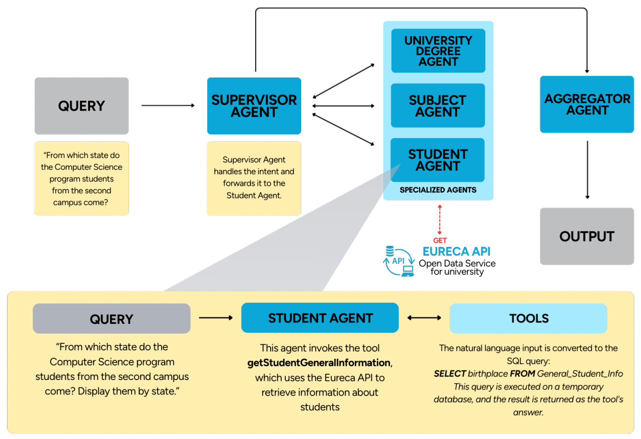
ISSN 2596-1683

2 SYSTEM ARCHITECTURE

Each agent was designed to fulfill a distinct role in the processing of user queries. At the core of the architecture lies the **Supervisor Agent**, which serves as the central coordinator of the system. This agent is responsible for receiving user input, interpreting the intent behind the query, and orchestrating communication among the other agents, as shown in Figure 1.

Upon receiving a query, the Supervisor Agent analyzes its content and delegates it to one of three specialized agents—the **University Degree Agent**, **Subject Agent** or **Student Agent**—based on the nature of the request. Each of these agents is equipped with dedicated tools capable of interacting with the *Eureca* API, a university-developed interface that exposes structured endpoints derived from the institution's central academic database and administrative system. The API acts as a controlled gateway, ensuring secure access to data such as academic programs, courses, student records, and administrative information, while allowing EurecaChat to retrieve only the endpoints relevant to each agent's scope.

Once a specialized agent retrieves or generates a response, the output is sent back to the Supervisor Agent, which evaluates whether further processing is necessary. In case of compound or follow-up queries, the Supervisor Agent may redirect the task to the same or a different specialized agent. Otherwise, the response is forwarded to the **Aggregator Agent**, whose role is to consolidate, format and articulate the final output in a natural language that is clear and user-friendly.



"From which state do the Computer Science program students from the second campus come?

Supervisor Agent handles the intent and forwards it to the Student Agent.

"From which state do the Computer Science program students from the second campus come? Display them by state."

This agent invokes the tool **getStudentGeneralInformation**, which uses the Eureca API to retrieve information about students

The natural language input is converted to the SQL query:
SELECT *birthplace* ***FROM*** *General_Student_Info*
This query is executed on a temporary database, and the result is returned as the tool's answer.

Figure 1: System architecture diagram

All specialized agents in EurecaChat share a modular internal architecture. Each agent begins by parsing the query passed from the Supervisor Agent, identifying key elements such as entities, parameters, and relevant intent. Based on this analysis, the agent may trigger a *tool call*—a mechanism through which the LLM invokes external functions dynamically. These tools correspond to specific `REST API` endpoints exposed by *Eureca* and are used to access university data.

Depending on the complexity of the query and the structure of the data, agents can employ RAG techniques based on information obtained from different endpoints of the same API. For example, one endpoint might provide degree or subject codes, which are then used as parameters in another endpoint to retrieve the associated knowledge content. This structured data is incorporated into the prompt, ensuring more accurate and contextualized responses, without relying solely on the model's internal knowledge.

The EurecaChat RAG system is built using a Sentence Transformer model (`all-MiniLM-L6-v2`) to generate embeddings from university data, database structure descriptions, degree names, abbreviations, and others. These are provided as contextual data to agents at runtime, along with user queries. Embeddings are also generated for user input. Cosine similarity is then applied by comparing the user query embedding with embeddings from the knowledge base. Matches are considered relevant when the similarity score exceeds a threshold. Tests with names and common degree or subject abbreviations showed that 0.7 was an optimal threshold.

Hence, the RAG approach helps the agent understand user needs and select the appropriate tools and API endpoints based on the user query. For example, to improve robustness against errors in user input—such as misspellings or non-standard terminology—some tools implement a RAG-based *normalization layer*. To illustrate, if a user inputs "computer science" and the official degree name in the database is "Computer Science - Bachelor", the normalization component helps match this variation to the correct value. Without this step, the strict value-matching required by many endpoints would cause the query to fail.

Furthermore, to ensure secure and context-sensitive responses, EurecaChat's agents comply with the access control policies implemented in the *Eureca* API. In addition to open endpoints, the API provides an authentication mechanism through which users log in with their institutional credentials. Upon successful authentication, the API issues a token tied to the user's profile (i.e., student, faculty, or staff), restricting queries to data permitted at that level. For instance, students can only access their own records and not personal information of professors or peers. All authorization checks are handled by the *Eureca* API.

2.1 User query example execution

To better illustrate how a user query is processed by the system, consider the following example:

> *"From which state do the Computer Science program students from the second campus come? Display them by state."*

Upon receiving this query, the Supervisor Agent identifies the user's intent—a request for demographic aggregation over a filtered student cohort—and delegates the task to the **Student Agent**, which is responsible for handling student-related queries.

The Student Agent first invokes a tool designed to retrieve general student records from the *Eureca* API, applying filters based on program name ("Computer Science") and campus ("second campus"). Given the potentially large volume of data returned—often comprising thousands of entries—this raw data is stored temporarily in an in-memory SQL database for efficient downstream processing.

To handle the user's request for state-wise aggregation, a natural language-to-SQL (`NL2SQL`) module is then activated. This component translates the original query into a structured SQL command that extracts the required information from the temporary database.

For this example, the SQL query would group student records by their home state and count the number of students per group. Next, the result of this query (i.e., a table of states with corresponding student counts) is then returned to the Student Agent. The response is passed back to the Supervisor Agent, which routes it to the Aggregator Agent for final formatting and presentation to the user in natural language.

Note that this form of SQL-based processing is employed only when the volume or structure of data fetched from the API is too large or complex to be processed reliably within the context window of the language model. This mechanism prevents LLM hallucinations. In contrast, for queries involving smaller or more targeted data, SQL translation and temporary storage are not required. In such cases, the specialized agent can retrieve and return information directly from the *Eureca* API using simpler tool calls.

2.2 Technology details

EurecaChat was developed using a combination of modern technologies tailored for multi-agent systems and LLMs. For agent implementation and inter-agent communication, the system leverages **LangChain** and **LangGraph**[3], two Python libraries designed to provide a robust framework for agent behavior modeling, communication protocols, tool invocation, and integration with RAG techniques.

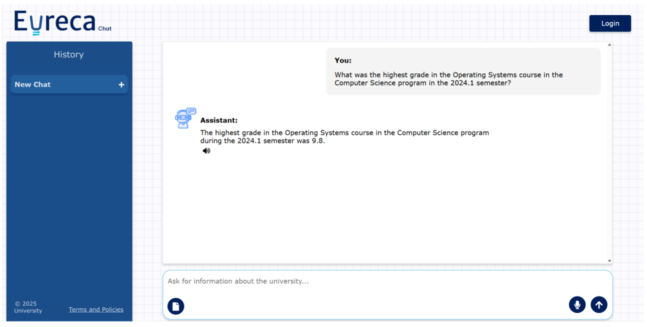
Furthermore, to support the management of large volumes of information—especially in scenarios where agents need to process extensive datasets without compromising accuracy or inducing hallucinations—a SQLite[4] database was used. This allows for temporary data storage and structured querying when necessary. Moreover, the user interface (frontend) was developed using HTML, CSS, and JavaScript to ensure an interactive and responsive user experience. The backend was built using the Quart Framework[5], providing RESTful endpoints and managing the communication between the frontend and agents.

In addition, the access to LLMs was facilitated through DeepInfra[6], a cloud-based platform that offers scalable access to cutting-edge language models via API. Two different models were used to optimize the system's performance for distinct roles. The Meta LLaMA 3.3 Instruct model (70 billion parameters) was employed by both the Supervisor Agent and the Aggregator Agent, as it showed superior performance in coordination and response synthesis tasks. For the specialized agents, the Qwen3 model (14 billion parameters) was selected due to its efficiency and reliability in executing tool-based operations.

3 USE CASE

This section presents several use case scenarios of EurecaChat, illustrating its effectiveness in supporting daily tasks carried out by university personnel and students. The objective is to showcase the capabilities and demonstrate how it can address the specific needs of different types of users. Users can initiate a chat by typing or speaking their questions. Also, users have the option to log in to access private data, if necessary. This allows them to retrieve sensitive information that should not be publicly available.

Figure 2 illustrates an interaction in which a student is looking for prerequisite information on a specific university subject they are interested in enrolling in. Additionally, Figure 3 shows an example of a Professor using the system to retrieve information about student grades for a particular subject.



You:

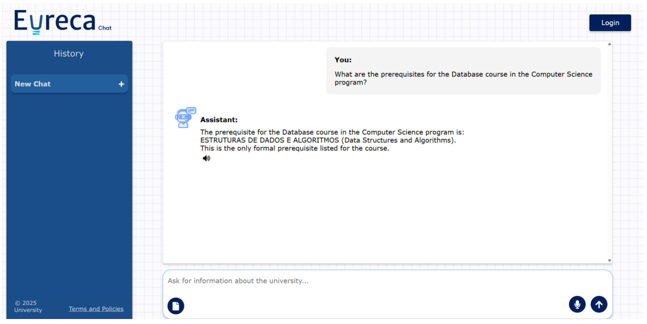
What was the highest grade in the Operating Systems course in the Computer Science program in the 2024.1 semester?

Assistant:

The highest grade in the Operating Systems course in the Computer Science program during the 2024.1 semester was 9.8.

Ask for information about the university...

Figure 2: Student query example (Information about university subject)



You:

What are the prerequisites for the Database course in the Computer Science program?

Assistant:

The prerequisite for the Database course in the Computer Science program is:
ESTRUTURAS DE DADOS E ALGORITMOS (Data Structures and Algorithms).
This is the only formal prerequisite listed for the course.

Ask for information about the university...

Figure 3: Professor query example (Information about grades in a specific subject)

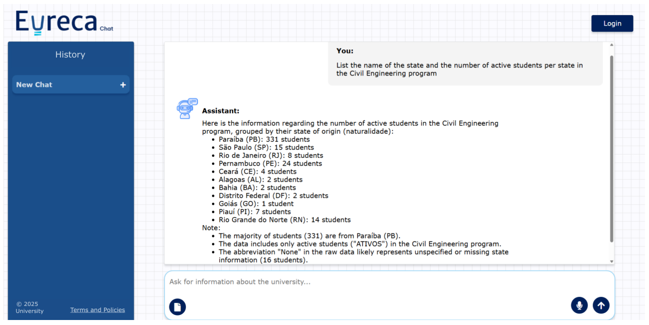
Finally, Figures 4 and 5 show a conversation likely initiated by an administrative staff member who is attempting to retrieve detailed personal information about students enrolled in a specific academic program. The response consists of text alongside a bar graph.

4 PRELIMINARY EVALUATION

A preliminary evaluation assessed EurecaChat's accuracy and reliability using 43 real-world queries submitted by students, professors, and administrative staff. Responses were double-reviewed for factual correctness against official university sources, with disagreements resolved by a third reviewer. Out of the 43 total queries, 36 were correct and 7 were incorrect, yielding a success rate of 84%, demonstrating a preliminary system's effectiveness. By group: coordination answered 12/15 correctly (80%), students 13/16 (81%), professors 10/11 (91%), and the administrative staff's single query was correct (100%).

Analysis of errors revealed three main types: Incomplete, where the response only partially addressed the query; System Error,

[3]https://www.langchain.com & https://www.langchain.com/langgraph
[4]https://sqlite.org/
[5]https://quart.palletsprojects.com/en/latest/
[6]https://deepinfra.com/



You:

List the name of the state and the number of active students per state in the Civil Engineering program

Assistant:

Here is the information regarding the number of active students in the Civil Engineering program, grouped by their state of origin (naturalidade):

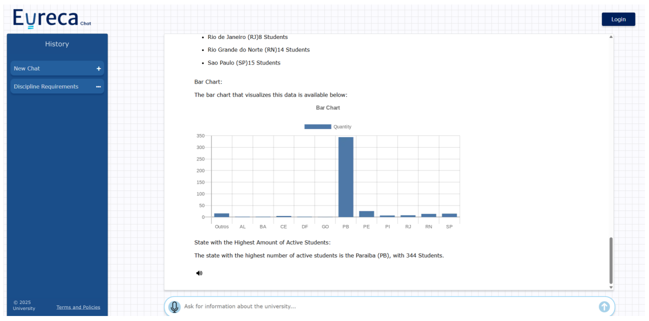
- Paraíba (PB): 331 students
- São Paulo (SP): 15 students
- Rio de Janeiro (RJ): 8 students
- Pernambuco (PE): 24 students
- Ceará (CE): 4 students
- Alagoas (AL): 2 students
- Bahia (BA): 2 students
- Distrito Federal (DF): 2 students
- Goiás (GO): 1 student
- Piauí (PI): 7 students
- Rio Grande do Norte (RN): 14 students

Note:

- The majority of students (331) are from Paraíba (PB).
- The data includes only active students ("ATIVOS") in the Civil Engineering program.
- The abbreviation "None" in the raw data likely represents unspecified or missing state information (16 students).

Ask for information about the university...

Figure 4: Staff query example (General information about students enrolled in a specific academic program - Part I)



- Rio de Janeiro (RJ)8 Students
- Rio Grande do Norte (RN)14 Students
- Sao Paulo (SP)15 Students

Bar Chart:

The bar chart that visualizes this data is available below:

State with the Highest Amount of Active Students:

The state with the highest number of active students is the Paraiba (PB), with 344 Students.

Ask for information about the university...

Figure 5: Staff query example (General information about students enrolled in a specific academic program - Part II)

caused by tool invocation issues, API malfunctions, or timeouts; and Hallucination. Most issues were attributable to tool configuration, data normalization, or API constraints, which can be mitigated through improvements in the RAG pipeline, data mappings, and error-handling mechanisms.

Despite these encouraging results, the evaluation remains limited in scope. The small sample size and lack of diversity question categories make it difficult to generalize findings. Thus, more extensive and diversified testing is needed to evaluate EurecaChat's effectiveness and ensure robustness under broader real-world conditions.

5 CONCLUSION AND FUTURE WORK

EurecaChat emerges as a versatile and powerful tool that supports diverse academic use cases, assisting students, faculty, and staff in routine tasks while improving efficiency. Its architecture combines state-of-the-art technologies—LLMs, multi-agent systems (MAS), retrieval-augmented generation (RAG), and integration with institutional information systems—transforming it from a simple query interface into a context-aware academic assistant.

Preliminary evaluations yielded encouraging results (84% accuracy), validating the system's initial effectiveness while also uncovering key limitations. Tests with real-world queries revealed edge cases and constraints that provide a road map for future improvements. Ongoing work will address these limitations by extending API integrations, introducing guardrails, and thus enhancing performance, robustness, and response accuracy. Broader user evaluations will further refine the system and ensure alignment with real academic needs.

In sum, EurecaChat represents a significant step forward in applying MAS and LLM-based technologies to the educational domain. By making academic data more accessible, not only strengthens decision-making within universities but also contributes to the broader democratization of public data.

6 LICENSE AND ACKNOWLEDGMENTS

The software described in this article is licensed under the Java Research License (JRL). Also, this work was funded through the **Agents4Good project**, a collaboration between **Kunumi**[7] and **the EMBRAPII UFCG-CEEI**.

REFERENCES

[1] Alaa Aloqayli and Hoda Abdelhafez. 2023. Intelligent Chatbot for Admission in Higher Education. *International Journal of Information and Education Technology* 13, 9 (2023), 1348–1357. https://doi.org/10.18178/ijiet.2023.13.9.1937

[2] Pica Atmauswan and Akibu Abdullahi. 2022. Intelligent Chatbot For University Information System Using Natural Language Approach. *Albukhary Social Business Journal* 3 (12 2022), 6. https://doi.org/10.55862/asbjV3I2a007

[3] S. Bieletzke. 2023. AI-CHATBOT-INTEGRATION IN CAMPUS-MANAGEMENT-SYSTEMS. In *EDULEARN23 Proceedings* (Palma, Spain) *(15th International Conference on Education and New Learning Technologies)*. IATED, 3574–3583. https://doi.org/10.21125/edulearn.2023.0971

[4] Omar Chamorro-Atalaya, Madison Huarcaya-Godoy, Víctor Durán-Herrera, Constantino Nieves-Barreto, Raul Suarez-Bazalar, Yreneo Cruz-Telada, Ronald Alarcón-Anco, Hilda Huayhua-Mamani, Ademar Vargas-Diaz, and Denisse Balarezo-Mares. 2023. Application of the chatbot in university education: A systematic review on the acceptance and impact on learning. *Int. J. Learn. Teach. Educ. Res.* 22, 9 (Sept. 2023), 156–178.

[5] Hoa Dinh and Thien Khai Tran. 2023. EduChat: An AI-Based Chatbot for University-Related Information Using a Hybrid Approach. *Applied Sciences* 13, 22 (2023). https://doi.org/10.3390/app132212446

[6] Igor Estrela, Kleydson Barbosa, Claudiny Brito, and Carlos S. Neto. 2024. Ferramenta CosmoBot: Um Chatbot de Apoio a Alunos em Avaliações de Algoritmos. In *Anais do XXXV Simpósio Brasileiro de Informática na Educação* (Rio de Janeiro/RJ). SBC, Porto Alegre, RS, Brasil, 380–391. https://doi.org/10.5753/sbie.2024.242585

[7] Debmitra Ghosh, Sayani Ghatak, and Hrithik Paul. 2023. A Proposed Cognitive Framework Model for a Student Support Voice based Chatbot using XAI. *Research Square* (2023). https://doi.org/10.21203/rs.3.rs-2888180/v1

[8] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large Language Model based Multi-Agents: A Survey of Progress and Challenges. arXiv:2402.01680 [cs.CL] https://arxiv.org/abs/2402.01680

[9] Nikita Kanodia, Khandakar Ahmed, and Yuan Miao. 2021. Question Answering Model Based Conversational Chatbot using BERT Model and Google Dialogflow. In *2021 31st International Telecommunication Networks and Applications Conference (ITNAC)*. 19–22. https://doi.org/10.1109/ITNAC53136.2021.9652153

[10] Qiaomu Li, Ying Xie, Sumit Chakravarty, and Dabae Lee. 2024. EduMAS: A Novel LLM-Powered Multi-Agent Framework for Educational Support. In *2024 IEEE International Conference on Big Data (BigData)*. 8309–8316. https://doi.org/10.1109/BigData62323.2024.10826103

[11] José Rodrigues Neto, Péricles Miranda, Rafael Mello, and André Nascimento. 2022. Chatbot to Support Frequently Asked Questions from Students in Higher Education Institutions. In *Anais do XIX Encontro Nacional de Inteligência Artificial e Computacional* (Campinas/SP). SBC, Porto Alegre, RS, Brasil, 591–601. https://doi.org/10.5753/eniac.2022.227553

[12] Neelkumar P. Patel, Devangi R. Parikh, Darshan A. Patel, and Ronak R. Patel. 2019. AI and Web-Based Human-Like Interactive University Chatbot (UNIBOT). In *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*. 148–150. https://doi.org/10.1109/ICECA.2019.8822176

[13] Michael Tamascelli, Olivia Bunch, Blake Fowler, Maryam Taeb, and Achraf Cohen. 2025. Academic Advising Chatbot Powered with AI Agent. In *Proceedings of the 2025 ACM Southeast Conference* (Southeast Missouri State University, Cape Girardeau, MO, USA) *(ACMSE 2025)*. Association for Computing Machinery, New York, NY, USA, 195–202. https://doi.org/10.1145/3696673.3723065

[7] https://www.kunumi.com/br