POLITECNICO DI BARI

DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING
Master Degree in Computer Engineering

Dissertation in Artificial Intelligence

Development of the Image Analysis Module of a Personal Assistant with Knowledge-Aware Reasoning Capabilities

Supervisor
Prof. Vito Walter Anelli

Candidate
Alessio Mattiace

Academic Year 2024 - 2025

Abstract

Current personal assistants are unable to answer complex queries that require cross-referencing user data fragmented across different apps and databases, such as photos, events, notes, etc. This thesis work proposes an architecture to overcome this limitation, centered on a Personal Knowledge Graph (PKG) that dynamically unifies personal data sources into a single database and can be queried very simply. Furthermore, an advanced Visual Analysis module is proposed to enable complex visual questions about images.

An orchestrator equipped with an LLM populates the graph with a single user's personal data, while a system of agents can query it using GraphRAG to provide a structured context to the LLM, anchoring responses to real data and mitigating hallucinations. If the user's question necessitates advanced reasoning over one or more images, the Visual Analysis module would be invoked to improve the system's performance. The evaluation demonstrates that visual analysis effectively fills informational gaps (over 82% positive responses) and that the multimodal integration is robust.

keywords: Personal Assistant, Personal Knowledge Graph, Visual Analysis, RAG

Contents

1 Introduction 1

2 State-Of-The-Art 5
- 2.1 Personal assistants, Knowledge Graphs e Personal Knowledge Graphs 5
- 2.2 PKG population 9
- 2.3 PKG utilization 12
- 2.4 Visual Analysis 15

3 Personal assistant architecture design 19
- 3.1 Architectural modules overview 19
- 3.2 Client architecture 25
- 3.3 EPANSA orchestrator architecture 30
  - 3.3.1 Extraction logic implementation 31
  - 3.3.2 Post-processing implementation 36
  - 3.3.3 Chat logic implementation 38
- 3.4 Chatbot test results 40

4 Visual Analysis Module 49
- 4.1 ReAct agent architecture 50
- 4.2 graphRAG img retrieval 55
- 4.3 visual reasoning e question answering 59

5 Testing and results discussion 63
- 5.1 Evaluation protocol 63
- 5.2 Test results 71
  - 5.2.1 PRIMARY VISION DATASET (PVD) results 71
  - 5.2.2 REDUNDANT VISION DATASET (RVD) results 78
- 5.3 Results Discussion 85
  - 5.3.1 PRIMARY VISION DATASET (PVD) results discussion . . 85
  - 5.3.2 REDUNDANT VISION DATASET (RVD) results discussion 91

Bibliography 103

Chapter 1

Introduction

In the contemporary technological landscape, we are observing a rapid evolution of conversational AI tools, which have radically transformed the way we interact with digital devices. Personal assistants have become increasingly powerful and, consequently, more present in our lives. They can execute commands and responding to both user-specific and more "generic" queries based on personal data with increasing efficiency. However, these systems still operate within a limited paradigm: They cannot answer questions that require complex, cross-referenced reasoning across multiple types of user data such as photos, events, notes, alarms, etc. Their knowledge is not fully contextualized, and their functionalities are still reduced, resulting in a poor ability to reason about the user's experiences and daily life, lacking an effective understanding of events and habits.

The true frontier of personal assistance, in fact, lies in quickly and easily providing answers to deeply personal questions, queries that today would require a manual search through a labyrinth of applications and functions on our smartphone or computer. Questions like "Do I already have commitments for next Saturday?" or "Did Sara call me before I arrived at work last Monday?" currently remain unanswered. In our digital lives, information is fragmented into isolated databases: calendar events, archived photographs, hastily written notes, work documents, call history, and address book contacts coexist without communicating. A unified system is missing that can collect this data and reason about it, understanding the implicit connections between events, people, places, etc.

The objective of this work is, therefore, to equip the assistant with the ability to answer more complex, yet extremely useful, everyday questions, queries that require a deep synergy between different sources of knowledge. An emblematic example is a question like: "What was I wearing at Marco's birthday?" The system could first identify the "Marco's birthday" event in the calendar to establish a temporal period, and subsequently retrieve the photographs taken during that time; finally, by activating a visual analysis module to inspect the content of those images, it could identify the person's clothing. This is not simple object recognition; it is multimodal reasoning that merges the structural knowledge of the personal graph with the direct perception of visual content.

To solve the challenges of this task, an architecture has been designed that utilizes specific technologies: the unique data source is the Personal Knowledge Graph (PKG). This concept represents an evolution of traditional Knowledge Graphs; while a generic KG models factual knowledge of the world through "triples" (subject-predicate-object), a PKG adopts this same structure to represent the private universe of a single user. The distinctive feature of a PKG is its focus on the individual: it places the user at the center of all connections and includes entities that are relevant only to them, such as friends, colleagues, personal items, or workplaces. These "non-famous" entities would never appear in a global knowledge base. The PKG thus transforms fragmented personal data into a coherent, navigable, and queryable knowledge network.

The construction of this personal graph occurs in parallel with the addition of new data on the device, hence dynamically; it is an aggregation from heterogeneous sources, both structured (like alarms, contacts, events, or calls) and unstructured (like documents, photos, or notes). The system processes these inputs differently: for already structured data, the transformation into graph nodes and relationships is a deterministic and simpler process, while for unstructured data, specialized modules come into play to detect the useful textual content to be inserted into the PKG. These components, based on the nature of the data, perform various operations such as reading textual content, segmentation into "chunks," textual description of an image, etc.

A Large Language Model (LLM) is then used to extract entities and relationships in the form of triples: the LLM does not operate freely but is constrained by a predefined schema and "disciplined" in populating the graph with coherent facts. For images, a visual description module first generates a textual summary, which is then processed by the LLM to extract significant triples, thus linking the photo to concepts like places or people. This factual base is further enriched by a post-processing phase that calculates vector embeddings and identifies, through community detection algorithms, thematic clusters within the graph, making the knowledge not only stored but also organized and ready for efficient retrieval.

Once the PKG is built, querying is done through an advanced Retrieval-Augmented Generation (RAG) methodology, specifically an evolution known as GraphRAG. This approach overcomes the limitations of traditional RAG, which often relies only on the semantic similarity of isolated text fragments. GraphRAG, instead, leverages the intrinsically connected nature of the graph; when the user asks a question, the system retrieves not just single nodes, but entire contextual subgraphs, multi-hop reasoning paths, and the most relevant thematic communities. This enriched and structured context, which represents a verifiable portion of the user's knowledge, is then provided to the LLM. The latter is constrained to formulate its response based exclusively on this evidence, drastically mitigating the risk of "hallucinations" and ensuring that every statement made by the assistant is faithful and anchored to the user's real data.

It is within this solid textual architecture that the Visual Analysis module is situated, designed not as a standalone component but as a specialized tool available to a larger decision-making agent. The entire reasoning flow is governed by an agent built according to the ReAct (Reasoning and Acting) paradigm, implemented via LangGraph. This agent acts as an intelligent coordinator, a state machine that, by analyzing the user's intent, decides which tool to invoke from its arsenal, which includes GraphRAG for textual queries and, indeed, the img_deep_analysis tool. Naturally, visual analysis is not the first choice, but a fallback strategy: the agent first attempts to answer using GraphRAG, and only if this graph-based search fails, does the agent decide to invoke the visual tool, recognizing that the answer might reside in the images.

This visual analysis process is divided into two macro-phases. The first is intelligent image retrieval: to avoid inefficiently analyzing the entire archive, the system reuses GraphRAG in a specialized mode. Instead of generating text, GraphRAG is instructed to return only a list of photo identifiers (photo_id) semantically relevant to the question. The second phase is the visual reasoning itself: the system takes the selected identifiers, resolves them into concrete file paths, and passes them, one by one, to a large Vision-Language Model (VLM). Here, the VLM does not perform a simple description but receives a specific prompt asking it to answer the user's original question based exclusively on the content of that single image. Positive responses are finally aggregated and returned to the ReAct agent, which synthesizes them for the user, completing the reasoning cycle.

To validate the effectiveness of this complex multimodal architecture, a rigorous evaluation protocol was defined. All experimentation was conducted by simulating a realistic use case, based on a single user and a unique consolidated PKG, with all interactions anchored to a fixed temporal instant (Monday, September 1, 2025) to ensure the reproducibility and consistency of temporal reasoning. Two distinct datasets were developed to measure different aspects of the system. The first, the Primary Vision Dataset (PVD), consisting of 29 questions, was designed to isolate and measure the added value of the visual analysis module in cases where GraphRAG alone was destined to fail. These are questions that require the identification of purely visual details, such as colors, patterns, or counting people, information not present in the textual graph. The second, the Redundant Vision Dataset (RVD), consisting of 14 questions, was instead conceived to test the robustness of the integration in scenarios where GraphRAG already possessed the necessary information. The goal here was to observe whether the intervention of the visual component improved, worsened, or left unchanged an already potentially correct answer, thereby evaluating the quality of the multimodal integration.

The effectiveness of this system is evident: in the Primary Vision Dataset, where only the visual module could answer, over 82% of the responses were judged

"Excellent" or "Good" by human evaluation. This demonstrates that the visual module is capable of effectively filling informational gaps in the graph, although a 14% failure rate indicated limits in perceiving very small details. The results on the Redundant Vision Dataset were equally encouraging: in 50% of cases, the visual contribution was neutral, confirming the coherence between textual and perceptual data without introducing noise; in 36% of cases, it led to a clear improvement, correcting conceptual errors in the graph or adding relevant details; and only in 14% of cases did it degrade the quality of the response, which suggests that the architecture is robust.

The work, therefore, not only outlines an architecture for a more capable personal assistant but also demonstrates the feasibility and effectiveness of a system that learns, remembers, and reasons in a complex way about the user's personal and visual world.

Chapter 2

State-Of-The-Art

This chapter analyzes the state of the art in the field of personal assistants, ranging from the methodologies for knowledge representation available to them, to the techniques employed for question answering and reasoning that enable user interaction.

In recent years, conversational artificial intelligence has shown a remarkable evolutionary path, enabling the transformation of personal assistants from basic tools that reacted passively to given commands by following simple rules, to sophisticated digital collaborators. Advances in the field of LLMs and in knowledge representation itself have been essential in charting this new trajectory in the scientific field. Among the most widely used representation methodologies are Knowledge Graphs (KGs), which can represent a very powerful tool and a pillar of factual knowledge for any conversational system.

A further development of KGs is represented by Personal Knowledge Graphs: the latest frontier of personalization. They build upon the core concept of a Knowledge Graph while providing detailed information relevant to a single user.

2.1 Personal assistants, Knowledge Graphs e Personal Knowledge Graphs

Personal assistants An AI-based Personal Assistant (PA) is intelligent software that can understand natural language and execute complex tasks on behalf of a human user. Well-known examples like Amazon Alexa, Siri, and Google Assistant have completely changed the way we perceive human-machine interaction. However, the effectiveness of these systems has always had to contend with major historical limitations related to the intrinsic complexity of Natural Language Processing (NLP). Current research focuses on three key issues that could transform PAs into systems truly useful in daily life: long-term memory, user modeling, and proactive assistance.

One of the biggest challenges in conversational AI has been its "goldfish memory," i.e., the inability to retain information across different sessions (a "stateless" nature). Innovative architectures have been proposed to overcome this limitation. The work by Yuan et al. [36] introduces a framework that gives the LLM an external evolving memory; the main innovation is the "Conditional Memory," which leverages the LLM itself to filter conversational noise and retain only the relevant information by separating the "fact" to be learned and memorized, and the surrounding dialogue that explains its context. Subsequently, Wang et al. [31] addressed the problem of the limited "context window" of LLMs with the LONGMEM framework; it uses a "frozen" LLM to encode memory and a trainable SideNet for data retrieval from the cache. In this way, it solves the problem of memory obsolescence that occurs when a model's internal representations change during training.

In the context of user modeling, attempts have been made to increase personalization by building a "user model" that goes beyond the simple retrieval of "relevant" facts; in fact, USER-LLM by Ning et al. [19] is a framework that aims to overcome the view of interaction history as a series of information to be inserted into the context via a prompt, but rather as a distinct modality. A dedicated "User Encoder" distills this history into "compact user embeddings," which are subsequently integrated directly into the LLM architecture, providing a contextual overview that acts as a kind of summary of user preferences.

An even more sophisticated approach is presented by Qiu et al. [22] with their Difference-Aware Personalization Learning (DPL) method; it constantly and actively compares the target user with other users in a scalable way, moving beyond the simple "isolated" view of the user. This allows for the identification of characteristics that are distinctive compared to other profiles ("differential profile"), creating a much more meaningful context that maximizes personalization during the LLM's responses.

The ultimate goal of personal assistants remains to proactively assist the user: according to Janagama [10], this means anticipating their real needs; this research highlights how an AI's ability to simulate empathetic behavior and understand emotional cues is important for establishing trust and a positive relationship (based on Mind Perception Theory and Parasocial Relationship Theory). The PaRT framework by Niu et al. [20] was built following this principle: PaRT implements a chatbot based on a user profiling module that builds a detailed profile, an intent-driven "query refiner" that analyzes the conversation to understand when is the moment to intervene proactively, and a Retrieval-Augmented Generation module that retrieves useful and relevant information in real time to be included in the conversation context.

In summary, the trajectory that personal assistants are following is clear: they are abandoning the idea of executing isolated commands and adopting techniques to leverage a persistent memory that makes the user a unique and complex entity, acting accordingly.

Knowledge Graphs The Knowledge Graph (KG) is a structured representation of knowledge used in various research fields to base reasoning on consistent and verifiable "facts"; the field of personal assistants is one of those where they can play a decisive role. The basic elements of a KG are the so-called "triples," which are subject-predicate-object relations that allow a given concept to be factually asserted.

Among the strengths of this representation is the reliability and navigability of the facts stored in such a system, which would allow personal assistants equipped with advanced AI to overcome the limits of purely statistical knowledge learned from textual data. KGs, ultimately, provide the necessary "groundings" for personalized, accurate, and verifiable answers, enabling complete and more complex reasoning. KGs have proven essential in specific contexts such as a dynamic and adaptive system; for example, the work of Kumar et al. [11] demonstrates how to integrate KG, LLM, and real-time event data to provide personalized responses. Here, the KG is a true "central nervous system" that provides all knowledge related to the individual's profile (e.g., geographical location, profession, or interests) in the form of triples and is aided by supplementary components such as a module for extracting real-time events to link to the existing info in the KG, and the LLM which enables the actual interaction by summarizing the question and adding everything useful to increase personalization and user experience to the context.

AGENTiGraph by Zhao et al. [38], on the other hand, is a platform that attempts to solve a well-known problem in this field: that of accessing the KG by non-expert users by allowing natural language queries. The framework transforms the user's questions into formal queries executable on the graph through a complex multi-agent system to obtain all useful information even without the need for real technical knowledge. This is a big step towards a more "democratic" interaction between the personal assistant and the user.

The effectiveness of Retrieval-Augmented Generation based systems, however, depends heavily on the quality of the retrieved data, as Mukherjee et al. [18] understand how the actual enhancement of personal assistants depends primarily on how the KG is constructed: this means that the correct construction of a KG from high-quality documents significantly improves the RAG's performance, as it is based on the vector similarity of text chunks. The KG-RAG framework is therefore introduced, which implements an advanced pipeline that includes assigning confidence scores to every "fact" extracted from the documents and tracking the provenance of each of them: this ensures noise reduction and an increase in reliability since, for every question, the system retrieves pertinent triples that are above all "verifiable." KGs, therefore, are a tool that acts as a bridge between unstructured language and structured databases that are difficult for LLMs to interpret: in effect, their use proves very useful, becoming the backbone of a personal assistant that needs fluent, but also factual and verifiable, conversations.

Personal Knowledge Graphs In recent years, KG research has expanded its horizons, leading to several specializations: one of these is Personal Knowledge Graphs. They differ from simple KGs based on the type of concepts they contain: PKGs contain personal and individual information for a specific user, and this allows a shift of focus from large general and collective knowledge bases to a personal one centered on a specific individual.

The implications of this paradigm shift are significant as they represent a new frontier of knowledge that places the user at the center of their digital universe; the main characteristics are outlined by Balog and Kenter [3] in their pioneering work that defines PKGs as "structured information resources about entities personally relevant to the user, including those that may not be globally important."

This definition, in the paper, is followed by two key concepts:

- The importance that "non-famous" entities, not present in known databases, could have for the user, such as personal objects, friends, or workplace.
- The "spiderweb" structure with the user always at the center and where all information is connected directly or indirectly to them.

Skjæveland et al. [26] add the fundamental concepts of ownership and control of personal data to this initial definition; in their vision of the PKG concept, the graph must be owned by the user, who has full read and write access, as well as all rights to grant this data to third parties. This very concept deeply distinguishes a PKG from a simple and generic KG. The authors also specify that the PKG concept is totally separate from that of a Personalized Knowledge Graph, which typically refers to a subset of a larger KG, curated and managed by an external service to reflect a user's interests, but without the user having full control over it.

The creation of a PKG is a complex process of aggregation from heterogeneous sources; these include private data (emails, calendars, local files), public data, and so-called "digital lifelogs" such as search history, social media posts, and wearable device data. However, the transformation of this raw data into structured knowledge requires sophisticated information extraction and entity linking pipelines. An important example of extracting this data from conversations between a user and an assistant is in the work of Li et al. [14]: they developed methods to extract personal facts (e.g., "my mother's name is Rosa") from conversational dialogues through assertion classification, relation detection, and slot filling.

Subsequently, regarding the management of the PKG itself, there are several challenges because personal data is intrinsically dynamic: PKGs require continuous bidirectional synchronization mechanisms with external data sources [3]. Furthermore, the temporal nature of a person's life must be modeled. This ranges from simple approaches, such as adding temporal validity intervals to relationships [5], to much more advanced methods like those proposed by Sansen et al. [23], which model the "fuzzy dates" typical of human memory (e.g., "about 10 years ago") using Gaussian functions, allowing for more flexible and natural temporal reasoning.

Personal Knowledge Graphs mark the decisive turning point toward an artificial intelligence truly at the user's service. Thanks to them, we move from being simple users of technology to becoming active managers of our digital knowledge heritage. Despite all the notable technical, ethical, and privacy difficulties, PKGs represent the foundations upon which we will build the next generation of personal assistants. We are talking about intelligent and proactive systems, capable of integrating deeply with our lives, of understanding us, supporting us, and evolving almost in symbiosis with us.

2.2 PKG population

The creation of a PKG (Personal Knowledge Graph) from structured and unstructured personal data of all types is a process known as PKG population; it consists of converting the chaotic, fragmented flow of personal data into a coherent and queryable knowledge network. Unlike the construction of large public knowledge graphs, which rely on stable and well-defined corpora like Wikipedia, populating a PKG is an intrinsically more complex and dynamic activity. It must draw upon private, heterogeneous, contextual, and continuously evolving information, which constitutes an individual's unique digital footprint.

Data Collection Mechanisms The process of building a PKG is an act of aggregation as it involves drawing from the vast and varied range of data streams that make up an individual's digital life. A useful initial classification of data sources, as proposed by Skjæveland et al. [26], divides them into three main categories: private data sources, public sources, and Linked Open Data. Private sources include all information to which only the PKG owner has access, such as personal files, emails, calendars, and chat histories. Public sources include any information accessible to everyone that may be relevant to the user, such as nutritional tables or health policies. Finally, Linked Open Data, such as DBpedia or Wikidata, represent already structured public knowledge graphs, which facilitates their integration and interconnection with the PKG.

The work of Chakraborty and Sanyal [6] illustrates how information can be extracted from a user's digital "lifelogs," which include a wide range of activities: social media posts, search histories, data from wearable devices, and conversations with voice assistants. User conversations represent an extremely rich but also challenging source; they are often unstructured, colloquial, and full of ambiguity, making the extraction of clear and concise facts difficult. Similarly, data stored in the system, such such as contact lists or schedules, offer already partially structured information, but even the structure of files and folders on a computer can be viewed as a personal classification scheme that reflects the user's habits.

The healthcare domain offers a powerful and concrete example of how these different sources can be collected and integrated. In their project for a Personal Health Knowledge Graph (PHKG), Hendawi and Li [9] outline a "collection" phase that draws upon a multitude of inputs. It starts with Electronic Health Records (EHRs), which contain formal medical history, and then enriches them with real-time data from IoT sensors and wearable devices, which track parameters like heart rate, blood pressure, and sleep patterns. Data from personal health apps, for managing diet or medication, are added to these. Crucially, their approach also includes parameters like economic status and education level, to create a truly comprehensive view of an individual's well-being.

This view is further refined by other researchers like Wang et al. [32], who emphasize an important distinction in how different types of data are managed. They propose that dynamic and real-time data, such as the continuous flow of information from a sensor, be managed by a time-series database optimized for rapid acquisition, while more static data, like past medical history, are more suited to the structure of the main knowledge graph.

At the same time, Lee and Song [12] offer a critique of the source of clinical information, comparing Electronic Health Records (EHRs), managed by healthcare institutions, with Personal Health Records (PHRs), managed directly by the patient. EHRs are created only during medical visits, while PHRs allow for continuous data collection, putting the patient at the center of the collection process and overcoming the limitations of traditional sources.

The versatility of these approaches is also demonstrated in completely different domains, such as the academic one; Chakraborty, Dutta, and Sanyal [5], in fact, illustrate how sources and collection mechanisms can be personalized to create a Personal Research Knowledge Graph (PRKG). Here, sources include the researcher's curriculum vitae, information collected through "direct elicitation" (a conversational agent asking questions about ongoing projects), and "passive tracking" (the analysis of downloaded papers or search queries). This hybrid collection model, which combines manual curation, automatic extraction from documents, and dynamic, active interaction, demonstrates how populating a PKG is a process adapted to the specific context of each user.

Information Extraction and Entity Linking Once the raw data has been collected, the most delicate and technically complex phase begins: its transformation into structured and interconnected knowledge. This process is divided into two main activities: Information Extraction and Entity Linking. The goal is to identify named entities (people, places, events), their attributes, and the relationships that connect them, and then map these elements to nodes and edges in the graph, correctly linking them to existing entities or creating new ones.

A pioneering approach in this field was that of Li et al. [14], who developed a statistical method to automatically build user-centered knowledge graphs from conversational dialogues. Their methodology is structured in three phases: first, Personal Assertion Classification filters user phrases to identify those containing personal facts. Subsequently, Relation Detection classifies these assertions into predefined categories (e.g., the "parent" relationship). Finally, Slot Filling precisely extracts the specific attributes (e.g., "name (parent): Rosa"). This complete pipeline, from phrase classification to semantic filling, laid the groundwork for populating PKGs from spoken language.

More recent systems, such as the one proposed by Vannur et al. [30], use even more complex pipelines, including Entity Recognition, Entity Classification, Relation Extraction, and, most importantly, Entity Resolution, a key task that ensures that different mentions of the same real-world entity (e.g., "mom" and "Rosa") are unified under a single identifier in the graph. Their approach leverages both rule-based annotators and Graph Neural Networks (GNNs) to infer missing relationships, demonstrating a comprehensive approach from information extraction to connection.

One of the most significant challenges is the dynamic nature of PKGs. Graphs are not static but constantly evolve with new conversations and information. The work of Mohanaraj and Laursen [17] directly addresses this problem with the concept of PKG Claim Linking. Their system must decide, for every new triple extracted from a conversation, whether the subject and object correspond to existing entities in the graph or if new ones need to be created. Their two-component architecture, a Personal Entity Classifier (PEC) and a Personal Entity Disambiguator (PED), is specifically designed to handle this challenge in an evolving conversational context.

Finally, it is important to recognize that complete automation is not always the optimal solution, especially when working with unconventional and noisy data sources. The "human-in-the-loop" approach proposed by Schröder, Jilek, and Dengel [24] for building PKGs from file names is a clear example. Their system uses heuristics to extract relevant terms, but the final decision on entity linking and creating new instances is guided by engineer feedback. This highlights the value of human supervision in the iterative PKG construction process, improving the accuracy of the extracted entities and relationships.

Schema Design and Data Standardization No population process, however sophisticated, can succeed without a solid upstream design: the graph schema. The schema acts as a conceptual model, defining the types of entities, attributes, and relationships that can be stored, ensuring that the collected information is organized in a coherent and meaningful way.

The need for detailed and domain-specific schemas was highlighted by Liu et al. [15] in their Question-Answering system (PKGQA) for mobile devices. They observed that existing schemas were either too generic or too coarse to capture the nuances of mobile personal data. To overcome this limit, they introduced a two-level granularity schema, centered on people and events, but with extremely detailed subcategories (e.g., twelve different types of relationship within the "Family" category). This level of detail is useful for allowing the system to answer complex questions.

Going beyond structure, data standardization is important for interoperability. Bernard et al. [4] propose a vocabulary for PKGs based on established web standards like RDF (Resource Description Framework). The use of RDF ensures that statements can be represented consistently and machine-readable. Furthermore, their approach integrates existing vocabularies such as SKOS for semantic relations and PAV for information provenance, a vital aspect for user privacy and control. Data quality validation is ensured by the use of SHACL (Shapes Constraint Language), which allows for defining structural rules for the graph.

In the healthcare domain, this need for standardization is even more critical; in fact, Hendawi and Li [9] base their PHKG schema on a consolidated medical standard, HL7 FHIR, and link their graph concepts to authoritative medical vocabularies via BioPortal. This ensures that the PKG is not an isolated data structure but a semantically interoperable system with the wider healthcare ecosystem, capable of integrating heterogeneous data into a unified and meaningful format.

PKG population, therefore, is an articulated and sophisticated process that goes far beyond simple data collection. It requires careful selection and integration of various sources, the application of advanced pipelines to accurately and dynamically extract and link information, and the design of detailed and standardized schemas for knowledge representation.

2.3 PKG utilization

After establishing the methods for the construction and completion of PKGs, the next step is to use them for reasoning and question answering processes. The main application in this area is Knowledge Graph Question Answering (KGQA), a discipline that aims to bridge the semantic gap between questions posed in natural language by users and the formal, symbolic nature of a knowledge graph.

The objective is no longer to query a universal knowledge repository but to dialogue with a structured representation of a user's individual reality. This chapter will trace the evolution of KGQA techniques, from traditional pipeline architectures to modern intelligent agent-based approaches, contextualizing how these technologies can be applied to effectively query a PKG and provide answers that are not only correct but also personalized, contextual, and reliable.

Taxonomy of KGQA Architectures The initial challenge in the KGQA research field is to translate an unstructured question into a formal query executable on a graph: a comparative analysis by Omar et al. [21] highlights the clear distinction between a traditional Question Answering System (QAS), such as their KGQAN, and a conversational AI model like ChatGPT. The traditional approach relies on a rigid pipeline: question understanding, linking entities and relationships to the KG, and finally executing a formal query like SPARQL. Although this method offers high precision and access to up-to-date data, it suffers from fragility and poor explainability. Conversely, the conversational approach based on LLMs is robust to input variations and offers a more human interaction, but it is limited by the static nature of its internal knowledge and a marked tendency towards hallucination.

Thus, research has moved towards hybrid systems: Sun et al. [28] propose a modern taxonomy for the integration of KGs and LLMs, defining two main paradigms based on the level of interaction between the components. The first, represents a weak coupling. In this model, typical of RAG systems, the KG acts as an external and passive knowledge source. The LLM translates the question into a query, executes it on the KG, and uses the results to generate the final answer. The second paradigm describes a tight coupling. Here, the LLM is not a simple translator but an intelligent agent that actively and iteratively explores the graph, using its reasoning capabilities to navigate nodes and relationships. This evolution, from rigid pipelines to interactive conversational models, marks a clear path towards increasingly powerful and flexible KGQA systems.

The "retrieve-then-reason" architecture has become the standard approach for mitigating LLM hallucinations, anchoring their responses to an external and verifiable source of truth like any KG. The KAPING framework, proposed by Baek, Aji, and Saffari [1], is an example of this. Its process begins with a "Knowledge Access step" where the entities mentioned in the question are linked to the KG and all associated facts (triples) are retrieved. Since this initial retrieval is often noisy, a "Question-Relevant Knowledge Retriever" filters these facts, selecting only those semantically most pertinent to the question via vector embeddings. These facts, verbalized in natural language, are finally injected into the LLM's prompt, which generates a response based on this enriched context. However, simple retrieval based on semantic similarity is often insufficient for complex questions that require multi-hop reasoning. For this, Sen, Mavadia, and Saffari [25] proposed a more sophisticated architecture, where the retriever is not a simple similarity module but one specifically trained for the KGQA task. This retriever learns to predict entire reasoning chains, identifying the relational paths necessary to answer complex questions. Once these chains of facts are retrieved, they are passed to the LLM, which acts as a zero-shot reasoner, interpreting the "facts" provided in the context to formulate the final answer.

Another line of research in the same KGQA field concerns how the information retrieved in the retrieval phase is presented to the LLM. Simple linearization of triples can create a noisy and semantically poor context. To overcome this limit, works like CoTKR (Chain-of-Thought Enhanced Knowledge Rewriting) by Wu et al. [33] integrate the principles of Chain-of-Thought (CoT) directly into the knowledge representation process. Instead of a single block of text, CoTKR generates a series of interleaving reasoning steps and pertinent knowledge fragments, decomposing the main question into logical sub-questions and verbalizing only the necessary facts to answer each one. This process produces a much more structured context aligned with the reasoning path.

If the RAG paradigm represents a significant step forward, it still maintains a clear separation between the KG (passive) and the LLM (interpreter). The agent-based paradigm, dissolves this separation, transforming the LLM into an active and intelligent explorer. The framework that theorizes this approach is Think-on-Graph (ToG), introduced by Sun et al. [28]. In ToG, the LLM does not receive a set of facts obtained from a previous retrieval phase but performs a beam search directly on the graph. The process is iterative: given a question, the LLM identifies the starting entities, explores promising relationships, retrieves the next step entities, and evaluates whether the collected information is sufficient to answer. This dynamic exploration allows the agent to actively navigate the PKG and use its internal knowledge to "jump" any missing links in the graph, combining the structured knowledge of the KG with the implicit knowledge of the LLM at each step. This approach was further extended in ToG-2, which allows the agent to draw not only from the KG but also from unstructured textual documents, enhancing hybrid reasoning.

A further example of this type of paradigm is Reasoning on Graphs (RoG) by Luo et al. [16], which structures the agent process into a "planning-retrieval-reasoning" pipeline. In the planning phase, the LLM generates an abstract "plan," meaning a path of relation types that could lead to the answer; subsequently, the retrieval module searches the KG for concrete instances of this path, and finally, the LLM reasons over these verified paths to produce the answer. This approach transforms reasoning into an explicit and planned process (it is no longer seen as a black box to the user).

Faithful and Verifiable Reasoning A personal assistant based on a PKG must not only be accurate but also and above all reliable; indeed, a correct answer generated for the wrong reasons is useless, if not harmful. For this reason, research is increasingly focusing on the faithfulness and verifiability of the reasoning process: an explanation is faithful if it "honestly" reflects the causal chain that led to the conclusion, without hallucinations. The RoG framework [16] takes the first step in this direction: the final reasoning of the LLM is constrained to the evidence retrieved according to an explicit plan, the process becomes transparent, and the chain of evidence is verifiable.

FiDeLiS by Sui et al. [27] builds on this concept by introducing a "Deductive-Verification Beam Search." In this agent-based framework, each path extension step of the reasoning is not only proposed by the LLM but must also pass a deductive verification check, performed by another instance of the LLM, which ensures that the new step follows logically and factually from the previous ones, guaranteeing that the entire reasoning path is based on KG facts and logically consistent from start to finish.

An even more rigorous approach is proposed by Toroghi et al. [29] with the Right for Right Reasons framework. Here, to answer a question, the LLM is first forced to generate an explicit, human-readable common-sense axiom (e.g., "If two locations are in countries without common border agreements, separate visas are required"). Subsequently, the system iteratively verifies every factual premise of this axiom against the KG. This method decouples common-sense reasoning (the axiom) from factual reasoning (verification on the KG), making every single step of the process completely transparent and verifiable.

An important contribution to this has been made by the Graph Retrieval-Augmented Generation (GraphRAG) framework, as while previous RAG architectures focused on transforming KG facts into simple textual triples, the GraphRAG approach explicitly recognizes and exploits the graph's relational structure to enhance both the retrieval and reasoning phases [8]. The authors emphasize how the intrinsic nature of a graph, comprising heterogeneous and relational information, makes it a very important resource during the retrieval phase.

Specifically, for querying a PKG, GraphRAG enables structural retrieval. The system does not limit itself to retrieving isolated facts based on semantic similarity but identifies and retrieves entire subgraphs or multi-hop reasoning paths most pertinent to the user's question thanks to the introduction of "super-nodes" called "Communities": this allows the LLM to reason over a much richer and more contextualized block of knowledge. Operating on these subgraphs allows for capturing the complex relationships between personal entities (e.g., professional projects linked to notes and calendar commitments), drastically reducing the risk of hallucination and ensuring that the answers are anchored to verifiable chains of evidence. The future direction for these frameworks sees the integration of advanced techniques, such as Graph Neural Networks (GNNs), to better encode and retrieve this context.

The field of KGQA, therefore, is undergoing a rapid transformation, thanks in part to the integration of LLMs, which has opened the way for paradigms that overcome the rigid pipelines of the past: enriched "retrieve-then-reason" techniques and the nascent agent-based approaches promise to provide more accurate and contextualized answers.

2.4 Visual Analysis

A sector of research to analyze for designing the key component of this architecture, namely the visual analysis module, is Visual Question Answering (VQA) or

Visual Reasoning. It delves into a complex topic like computer vision and natural language processing (NLP). The objective is to develop systems capable not only of "seeing" an image but of "understanding" it at a deep level, answering questions that require articulated reasoning processes. This ability ranges from simple object recognition to complex logical inferences, which may involve understanding spatial relationships, integrating external knowledge, and decomposing a problem into intermediate steps. The recent evolution of the sector has been driven by the rise of Large Multimodal Models (LMMs), which merge powerful visual encoders with the architecture of language models to achieve strong synergy.

Main Architectures Early architectures mainly focused on the fusion of visual and textual features. The first models explored different techniques to combine the vectors extracted from an image and a question, for example through bilinear pooling operations; although effective for basic tasks, these approaches showed limitations in managing complex interactions.

A significant step forward was made with the introduction of attention mechanisms, such as in MCAN [35]. They allowed the system to dynamically focus on the most relevant regions of the image based on the question, making it possible to obtain more accurate and contextualized answers. Attention then evolved into self-attention and co-attention mechanisms, which allowed for modeling long-range dependencies both within a single modality and between the two, paving the way for a more granular understanding of the visual scene.

In parallel, to tackle problems that required more structured reasoning, graph-based models emerged. Architectures like ReGAT [13] represent the image as a scene graph, where nodes correspond to objects and edges to their relationships. This approach allows the model to explicitly navigate the entities and their connections to build a chain of logical inferences, an ability useful for solving complex questions involving multiple objects and their interactions.

Subsequently, the advent of transformers and large-scale pre-training marked a turning point, giving rise to the Large Multimodal Models (LMMs) that now dominate the state-of-the-art. Models like Qwen2.5-VL [2] have raised the bar by integrating high-resolution visual encoders with powerful language models, dramatically improving the ability to merge vision and language for more sophisticated reasoning. This new generation of models does not just merge features but orchestrates the entire reasoning process.

An emblematic example of this evolution is Griffon-R [37], which employs a unified "understand-think-respond" mechanism. This approach allows the model to perform complex compositional reasoning in a single pass, autonomously gathering the necessary visual and textual cues and articulating step-by-step thinking before formulating the answer. This self-reflection capability reduces hallucinations and improves efficiency, allowing it to handle a wide range of tasks without the need for external tools.

Other architectures, such as DeepSeek-VL2 [34], leverage paradigms like Mixture-of-Experts (MoE) to specialize different parts of the model on specific tasks, excelling in activities that require fine perception, such as Optical Character Recognition (OCR) in the context of an image or precise object localization. The versatility of these models is further demonstrated by applications in specialized domains.

One of the most arduous challenges for visual reasoning is the integration of external knowledge, which is information not directly present in the image but necessary to correctly answer a question. The most recent models are beginning to address this problem more systematically. Although early attempts explicitly queried structured knowledge bases, today's LLMs seek to internalize this knowledge during pre-training. Nevertheless, multi-hop reasoning, which requires concatenating multiple pieces of information, remains a significant obstacle even for highly powerful systems like GPT-4o. Progress in this area is often linked to targeted fine-tuning techniques, as demonstrated by the notable improvements achieved by models like PaliGemma-2-3B after specific training on this type of task.

Another promising research direction is the evolution towards agentic artificial intelligence. Instead of passively answering a question, new systems aspire to become visual agents capable of "seeing, understanding, and acting." This vision, explored in models like WebWatcher [7] that uses reinforcement learning for complex information retrieval, paves the way for more natural interfaces capable of executing real-world tasks based on multimodal instructions. Despite progress, current models do not truly "reason" but merely exploit statistical shortcuts present in the training data. For this reason, research is shifting towards models that generate textual or visual justifications for their answers.

Chapter 3

Personal assistant architecture design

This chapter presents the detailed architectural design of the EPANSA system, describing the software components that enable the creation, management, and querying of a dynamic Personal Knowledge Graph. The analysis begins with an overview of the fundamental modules, focusing on the strategic role of the Orchestrator as the central hub for managing security, multi-tenancy, and coordination between distributed services and the Neo4j database.

Subsequently, the focus shifts to the client side, illustrating the architecture of the mobile application developed in Flutter, which is designed not only as a conversational interface but as a proactive tool for personal data synchronization. Then, the implementation logic of the backend is explored in depth, examining the PKG extraction processes, the graph construction pipelines via LLM, and the advanced retrieval strategies based on GraphRAG for response generation.

Finally, the experimental results obtained through a dedicated test dataset are discussed, aimed at validating the factual correctness, temporal stability, and complex reasoning capabilities of the assistant.

3.1 Architectural modules overview

The architecture of the EPANSA framework revolves around a central element that holds together all parts of the system and allows the personal assistant to behave as a single coherent entity, even though it is composed of different, specialized modules often distributed across multiple services. This element is the Orchestrator, a FastAPI-based backend that exposes a REST API to the client and coordinates interactions with the cognitive graph-construction engine (LLM-Graph-Builder), with the Personal Knowledge Graph in Neo4j, and with the RAG and recommendation subsystems.

Its fundamental task is to transform HTTP requests coming from a Flutter app into a series of well-defined operations on the internal modules, ensuring security, multi-tenancy, state management, and isolation among different users, so that every call remains traceable and linked to the correct context. From the client's point of view, the Orchestrator is a web server that provides authenticated endpoints, and every time the app sends a request, whether it is to upload a document, synchronize phone contacts, ask the chatbot a question, or request a travel suggestion, the Orchestrator interprets the payload, enriches the request with the user context, and decides which internal components must be involved and in what order; the client always interacts with a single API surface.

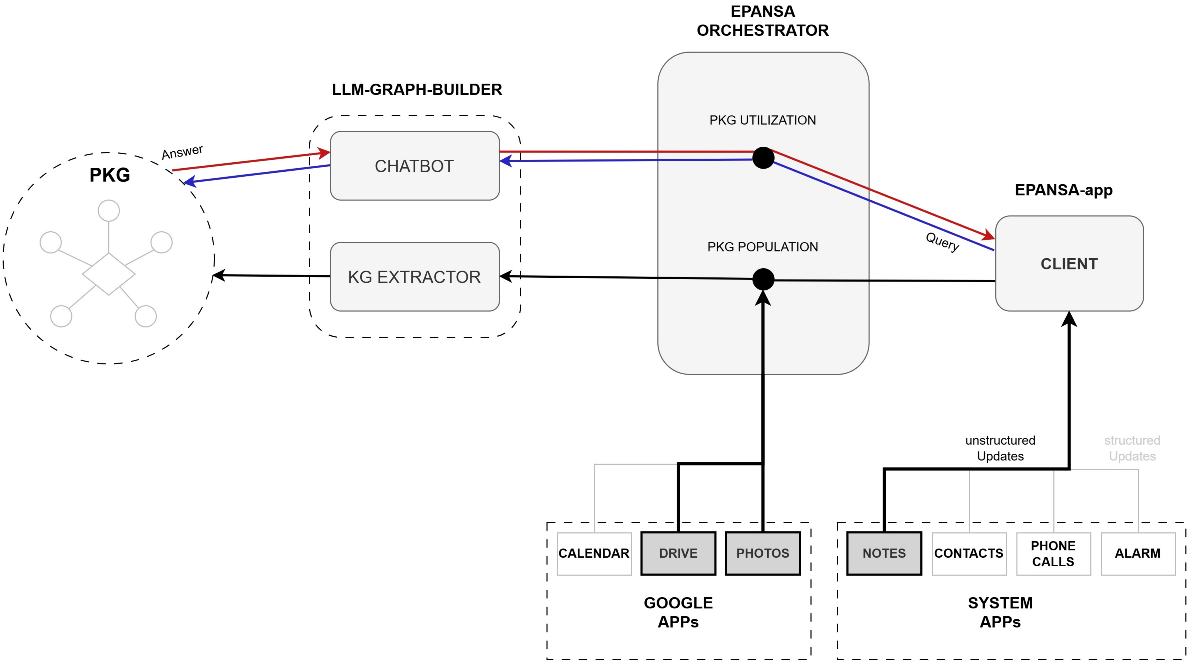


Figure 3.1: EPANSA App architecture.

The role of the Orchestrator in the ecosystem To understand how the Orchestrator fits into the architecture, it is useful to imagine it as the "central node" of a network in which the branches are represented by three major families of components: on one side, the client app, which runs on the user's device and collects data or commands; on another, the LLM-Graph-Builder, responsible for ingestion, knowledge extraction, and enrichment; and finally, the persistence layer, represented by the Personal Knowledge Graph in Neo4j, which stores all personal information in a structured and verifiable way. The Orchestrator does not store knowledge directly, but decides when and how it should move among these elements, ensuring that every document, image, or synchronized event is transformed into coherent nodes and relationships, and every query is resolved by querying the graph using the most appropriate strategies.

On the client–server communication side, the flow is entirely based on HTTP and JSON: the Flutter app sends requests to REST endpoints exposed by the Orchestrator, attaching a JWT token that represents the user's identity and associated permissions. At the middleware level, before the request even reaches the application logic, this token is verified and used to derive, among other things, the name of the Neo4j database representing the user's personal graph. This information is stored in a request context (for example, in request.state.db_name), which becomes the guiding thread used by all downstream functions to know where to read and write, avoiding any mix-up between different users' data.

End-to-end request flow If we consider a typical end-to-end interaction, such as when the user uploads a new document to analyze, the Orchestrator receives the call from the client, verifies authentication, temporarily stores the file in an ephemeral filesystem area, and then invokes the LLM-Graph-Builder engine through a dedicated layer. This layer consists of a set of high-level functions, exposed as a stable interface, that hide the implementation details of the cognitive pipeline; from the Orchestrator's perspective, the available operations conceptually correspond to "upload document," "extract knowledge graph," and "run post-processing and indexing," while the internal implementation details (chunking, entity extraction, vector and full-text index creation) remain confined within the builder.

During this ingestion pipeline, the Orchestrator behaves like a coordinator of a high-level transaction: it calls builder functions in sequence, interprets return codes, and, in case of partial errors, may decide to trigger rollback or recovery logic, for example, deleting a document that was uploaded but not indexed correctly, or retrying only the last failed phase. In this way, PKG consistency is preserved, and the user does not end up with "half-processed" content in their graph. Communication between Orchestrator and builder remains purely server-to-server, typically through internal calls not exposed externally, precisely to maintain a clear separation between the public API perimeter and the internal cognitive domain.

A similar but mirrored flow occurs when the user asks a question to the personal assistant through the chat. In this case, the client sends a textual request to the Orchestrator, which forwards it to the conversational backend. The backend then queries the RAG pipeline that operates on the PKG: the query may be rewritten, the most appropriate retrieval strategies are selected (combinations of vector search, full-text search, and entity- or community-based graph search), and the resulting context is passed to the generative model that produces the answer, constrained to use only material coming from the personal graph. The Orchestrator, again, does not intervene in the linguistic reasoning but establishes operational parameters such as token budget, latency limits, and maximum retrieval fan-out, acting as a “controller” of the process rather than an executor of neural transformations.

Application interfaces and routers Inside the Orchestrator, the API exposed to the client is organized through thematic routers that act as logical front-ends for different use cases. One router handles authentication operations, implementing the OAuth2 flow with external providers and issuing JWT tokens that will then be used by all other requests. A second router focuses on functional endpoints, those that allow the app to upload documents, start ingestion, query the graph, or request recommendations, and is protected by dependencies that verify token presence and validity. A third router is dedicated to runtime and periodic synchronization operations, such as starting jobs that import calendar events or synchronize with Google services.

These routers are not simple endpoint groupings but actual “gateways” to distinct application domains. The functional router, for example, receives a file upload request from the client, validates the payload, builds a call context that includes the tenant’s database and disk paths to use, then invokes the ingestion service, logs results and timings, and returns a comprehensible status to the client (e.g., “document being processed” or “document successfully indexed”). The runtime router, instead, often operates asynchronously: it starts scheduled jobs via an internal scheduling component, configures time windows, request limits, and backoff policies, and records metadata such as start and end timestamps, number of attempts, and caller identity, enabling later analysis of throughput and failure patterns.

From the client’s perspective, all this is compacted into a very linear REST contract: for example, the app’s synchronization screen does not need to know the existence of a scheduler or internal jobs; it simply sends a “start sync” request, receives a result, and shows the status to the user, while the Orchestrator translates that command into persistent, observable, repeatable jobs.

Multi-tenancy, security, and configuration A crucial aspect of the Orchestrator architecture is multi-tenancy handling, the ability to serve many isolated users using the same infrastructure. Each EPANSA user has their own separate Neo4j database serving as their personal PKG; when a request reaches the backend, the JWT middleware decodes the token, extracts the application identity, and resolves the database name to use, storing it in the request context. All subsequent calls to the builder or Neo4j use this parameter to limit operation visibility to that user’s graph only, ensuring that two people’s data cannot end up on the same graph or contaminate one another.

This logic relies on a centralized system configuration that explicitly defines endpoints, credentials, storage paths, and temporary directories. Variables such as STORAGE_BASE_PATH or DATABASE_URL identify areas for persistent data, whereas constants like TMP_DIR or SYNC_DIR represent ephemeral areas where the Orchestrator can store temporary files or intermediate snapshots. The distinction is not just organizational: it allows the application of different cleanup, backup, and security policies, for example, deleting all temporary files after a crash or limiting access to persistent areas to specific processes. The entire configuration is designed to stay aligned across environments, so that moving from development to production requires no code changes, only replacing .env files or secret manager entries, while preserving the stability of the Orchestrator interface.

HTTP perimeter security is further reinforced by security policies configured at middleware level, defining which origins can call the API, how preflight requests must be handled, and which credential patterns are allowed. This is especially important in multi-client scenarios where the Flutter app may be distributed across different platforms and potentially executed from different domains; by centralizing exposure rules in the Orchestrator, security logic is not duplicated across endpoints and control over how and from where the system can be invoked remains consistent.

State management, concurrency, and observability Although EPANSA is designed as a distributed and highly modular system, the user experience requires significant continuity, making state management a central topic. The Orchestrator not only preserves authentication state but must also track the state of documents and ingestion jobs: a document may be in different phases (uploaded, extracting, indexed, enriched), and these phases are reflected in a sort of state machine persisted in the graph or related structures. The Orchestrator uses these states to determine whether a “repeated” request should be treated as a retry, a resume from an intermediate point, or a new flow, thus maintaining API-level idempotency: calling the same update endpoint twice should not generate duplicate graphs but lead the document to a coherent final state. At the same time, the orchestration layer must govern concurrency among jobs to prevent too many processes from working concurrently on the same graph or the same document.

The recommended logic suggests dividing work units into idempotent segments and scheduling them to minimize conflicts on “hot” graph partitions: updates on the same document or tenant tend to be serialized, while operations on different tenants can run in parallel, taking advantage of their isolated Neo4j databases. When external services with quotas or limitations are involved (e.g., sync APIs or LLM providers), the Orchestrator applies admission-control rules, limiting the number of simultaneous calls per provider or tenant and favoring completion of already-started work before accepting new requests, avoiding system saturation.

All this is accompanied by a detailed observability layer: the builder produces telemetry on retrieval times, counts of added nodes and relationships, community statistics; the Orchestrator enriches this information with correlation identifiers, user context, and routing data, and sends it to structured log and time-series systems. This enables long-term performance analysis, detection of regressions caused by model or index changes, and, when necessary, recalibration of orchestration policies (e.g., reducing token budgets or adjusting similarity thresholds).

Extensibility Finally, the Orchestrator architecture is designed to be extensible: new modules can be added to the system without requiring deep changes to the client or the PKG structure. For example, if a new type of data source were introduced (such as application logs or sensor data), it would be enough to define a new ingestion path which later relies on the same mechanisms for graph construction, indexing, and RAG. Likewise, a new recommendation or predictive-analysis layer could be seen by the Orchestrator as a "specialized service" invoked in response to certain recognized intents, just as today happens for the point-of-interest recommendation engine.

This flexibility is made possible precisely by the clear separation of roles: the client handles data collection and user interface, the PKG provides a structured and verifiable knowledge base, the builder and RAG perform intelligent transformations, while the Orchestrator decides how to connect these worlds, when to activate them, how to control resource consumption, and how to expose everything as a single coherent service.

GraphRAG The entire architecture, as already mentioned, culminates in the module that enables information retrieval and the subsequent answering of user questions about their personal data stored in the PKG; this is done through a RAG capable of retrieving useful information while also considering the graph's connectivity and topology.

This type of RAG is called GraphRAG, developed by Han et al. [8]; when the user asks a natural-language question, this subsystem orchestrates a sophisticated information-retrieval pipeline. Unlike a traditional RAG, retrieval is not limited to vector similarity search on chunks; instead, it activates a hybrid method combining multiple signals: semantic vector similarity, exact full-text index matches, and, above all, topological paths within the graph. It can retrieve not only text fragments but entire subgraphs representing an entity's context or relationships between concepts. The retrieved material is then processed by a contextual compression module that filters redundancies and maximizes information density before passing it to the generative model. The last and most important step is the answer generation: the LLM receives a system prompt constraining it to formulate its answer based solely on the provided context. This transforms generation into a controlled synthesis of facts, making the answer faithful to the data, verifiable, and traceable back to its original sources in the graph.

3.2 Client architecture

The architecture of the EPANSA client represents the point of contact between the user and the system's sophisticated orchestrator. Built using the Flutter framework, the client is conceived from the outset to be inherently multi-platform, ensuring a consistent and coherent user experience across a wide range of mobile operating systems. Its primary interaction paradigm is that of an advanced personal assistant, where the mobile application does not act as an autonomous entity but rather as an intelligent terminal that provides the physical and contextual point of presence for a far more powerful remote intelligence. This strategic choice is rooted in a well-defined client–server architecture, in which complex decision-making logic, orchestration of "capability-tools" (such as Google Calendar management or web searches), and knowledge persistence are delegated to the backend. The client is therefore responsible for encapsulating the complexity of API communication and presenting the user with a fluid, natural, and multimodal interface capable of handling not only textual inputs but also voice commands and displaying responses enriched with media such as images.

The client assumes a proactive and critical role both in the authentication process, coordinating the OAuth 2.0 authorization flow with Google, and in populating the PKG, acting as a key data provider for all personal information, such as phone contacts or local alarms, which does not reside in Google's cloud ecosystem but is essential to enrich the backend knowledge graph with deeply personal and device-specific context.

Initialization To fully understand the architecture of the EPANSA client, it is necessary to analyze its foundational files, which define its structure, technological dependencies, and initial lifecycle. The formal starting point is the pubspec.yaml file, which acts as the project's identity card within the Dart and Flutter ecosystem. This file not only declares the application's identity but also lists its external dependencies, whose selection dictates the key architectural patterns. The inclusion of libraries such as flutter_bloc and bloc clearly indicates the adoption of the BLoC (Business Logic Component) pattern as the primary strategy for state management. This approach enforces a clean separation between the user interface (UI) presentation logic and business logic, drastically improving code testability and maintainability. Other dependencies, such as http and google_sign_in, define the role of the application as a network client that communicates with both the EPANSA backend and Google's authentication services, while the use of flutter_secure_storage provides a robust mechanism for persisting sensitive data, such as authentication tokens, in an encrypted, platform-specific storage area.

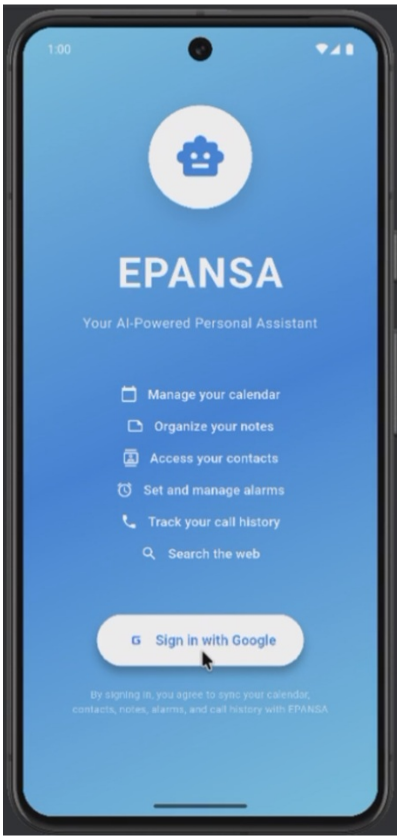
The application is designed to decouple its configuration from its source code, an extremely important practice for flexibility and maintainability: this is made possible through the interaction between the assets/config/config.json file, which stores environment-specific values such as the backend API base URL, and the lib/config/app_config.dart class. The latter acts as a configuration provider, loading the JSON file at startup and offering type-safe access to its values throughout the application, allowing behavioral changes without requiring recompilation. In parallel, the lib/config/theme.dart file centralizes the visual design of the application, defining a formal color palette system to ensure a better user experience. The true entry point of the application and its initialization logic are found in lib/main.dart, which contains the main() function. This function orchestrates the bootstrap sequence: it ensures the Flutter framework is initialized, loads the configuration, and builds the root widget tree. A key role of main.dart is the implementation of dependency injection, using BlocProvider widgets to instantiate and provide global, long-lived BLoCs, such as the AuthBloc, to all descendant widgets. This allows any screen to access shared state in a decoupled way, establishing from the start a reactive, stateful architecture in which the UI is rendered as a direct function of the application's business state.

User Session Management Immediately following the initialization phase, the first critical flow handled by the EPANSA client is authentication. This process is not a simple credential check, since it involves the user interface, business logic, data abstraction, and communication services with both external platforms and the proprietary backend. The presentation interface contains a screen for unauthenticated users that displays interactive elements such as the "Sign in with Google" button. In adherence to the BLoC pattern, when the user interacts with this button, the widget does not contain the logic to perform authentication; instead, it simply notifies the user's intent to the business logic layer. This decoupling is achieved by creating and dispatching an event, such as GoogleSignInRequested, to the AuthBloc, the stateful component that orchestrates the entire process.

Once the event is received, the AuthBloc, immediately transitions to an AuthenticationInProgress state. This state change is reactively communicated to the UI, which updates to show a loading indicator and disable further interactions. The AuthBloc does not directly implement external service calls but delegates this responsibility to a higher-level abstraction layer, the repository, by invoking a method such as logInWithGoogle on the AuthRepository. Control then passes to lib/repositories/auth_repository.dart, which represents an architectural layer abstracting data sources and communication logic.

The repository orchestrates the sequence of necessary calls: it first invokes a service that encapsulates direct interactions with Flutter's google_sign_in package, initiating Google's native login flow. Once the user successfully completes this step, the service extracts the critical credential returned by the SDK: the server-side authorization code (serverAuthCode). With this code in hand, the AuthRepository proceeds to the next step: authentication with the EPANSA backend. Using a generic API service, it issues a POST request to the /auth/google/exchange_code endpoint, sending the authorization code.

The backend exchanges this code and returns an application-specific JSON Web Token (JWT). The final step of the flow once again resides within the AuthRepository, which must ensure the persistence of the JWT to maintain the user session across application restarts, saving the token securely using a SecureStorageService. Only after the token has been saved does the repository notify the AuthBloc of success, which in turn emits a new state, AuthenticationSuccess. The UI, listening reactively, receives this notification and rebuilds to navigate the user from the login screen to the main application screen, completing a robust, well-layered cycle.

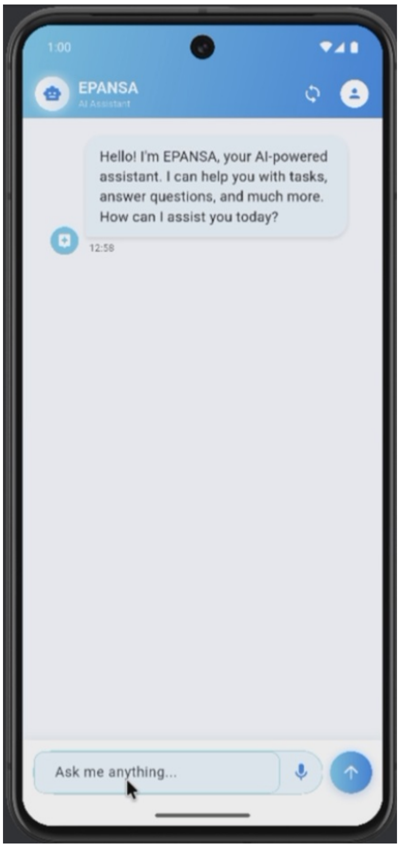


By signing in, you agree to sync your calendar, contacts, notes, alarms, and call history with EPANSA

Figure 3.2: EPANSA app, auth screen

The Conversational Interface Once authentication is complete, the user gains access to the primary function of the EPANSA application: the conversational interface. This is not merely one of many features but the primary interaction paradigm through which the user communicates with the remote artificial intelligence agent. The architecture of this component consistently follows the layered BLoC pattern (UI → BLoC → Repository → Service), ensuring a reactive data flow and clear separation of responsibilities, essential for managing the dynamic and asynchronous nature of a real-time conversation.

The visual entry point is the widget defined in lib/screens/chat_screen.dart, which serves as the main container. Using a BlocBuilder or BlocListener tuned to the ChatBloc, the screen subscribes to changes in the conversation state. Inside it, a ListView widget manages the rendering of the message history, instantiating for each message a specialized, reusable widget, lib/widgets/chat_bubble.dart, which handles the visual presentation of a single chat "bubble." User interaction, namely composing and sending a new message, is handled by another reusable widget, lib/widgets/message_composer.dart, which captures text and dispatches a SendMessageRequested event to the ChatBloc.

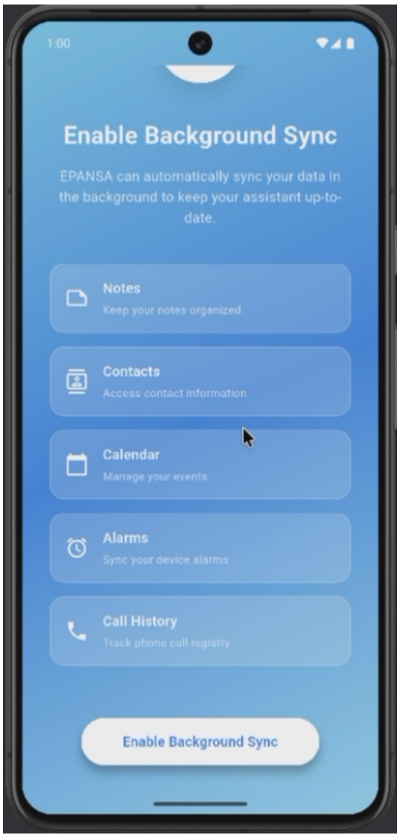


Hello! I'm EPANSA, your AI-powered assistant. I can help you with tasks, answer questions, and much more. How can I assist you today?

Figure 3.3: EPANSA app, chat screen.

Data ingestion Beyond real-time conversational interaction, a functional pillar of the client is its ability to act as a provider of personal data, enriching the back-end's Personal Knowledge Graph with information that resides exclusively on the user's mobile device. This local data-synchronization process includes the following data types: documents (pdf, doc, or docx), photos (jpeg, png), calendar events, alarms, notes, phone contacts, and call logs.

The initiation of this process is typically delegated to the user through interaction elements present in management or configuration screens, such as an "Enable Background Sync" button. In line with the application's architecture, pressing this button does not directly start synchronization but simply dispatches an event, such as SyncStarted, to the responsible BLoC, lib/blocs/sync/sync_bloc.dart. The latter, upon receiving the event, transitions to a SyncInProgress state and delegates the complex and potentially lengthy logic to the data-abstraction layer, the SyncRepository. The repository coordinates the various phases of the process, beginning with accessing native device data. To do this, it does not interact directly with platform APIs but instead invokes a specialized service, lib/services/local_data_service.dart, which encapsulates the logic for querying the address book or alarm manager through specific Flutter packages.



Enable Background Sync

EPANSA can automatically sync your data in the background to keep your assistant up-to-date.

Figure 3.4: EPANSA app, Enable background synchronization screen

Before any access to this sensitive data, it is essential that the application has the required permissions, handled by another utility service, lib/services/permission_service.dart, which abstracts the complexity of runtime permission requests. Once the raw data is retrieved and normalized, the SyncRepository sends it to the backend via the generic API service, completing an interaction cycle that spans all layers of the application, from native device APIs to the remote backend.

Coherence, Portability, and Robustness Beyond specific functional flows, the architecture of the EPANSA client is supported by a set of cross-cutting components and a state-based navigation logic that collectively ensure coherence, robustness, and usability. The first and most critical of these components is the initial navigation logic, whose epicenter lies in lib/screens/splash_screen.dart. This screen, typically the first to be rendered at application startup, acts as a gatekeeper and decision router. Its role is to determine the user's initial authentication state and redirect them accordingly to the appropriate entry point. To fulfill this role, the SplashScreen reactively subscribes to the AuthBloc state and awaits the outcome of the attempt to retrieve a valid session token from the SecureStorageService. If the AuthBloc emits an AuthenticationSuccess state, the SplashScreen imperatively navigates to the authenticated user's main screen; otherwise, navigation is directed to the login screen. This mechanism ensures that the user always lands in the correct context without disruption.

Once the user is authenticated, lib/screens/home_screen.dart becomes the hub of their experience, acting as the central navigation point. Typically implemented with Flutter's Scaffold widget, it provides the persistent structure of the interface, such as a BottomNavigationBar or Drawer, that allows the user to seamlessly switch among the main sections of the application, such as the ChatScreen and the SettingsScreen. Finally, an important aspect for robust interaction is the uniform handling of significant actions and errors, a responsibility that falls to cross-cutting utility components such as those defined in lib/utils/dialogs.dart.

The choice of Flutter as the primary technological tool is an architectural strategy guided by the intention to provide a consistent user experience across a wide range of operating systems. The goal is not simply to be multi-platform but to be deeply and correctly integrated with each ecosystem, ensuring that the EPANSA client is perceived as powerful and appropriate regardless of where the user chooses to access it.

3.3 EPANSA orchestrator architecture

In this subsection, each essential component of the orchestrator presented in the overview will be analyzed from an implementation perspective, starting from the modules responsible for extracting facts to be inserted into the PKG, and concluding with the chat module that enables communication with the user.

3.3.1 Extraction logic implementation

During the PKG population phase, events generated by applications, background synchronizations, or direct interactions with the client are translated into new portions of the PKG or lead to the removal or update of existing ones within a carefully orchestrated flow that involves the orchestrator, Neo4j, and, centrally, the llm-graph-builder module.

Every PKG population is triggered by a change detected by the backend, which may vary widely in nature: deletion of an entity such as an alarm or a photo; insertion of a new object, such as a PDF document uploaded by the user; or the update of an already existing piece of data, such as a modification to the text of a note or a file's metadata. Starting from this elementary event, the system never behaves in a "local" or ad hoc fashion but always follows a conceptually unified pipeline that branches into three major operational trajectories, deletion, insertion, update, and whose final product is a coherent set of semantic triples (nodes, relationships, properties) materialized in the user's personal graph.

In the case of deletion, the path is relatively linear and completes by calling the service dedicated to removing triples associated with the entity to be eliminated, ensuring that no orphaned or inconsistent traces remain in the PKG. In the case of insertion or update, however, the flow converges on the semantic extraction service, which must interpret the nature of the incoming data and decide whether it can be treated as structured metadata or whether the more sophisticated path involving the llm-graph-builder must be activated.

When the input data is purely structured, such as a JSON object describing a recurring alarm or a contact entry, PKG population can proceed without neural intelligence, relying on a relatively direct transformation from the source schema to the graph schema. In this scenario, the semantic extraction service behaves as a deterministic translator: it maps JSON fields into typed nodes and relationships, applies domain modeling rules (e.g., choice of node labels and relationship types), and produces a set of triples that can be immediately persisted in Neo4j. The example of the recurring alarm is emblematic: from metadata containing the alarm identifier, time, weekly frequency, and active days, the system generates an Alarm node with a stable identifier and a set of relationships to DayOfWeek nodes representing Monday, Tuesday, and so on, enabling precise temporal queries such as finding all alarms active on Monday or those recurring on a specific day of the month.

Below is an example of JSON containing the metadata of an alarm:

```
{
    "source_app": "alarm",
    "alarm": "recurrentAlarm_328",
    "reurrence_type": "recurrent",
    "metadata": {
        "label": "Work",
        "time": "06:00",
        "repeat_frequency": "weekly",
        "on": "Monday, Tuesday, Wednesday, Thursday, Friday"
    }
}
```

Much more interesting for understanding the role of the llm-graph-builder is the case in which the information to be inserted into the PKG has a hybrid nature, combining a structured component, such as file metadata, with intrinsically unstructured content, such as the full text of a document or the visual content of a photograph. In these cases, the population pipeline deliberately bifurcates, because the two parts of the data require different transformations before they can be converted into homogeneous triples.

Metadata follow a path like that of purely structured examples and are translated into nodes and relationships describing the high-level object (document, photo, calendar event). Meanwhile, the unstructured component is handled by a neural processing chain in which the llm-graph-builder acts as the extractor of textual knowledge, possibly preceded by an additional “translation” stage for non-textual modalities, such as the img_description module used to convert images into verbally rich semantic descriptions.

From an architectural perspective, the llm-graph-builder is positioned as a specialized service within the backend, invoked by the orchestrator, implemented on FastAPI, whenever heterogeneous content must be transformed into a graph structure. It is responsible exclusively for the phases of building and enriching the PKG, while query-time reasoning and dialog management are handled by a separate runtime based on GraphRAG. Every call to the builder is strictly constrained to a single user through association with a dedicated Neo4j database, ensuring the resulting graph is isolated and private.

The entire data layer is executed in containers and uses Neo4j 5 Enterprise with the APOC plugin, which provides advanced primitives for hybrid index management and graph-analysis algorithms. In this framework, PKG population is not a “monolithic” operation but a composition of orchestrated micro-stages in which the llm-graph-builder receives content, segments it, interprets it according to a predefined schema, and finally materializes it in the graph, while always maintaining the ability to repeat or resume execution without corrupting existing state.

Inside the llm-graph-builder, the phase relevant to PKG population corresponds to Graph Construction, the neuro-symbolic core of transforming unstructured data. This pipeline is designed to be deterministic and restartable and is articulated into three main steps which, although conceptually distinct, are intertwined with a series of invariants: preservation of editorial context, traceability of origins, and idempotence of graph writes.

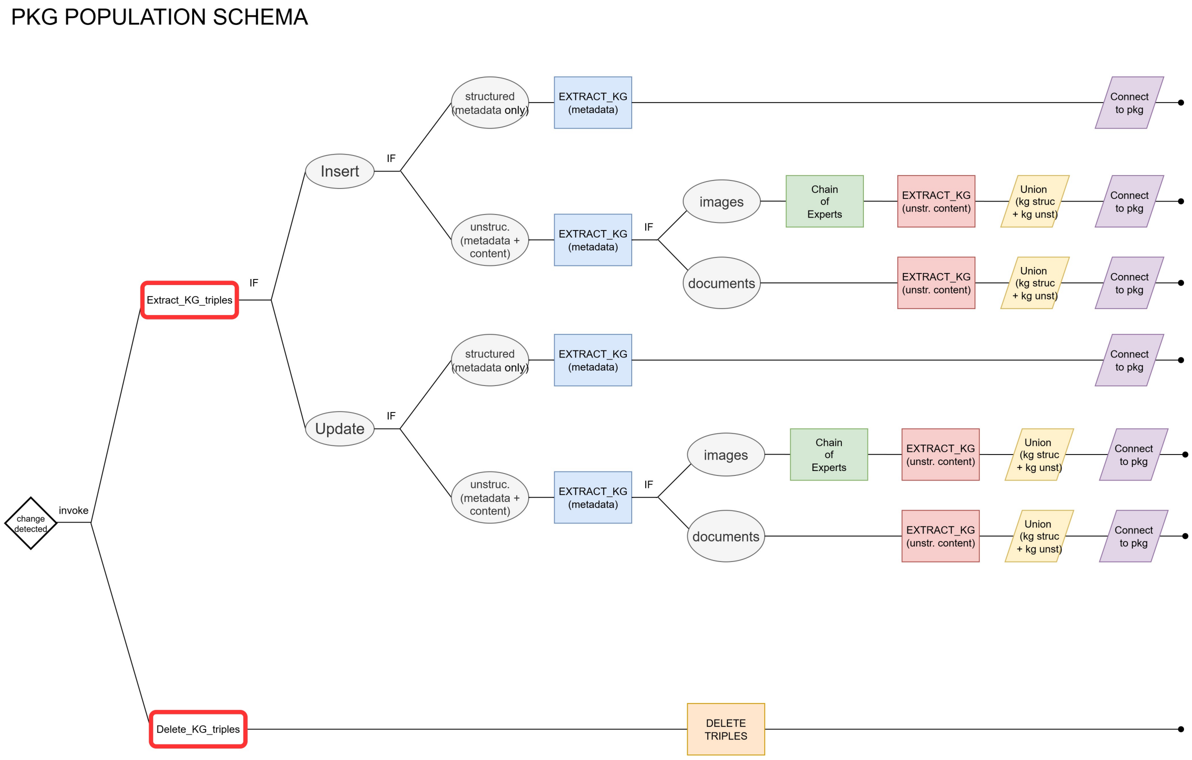


Figure 3.5: Detailed PKG population process based on the input information type to KG extraction module and mode (update or insert).

The first step is text segmentation, applied to any document processed: the content is broken into elementary units called chunks using the logic implemented in the chunk creation module, which does not merely split text mechanically but seeks a balance between length, semantic coherence, and reuse of local context. Each chunk is assigned a deterministic identifier derived from its content so that two identical copies of the same fragment, encountered at different times or in different files, can be recognized and treated as references to the same conceptual unit, supporting both deduplication and safe pipeline re-execution.

On this foundation operates the neural component for extracting entities and relationships, implemented in the module that queries the language model. Here, the LLM is not left free to generate text creatively but is constrained by a prompt and output schema requiring it to return structured lists of entities and relationships built exclusively using node and edge types permitted by the system's ontology.

Practically, the model receives as input one or more chunks, possibly combined in sliding windows to capture context, and must produce a set of triples suitable for translation into labeled nodes and typed relationships in Neo4j. This constraint turns the LLM from a prose generator into a disciplined structure generator, reducing hallucination risk and ensuring every element of the resulting graph maps to a well-defined semantic category. The outputs then undergo syntactic and semantic validation, where artifacts are removed, type strings are normalized, and any relationships that violate ontology constraints are discarded, ensuring that no local irregularity propagates into the PKG schema.

Once the list of validated entities and relationships is obtained, the graph creation and linking phase begins, using Neo4j access functions designed to operate idempotently via the MERGE construct. Instead of always creating new nodes and edges, the system looks up key combinations of properties and labels and reuses them when they already exist, avoiding duplication and favoring the progressive construction of a coherent network. In this stage, three families of "base" relationships are also established, providing a minimal yet navigable structure for the graph:

- :PART_OF — links each chunk to the document node from which it was extracted, preserving the relationship between atomic content and its editorial container
- :NEXT_CHUNK — sequentially orders chunks and allows reconstruction of the original narrative flow
- :HAS_ENTITY — links each chunk to the conceptual entities appearing in it, creating an explicit bridge between text and concept

To these infrastructural relationships are added conceptual relationships among the entities themselves, e.g., an instance of :MANUFACTURED_BY between a car model and its manufacturer, derived directly from the LLM-extracted triples. The systematic use of deterministic chunk identifiers and MERGE operations ensures that, in case of error or interruption, the entire process can be restarted from the stopping point without compromising PKG integrity, since already persisted parts are neither duplicated nor inconsistently altered.

To understand how these abstractions materialize in PKG population, it is useful to reconsider examples of unstructured input through the internal behavior of the llm-graph-builder. In the case of a photo, e.g. photo_20250616, the system receives two inputs: structured metadata describing file path, creation date and time, and a localization string; and the binary content of the image, which the builder cannot interpret directly.

The img_description vision module analyzes the visual content and produces a textual description such as "a photo of the Columbus Monument in Barcelona," enriched with attributes or contextual details. This description, together with the explicit localization contained in the metadata, is then provided to the llm-graph-builder as input text. The chunking module segments it (often into a single chunk), the schema-aware LLM identifies entities such as the monument and the city, as well as relationships like DEPICTS between the photo and the monument and LOCATED_IN between the monument and the city, and finally the materialization phase creates or updates a Photo node, a Monument node, a City node, and their corresponding relationships. In this way, the photo does not remain a mere file with a path and coordinates but becomes an entry point into the PKG connected to broader concepts such as a place or monument, potentially linked in the future to other content referencing the same city or site.

A similar process occurs for a document (e.g., doc_598) containing information about car models, manufacturers, and technologies. Here, the llm-graph-builder must handle potentially long text segmented into many chunks, each identified by a deterministic hash of its content. For each chunk, the language model extracts conceptual entities such as specific car models, manufacturers, and technological categories like "electric vehicle," as well as relationships such as IS_A or MANUFACTURED_BY.

Materialization produces Document nodes representing the file itself, Concept or Manufacturer nodes representing extracted concepts, MENTIONS relationships linking the document to concepts, and direct conceptual relationships between concept and manufacturer nodes. The result is that the PKG acquires a component of technical knowledge that is no longer limited to the PDF's existence but exists in structured form, reusable for future queries and linkable to other pieces of information, such as user preferences for certain brands or vehicle types inferred elsewhere in the graph.

A very important element of PKG population is the unification between what is derived from structured metadata and what is extracted from unstructured content. This step ensures that the final graph does not represent two parallel worlds, one for the "technical sheet" of the object and one for its conceptual meaning, but a single integrated representation. The unification typically occurs at the node representing the primary entity (photo, document, alarm) and results in a single set of triples linking that entity both to its low-level attributes and to the conceptual entities derived from it. In other words, a "neck" is constructed in the graph where the two flows meet, enabling navigation from concept to originating file and, conversely, from file to its semantic implications.

Applied uniformly to insertions and updates, this same mechanism implements upsert-like behavior: each new pipeline execution on the same object does not create duplication but refinement or extension of the corresponding subgraph, while maintaining global PKG integrity. When PKG population is viewed as a whole from an operational perspective, it becomes clear that the llm-graph-builder is designed for repeatability and robustness: graph construction stages are conceived as restartable sequences that can be interrupted and resumed without coherence loss thanks to deterministic chunk identifiers and idempotent write operations. Moreover, since each call to the builder is isolated in the user's Neo4j database,

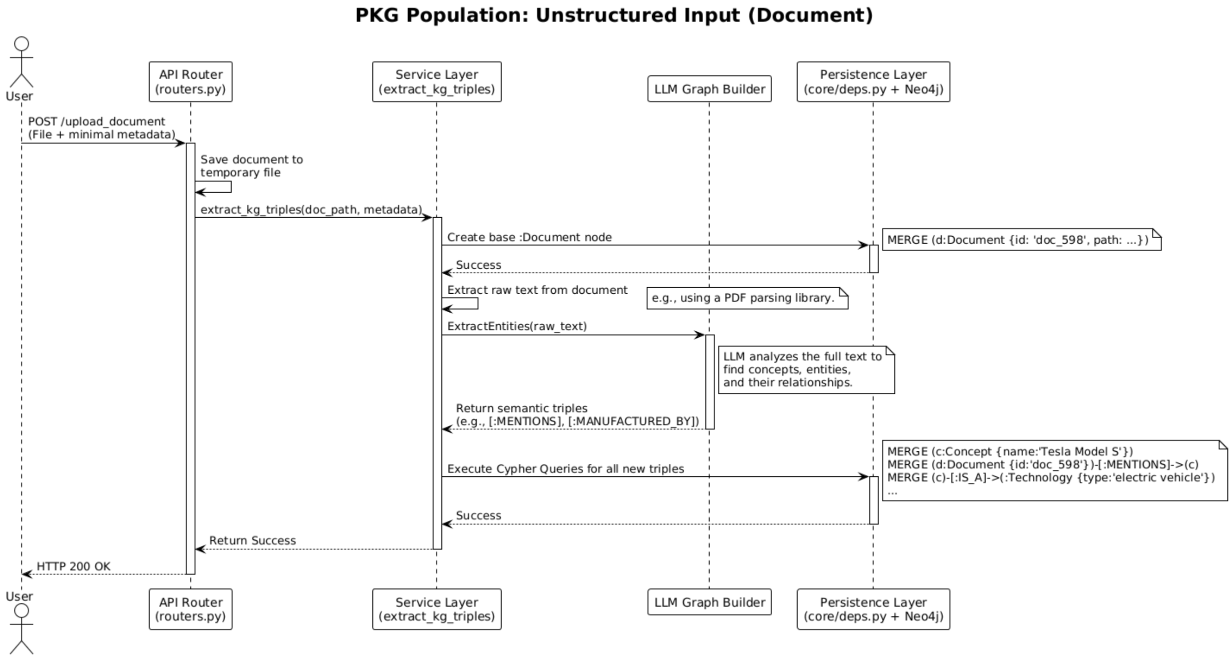

PKG Population: Unstructured Input (Document)
User
API Router (routers.py)
Service Layer (extract_kg_triples)
LLM Graph Builder
Persistence Layer (core/deps.py + Neo4j)
POST /upload_document (File + minimal metadata)
Save document to temporary file
extract_kg_triples(doc_path, metadata)
Create base :Document node
MERGE (d:Document {id: 'doc_598', path: ...})
Success
Extract raw text from document
e.g., using a PDF parsing library.
ExtractEntities(raw_text)
LLM analyzes the full text to find concepts, entities, and their relationships.
Return semantic triples (e.g., [:MENTIONS], [:MANUFACTURED_BY])
Execute Cypher Queries for all new triples
MERGE (c:Concept {name:'Tesla Model S'})
MERGE (d:Document {id:'doc_598'})-[:MENTIONS]->(c)
MERGE (c)-[:IS_A]->(:Technology {type:'electric vehicle'})
...
Success
Return Success
HTTP 200 OK
User
API Router (routers.py)
Service Layer (extract_kg_triples)
LLM Graph Builder
Persistence Layer (core/deps.py + Neo4j)

Figure 3.6: Temporal diagram for adding a document from the user perspective.

the architecture supports a multi-tenancy model in which indexes, statistics, and auxiliary structures grow in relation to each personal graph rather than the entire user base. Conceptually, this means that PKG population is a continuous process of cognitive maintenance of the graph, in which each new event is interpreted, integrated, and, when necessary, reconciled with the pre-existing PKG.

3.3.2 Post-processing implementation

The population of the Personal Knowledge Graph does not end with the extraction of triples from the source content. Instead, it includes a post-processing phase triggered by the PKG population service every time a new extraction or update is performed, ensuring that the graph does not remain a simple collection of facts but is progressively transformed into an organized structure, optimized for search and ready to support the reasoning engine based on GraphRAG.

The post-processing phase takes as input a PKG already populated with document nodes, chunk nodes, and entity nodes, as well as the elementary relationships produced during the extraction phase. Its first task is to project this symbolic base into vector space by computing an embedding for each textual chunk and for other conceptual entities. These embeddings are stored as properties of the nodes within the Neo4j database. Once the content has been projected into vector form, post-processing addresses the problem of how to organize the mass of entities into more abstract thematic structures to enhance the capabilities of the model that

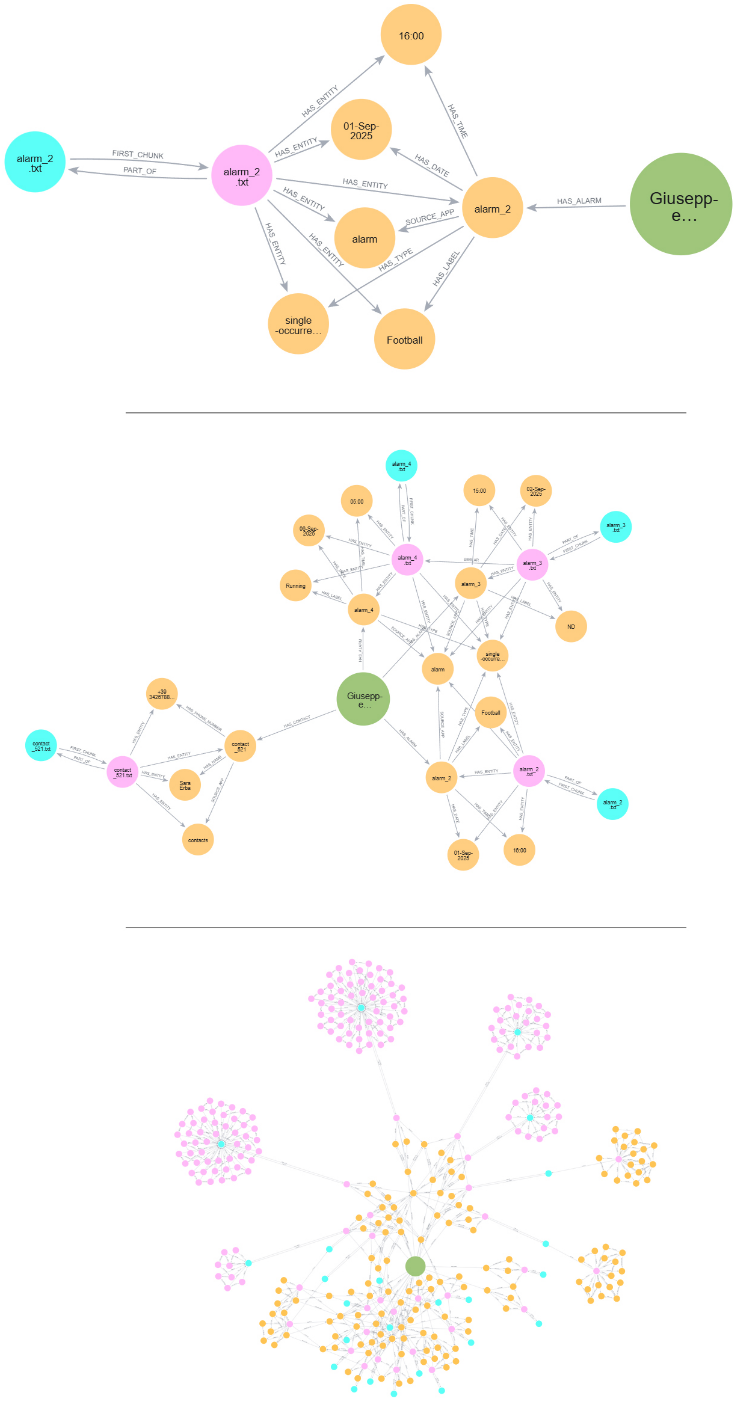


Figure 3.7: Frames of the PKG population process.

will perform reasoning.

The module therefore builds an entity–entity projection weighted by co-occurrences and applies community-detection algorithms provided by the Graph Data Science library, such as Leiden, to identify coherent clusters of concepts reflecting the emerging themes in the user's data. Each detected community is interpreted as an intermediate semantic unit connecting atomic graph nodes to more stable sets of topics, enabling reasoning in terms of "zones" of the PKG rather than isolated entities.

To make these communities truly usable by the rest of the system, post-processing introduces an additional layer of neural enrichment: for each cluster, an LLM is invoked that, constrained by strict output templates, produces a synthetic title and a summary of the set of entities belonging to it. This summary does not replace the original data but serves as a projection of the structural pattern identified by the algorithm. Title and summary are then stored in the PKG as properties of new nodes of type :Community, which explicitly represent these thematic units and are also assigned an embedding computed from the generated text. This allows the same hybrid-indexing logic applied to chunks to be extended to the scale of communities, enabling search to be guided both by similarity to document fragments and by proximity to entire themes.

It is important to note that this phase forms a junction between the graph-population process and reasoning itself: only thanks to the combined presence of updated embeddings, hybrid indexes, and communities can the GraphRAG engine later extract contexts that combine text fragments, specific entities, and thematic summaries, providing the generative model with dense, structured, and traceable informational frames. In other words, after each knowledge-graph extraction, post-processing transforms a set of correct but still "flat" triples into a navigable body of knowledge, where the personal assistant can move along multiple dimensions, lexical, semantic, topological, without ever losing connection to the original sources, thereby meeting the system's reliability objective.

3.3.3 Chat logic implementation

The chatbot architecture is a sophisticated, multi-layered system, designed to provide a stateful user experience supported by the "facts" stored in the PKG, going beyond simple question-and-answer interactions. Its functionality does not reside in a monolithic block of code but is distributed across a dedicated API control layer, a central orchestration service, and a specialized subsystem for deep reasoning over the user's data. This separation of responsibilities is essential for modularity and maintainability, allowing each component to execute its specific function.

The entry point for any conversational interaction is defined in the API router, specifically at the POST /chat endpoint. This endpoint is not just an access point but a secure gateway that serves as the first guardian of system integrity. Each request is intercepted and validated via a security dependency based on the previously mentioned JWT. This mechanism not only authenticates the user but extracts essential contextual information from the token, such as the user's unique identifier and, most importantly, the name of their dedicated multi-tenant database. This information is injected into the request.state object, making user context available to all subsequent layers of the process and ensuring complete data isolation from the very beginning.

The process_user_command function, located in the conversation.py module, is triggered directly by the APIs. It takes the user's question and processes it through a complex Agent, a system implemented in LangGraph capable of proactively using different tools (e.g., Recommendation, RAG, Visual analysis, etc.) powered by an LLM. Our agent is a true state machine acting according to the state of the conversation. Its first task is user-intent analysis: by examining the natural-language query, the agent determines what the user wishes to accomplish and selects the tool most appropriate to fulfill the request. Instead of trying to answer directly, the agent acts as a coordinator that knows which specialist to call. For recommendation requests, it activates the PoI_recommender tool, while for any question that requires access to personal information, it relies on the graph_RAG tool. In the following sections we will examine in detail the role of the "img_deep_analysis" tool, which is activated only if the graph_RAG tool fails to produce an accurate answer.

For complex commands that combine multiple intents, the agent can sequentially invoke different tools to construct a complete and coherent response. But the true strength of this system lies in its ability to maintain state across multiple API calls.

This is enabled by two key mechanisms:

- **MemorySaver:** persists the agent's message history
- **custom agent_cache:** stores initialized agent instances.

When a user sends their first message, the agents are created and cached. For subsequent messages, the system retrieves these instances, avoiding the overhead of new initialization and ensuring fast responses. Each conversation is tied to a unique thread_id for the user, allowing the MemorySaver to load and save the correct conversational state.

When the agent determines that the user's request requires deep reasoning over the PKG, the GraphRAG subsystem comes into play. GraphRAG is not a simple vector search but an advanced technique that leverages the interconnected nature of the user's data. Instead of treating the PKG as a flat collection of text fragments, GraphRAG builds and queries a semantic index representing entities, relationships, and thematic communities as a structured graph. This approach retrieves a richer, multidimensional context, traversing the graph's connections to gather not just directly relevant information but also related concepts and summaries of entire thematic areas. The result is an exceptionally high-quality context provided to the LLM, enabling it to generate more nuanced, accurate, and factual responses while drastically reducing hallucination risk.

The implementation of GraphRAG is a two-phase process that further demonstrates separation of responsibilities. The first phase is index construction, orchestrated by the initialize_graphRag.py script. This preliminary, one-time operation for a given PKG state acts as a bridge between the dynamic Neo4j database and a static query-optimized index. The script connects to the user's tenant database, executes complex Cypher queries to extract all entities, relationships, textual units (Chunk nodes), and pre-computed community structures, and stores these data in a set of Parquet files. It then invokes the GraphRAG command-line interface to complete indexing, an operation that includes generating textual summaries for each community via an LLM and creating dense vector embeddings stored in a local LanceDB instance.

The second phase is real-time query execution, handled by the graphRag_chat.py module. When the graph_RAG tool is called, this function loads the pre-built index stored in Parquet files and the LanceDB store and initializes the GraphRAG query engine. Upon receiving the user's query, the search engine performs a structured query over this local index, retrieving relevant entities, relationships, and community reports. This rich context is then synthesized and used to generate the final grounded response from the LLM.

3.4 Chatbot test results

The experimentation and verification phase of the EPANSA prototype was entirely focused on validating the system as a complete cognitive infrastructure, going beyond the simple evaluation of a generic language model. For this purpose, a specific test dataset was developed and administered, designed to test the limits of the system and any flaws in logical and temporal reasoning. The essence of this methodology is because a Personal Knowledge Graph (PKG) is used as the sole factual basis for the LLM's response generation.

To ensure maximum consistency in the answers, the entire evaluation was conducted by simulating a specific use scenario, where a single user interacts with a consolidated PKG at a fixed and predefined temporal instant, using an established global context, such as a reference date and time (GLOBAL_DATE/TIME). This approach is useful for the normalization of deictic expressions, such as "today" or "next week," and to allow for precise and unambiguous temporal reasoning. The graph, in this fixed scenario, covers a wide range of heterogeneous personal data, organized into five key dimensions that reflect the cognitive activities of a personal assistant. These include Planning, which tests the management of routines and the detection of scheduling conflicts: specifically, the date was fixed on Monday, September 1, 2025 (1:00 PM).

The actual evaluation dataset consists of 103 carefully crafted queries, each designed to elicit specific complex and realistic cognitive behaviors. The validation targets three fundamental criteria: factual correctness, the ability to link data from different domains, and temporal stability with respect to the fixed reference instant.

To ensure exhaustive coverage of the system's capabilities, the questions were divided into distinct functional classes:

- **Temporal and Conflict Reasoning (Time-Awareness):** These tests verify the system's management of state, event sequencing, and detection of overlaps.
- **Correlational and Multi-Hop Reasoning (Verifiability):** These tests verify that each answer is generated from a justifiable subgraph and correctly integrates information across multiple domains.
- **Grounding and Epistemic Consistency (Anti-Confabulation):** These tests verify that the LLM adheres to the constraint of answering exclusively based on the context provided by the PKG, avoiding confabulation.

Below are some of the 103 questions administered to the system, along with their answers and ratings; 19 questions were selected for which the system was able to perform sufficiently complex reasoning to arrive at the answer.:

Table 3.1: Examples of questions with desired answers, actual answers and human evaluation with a score of correctness.

User Query	**Desired Answer**	**Actual Answer**	**Human Evaluation**
Why there is an alarm set for 5:00 on Saturday morning, did I forget something?	you may have forgotten the "run" event at 6.00 on the calendar	"The alarm set for 5:00 AM on Saturday, September 6, 2025, is a one-time alert with no associated description. While there's no explicit link to a specific event, here's what we know: You have a scheduled ""Running"" activity at 6:00 AM that day. The alarm could be a preparatory reminder."	CORRECT target information. Added some non-essential but HARMLESS details. [5/5]

Table 3.1 – continued from previous page

User Query	**Desired Answer**	**Actual Answer**	**Human Evaluation**
If I started waking up an hour later every day, what recurring commitments would I risk skipping, based on my typical schedule?	your alarm is at 6.00, if you moved it to 7.00 you would risk skipping the gym on Monday, Wednesday, Friday and you could arrive late at work.	"If you started waking up an hour later daily, the recurring commitments most at risk would include: High-Risk (Before 8:00 AM) - 6:00 AM Work Alarm (Weekdays). - 7:00 AM Gym Sessions (Mon/Wed/Fri). Potential Conflicts (9:00 AM Start Times) - 9:00 AM Work Start (Weekdays)."	CORRECT target information. Added some non-essential but HARMLESS details. [5/5]
I have a meeting on September 19th, tell me if I have another overlapping commitment	yes, the meeting overlaps with the flight to Milan from 16:00 to 18:00	"You have a conflict on September 19, 2025. Your ""Job Meeting"" (17:00–18:00) overlaps with your ""Flight to Milan"" (16:00–18:00). The flight overlaps with the final hour of your meeting."	CORRECT target information. No unnecessary details has been added. [5/5]
how did I feel after the last business lunch we had?	you felt calm and serene	Based on the analysis of your personal data, the conclusion remains that you felt calm, tranquil, and positive after the last business lunch.	CORRECT target information. Added some non-essential but HARMLESS details. [5/5]
on what day and at what time did I write the note regarding the Paris itinerary?	at 17.00 on 22/07	"The note titled ""Paris itinerary"" was written on July 22, 2025, at 17:00. This timestamp is recorded in the metadata for ""note_4"" in your system."	CORRECT target information. No unnecessary details has been added. [5/5]

Table 3.1 – continued from previous page

User Query	**Desired Answer**	**Actual Answer**	**Human Evaluation**
who held the August 29 meeting on sustainability?	Luca Bianchi	"The information available indicates that Luca Bianchi authored notes related to an environmental sustainability project, but there is no explicit connection between this project and the ""Job meeting"" held on August 29, 2025."	CORRECT target information. Added some non-essential but HARMLESS details. [5/5]
tell me Luca Rossi's phone number	the number is +39 3416754908	"Luca Rossi's phone number is +39 3416754908. This number is linked to his contact profile and is associated with phone calls recorded between August 28, 2025, and September 1, 2025."	CORRECT target information. No unnecessary details has been added. [5/5]
how long do calls with Luca Rossi last on average?	about 38 minutes and 31 seconds	"The average call duration with Luca Rossi is 38 minutes and 31 seconds, calculated from 4 calls between August 28 and September 1, 2025."	CORRECT target information. No unnecessary details has been added. [5/5]
today Sara Erba called me before or after I arrived at work?	today you arrived at work at 10.50 late due to traffic and Sara Erba called you twice before your arrival	"Assuming your arrival at work at 10:50 AM was on September 1, 2025, the calls from Sara Erba occurred before you arrived. The calls took place at 10:00:10 and 10:30:00, both concluding before 10:50 AM."	CORRECT target information. Added some non-essential but HARMLESS details. [5/5]

Table 3.1 – continued from previous page

User Query	**Desired Answer**	**Actual Answer**	**Human Evaluation**
Where was I on June 13th?	You were in Barcelona, you took a photo of the cathedral	"On June 13, 2025, you visited the Catedral de la Santa Creu i Santa Eulàlia (Barcelona Cathedral) in Barcelona, Spain. This is confirmed by the metadata of your photo taken at 10:00 AM."	CORRECT target information. No unnecessary details has been added. [5/5]
Did we take a group photo in Barcelona?	Yes, there are 2 group photos related to your Spanish stay	"We took two group photos in Barcelona: 1. Cathedral of Barcelona (June 13, 2025). 2. Camp Nou Stadium (June 17, 2025)."	PARTIALLY WRONG target information (it counts 7 people in the group instead of 9). No unnecessary details has been added. [4/5]
What did I do on the fourth day in Barcelona?	You went to Barceloneta and la Rambla	"On your fourth day in Barcelona (June 15, 2025), you visited La Rambla at 1:00 PM and the Barceloneta neighborhood at 8:00 PM. This is confirmed by photo metadata."	CORRECT target information. No unnecessary details has been added. [5/5]
What days have I been to Barcelona?	You were in Barcelona from June 12th to 17th based on photos	"You were in Barcelona, Spain, from June 12, 2025, to June 17, 2025. Your itinerary included visits to the Sagrada Família, Barcelona Cathedral, Casa Batlló, La Rambla, Barceloneta, the Columbus Monument, and Camp Nou."	CORRECT target information. No unnecessary details has been added. [5/5]

Table 3.1 – continued from previous page

User Query	**Desired Answer**	**Actual Answer**	**Human Evaluation**
Can you tell me where I saved the return plane ticket for my trip to Paris?	The file you are looking for is located in downloads/Flight_ticket_back.pdf	"Your return plane ticket for the trip to Paris is saved in your Downloads folder with the filename ‘Flight_ticket_back.pdf‘."	CORRECT target information. Added some non-essential but HARMLESS details. [5/5]
Can you tell me the information about the programs contained in the document to use the household appliance in my photos?	Quick settings are colorful, delicate programs etc.	"Based on the document for your Bosch appliance: Mixed/Colors Laundry is Synthetics cycle at 40°C. Sheets and Towels is Cottons cycle at 60°C. Wool/Delicates is Wool/Hand Wash cycle on Cold."	CORRECT target information. No unnecessary details has been added. [5/5]
what is the most expensive boat in the boat list document added last week?	The XK1 X for almost 13 thousand dollars	"The most expensive boat in the ""Boat Prices 2025"" document is the XK1, priced at £12,595.00."	CORRECT target information. No unnecessary details has been added. [5/5]
did I wake up before the alarm last Sunday?	I have no information to answer this question	"There is no record of an alarm being set for last Sunday. The available data only includes a recurring ""Work"" alarm for weekdays. Since no Sunday alarm exists, it is impossible to determine whether you woke up before an alarm."	CORRECT target information. Added some non-essential but HARMLESS details. [5/5]

Table 3.1 – continued from previous page

User Query	Desired Answer	Actual Answer	Human Evaluation
on my trip to Paris did I respect the itinerary for visiting the eiffel tower?	Yes, because you visited the Eiffel Tower on the day scheduled in the itinerary	"You successfully followed your itinerary for visiting the Eiffel Tower. A photo taken at 22:00 on July 23 at the Eiffel Tower's location confirms the planned activity."	CORRECT target information. No unnecessary details has been added. [5/5]
On my trip to Paris did the plane land on time in Bari?	Yes, because there is a photo taken shortly after the scheduled arrival time	"Your Ryanair flight from Paris to Bari on July 29, 2025, landed on time at 23:00 as scheduled. This is confirmed by your itinerary, a photo timestamped at 23:00 at Bari Airport, and a calendar event."	CORRECT target information. No unnecessary details has been added. [5/5]

Test results analysis The functional tests show that the system is highly competent and robust, capable of managing a wide variety of complex user requests. The analysis of the 81 test cases highlights overall solid performance: 88.9% obtained an "Excellent" or "Good" rating according to human evaluation. The distribution of scores is as follows:

- **Excellent (5/5):** 82.7%
- **Good (4/5):** 6.2%
- **Sufficient (3/5)**: 4.9%
- **Poor (2/5 or 1/5):** 4.9%
- **Failure (0/5):** 1.2%

The main strengths of the system clearly emerge in all reasoning categories:

- **Temporal Reasoning:** The system manages time-related requests consistently and accurately. It correctly detects overlaps between events and schedule changes, reconstructs weekly habits from recurring events, and correctly deduces the sequence of events, such as establishing whether the user woke up before the alarm using call logs.

- **Correlational and Multi-Hop Reasoning:** This is one of the most evident capabilities. The system easily links distant information, retrieves specific details from complex documents (e.g., identifying the most expensive boat in a PDF), reconstructs multi-day travel itineraries from scattered notes, and cross-references photo metadata with calendar events to verify the user's location and activity.

- **Correlational and Multi-Hop Reasoning:** This is one of the most evident capabilities. The system easily links distant information, retrieves specific details from complex documents (e.g., identifying the most expensive boat in a PDF), reconstructs multi-day travel itineraries from scattered notes, and cross-references photo metadata with calendar events to verify the user's location and activity.

- **Adherence to Data and Epistemic Consistency:** The system remains firmly anchored to the available information, avoiding the invention of content. In many cases of missing data, it correctly stated its limitations. It is also capable of verifying user actions by comparing different sources, for example, confirming the punctual arrival of a flight by cross-referencing the ticket with a timestamped photo.

Despite the high success rate, the evaluation revealed some anomalies and limitations, especially in cases requiring deep content understanding or more complex inferences. These points represent strategic areas for intervention:

- **Difficulty in Deep Content Understanding:** The worst performance was related to requests that involved critical analysis of documents and notes. The system incorrectly stated that a note was a complete summary of an article and failed to pinpoint the specific contributions of the text. Similarly, it did not correctly identify the topic of a meeting or the content of a note on emotional status, receiving "Failure" (0/5) and "Poor" (1/5) scores. This suggests that, while skilled at retrieving metadata, the system has limits in reasoning about the deep semantic content of certain documents.

- **Incorrect Correlations and Incomplete Reasoning:** In several cases rated as "Sufficient" or "Poor," the system constructed wrong or partial correlations. It failed to consider all notes on emotional status when analyzing stress patterns, could not perform a simple date-to-day conversion to confirm a birthday, and erroneously deduced that the user's "home" was a visited location. These errors indicate a difficulty in constructing a complete picture before formulating an answer.

- **Minor Factual and Contextual Errors:** A smaller group of anomalies relates to less severe inaccuracies, such as an incorrect count of people in a photo, inability to identify a specific location from the image context, or minor logical errors in temporal reasoning. While not severely compromising functionality, these errors reduce overall reliability.

Chapter 4

Visual Analysis Module

This chapter analyzes the architecture and functioning of the Visual Analysis Module, a core component of the system, designed to answer questions that require the interpretation of visual content. In a system whose goal is to act as a personal assistant based on a PKG, the ability to handle only textual information represents a significant limitation. Much of personal memory and information is inherently visual: photographs of events, places visited, and people met. The Visual Analysis Module was conceived to fill this gap, equipping the assistant with the ability to "look" at the user's images and extract factual information from them to answer specific queries.

The integration of this capability, however, is not trivial. It is not simply a matter of connecting a Vision-Language Model (VLM) to the chat interface. The true engineering challenge, which this chapter will address in detail, lies in the orchestration: how and when does the system decide that an answer requires visual analysis? How does it avoid analyzing hundreds of images inefficiently? And how does it fuse the visual information with the textual and relational context already present in the user's graph?

The answer to these questions lies in the chosen architecture: the visual analysis module is not an autonomous component, but a specialized tool available to a broader ReAct (Reasoning + Acting) agent: the entire decision-making flow is governed by an agent built with LangGraph. This agent coordinates various tools, including GraphRAG for textual searches and, precisely, the img_deep_analysis module. The fundamental aspect of this design is that visual analysis is not the first choice, but a fallback strategy. The agent first attempts to answer using GraphRAG; only if this textual search fails (a failure actively monitored by an internal "evaluation" node) does the agent decide to invoke the visual tool, recognizing that the answer might be found within the images.

Once the agent has established the need for visual analysis, the process is divided into two distinct macro-phases, which correspond to the subsequent sections of this chapter:

- **The first phase** is Intelligent Image Retrieval. To avoid analyzing the user's entire photo archive, the system reuses the GraphRAG engine in a specialized mode. Instead of generating text, its task becomes that of a "structural search engine": leveraging the PKG connections (links between photos, events, people, and places), GraphRAG selects a small set of candidate photos that are semantically and relationally most relevant to the user's query. This step transforms the LLM from a generator to an entity selector.
- **The second phase** comprises Visual Reasoning and Question Answering (VQA) and begins where the first one ends. The system takes the IDs of the photos selected by GraphRAG, resolves them into concrete file paths, and passes them, one by one, to a Vision-Language Model (VLM). Here, the model does not perform simple description (captioning), but receives a specific prompt asking it to answer the user's original question based exclusively on the content of that single image. The positive answers are finally aggregated and returned to the ReAct agent, which presents them to the user.

This chapter, therefore, describes not just a vision model, but an entire end-to-end pipeline: from the strategic decision of a ReAct agent to use visual analysis, through the intelligent selection of images via GraphRAG, up to the applied multimodal reasoning for answering specific questions.

4.1 ReAct agent architecture

The visual analysis module, as already mentioned, is a tool available to a ReAct agent that governs all the system's "intelligent" decisions; to understand when and how the image analysis component intervenes in the pipeline, therefore, one must first look at how the agent itself is constructed. The core idea is that there is no single model answering the user's question, but an agent coordinating several specialized tools, including GraphRAG, the point-of-interest recommender, and the img_deep_analysis tool, following a state-based logic and using the language model not only to generate text but also to decide what to do step by step. This setup is characteristic of the ReAct paradigm: natural-language reasoning and tool-based actions alternate in an iterative cycle until the agent believes it has gathered enough information to return a satisfactory answer.

From a flow perspective, everything begins when the user sends a message to the /chat endpoint of the API. The process_user_command function acts as the bridge between the client and the conversational graph. The message is first stored in the history through the chat_history_service, which both records the conversation history and injects a system prompt with temporal context. This means that even before the ReAct agent is activated, the interaction has already been made stateful at the database level: each user has a sequence of stored messages that can be

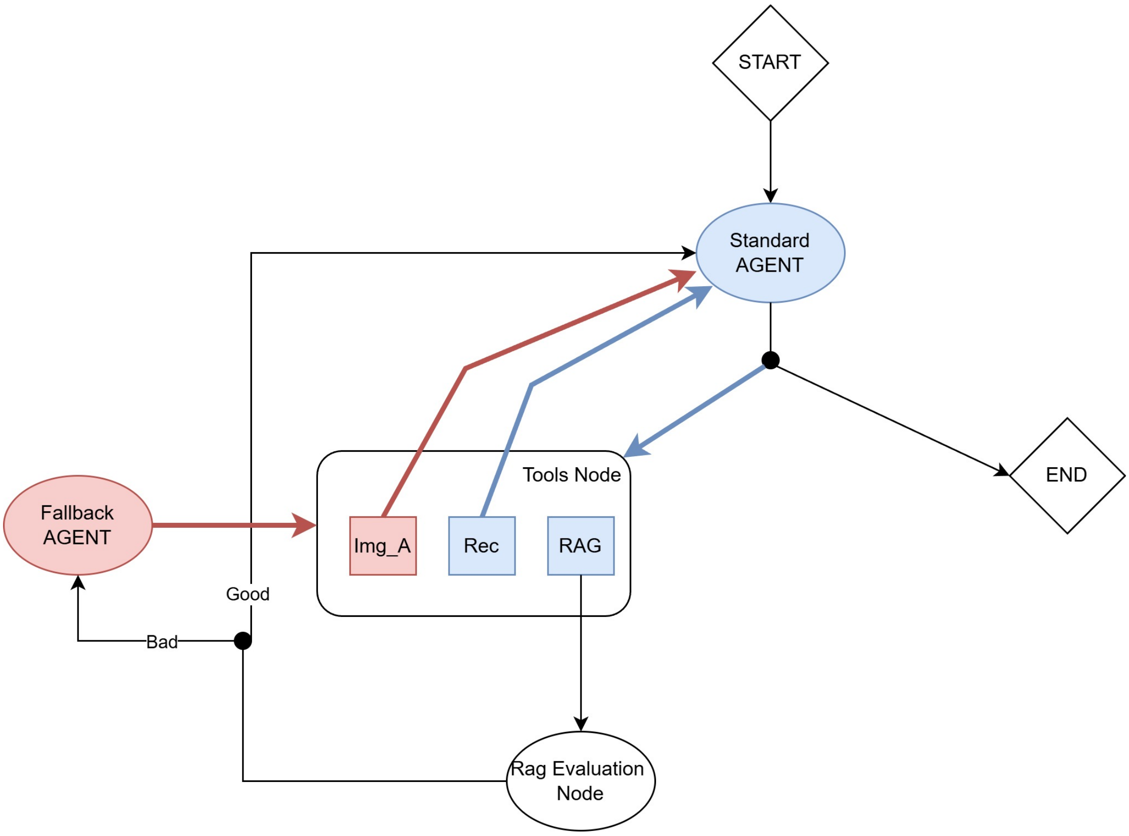


Figure 4.1: Agent's graph structure with nodes (tools and agents) and edges between them.

retrieved and used when needed by any component, including GraphRAG and the image analysis module, independently.

Once persistence is handled, process_user_command retrieves or initializes the actual agent, and here the agent_cache and LangGraph's MemorySaver come into play. The cache is a FixedSizeDict mapping user_id to a tuple composed of the agent graph and the RecommenderAgent; if a graph already exists for a user, it is reused, otherwise a new one is created. MemorySaver functions as a checkpoint system for LangGraph: it allows the graph to remember, between invocations, the internal state associated with a given thread_id, which in this case coincides with the user's identifier. This dual mechanism (high-level cache plus graph memory) enables a truly persistent agent in which not only messages are saved to the database, but the state machine making decisions can continue its flow by considering earlier turns.

During agent creation, a crucial component called WorkflowConfigs is provided as an argument. It contains the tools specifically initialized (based on a given user_id) to define the environment in which the agent will operate. It receives as input the user_id and the already initialized RecommenderAgent, and from these elements builds the tools that the agent may invoke; the tools are implemented as asynchronous functions nested inside WorkflowConfigs and decorated with @tool. Each tool can easily access the user-specific resources (multi-tenant database, embedding model, temporary directory with GraphRAG output) while maintaining an abstract interface compatible with the model's tool-calling mechanism.

Having outlined the ReAct agent's general functioning, this section focuses on the img_deep_analysis tool, which is the entry point of the entire visual analysis flow. From the agent's perspective, this tool is described in very generic terms: "Useful when the user requests images analysis. Takes in input the user question and, first searches the best images to analyze between those stored in DB (matching the user query), then analyze them with a Visual Analyzer AI model and returns a response." Once invoked, it never competes with other tools: it is always called only after GraphRAG fails to provide a satisfactory answer, so the agent is never faced with choosing between it and other tools.

Next, the agent reads the user's conversational history from chat_history_service and passes it to the GraphRAG engine in a special mode, mode="img_deep_analysis". In this mode, GraphRAG is not used to generate a discursive answer but to select the most relevant photos for the question, using a dedicated system prompt that forces the model to return only a list of photo identifiers. Therefore, the agent does not "see" images directly: it delegates to GraphRAG the task of identifying which photos might contain the information requested by the user. Once the search is done, img_deep_analysis extracts the photo_id values from the returned bullet list and passes them to the img_analyzer, which performs the actual visual reasoning.

Alongside the image analysis tool, WorkflowConfigs also defines the two tools considered "primary" from the agent's perspective because they may be used in the first phase of the graph: graph_RAG and PoI_recommender. The former encapsulates the standard use of the GraphRAG subsystem, invoked with mode="rag", and thus focuses on retrieving and synthesizing information from the user's personal PKG: documents, events, notes, and so on. The latter relies on the RecommenderAgent to produce points-of-interest and travel-destination suggestions using its own model and dataset. In the ReAct architecture, these two tools represent the agent's "main moves," the ones it tries first. The img_deep_analysis tool, instead, is registered as a fallback_tool: it is not part of the default strategy but belongs to a second line of intervention designed for cases where a purely textual answer based on the PKG is insufficient.

Once the tools are defined, the ReactAgent class constructs the agent's logical graph: the constructor receives WorkflowConfigs, extracts the list of primary tools and the fallback one, and initializes the language model based on AGENT_LLM_CONFIG through the get_llm function. This function instantiates the correct client depending on the configured provider (OpenAI, Groq, Gemini, Anthropic, Ollama, etc.), setting common parameters such as temperature zero for deterministic behavior. The tools are then bound: the LLM is "augmented" with the tools through bind_tools, producing two distinct versions of the model, one aware only of the primary tools and one aware only of the fallback tool. This allows the agent to precisely control which tools are available in each flow phase, reducing the risk of the model calling unwanted tools at inappropriate times.

The core of the ReAct behavior lies in the graph nodes defined in ReactAgent: the standard_agent node is the logical starting point because it receives the current state, which contains the sequence of messages and a possible previous rag_evaluation, and passes it to the LLM augmented with the primary tools. From there, the model may generate a direct answer or a tool call. A peculiarity of the LangGraph framework is that the model's output is reinserted into the state as a new message, and the routing logic interprets it. In this sense, standard_agent fully embodies the ReAct philosophy: the LLM reasons on the linguistic context, decides whether it needs to act, and, if so, specifies which action (which tool) to perform and with which parameters.

The node that executes actions is ToolNode, added to the graph during construction. Unlike standard_agent, which calls the model, ToolNode performs no reasoning: it inspects the latest message produced by the model and, if it contains one or more tool_calls, invokes the corresponding Python functions, collecting their results as new messages. This split between a "cognitive" node and an "executive" node is key to keeping the flow transparent and controllable. The router router_after_llm orchestrates the transition: after each standard_agent execution, it checks whether the last message contains tool calls; if so, it directs the graph to the tool node, otherwise it ends the flow. If the model believes it already has a satisfactory answer, the interaction ends; if it decides it must consult an external source (GraphRAG or the Recommender), the flow continues with the action phase.

Once a tool has been executed, control passes to the router router_after_tools, which decides the next step based on which tool was just called. The idea is to distinguish between responses produced by graph_RAG and those produced by other tools. If the last message comes from graph_RAG, the router does not return immediately to standard_agent but diverts to evaluate_rag. In all other cases, for instance, after a PoI_recommender call, the flow returns directly to the main node, allowing the model to integrate the tool's output into its final user answer. This distinction reflects a specific architectural choice: the system considers GraphRAG responses potentially fallible and subject to quality-checking by the visual analysis component.

The evaluate_rag_response node is an internal "judge," also LLM-based, that evaluates the quality of the latest GraphRAG answer. It takes the user's original question and the tool's returned text, builds an evaluation prompt instructing the model to classify the answer as "good" or "bad," and includes examples of vague or unsatisfactory replies. The LLM is thus used not to generate a user answer but to assess whether another component did its job. The result, one word, is cleaned of any artifacts (e.g., thinking tags) through a regular expression and stored in the rag_evaluation field of the state. If the response is not explicitly “good” or “bad,” an exception is raised, showing how strongly the whole flow relies on well-defined textual contracts with the model.

Below is the prompt used in the RAG response evaluator:

```
‘‘Given the user's original question and the response provided by an AI agent,
evaluate whether the agent managed to answer the question or not.
User Question: "{user_question}"
AI agent Response: "{tool_output}"
If the AI agent says it cannot answer the question due to lack of information,
or gives a vague answer, respond with the word ‘‘bad.’’
If it manages to fulfill the user’s request, respond with the word ‘‘good.’’
Here are some examples of unsatisfactory or vague answers:
●       ‘‘I don't have information about...’’
●       "There's no mention about ..."
●       "There's no data about ..."
●       "I'm sorry but ..."
●       "It seems there was an issue retrieving the data ..."

Reply with a single word only: ‘‘bad’’ if the response is unsatisfactory,
or ‘‘good’’ if it is satisfactory.’’
```

It is precisely from this evaluation that the visual analysis module’s entry point in the ReAct flow emerges. The router router_after_evaluation checks the rag_evaluation value: if it is “bad,” the state machine determines that the PKG-based textual search did not adequately answer the question, and hands control to the fallback_agent node. If instead the response is “good,” the flow simply returns to standard_agent, which may conclude the conversation or invoke other tools. The fallback_agent node is itself a small LLM agent but tied exclusively to the img_deep_analysis tool. Before calling the model, it adds an artificial human message explaining explicitly what happened: “The previous search for information failed or yielded a vague answer. You must now use the ’img_deep_analysis’ tool to find a better answer to the user’s original question.” instructing the model to use the image analysis tool to find a better answer to the user’s question. Again, the behavior is aligned with ReAct: the LLM reflects on the previous failure and decides a new action, but the set of available actions is restricted to the only tool suitable for visual analysis, reducing ambiguity.

All this logic resides in the create_agent method, where the actual LangGraph graph is built. The standard_agent, tools, evaluate_rag, and fallback_agent nodes are added, each with its corresponding function. The control flow is then defined: from the start node to standard_agent; from there, depending on the output, to tools or termination; after tools, the router leads either to evaluate_rag or back to standard_agent; after evaluate_rag, it chooses between fallback_agent and standard_agent; finally, from fallback_agent it returns to tools to execute the fallback tool call. The graph is compiled with the provided checkpointer (in this case, the MemorySaver), producing an executable object that can be invoked with an initial state and a configuration specifying the thread_id. Each invocation of the graph is thus a mini "reasoning episode," but the use of checkpoints links episodes together for the same user.

Considering the broader picture, the ReAct agent architecture enables the visual analysis module: the img_deep_analysis tool becomes a specific action the agent may decide to perform when it recognizes that the "textual" path is insufficient. The decision is not hard-coded but emerges from a sequence of steps guided by the model itself: the choice to use GraphRAG first, then the evaluation of its answer, and finally the fallback. This makes the system more flexible but also more dependent on the quality of the instructions provided to the LLM. If, on one hand, the modularity facilitates extending the system to other tool types (for example, future audio or video analysis modules), on the other it requires careful design of prompts and model contracts, since a wrong decision may prevent visual analysis from being used when needed, or cause it to be invoked unnecessarily. Overall, however, the ReAct agent provides a coherent and scalable structure within which the visual analysis module can operate as an integral system component rather than an isolated element.

4.2 graphRAG img retrieval

In the visual analysis module, image retrieval is the first of the two macro-operations required to obtain the final answer. In this phase, GraphRAG is not used to directly produce a natural-language answer, but rather as a "structural search engine" that determines which photos are truly worth analyzing in depth. This represents an important shift in perspective: instead of asking the visual model to indiscriminately inspect all the user's images, the system leverages the graph to select a small set of candidate photos, maximizing both relevance to the question and computational efficiency.

To understand how this variant works, it is useful to recall the general structure of GraphRAG described in the survey by Han et al. [8], where the system is decomposed into query processor, retriever, organizer, generator, and data source. In a "classic" GraphRAG setup, the retriever takes the processed query, queries a graph-structured data source, and returns structured content that is later transformed into textual context for the LLM. This process is richer than traditional RAG because it does not rely solely on semantic similarity over chunks of text, but uses entities, relationships, paths, and communities in the graph to capture semantically rich information. A similar mechanism happens in our chatbot, but with a more specific objective: not to answer the question, but to identify the "photo" nodes in the graph that are most likely to contain the answer when visually analyzed.

From an implementation standpoint, this role is delegated to the chatbot function of the graphRag_chat.py module, which represents the sole entry point to the local search engine based on the GraphRAG library. The function is called both in standard mode (mode="rag") when the graph_RAG tool must answer a textual query, and in the specialized mode (mode="img_deep_analysis") when the img_deep_analysis tool needs a shortlist of photos to pass to the visual analyzer. The key detail is that, in both cases, the underlying retrieval pipeline is the same: a local context is built around the user's graph and combined with the conversational history to guide the LLM. What changes are the system prompt and the expected type of output, and this single choice completely transforms the role the retriever plays within the system.

When the chatbot function is invoked, it first initializes two models, a chat model and an embedding model, using the LanguageModelConfig class of the GraphRAG framework. The model information (provider, model name, API key) is retrieved from configuration variables, like the LLMs used in the LangGraph agent, but here the focus is exclusively on the retrieval and generation components of the GraphRAG subsystem. In parallel, a tokenizer compatible with the chat model is retrieved; this detail matters because the local search must balance how much of the graph can be compressed into a single prompt while respecting the model's token limits.

Immediately after, the chatbot function loads from the filesystem the various Parquet tables generated during indexing:

- communities.parquet
- community_reports.parquet
- entities.parquet
- relationships.parquet
- text_units.parquet

These files represent the different "projections" of the PKG built from the Neo4j database: entities (people, events, places, photos, etc.), relationships among entities, chunks linked to entities, communities, and their corresponding summary reports. GraphRAG provides adapters (read_indexer_entities, read_indexer_relationships, read_indexer_reports, read_indexer_text_units) that transform these tables into internal data structures optimized for search. In parallel, a LanceDB store is opened for the embeddings of entity descriptions, indexed through LanceDBVectorStore with a dedicated index name. This store enables the retriever to perform similarity search not only on plain text but also on the dense representations of entity descriptions, which is particularly useful when the query mentions a person, event, or object that exists in the graph with a rich but not necessarily literal description.

On top of these components, the LocalSearchMixedContext object is built, which constitutes the core of the local search. The idea is to combine all this information so the retriever can move beyond simple semantic similarity: it may take into account which entities are central in a community, how they are connected, which relationships are weighted more strongly, and even how the question fits within the flow of previous conversation. The local_context_params configuration finely regulates this process: some parameters control the mix between text units and community reports (e.g., text_unit_prop and community_prop), others limit the number of candidate entities and relationships included (top_k_mapped_entities, top_k_relationships). Additional parameters determine how much previous conversation influences the context (e.g. conversation_history_max_turns, conversation_history_user_turns_only) and flags specify whether to add metadata such as entity rank, relationship weight, or community rank. Finally, max_tokens imposes an upper bound on the length of the generated context, forcing the algorithm to select only the most relevant portion of the graph. This configuration represents a delicate trade-off: if parameters are too permissive, the model becomes overwhelmed by irrelevant information; if too restrictive, it risks missing the "photo" entities that contain the answer. This trade-off is important to acknowledge explicitly because it directly impacts the quality of image selection.

Conceptually, LocalSearch implements the typical GraphRAG pattern: given the input query text and the conversation, it queries the context_builder to obtain a relevant subgraph and a set of selected textual units, organizes them into a tabular structure (the "Data Tables" included in the system prompt), and finally builds a prompt for the LLM containing both the question and the structured context. The model does not see the graph in node-edge format but as tables of entities, relationships, communities, and associated text: this is how the framework enables the LLM to "reason" over the graph structure while staying within a text-in/text-out paradigm.

The fundamental difference between rag mode and img_deep_analysis mode lies precisely in how LocalSearch is parameterized, and especially in the system prompt. In rag mode, LocalSearch is constructed without specifying a custom system prompt and with response_type="multiple paragraphs", meaning the model is guided to produce a discursive natural-language answer that will be directly returned to the user (and evaluated by the ReAct agent through the evaluate_rag_response node). In other words, in rag mode GraphRAG plays its classic RAG-on-graph role: it decides what to retrieve from the PKG and synthesizes a textual response.

In img_deep_analysis mode, however, LocalSearch is initialized with a dedicated system prompt, IMG_DEEP_ANALYSIS_SYSTEM_PROMPT, and with response_type="bullet points". The prompt specifies that the assistant must reply only with the list, without additional comments, and that photos must be named strictly using their internal graph identifiers in the photo_id format. A concrete example is also provided. Finally, the prompt covers the case where no useful photo exists and instructs the assistant to output “No photo matches the question”.

The complete prompt is shown below:

```
"---Role---
You are a helpful assistant that selects a list of photos that could be useful
to answer the question.
---Goal---
Generate a response that contains a list of up to 3 photos that could be useful
to answer the question if analyzed in-depth, using the context provided in the
Data Tables.
You must reply only with the list of photos and nothing else, avoid other
unnecessary comments.
The photo entities in the Data Tables have a name in the following format:
photo_id.
If there are fewer than 3 "photo" entities that can be useful for the
question, list fewer.
If there are no photos useful for the question, reply "No photo matches the question".
---Examples---
"user": "Was I wearing blue shoes to Kevin's birthday?"
"assistant": "- photo_5231
- photo_7343
- photo_5241"
---Data tables---
{context_data}
---Target response length and format---
{response_type}"
```

This design has important implications: it transforms the LLM from a generator of answers into a selector of entities, since the output is no longer argumentative text but a list of keys that allow subsequent components to physically retrieve the images. The LLLM is encapsulated into a more controlled task that reduces the space for classic textual hallucinations, while still leaving room for selection errors if the graph is noisy or incomplete. Moreover, the requirement to use only photo_id identifiers ensures the output is easily parsable by the code.

From the retrieval perspective, however, the behavior of local search remains conceptually identical. Starting from the question, the context_builder selects relevant entities and relationships in the graph, using both similarity between entity-description embeddings and structural signals such as community centrality or degree of connection. In our system, “photo” entities are not treated specially by the retriever code; rather, they naturally emerge as graph nodes connected to events, people, places, and texts. This means the success of the “GraphRAG image retrieval” depends on an assumption worth stating explicitly: the graph-building phase must extract and connect photos correctly. If images are not represented as well-linked photo entities associated with events and people, the local search may judge no photo relevant, even if the correct one is present in the PKG.

This architecture has the advantage of reusing the same GraphRAG index for two different purposes, but it also has a conceptually interesting consequence: the image is never retrieved directly from Neo4j or the filesystem via manual queries. It is always mediated by GraphRAG, which determines whether the photo is relevant in light of semantic and relational context. This differs significantly from metadata-based or tag-based search approaches; the goal here is to use the graph structure to capture implicit relationships that would not surface through purely textual search.

Once GraphRAG returns the list of photo_id values, the img_deep_analysis tool passes it to the analyzer module, which resolves the filesystem paths of the images and invokes the visual model through img_description. If GraphRAG selects good photos, the multimodal model can answer very specific questions about the user’s past robustly; if the selection is weak or incorrect, the visual analysis, however powerful, cannot compensate for the lack of relevant images. The next chapter, dedicated to visual reasoning and question answering, completes this picture by showing how descriptions of individual images are integrated into a coherent final answer.

4.3 visual reasoning e question answering

The visual reasoning phase fits exactly where the flow described in the previous paragraph ends: the LangGraph agent, after querying the GraphRAG subsystem in “img_deep_analysis” mode, receives as output a structured string, photo_list_str, which is a textual list of photo_id values that, according to the graph, are the most promising images for answering the user’s question.

The concrete starting point is the img_deep_analysis function defined in conversation.py within the class that configures the agent. This tool is seen by the LangGraph graph as one of the possible “actions,” and it is selected by the ReAct agent when it detects an intent explicitly related to images. After the GraphRAG phase, img_deep_analysis receives in input the natural-language question and the conversational context and calls the asynchronous chatbot function with mode= "img_deep_analysis". It is this mode that, as explained earlier, instructs GraphRAG to return a list of photos in the form of bullet points, each line corresponding to a single photo_id.

Once photo_list_str is obtained, the tool does not immediately proceed to answer generation: it first converts the string into a “clean” list of identifiers. The logic is intentionally simple but strict: the text is split into lines, and each line is stripped of its prefix to extract the actual image id. This encodes a protocol between GraphRAG and the visual module: the retrieval system must respond only with a bullet list of ids, without any additional text. This makes the state transfer between modules very robust.

The output of this transformation is a photo_ids list containing only strings such as photo_123, photo_456, and so on. At this point, the actual visual analysis module enters the scene: img_deep_analysis delegates the rest of the processing to img_analyzer, a function defined in analyzer.py, to which it passes three critical pieces of information: the user's original question, the list of selected photo identifiers, and the user's numeric identifier (necessary for resolving the tenant and temporary directories). This is where the actual visual reasoning flow begins.

The module's first subcomponent is imgs_temp_path_resolver, an asynchronous function responsible for translating logical identifiers into real file paths accessible by the model. Photo_id values are not filesystem paths; they are nodes in a Neo4j graph. Therefore, for each id, a Cypher query is executed through the "cypher" service, using a predefined query template (QUERY_TO_RESOLVE_IMG_NAME) and passing the photo_id as parameter. The result returns the filename associated with that node, typically something like IMG_2342.jpg or family.png, retrieved from the "fileName" property of any Chunk node connected to the photo.

This resolution step is the bridge between the symbolic dimension of the PKG and the "physical" layer of the system. As long as one works with logical ids, reasoning remains lightweight but abstract; when visual reasoning is required, a concrete correspondence between graph and filesystem is needed. It is imgs_temp_path_resolver that constructs this correspondence, concatenating the user's database name, the temporary directory TMP_DIR, and the filename obtained from Neo4j. The result is a list of paths such as ./data/tmp/user-42/family.jpg, which the visual description module can directly open from the server storage.

Once this resolution is complete, img_analyzer constructs the prompt to pass to the vision-language model. Here the system makes a clear design choice: instead of sending the original question to the VLM as-is, it creates a composite prompt, IMG_DEEP_ANALYSIS_PROMPT, which embeds the question inside a frame of instructions. The prompt is shown below:

```
‘‘QUESTION: {question}
Given this question analyze the image and, if it's possible,
return a clear and brief answer.
If not, only respond ‘I can't respond to this question.’’’
```

This design has two interesting implications. On one hand, it steers the model toward a more controlled behavior: instead of producing "generic captioning," the VLM is pushed to reason in a question-answering mode, searching for specific relationships between the query and the visual content. On the other hand, it introduces a form of self-assessment: the model is explicitly allowed to declare its uncertainty. This mechanism can reduce fabricated answers when the visual signal is ambiguous or irrelevant.

At this stage, img_analyzer enters an iterative phase: for each resolved image path, it calls the get_img_description function defined in img_description.py, passing the deep analysis prompt, the image path, and the configuration string of the vision-language model, IMG_ANALYSIS_VL_CONFIG. Each image is analyzed independently, using the same question but different visual content. This "one-by-one" approach has a higher computational cost than joint analysis but greatly simplifies the design: multimodal batching is unnecessary, and the fusion logic is left to img_analyzer, which can decide how to combine or filter responses.

Inside get_img_description, the actual interaction with the model takes place. The function opens the image file, reads it in binary form, encodes it in base64, and constructs a data URI that can be passed to a multimodal endpoint. The model configuration is extracted from VL_model_env_value, a string encoding the provider, model name, possible extras, and API key. This is where the model choice comes into play: in the deployment configuration, the model *meta-llama/llama-4-maverick-17b-128e-instruct* exposed by Groq is used as a multimodal chat model due to its versatility provided by a Mixture-of-Experts of 128 experts.

The call to the Groq client uses the chat.completions.create endpoint, setting the model_name and passing a single user-role message whose content is a list composed of two parts: a text block containing the prompt built by img_analyzer and an image_url block containing the data URI with the encoded image. In this way, the model receives both the question text and the associated image, enabling multimodal reasoning. No streaming or custom stop sequences are used, and the output is extracted directly from the first result's content field.

Conceptually, at this stage the system delegates the entire "visual reasoning" task to the VLM. All graph-structural and user-context information has been used earlier to select which images to analyze; but once get_img_description is invoked, the VLM only sees the prompt text and the image content. It has no access to the PKG's relationships, the graph structure, or the conversation history beyond what is embedded in the prompt. This is possible because GraphRAG acts as a semantic filter and context selector, while the VLM acts as the "eye" on the visual content.

The downside is that the reliability of the final visual answer becomes highly dependent on the chosen model's capabilities. *meta-llama/llama-4-maverick-17b-128e-instruct* is a state-of-the-art model offering strong multimodal abilities, but if it were replaced with a less robust or less aligned model, the overall system behavior would degrade at this stage even though the rest of the architecture remains identical.

Once the answer for a single image is obtained, get_img_description returns the string to the caller. img_analyzer filters it: if the answer is exactly "I can't respond to this question.", it is discarded; otherwise, it is added to a list of descriptions. This introduces an additional robustness layer: if some images are misleading or irrelevant, the model may correctly declare its inability to answer, and that evidence will be ignored in the synthesis. It is a mechanism used to discard an image that is not relevant to the final reasoning.

After analyzing all images, the module must transform the individual responses into a single text answer. If no valid description has been collected, img_analyzer returns a fallback message communicating the inability to answer the question based on the available images: it does not attempt to invent anything based solely on text, since this tool is explicitly invoked for visual analysis; if no useful visual signal exists, the pipeline prefers not to answer rather than mislead the user.

If at least one description has been produced, the responses are concatenated into a single structured string: the string begins with “Answers by analyzing different pictures:” followed by the list of descriptions generated by the VLM to be passed to the agent for the final answer. The implicit idea is that the photo_id values selected by GraphRAG are coherent enough to produce compatible descriptions, and that the agent’s final answer can rely on multiple simultaneous outputs from the vision model, yielding a unified and satisfactory response.

From a question-answering perspective, this schema produces responses that already contain “reasoning” within the visual domain. Although the VLM’s task is to try to answer that specific question rather than generically describes the scene or other parts of the PKG, the symbolic reasoning about which photo to analyze is delegated to GraphRAG, which already has all the necessary context to answer precisely and factually. The final natural-language fusion is a simple assembly process delegated to the standard_agent, whose role is to deliver the answer to the user based on the output of the visual analysis system.

The end-to-end flow of visual reasoning and question answering, therefore, begins with the LangGraph agent, which activates img_deep_analysis when it detects a visual intent; GraphRAG, in its specialized mode, returns a list of potentially relevant photos; the parser in conversation.py extracts the ids and passes them to img_analyzer; the resolver translates these ids into real image paths; for each image, a multimodal prompt is built and sent to the vision model; and the responses are filtered, aggregated, and returned as the tool’s final output. The overall reliability of this chain therefore depends on three main factors: the quality of GraphRAG’s selection, the correctness of the path resolution, and, above all, the ability of the chosen VLM to interpret the images in a manner coherent with the question. If any of these links is weak, the final visual answer will be unhelpful or misleading.

Chapter 5

Testing and results discussion

This chapter is dedicated to the testing and evaluation of the personal assistant extended with the visual analysis module, with the goal of quantifying and analyzing the added value introduced by the multimodal component. Two datasets of questions were created following the methodological approach defined in the previous chapters: basing the evaluation on a single user and a single simulated Personal Knowledge Graph, with all interactions anchored to a fixed temporal instant to ensure reproducibility and consistency of the reasoning process.

5.1 Evaluation protocol

The goal of the evaluation protocol was to systematically verify not only the correct functioning of the language module, but the overall quality of the personal assistant once extended with the visual analysis module. All experimentation was designed in continuity with the setup defined in the previous chapters: a single user, a single Personal Knowledge Graph as factual base, and a fully specified usage scenario fixed at a global temporal instant. All interactions are anchored to the same reference date and time, Monday, September 1st, 2025, at 13:00, used as GLOBAL_DATE/TIME. As stated, this constraint makes it possible to normalize deictic expressions such as “today,” “next week,” or “yesterday,” and to make the assistant’s temporal reasoning verifiable and reproducible, eliminating ambiguities related to the passing of real time.

The PKG, constructed before the testing phase, collects heterogeneous data about a single user across multiple dimensions: calendar events, notes, documents, photos, calls, phone contacts, alarms, and calendar entries. It has been described how, from this knowledge base, a dataset of 103 textual queries was created to probe the limits of the system in terms of temporal reasoning, correlational reasoning, and coherence. These questions were designed to be solvable exclusively using the PKG, without relying on visual capabilities or external sources, so the assistant should be able to answer based solely on the PKG-provided context. On this dataset, the system was evaluated using a five-level scale, from “Failure” to “Excellent,” where a numerical score between 0 and 5 expresses the degree of factual correctness, completeness, and adherence to the available knowledge. The results of this first campaign, which show robust behavior, form the baseline for understanding the added value of the visual analyzer.

The introduction of the visual analysis module required revisiting the evaluation protocol to answer a different question: in which cases do direct access to images allow the system to overcome the limitations of the PKG and traditional GraphRAG, and in which cases is visual analysis alone still insufficient? The visual questions were therefore designed in strict continuity with the existing scenario, leveraging the same events, places, and photos already present in the PKG, but modifying the type of information requested to truly test the system’s ability to see and reason over images.

Starting from the original dataset of 103 queries, two distinct but complementary sets of questions were constructed: the first, called “PRIMARY VISION DATASET (PVD) – Visual questions to test the Visual Analyzer in cases in which GraphRAG fails,” and the second, “REDUNDANT VISION DATASET (RVD) – Visual questions to test the Visual Analyzer in cases in which GraphRAG has enough data.”

PRIMARY VISION DATASET (PVD) This dataset contains 29 questions designed to be impossible to solve for a system relying exclusively on the PKG. These are queries whose answer necessarily requires direct analysis of one or more images: the color of a piece of clothing, the number of people visible in a photo, the presence or absence of a specific object in the background, the direction in which an individual is looking, and so on. These details are not encoded in the graph because the img_description module, which generates descriptions later converted into triples for the PKG, does not always capture every useful detail, and therefore GraphRAG is destined to fail.

The creation of these 29 questions was not arbitrary. Starting from the same narrative scenario of the textual dataset, situations were identified in which the PKG provides only a coarse-grained context (e.g., “A group of tourists pose in front of the iconic Cathedral of Barcelona…”). By asking questions that require deeper visual analysis (e.g., “During our trip to Barcelona, were there more males or females in our group?”), the PVD directly measures how far the visual analysis module can go in filling an informational gap in the graph. To make this structure explicit, each question is accompanied by the natural-language query, the expected answer defined a priori based on the scenario and images, the actual answer produced by the system, and a human evaluation of correctness.

The structure of the dataset clearly reflects its role and is as follows:

- **The first column** contains the natural-language formulation of the question ("USER QUERY"), consistent with real user requests.
- **The second column** contains the "DESIRED ANSWER," manually defined as the ground truth against which system outputs are compared.
- **The third column** records the "ACTUAL ANSWER" produced by the system when the visual analysis module is given access to the relevant images and can combine their content with the rest of the pipeline.
- **The fourth column**, "HUMAN EVALUATION," contains the human qualitative evaluation of the response based on strict criteria, with a score from 0 to 5.

The evaluation criteria remain the same used for the textual dataset: the primary focus is on the correctness of the "target information" and on the absence of invented or unjustified details unsupported by either the PKG or the image. A score of 5 indicates a fully correct answer that reports exactly the expected information without adding conjectures. Intermediate values such as 4 or 3 are used when the answer identifies the main point but contains omissions, imprecise formulations, or mild ambiguities that do not entirely compromise usefulness. Low scores between 0 and 2 indicate errors in interpreting the visual content, incorrect attribution of attributes (such as a wrong color or object confusion), or evasive responses that do not address the core question.

REDUNDANT VISION DATASET (RVD) The second dataset, called "REDUNDANT VISION DATASET (RVD) – Visual questions to test the Visual Analyzer in cases in which GraphRAG has enough data," was designed with a conceptually different objective. While the PVD aims to evaluate system performance when RAG lacks sufficient information and therefore depends entirely on visual analysis, the RVD evaluates the robustness of the visual analyzer when used as a redundant instrument: what happens when the visual analyzer is used even when it is not strictly necessary, because GraphRAG already has all the required information to answer satisfactorily? In other words, this dataset tests not only the power of the vision module but especially its ability to integrate itself in a useful, and not harmful-manner within a reasoning pipeline that is already competent.

The evaluation in this second phase does not simply judge the final multimodal system output as if it were independent: rather, it explicitly centers on the contribution of the visual analyzer. When assigning evaluations, having access both to the GraphRAG answer and to the multimodal answer, the key question becomes whether and how the visual module modified the quality of the response. Practically, the relevant questions are: Did the visual analyzer add genuinely useful information, such as concrete details not explicitly stated in the PKG but nonetheless coherent with the scenario? Did it reinforce the confidence of the answer by visually confirming what the graph had already inferred? Or did it introduce irrelevant, redundant, or even incorrect information, degrading clarity and reliability? It is also possible that the visual analyzer was effectively ignored by the model, producing an answer nearly identical to GraphRAG's; in this case, its contribution is neutral, and the presence of the visual component does not change the substantive evaluation.

The redundancy is therefore very useful for deepening the analysis: each answer (dataset row) is evaluated by following these questions:

- When does the Visual Analyzer help RAG in answering?
- When does the Visual Analyzer introduce only noise?
- When do the answers of the two tools agree?
- When do both tools fail?

The construction of the dataset is based on a strong constraint: the 14 selected questions must be theoretically solvable by GraphRAG alone, since all necessary information is already present in the Personal Knowledge Graph. This means that each query is designed so that the correct answer is already "implicitly" or "explicitly" encoded in the entities, relationships, or attributes of the PKG. In the REDUNDANT VISION DATASET, the structure of each row is duplicated with respect to the PRIMARY VISION DATASET to allow comparison between GraphRAG and the Visual Analyzer.

For each query, the dataset includes:

- **The first column:** natural-language formulation of the question ("USER QUERY").
- **The second column:** "DESIRED ANSWER," the manually defined ground truth.
- **The third column:** "GRAPH_RAG ANSWER," the answer produced solely by GraphRAG observing only PKG data.
- **The fourth column:** "GRAPH_RAG ANSWER HUMAN EVAL," reporting the human qualitative evaluation and its 0–5 score.
- **The fifth column:** "GRAPH_RAG + VISUAL ANALYSIS ANSWER," reporting the answer produced using both tools combined.
- **The sixth column:** "GRAPH_RAG + VISUAL ANALYSIS EVAL," reporting the human evaluation of the combined output with its 0–5 score.

For each row (user query), the evaluation protocol associated with this dataset unfolds in two consecutive phases:

- **In the first phase**, the query is submitted to the system configured in “GraphRAG only” mode. The model accesses only the PKG and uses the graph-based retrieval and reasoning mechanisms to construct the answer. For each output, a human evaluator compares the response with the desired one and assigns a score on a 0–5 scale, interpreted consistently with the evaluation method adopted in the first set of experiments. A high score indicates that GraphRAG not only retrieved the correct facts but also integrated them into a coherent, complete, and logically grounded answer relative to the context. This first phase aims to establish a “clean” baseline that shows what the system is already capable of doing without any visual support.
- **In the second phase**, the exact same set of questions is submitted to the system in a different configuration: GraphRAG is still invoked first, but the processing pipeline is forced to call the visual analysis module as well. This means that, regardless of whether the graph already contains sufficiently detailed information, the system must still analyze one or more images linked to the query (we have already ensured that at least one image must be analyzed in order to answer that question). This constraint is not neutral: it forces the system to integrate the visual content, at least potentially, into the answer-generation phase. It is precisely this “artificial constraint” that makes the dataset redundant.

For proper evaluation, the protocol adopts a scoring logic in which the final score, in the context of the REDUNDANT VISION DATASET, is assigned based on the added value, or the damage, introduced by the visual module, not solely on the absolute correctness of the answer. This is a crucial point that is easy to misunderstand: if an answer is perfectly correct but that correctness derives exclusively from GraphRAG, while the visual analyzer contributed nothing meaningful, or introduced noise, the score attributed to the visual module must remain relatively low. Conversely, if the multimodal answer becomes slightly more articulated, clearer, or more justified thanks to evidence extracted from the image, the evaluation will tend toward higher scores. Finally, if the visual analyzer introduces errors, contradictions, or noise, the answer may be judged inferior to the one produced only by GraphRAG, with a very low score, even in the presence of a strong baseline.

In this sense, the REDUNDANT VISION DATASET does not simply measure “how good the system is at analyzing images,” but tests its ability to perform multimodal reasoning responsibly. It forces the model to demonstrate that adding an additional information channel is not useless or destabilizing. If integrating the visual component frequently results in neutral or negative contributions, this would indicate that the system does not yet have a good strategy for weighting and critically selecting information coming from images compared to what is already present in the graph. On the contrary, if in many cases the visual analyzer manages to enrich the answer with coherent and pertinent details, one can argue that the vision module is not only technically competent but also cognitively integrated into the behavior of the personal assistant.

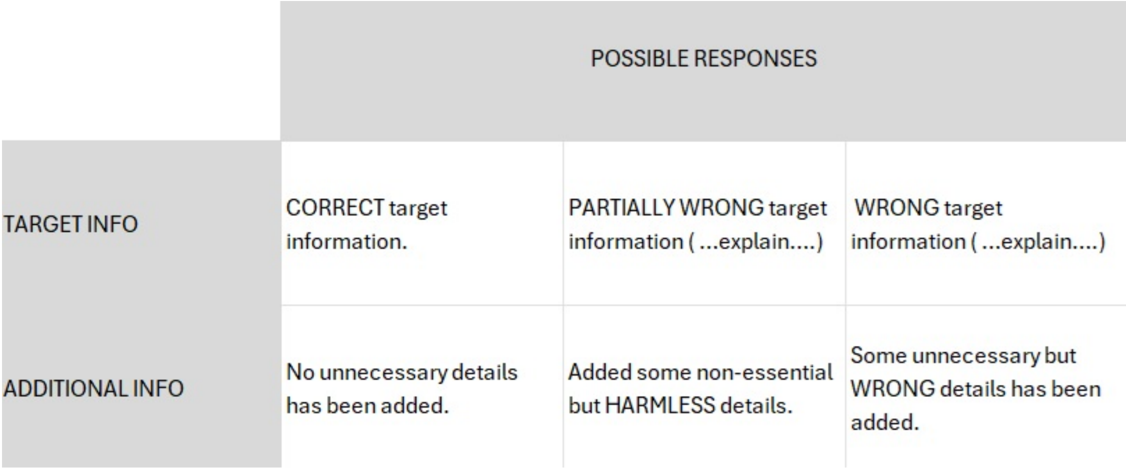
From an experimental documentation perspective, the presence of both answers enables more refined analyses than a simple average score. One can distinguish, for example, cases in which the multimodal answer coincides with that of GraphRAG, cases in which the answer remains conceptually identical but is formulated in a richer way, cases in which new but correct details are introduced, and cases in which the answer is distorted or degraded. This qualitative analysis allows going beyond numerical metrics, highlighting typical behavioral patterns of the system. It is plausible, for example, that the vision module proves particularly useful when the image provides redundant but more "salient" clues compared to what is encoded in the graph, whereas it may prove more fragile when the image is ambiguous or contains potentially misleading elements.

In a sense, this is a qualitative test bench, rather than a quantitative one, designed to understand how the system behaves in specific situations and to identify patterns and failure modes. The goal is to observe the dynamics of multimodal integration and evaluate whether the system's architecture and reasoning heuristics leverage the new information channel sensibly. This protocol forces one to verify whether the visual component is truly useful or whether, under certain conditions, it introduces unnecessary complexity. It is a more demanding approach than a simple capability test because it requires the system not only to show that it can do something extra but also to show that it knows when that "extra" makes sense and when it is better not to alter what the graph already handles well.

Human evaluation method and scoring A key component of the protocol is the human evaluation process, which does not simply judge the correctness of the answers but analyzes the inferential pathways and the coherence of the system in detail. For each question, the evaluator compares the response generated by the system with the desired one and assigns a score from 0 to 5. A response receives the maximum score only if it satisfies three core criteria: factual correctness, justifiability of the inferences, and strict adherence to the set of information available in the PKG and in the images. A response is considered incorrect not only when it provides false content but also when it omits essential information, draws unverifiable conclusions, or fails to recognize epistemic limitations due to missing data. This type of evaluation reflects the broader goal of the project: to build an assistant that is not simply capable of generating plausible responses but one that adopts a form of controlled and verifiable reasoning.

The human evaluation method and scoring process is based on an explicit decomposition of each answer into two conceptually distinct components: on one hand, the target info, meaning the minimal core information that the answer must necessarily contain in order to be considered correct; and on the other, the additional info, meaning all accessory, contextual, or elaborative elements that the system decides to include. The underlying idea is that an answer should not be judged solely on a binary "correct/incorrect" basis, but also in terms of how well it remains "disciplined," avoiding invented details and unnecessary noise.

The reference table used for evaluation formalizes this approach. In the first row appear the categories associated with the target info: the answer may contain fully correct information, partially incorrect information, or completely incorrect information. In the second row, the same structure is applied to the additional info but interpreted in terms of the quantity and quality of unnecessary details: the system may add no unnecessary detail, may add non-essential but harmless details, or may introduce incorrect additional details. In this way, the evaluator does not simply declare whether the answer is "right or wrong," but analyzes how the model handles both the required core information and the optional extension beyond the minimum necessary.



	POSSIBLE RESPONSES		
TARGET INFO	CORRECT target information.	PARTIALLY WRONG target information (...explain....)	WRONG target information (...explain....)
ADDITIONAL INFO	No unnecessary details has been added.	Added some non-essential but HARMLESS details.	Some unnecessary but WRONG details has been added.

Figure 5.1: The reference table used to evaluate the answers.

For the target info, the main criterion is correspondence with the ground truth and the PKG. A response is considered "correct" when it includes all essential facts required by the question, without conceptual errors or omissions of decisive elements. "Partially incorrect" refers to intermediate situations: the system captures part of the correct content but integrates it with an error or omits a crucial aspect whose absence compromises full adherence to the expected answer. Finally, the target info is deemed "incorrect" when the system completely misses the mark: it provides a wrong fact, attributes an incorrect value or date, confuses entities or events, or answers a different question altogether. In these latter cases, the quality of the answer is fundamentally compromised, regardless of how well-formulated secondary details may be.

The additional info is evaluated with a different but complementary logic. A response that adds “no unnecessary detail” is not a minimalistic or impoverished answer but one that remains focused on the point, limiting itself to the information required to satisfy the query. In many cases, this is the ideal behavior: the system demonstrates that it can stop once it has provided what is needed, without filling the space with banalities or generic content. The category “non-essential but harmless details” refers to responses in which the model adds marginal contextual elements that were not required but are not harmful, such as a redundant paraphrase, a brief extra explanation, or a superfluous but coherent clarification. Finally, the most problematic situation occurs when the additional info contains incorrect content: the system, while perhaps hitting the target with the core information, introduces invented details, incorrect references, or interpretations unsupported by data. This is a direct sign of hallucination and is significantly penalized because it reduces the overall reliability of the answer.

The numerical score from 0 to 5 is always derived from a combined reading of these two dimensions. The evaluator does not apply a mechanical formula but uses the table as a conceptual grid to place the judgment. A response that contains correct target info and no unnecessary or incorrect additional details tends toward the highest score: this represents the ideal scenario, in which the system retrieves exactly what is needed, formulates it clearly, and avoids unsupported inferences. If correct target info is accompanied by non-essential but harmless details, the score remains high, but the evaluator may choose whether to award the maximum or a slightly lower value depending on the perceived utility of these details: if they enrich the answer without introducing noise, the effect is neutral or slightly positive; if they weigh down the answer without adding value, the score may be moderately reduced.

When the target info is only partially correct, the score tends to fall into an intermediate range. In these cases, the evaluator analyzes how much the correct portion still allows the user to orient themselves and how much the remaining errors may cause misunderstanding or poor decisions. If the errors are minor and the additional info does not introduce misleading elements, the system can receive a “sufficient” evaluation: the answer is imperfect but maintains some practical usefulness. If, on the other hand, the partiality concerns the core of the required information, or if partially correct target info is combined with incorrect additional details, the score moves toward the lower end, because the user is exposed to confused and unreliable content.

The worst situation arises when the target info is incorrect: in these cases, the evaluator typically assigns the lowest scores, as low as complete failure, because the answer not only fails to address the question correctly but constructs a fictitious narrative that may be harmful. Even if the linguistic form is fluent or convincing, the criterion remains anchored to the truthfulness of the content: in such cases, evaluating additional info is pointless.

This two-dimensional criterion is applied systematically both to the responses generated by GraphRAG alone and to those produced by the multimodal system integrating the visual analyzer. In the case of the REDUNDANT VISION DATASET, as stated, the table is interpreted slightly differently because the goal is to evaluate the specific contribution of the visual module. A response that GraphRAG would have already answered correctly on its own but that is degraded by the inclusion of incorrect visual details is therefore penalized more heavily, precisely because it reveals that multimodal integration introduced noise instead of value.

Conversely, the same framework allows identifying virtuous cases. If the multimodal system provides the same correct target info as GraphRAG but adds accessory details derived from images that are coherent, precise, and useful to the user, the evaluator may justify a very high score, highlighting that the visual module genuinely enriched the answer. In this sense, the distinction between target and additional info is not only a punitive mechanism against errors but also a means to make visible and measurable the added value of well-calibrated multimodal behavior. Overall, the human evaluation and scoring method based on target info and additional info transforms what could be a purely subjective numerical scale into a reasoned, transparent, and reproducible process.

5.2 Test results

The following section presents the test results aimed at evaluating the effectiveness of the multimodal GraphRAG system, with a specific focus on the contribution of the visual analysis module. The analysis is divided into two parts, listing the results for each of the two datasets.

5.2.1 PRIMARY VISION DATASET (PVD) results

The questions and their corresponding answers from the dataset constructed with the explicit goal of testing exclusively the contribution of the visual analysis module—isolating cases where the PKG alone is unable to provide an adequate answer, are illustrated below. The dataset collects 29 questions specifically designed so that their solution necessarily requires the direct inspection of one or more images: the number of people present in a photo, the color of specific objects or clothing items, the direction of an individual's gaze, the presence or absence of elements in the background, and so on. These are, therefore, pieces of information that typically do not emerge from the textual description generated by the img_description module and translated into triples in the PKG but are instead accessible only through a finer understanding of the visual content.

Starting from the same narrative scenario underpinning the textual dataset, as previously seen, the PVD selects situations where the graph provides a gross context and formulates queries that require a higher level of detail, thereby measuring "how deep" the system can go in extracting knowledge from images and filling in the PKG's informational gaps. For each question, the natural text of the query, the ground truth answer, the system's actual output, and a human evaluation of correctness, based on the same criterion adopted for the textual dataset, are recorded.

Below are some of the answers to the 29 questions; the most significant ones have been selected:

Table 5.1: PRIMARY VISION DATASET (PVD) - Visual questions to test Visual analyzer in cases in which GraphRAG fails.

User Query	**Desired Answer**	**Actual Answer**	**Human Evaluation**
What kind of dress was I wearing on June 1st?	A dress with vertical black and white stripes.	Based on the available tools and data, there is no information about the specific dress you wore on June 1st. The photo taken that day ('photo_20250601') lacks visual analysis results or metadata describing clothing. If you have additional context or sources, feel free to share!	WRONG target information (small detail not detected) [0/5]
Was I wearing a striped shirt that time at the pizzeria in June?	Yes, black and white stripes.	Based on the image analysis, the person in the photo is wearing a black-and-white striped shirt while sitting at a table with a pizza. While the image does not explicitly confirm the location as a pizzeria or specify the date in June, the combination of the striped shirt and the pizza context suggests a **likely "Yes"** to your question. For more precise details (e.g., exact location or date), additional data would be needed. **Answer:** Yes, you were likely wearing a striped shirt at a pizzeria in June.	CORRECT target information. No unnecessary details has been added. [5/5]

Table 5.1 – continued from previous page

User Query	**Desired Answer**	**Actual Answer**	**Human Evaluation**
In the photo taken at the pizzeria on Via Re David, was the pizza still whole?	Yes	Based on the analysis of the photo, the pizza at the pizzeria on Via Re David was still whole. The image shows an intact crust with no visible slices or cuts, and the toppings are evenly distributed across the entire pizza. **Answer:** Yes, the pizza was still whole.	CORRECT target information. No unnecessary details has been added. [5/5]
What color is the laundry basket at my house?	Dark green.	The laundry basket at your house is **green**, as noted in your personal information. It's located in the laundry area, placed on the floor next to a white Bosch washing machine and filled with clothes. Would you like to know anything else about it?	CORRECT target information. No unnecessary details has been added. [5/5]
Among the photos of the Sagrada Familia, was the first one I took from the front or the back of the church?	From the front.	The first photo of the Sagrada Familia you took was from the **front** of the church. The image analysis indicates that the photo captures the main facade and intricate spires, which are characteristic of the front view. The back of the church typically has a more subdued design, so the details described in the analysis confirm the front perspective.	CORRECT target information. No unnecessary details has been added. [5/5]

Table 5.1 – continued from previous page

User Query	**Desired Answer**	**Actual Answer**	**Human Evaluation**
Among the photos of the Sagrada Familia, is any of them free from construction cranes?	Yes, one of them	Based on the analysis of the provided images: - **Image 1**: Construction cranes are visible. - **Image 2**: No construction cranes are present. At least one photo of the Sagrada Familia (the second image analyzed) is free from construction cranes. Therefore, the answer is **yes**.	CORRECT target information. No unnecessary details has been added. [5/5]
When we visited the Barcelona cathedral, was there a woman with white hair in our group?	Yes	Based on the image analysis, **yes**, there was a woman with white/gray hair in your group during the Barcelona cathedral visit. She was positioned **fourth from the left** in the photo.	CORRECT target information. Some unnecessary but PARTIALLY WRONG details has been added (the woman is the fifth from the left). [4/5]
When we visited the Barcelona cathedral, was there anyone in shorts in our group?	Yes, a man with sunglasses (the second from the right).	Based on the image analysis, **yes**, there was someone in shorts in your group during the visit to the Barcelona cathedral. The analysis identified a man (third from the right in the photo) wearing shorts. While the initial records only noted the group was "dressed casually," the visual confirmation from the image provides this specific detail.	CORRECT target information. Some unnecessary but PARTIALLY WRONG details has been added (the man is the second from the right). [4/5]

Table 5.1 – continued from previous page

User Query	**Desired Answer**	**Actual Answer**	**Human Evaluation**
In our trip to Barcelona, were there more males or females in our group?	There were more females in your group: 5 females and 4 males.	Based on the analysis of the group photo taken during your Barcelona trip: **Final Answer**: There were **5 females** and **4 males** in the group, so **more females**. This conclusion comes from counting the 9 individuals visible in the image, with gender distribution as follows: - Females: 5 - Males: 4	CORRECT target information. No unnecessary details has been added. [5/5]

Table 5.1 – continued from previous page

User Query	**Desired Answer**	**Actual Answer**	**Human Evaluation**
During the trip to Barcelona, did we visit Barceloneta at sunset?	Yes	let's analyze the given image. 1. **Identify the Location**: The image shows a beach scene with a distinctive building on the right side. The building is likely the Hotel Arts or a similar structure, which is known to be near Barceloneta beach in Barcelona. 2. **Analyze the Scene**: The image depicts a beach with people on it and a body of water, which matches the description of Barceloneta, a popular urban beach in Barcelona. 3. **Sunset Indication**: The sky in the image is painted with hues of yellow and orange, typical of a sunset. The lighting on the water and the beach also suggests that the sun is setting. 4. **Conclusion**: Given that the image appears to be of Barceloneta and it shows characteristics of a sunset (the color of the sky, the lighting), it can be inferred that the image was taken at sunset in Barceloneta. Therefore, based on the information provided by the image, the answer to the question is: **Yes**. Answer: Yes, you did visit Barceloneta around sunset.	CORRECT target information. No unnecessary details has been added. [4/5]

Table 5.1 – continued from previous page

User Query	**Desired Answer**	**Actual Answer**	**Human Evaluation**
When we visited Camp Nou, was anyone of us wearing a jacket?	No	Based on the analysis of the photo description and available details, **no one in the group was wearing a jacket** during the visit to Camp Nou. Here's why: 1. **Casual Attire**: The group was described as "casually dressed," with one person explicitly wearing a "white sleeveless top." This suggests a warm, relaxed outfit rather than layered clothing. 2. **Weather Context**: The visit occurred on **June 17th**, a time of year when Barcelona typically experiences warm weather, making jackets unnecessary. 3. **No Mention of Outerwear**: The detailed description of the photo does not reference any jackets or outerwear for the group.	CORRECT target information. No unnecessary details has been added. [5/5]

Table 5.1 – continued from previous page

User Query	Desired Answer	Actual Answer	Human Evaluation
In the photo at the stadium a few months ago, what color were the stands?	They were blue and red, the official colors of Barcelona.	The analysis of the image suggests that the stands in the photo are **red, blue, and maroon**. However, it's worth noting that Camp Nou, FC Barcelona's stadium, is traditionally associated with **red and blue** (the team's colors). The mention of "maroon" could be due to lighting, shadows, or image quality affecting color perception. For precise confirmation, reviewing the original photo or cross-referencing with additional sources might help clarify.	CORRECT target information. No unnecessary details has been added. [5/5]

5.2.2 REDUNDANT VISION DATASET (RVD) results

The questions and their corresponding answers from the dataset are illustrated below, with the goal of understanding what happens when the visual component intervenes in situations where GraphRAG is already perfectly capable of producing an adequate answer. In the REDUNDANT VISION DATASET, every question was selected, as previously mentioned, based on whether the PKG already contains all the essential informational elements to answer it; consequently, the behavior of the multimodal system allows us to observe how the vision module integrates into a complete reasoning process.

In this sense, the dataset becomes a tool for evaluating the maturity of multimodal integration. Through a direct comparison between the output of GraphRAG alone and the output of the system combining RAG and the visual analyzer, it is possible to identify when the visual module positively supports reasoning, when it merely generates noise, and when it is ignored by the model. For each question, as anticipated, the natural text of the query, the ground truth answer, the actual output of the GraphRAG tool (with human evaluation of correctness), and the actual output of the GraphRAG tool supported by the visual analyzer (with human evaluation of correctness) are recorded.

Below are some of the answers to the 14 questions; the most significant ones have been selected:

Table 5.2: REDUNDANT VISION DATASET (PVD) - Visual questions to test Visual analyzer in cases in which GraphRAG has enough data.

User Query	**Desired Answer**	**GraphRAG Answer**	**GraphRAG Answer Human Eval**	**GraphRAG + Visual Analysis Answer**	**GraphRAG + Visual Analysis Human Eval**
What kind of pizza I ate in June?	It was a margherita pizza, with tomato sauce, mozzarella, and basil.	The system references a **Pizza Margherita** (mozzarella, tomato sauce, basil) as part of its knowledge. This classic Italian pizza is tied to a culinary community in the data, but there's no confirmation it was eaten by you. Would you like to: 1. Explore more about the Pizza Margherita? 2. Check if you have any unrelated records about June meals? 3. Discuss other pizza types or recommendations? Let me know how I can help clarify!	CORRECT target information. No unnecessary details has been added. [5/5]	Given the information in the image, the kind of pizza is likely a Margherita pizza, which is a classic Neapolitan-style pizza topped with tomato sauce, mozzarella cheese, and fresh basil leaves. Therefore, the answer is: Margherita pizza.	CORRECT target information. No unnecessary details has been added. [5/5]

Table 5.2 – continued from previous page

User Query	**Desired Answer**	**GraphRAG Answer**	**GraphRAG Answer Human Eval**	**GraphRAG + Visual Analysis Answer**	**GraphRAG + Visual Analysis Human Eval**
What brand is my washing machine?	BOSCH	Your washing machine is a **Bosch**. It's described as a front-loading washing machine placed in a laundry/storage room setting, next to a laundry basket and with various cleaning products nearby. This suggests it's part of a typical home household appliance setup. If you need, I can also help you look up common features or manuals for Bosch washing machines!	CORRECT target information. No unnecessary details has been added. [5/5]	The brand of your washing machine is **Bosch**. The machine appears as a front-loading washing machine with a circular glass door, typical of Bosch models. It is located in a utility or laundry room with cleaning products and other household items nearby, and there is a laundry basket positioned close to the machine. The visual details, such as the layout and styling of the control panel, are consistent with Bosch's design language.	WRONG target information from the analyzer tool (small detail not detected, it retrieves right info only with GraphRag: the analyzer adds noise). [0/5]

Table 5.2 – continued from previous page

User Query	**Desired Answer**	**GraphRAG Answer**	**GraphRAG Answer Human Eval**	**GraphRAG + Visual Analysis Answer**	**GraphRAG + Visual Analysis Human Eval**
What was the weather like when we visited the Sagrada Familia in Barcelona in June?	It was partly clouded.	The weather during your visit to the Sagrada Familia in Barcelona on June 12, 2025, was **pleasant and sunny**, with clear blue skies. The photos from that day capture the cathedral under ideal sightseeing conditions, highlighting its iconic architecture against a bright backdrop. Additional context from the response notes the ongoing construction (with cranes visible) and the site's status as a UNESCO World Heritage Site, but the primary focus on weather is confirmed. Let me know if you'd like further details!	PARTIALLY WRONG target information (it can't see clouds). [4/5]	The weather during your visit to the Sagrada Familia in Barcelona appears to have been **pleasant overall**, with **varying cloud cover**. - Some photos suggest **clear, sunny conditions** with a bright blue sky. - Other images indicate **partly cloudy skies** with visible white clouds. This suggests a day of generally good weather, ideal for sightseeing, though with occasional cloud cover. No signs of rain or extreme conditions were noted.	CORRECT target information. No unnecessary details has been added. [5/5]

Table 5.2 – continued from previous page

User Query	**Desired Answer**	**GraphRAG Answer**	**GraphRAG Answer Human Eval**	**GraphRAG + Visual Analysis Answer**	**GraphRAG + Visual Analysis Human Eval**
What were we wearing during our visit to Camp Nou?	You were casually dressed; the boy was wearing a T-shirt and the girls sleeveless dresses.	During our visit to Camp Nou on **June 17, 2025**, the group was photographed wearing **casual attire** suitable for attending an event at the stadium. Specific details from the photo include: - **One individual** was wearing a **white sleeveless top**. - The **rest of the group** was dressed in **casual clothing**, though specific items (e.g., jeans, t-shirts) aren't described. Let me know if you'd like further details!	CORRECT target information. Added some non-essential but HARMLESS details. [5/5]	During our visit to Camp Nou on June 17, 2025, the group's attire, as captured in the photograph, was as follows: 1. **Man on the left**: Wore a **white t-shirt**. 2. **Woman next to him**: Dressed in a **white sleeveless top**. 3. **Woman in the middle**: Wore a **brown sleeveless top paired with a black skirt**. 4. **Woman on the right**: Dressed in a **blue patterned dress**.	CORRECT target information. No unnecessary details has been added. [5/5]

Table 5.2 – continued from previous page

User Query	**Desired Answer**	**GraphRAG Answer**	**GraphRAG Answer Human Eval**	**GraphRAG + Visual Analysis Answer**	**GraphRAG + Visual Analysis Human Eval**
In which section of the supermarket was the photo taken a few days ago?	In the pasta section.	The photo was taken in the **snacks and cheeses section** of a Eurospin supermarket in Bari, Italy. The specific aisle highlighted tortilla chips and other snack items arranged on shelves. However, there is an inconsistency in the timeline: the photo is dated August 30, 2025 (a future date relative to the current year, 2023). If this is a recent photo, the date might be an error. Would you like to clarify or verify this further?	PARTIALLY WRONG target information (it confuses pasta with chips). No unnecessary details has been added. [3/5]	The photo was taken in the **snack aisle** of the supermarket, specifically in the section dedicated to **tortilla chips and snacks**. This is confirmed by the organized display of chip varieties and the context provided in the analysis.	PARTIALLY WRONG target information (it confuses pasta with chips). No unnecessary details has been added. [3/5]

Table 5.2 – continued from previous page

User Query	**Desired Answer**	**GraphRAG Answer**	**GraphRAG Answer Human Eval**	**GraphRAG + Visual Analysis Answer**	**GraphRAG + Visual Analysis Human Eval**
In the photo taken at Barceloneta, were there any boats?	Yes.	Based on the available information, it is likely that there were boats in the photo taken at Barceloneta. While the photo's metadata does not explicitly describe its contents, contextual clues and a related source describing a sunset scene at the location mention "several small boats floating gently" in the beach setting. The photo was taken at 20:00 on June 15, 2025, a time consistent with the described serene beach atmosphere.	CORRECT target information. Added some non-essential but HARMLESS details. [5/5]	Yes, there were boats in the photo taken at Barceloneta. The image analysis revealed ships visible on the horizon as part of the beach scene.	CORRECT target information. No unnecessary details has been added. [5/5]

Table 5.2 – continued from previous page

User Query	**Desired Answer**	**GraphRAG Answer**	**GraphRAG Answer Human Eval**	**GraphRAG + Visual Analysis Answer**	**GraphRAG + Visual Analysis Human Eval**
What color was the cat in yesterday's photo?	striped brown and black	The images describe a **Brown and black striped cat** on a white surface. Let me know if you'd like further assistance!	CORRECT target information. Added some non-essential but HARMLESS details. [5/5]	The cat in the analyzed image has a coat with brown and black stripes. While the image analysis confirms the coloration, the reference to "yesterday's photo" cannot be verified due to the lack of temporal context in the provided data. The described cat (brown and black) is from an image in the dataset, but its specific date is unspecified.	CORRECT target information. Some unnecessary but WRONG details has been added (caused by Visual Analyzer's lack of temporal info). [4/5]

5.3 Results Discussion

This chapter presents the in-depth analysis of the results obtained from the human evaluation of the multimodal Question Answering system, focusing in particular on the role and effectiveness of the Visual Analysis module; the results refer to the two different datasets.

5.3.1 PRIMARY VISION DATASET (PVD) results discussion

Each question in this dataset requires identifying a concrete visual detail: the colors of objects, the presence or absence of elements in a photo, the state of an object, the number or position of people, signs of construction work, and so on. In this context, the visual analyzer is not optional but the only path to the correct answer: if the visual component fails, the entire system fails.

A first group of questions concerns domestic scenes (clothing, washing machine, laundry basket, ironing board, cat), which test the system's ability to detect small color details or patterns (stripes, specific colors). A second group is linked to the trip to Barcelona and includes different subgroups: photos of the Sagrada Família (camera angle, presence or absence of cranes), those of the Gothic cathedral and the group of people (bags, shorts, white hair, gender distribution), scenes from tourist locations (Casa Batlló, La Rambla, Barceloneta), and finally photos taken at the Camp Nou stadium. There are also some isolated questions about other contexts (the illuminated Eiffel Tower, the supermarket, another stadium). This design is intentional: it allows measuring system behavior both on minute details and on more global scene properties (crowd size, relative majority, presence of specific object categories).

Below is an explanation of the evaluations assigned to the questions:

Question 1 – dress from June 1st (vertical stripes) The question requires recognizing a precise pattern on the dress (black and white vertical stripes). However, the system does not provide this information and merely states that it does not have access to the necessary data. From the standpoint of target info, this is a complete failure: the central detail is neither retrieved nor reconstructed, and the answer does not help the user. There is no hallucinated additional info, but the total absence of useful content justifies the score of 0.

Question 2 – striped shirt in the pizzeria Here the question is whether the person in the pizzeria photo was wearing a striped shirt. The system correctly confirms the presence of stripes and aligns with the ground truth without adding unnecessary details. The target info is entirely correct and complete, and additional info is essentially absent: this is exactly the ideal scenario expected by the protocol, motivating the score of 5. This means that the model was able to detect the clothing pattern only when prompted.

Question 3 – pizza: whole or already sliced The question verifies the state of an object: was the pizza still whole? The system correctly identifies the situation in the image and answers concisely and appropriately. The target info is accurate, there are no unnecessary expansions, and the answer is easily verifiable from the image. Again, the behavior is "disciplined," and the score of 5 is coherent.

Question 4 – cat's eye color This is a subtle visual detail (eye color "tending toward yellow"). The system manages to pick up this chromatic nuance and return it without superfluous commentary, demonstrating good sensitivity to color tones.

The target info is correct, additional info is minimal or absent, and no confusion is introduced: the score of 5 indicates that the visual module handles this type of description well.

Question 5 – color of the laundry basket The question concerns a static household object with a specific color. The system correctly identifies the color and does not elaborate beyond what is necessary. The answer aligns directly with the ground truth, score 5.

Question 6 – color of the clothes inside the washing machine Again this is a color-based detail, but in a more chaotic context (clothes inside the washing machine). Nonetheless, the system provides a description consistent with the ground truth and avoids superfluous interpretation. The evaluator recognizes the full correctness of the core information and the absence of problematic additional info, score 5.

Question 7 – color of the ironing board This question is conceptually similar to the previous ones (color of a domestic object), but here the system fails. Instead of reading the color of the ironing board, the answer remains vague and does not provide the required detail. From the protocol's perspective, this is a case of missing target info: the visually small detail is not detected, and the user does not receive the key information. The score is 0.

Question 8 – Sagrada Família: first photo taken from the front or the back The question asks the system to distinguish the viewpoint (front/back) in the first picture of the Sagrada Família. The system correctly recognizes the camera angle and provides a direct answer. This is a global property of the scene, which the visual module seems able to handle well. Correct target info and no misleading details lead to a score of 5.

Question 9 – Sagrada Família: presence of cranes Here the task is to check whether at least one photo contains construction cranes. The system correctly identifies their presence and provides an answer consistent with the ground truth, score 5.

Question 10 – Sagrada Família: photo without cranes The "specular" question asks whether there is at least one photo completely free of cranes. The system manages to distinguish the different shots and detect the absence of cranes in at least one image, answering correctly. Again, target info is accurate, and no additional invented details appear, so the evaluator assigns a score of 5.

Question 11 – boy with sling bag at the cathedral This question concerns the presence of a boy with a sling bag in the group. The system identifies him correctly in the image and provides a clear answer without adding unnecessary descriptions. The target info is fully aligned with the ground truth, and the additional info does not introduce noise: score 5.

Question 12 – color of the shirt of the boy with the sling bag Here the difficulty increases: it is not enough to identify the boy; the system must also detect the color of his shirt. The system does so correctly. Once again, correct target info and no additional info: score 5.

Question 13 – presence of a woman with white hair in the group The question asks only whether there was a woman with white hair. The system correctly answers "yes" and adds a positional detail ("fifth from the left"); the additional detail is superfluous and incorrect: she is the fourth from the left. The score is slightly reduced to 4.

Question 14 – someone wearing shorts in the group Again, the system correctly answers the question ("yes, someone is wearing shorts") and adds a positional indication. The additional info is not essential and slightly harmful since the subject is not the third from the right but the second. The core information is fully correct, so the score is high but fixed at 4.

Question 15 – construction work at the Gothic cathedral (visit) Here the question asks whether, during the visit to the Gothic cathedral, there were scaffolding or construction work. Instead of relying on the specific image, the system seems to slip into an "encyclopedic" type of answer, speaking in general terms about the Sagrada Família and its well-known ongoing construction features. This response does not correspond either to the exact location or to the situation shown in the photo, and it is explicitly annotated as a case in which the visual analysis tool was not actually invoked. The result is a complete error, score 0.

Question 16 – construction work in the photo taken below the Gothic cathedral The question is similar to the previous one but refers to a specific photo taken "below" the cathedral. Although the system mentions the Gothic cathedral, it fails to correctly recognize the presence of scaffolding in the specific image. Again, the visual detail is missed, and the target info is wrong. The evaluation classifies this as a "small detail not detected": the score is 0.

Question 17 – more males or more females in the group (cathedral) This question involves a global property of the photo: the gender distribution within the group. The system correctly identifies it and responds in line with the ground truth, demonstrating good ability to synthesize approximate numerical information (relative majority). There is no misleading additional info, so the score of 5 is fully justified.

Question 18 – more males or more females in the group (entire trip) The same question is extended to the entire trip: considering all photos, were there more men or more women? The system remains consistent with the available data and provides the same correct answer, showing robustness with respect to a change in formulation. Once again, correct target info and no added noise led to a score of 5.

Question 19 – how crowded the Casa Batlló photo is The question evaluates crowd perception: is the photo very populated? The system correctly interprets the level of people density in the image and provides a coherent assessment. Score 5.

Question 20 – Rambla crowded or not The situation is similar: the question asks whether La Rambla was very crowded in the photo. The system recognizes a high influx of people and describes the scene accurately. The answer is useful to the user and does not add extra details: target info fully correct, score 5.

Question 21 – Barceloneta at sunset Here the property to be recognized is atmospheric: the system exploits light features and sky color to correctly confirm that the photo was taken at sunset. This is a more subtle form of visual reasoning and is handled well: score 5 because the target info is precise and no additional details are introduced.

Question 22 – someone swimming at Barceloneta The question explicitly asks whether someone is "swimming" in the water. The system observes people in the sea and concludes that yes, someone is swimming, but the ground truth distinguishes between "being in the water" and "actually swimming." The target info is therefore only partially correct: the interpretation of the action is too coarse. Score 4.

Question 23 – someone with a mask at Camp Nou The question asks whether anyone in the group was wearing a mask. In the photo, no one is wearing it properly, but one person has it around their neck. The system notes that no one has their face covered and concludes that no one is wearing a mask, ignoring the detail of the mask worn at the neck. The target info is therefore partially incorrect: the error is neither total nor marginal, justifying the intermediate score of 3.

Question 24 – masks worn or at the neck The question is reformulated in a more explicit way, including masks worn around the neck. In this case the system manages to correctly interpret the scene and recognize the presence of the non-worn mask. The fact that the reformulation helps the model shows how linguistic precision influences visual reasoning. With correct target info and a “clean” answer, the score is 5.

Question 25 – what we were pointing at in Camp Nou Here the question asks what the group was pointing at with their hands in the stadium photo. The system correctly identifies the element (the “Més que un club” slogan). This is a good example of integration between text recognition in the image and contextual understanding. Perfect target info, no superfluous additional info: score 5.

Question 26 – someone wearing a jacket at Camp Nou The question checks for the presence of a jacket among group members. The system correctly observes the clothing in the image and responds consistently with the ground truth. No hallucinations or unnecessary expansions appear, so the score is 5.

Question 27 – color of the Eiffel Tower lighting This involves a chromatic detail in a nighttime context: the color of the Eiffel Tower’s lights. The system recognizes the yellow/orange tone of the illumination and reports it faithfully. The evaluation is therefore fully positive, score 5.

Question 28 – number of shelves in the supermarket Here the system must count the number of shelves in a supermarket aisle. The count is off by one: the model “sees” five shelves where there are six. The error is small but affects the target info directly, which is an exact quantity, classified as “partially wrong.” Score 4.

Question 29 – color of the stadium stands The final question concerns the color of the stands in a stadium photo. The system correctly identifies the dominant colors and describes them coherently with the ground truth. Once again, no superfluous expansion: correct target info, neutral additional info, and a score of 5.

The results on the PRIMARY VISION DATASET indicate that the visual analysis module is overall effective but still shows room for improvement in specific scenarios. The analysis of the 29 test cases shows generally solid performance: 82.8% of the answers received a “Excellent” or “Good” rating according to human evaluation, while a non-negligible portion resulted in complete failure, indicating situations in which the system could not recover the required target info at all. The score distribution is as follows:

- **Excellent (5/5):** 69.0%
- **Good (4/5):** 13.8%
- **Sufficient (3/5):** 3.4%
- **Poor (2/5 or 1/5):** 0.0%
- **Failure (0/5):** 13.8%

Looking more closely, the first notable point is the high concentration in the upper range: 20 out of 29 questions scored 5, and 4 out of 29 scored 4. Summing these categories yields 24 very good answers out of 29, approximately 82.76%. Including the single 3/5 response, the number of at least sufficient answers rises to 25 out of 29, or about 86.21%. Numerically, this means that in the vast majority of cases the module succeeds in identifying the essential target info and keeping additional info under control, avoiding hallucinations.

On the other hand, the 4 failures (0/5) are not negligible: they represent almost 14% of the dataset. This does not merely indicate "wrong answers" but actual performance gaps, in which the target info is completely missed, or the answer is misleading relative to the question. In some cases, the system explicitly declares that it cannot answer; in others it relies on generic background knowledge instead of reading the image. From the project's point of view, these zeros should not be treated as simple outliers but as clear indicators of conditions under which the visual analyzer is either not properly engaged or not yet capable of handling particularly subtle visual details.

Another interesting aspect is the shape of the distribution: the total absence of (1/5) and (2/5) scores suggest a kind of "two-state" behavior. When the system manages to extract the target info, it generally does so clearly, resulting in high scores (4 or 5), and only marginally drops to 3 when imprecisions are more noticeable. When it fails, however, the evaluation tends to collapse directly to 0, because the answer provides no usable information to the user. This pattern is coherent with the nature of PVD questions: since each query requires a specific visual detail, there is little room for intermediate compromises; either the detail is detected, or it is not.

5.3.2 REDUNDANT VISION DATASET (RVD) results discussion

This dataset makes it possible, as already shown, to observe what happens when the visual analyzer is invoked even in situations where GraphRAG already has, at least in theory, all the information needed to answer the question. In this scenario the vision module is not necessary as in the PVD, but redundant: its task is to add visual evidence, confirm or contradict what is retrieved from the PKG, and do so without degrading the overall quality of the answer. For this reason, human evaluation is centered on the marginal contribution of the visual module: it is not enough for the final answer to be correct; one must understand whether it is correct thanks to (or despite) the visual analyzer.

The questions to consider during evaluation, therefore, are: In which cases does the visual analyzer add useful elements? When does it introduce superfluous or inaccurate details, degrading the answer? And when, instead, does it not substantially modify the output, resulting in a neutral contribution?

Below are the explanations of the evaluations assigned to each question:

Question 1 – Type of pizza eaten in June In the first case both GraphRAG and the multimodal system correctly identify the type of pizza (margherita), both receiving the maximum score. The vision module recognizes the main ingredients in the image and aligns perfectly with the knowledge already present in the PKG, without introducing superfluous or contradictory details. Here the visual analyzer adds no new information, but robustly confirms what the graph already knew: the redundancy is entirely "healthy," because it does not change the score but makes the answer more reliable and coherent.

Question 2 – Location of the cat photo from a few months ago For this question GraphRAG makes a conceptual mistake: it tends to interpret "where" in terms of geographical location rather than as a concrete physical environment (the room or office where the photo was taken). Human evaluation assigns an intermediate score, precisely because the system confuses geographical level and environmental context. The multimodal system, thanks to the visual analyzer, manages instead to describe the environment in the photo (office, desk, printer, work-related objects) in a way that aligns with the desired answer, raising the score. Some additional details remain, inherited from the textual component, which are judged unnecessary or imprecise. The result is a clear improvement on the main target, but with some traces of noise due to the integration of the two channels.

Question 3 – Brand of the washing machine This is one of the most critical cases. GraphRAG correctly retrieves the washing machine brand (BOSCH) from the PKG and receives the maximum score; the answer is precise and essential. When the visual analyzer enters the process, however, performance collapses: the visual component leads the model to associate the washing machine with an incorrect or non-coherent brand, effectively overwriting the correct information provided by the graph. The final result is a wrong answer, with the minimum score for the multimodal system. Here the visual analyzer not only fails to help, but actively damages an answer that was already correct, showing how multimodal redundancy can be risky when the visual signal is ambiguous or interpreted too dominantly compared to symbolic knowledge.

Question 4 – Weather during the visit to the Sagrada Família In this scenario GraphRAG manages to reconstruct only partially the weather conditions of the visit: the context of the trip is known, but perceptual details are missing (such as the presence of clouds), resulting in an intermediate score. With the visual analyzer, however, the system reads the scene directly, describing the weather precisely and producing a fully aligned answer without adding irrelevant elements. The multimodal system therefore reaches the maximum score. This is a clear example of vision completing incomplete narrative information, transforming a "partial" response into a fully satisfying one.

Question 5 – Weather during the visit to the Barcelona Cathedral The pattern mirrors the previous case. GraphRAG has correct but incomplete knowledge of the weather conditions, reflected in a medium score. The multimodal system, using the image associated with the visit, accurately captures the conditions of the sky and the environment, offering a coherent and complete description of the weather. Again, the score rises to the maximum thanks to the visual analyzer, which proves particularly useful for all questions where the graph provides only narrative context and not detailed perceptual properties.

Question 6 – Number of people present at Camp Nou In this case both GraphRAG and the multimodal system achieve the maximum score. The number of people involved in the visit is correctly encoded in the PKG, and the textual system retrieves it without difficulty. The visual component, analyzing the photo, confirms the count without introducing errors or complications. The contribution of the visual analyzer is therefore neutral regarding the score but positive in terms of robustness: it shows that the model can recognize and count people consistently with what is already known in the graph, and that redundancy does not generate conflicts.

Question 7 – What we were wearing at Camp Nou This question combines narrative information (the visit) and perceptual details (clothing). GraphRAG already provides a correct and fairly rich answer, based on descriptions encoded in the PKG, including some non-essential but harmless details. The multimodal system fully confirms these contents, describing clothing and colors accurately, and reduces the tendency to insert marginal details. The score remains the maximum for both systems, but the multimodal answer is slightly more focused on visually relevant elements, with less verbal redundancy. This is therefore a case in which vision does not change the evaluation.

Question 8 – Weather during the visit to Paris Here the PKG appears to contain a sufficiently precise representation of the weather conditions for the visit: GraphRAG answers correctly and receives the maximum score. The multimodal system confirms this information through the image, verifying that the sky and environment correspond to what the graph describes. No contradictions or off-topic details emerge. The visual analyzer therefore does not provide a score advantage but reinforces consistency between symbolic and perceptual channels.

Question 9 – Crowding level at Bari airport when returning from Paris This question highlights one of the most interesting cases in favor of the visual component. GraphRAG, based solely on the PKG, strongly misinterprets the level of crowding, effectively describing a scenario that does not match the photo (e.g., an almost empty airport). As a result, the textual system receives the lowest score. The visual analyzer, however, recognizes the presence of many people in the image and provides a response aligned with the desired answer, namely that the airport was indeed crowded. The multimodal score thus jumps from minimum to maximum. This is an emblematic case where the RAG alone fails, while the visual analyzer "rescues" the answer.

Question 10 – Aisle in the supermarket photo from a few days ago In this case both GraphRAG and the multimodal system receive an intermediate score. The textual system correctly identifies the general setting (supermarket) but misclassifies the aisle type, confusing two product categories (e.g., pasta and snacks). The visual analyzer, despite having access to the image, cannot correct this ambiguity: the multimodal answer essentially reproduces the same incorrect interpretation. No improvement or degradation is observed; both tools make the same classification error, showing that redundancy does not automatically correct semantic mistakes if the model fails to exploit perceptual evidence effectively.

Question 11 – Location of yesterday's cat photo GraphRAG completely misinterprets the type of information required, insisting on the geographical dimension instead of describing the concrete physical environment in which the cat appears; consequently, it receives the minimum score. The multimodal system, in contrast, thanks to the visual analyzer, provides a correct description of the environment (room, furniture, domestic context) in line with the desired answer, and receives a high score. Here again the outcome is clearly positive: the visual module corrects a conceptual error in the graph and shifts the answer from completely wrong to substantially correct.

Question 12 – Presence of boats in the Barceloneta photo For this question both tools converge. GraphRAG already knows that boats are present in the scene and expresses this correctly, with perhaps some additional but harmless details. The multimodal system, directly inspecting the image, confirms the presence of the boats and produces an essential, focused answer. Both receive the maximum score.

Question 13 – Cat color in yesterday's photo This is one of the few cases in which the visual analyzer slightly lowers the score. GraphRAG correctly describes the cat's coat (e.g., brown and black striped) and receives the maximum score. The multimodal system recognizes colors and fur patterns correctly, but the answer shows slight uncertainty in linking with absolute certainty that specific image to the temporal reference "yesterday," since such information is not visually explicit. From a perceptual standpoint the content is correct, but this minor contextual ambiguity yields a slightly lower score.

Question 14 – Description of the building photographed on June 14 in Barcelona The last question concerns a richer and more open description: the building is Casa Batlló. GraphRAG correctly identifies the name and style of the building, providing a description consistent with its known characteristics (articulated façade, vibrant colors, elements typical of Catalan modernism), and receives the maximum score. The multimodal system confirms this interpretation and enriches the answer with visual details present in the photo (organic shapes, mosaics, specific decorations), without exaggerating or introducing unsupported elements. Here again both tools receive the maximum score.

The system is overall stable, capable of integrating the visual module in a mostly safe way even when GraphRAG already possesses the necessary information. Overall performance is positive: 85.7% of multimodal answers were rated "Excellent" or "Good," confirming that visual redundancy does not compromise response quality and, in a significant number of cases, even improves it. The distribution of final scores for the multimodal system is as follows:

- **Excellent (5/5):** 64.3%
- **Good (4/5):** 14.3%
- **Sufficient (3/5):** 14.3%
- **Poor (2/5 or 1/5):** 0.0%
- **Failure (0/5):** 7.1%

Beyond the absolute response quality, a central aspect concerns the direct comparison between GraphRAG and the multimodal system. The analysis reveals three distinct situations:

- **Improvement thanks to the visual analyzer:** 5 cases (35.7%)
- **Neutrality (unchanged result):** 7 cases (50%)
- **Worsening caused by the visual analyzer:** 2 cases (14.3%)

The overall picture emerging from the 14 cases is that of a multimodal integration that tends to be more helpful than problematic. The percentage of responses rated “Excellent” or “Good” indicates solid performance even in a context where the visual module is not strictly necessary. This is an important result, because it demonstrates that adding the visual analyzer does not introduce a systematic degradation risk: in most cases the system not only avoids new errors but mainerrors butor consistent with what GraphRAG already handles well.

The 64.3% of “Excellent” responses represents the most populated category. This suggests that in most scenarios the visual analyzer provides confirmations consistent with knowledge already encoded in the PKG, without introducing noise or harmful redundancy. The 14.3% of “Good” answers is also noteworthy: in many of these cases the visual analyzer contributed positively but not in a way entirely free from marginal details or minor uncertainties. The absence of any “Poor” (1–2/5) answers signals strong robustness against severe errors.

Shifting from absolute scores to the direct comparison between GraphRAG and the multimodal system, interesting dynamics emerge. Cases in which the multimodal system improves over GraphRAG number 5 out of 14 (35.7%). This means that in more than a third of situations vision fills gaps in the graph and corrects conceptual errors , often related to environmental descriptions, weather conditions, or contextual interpretations that the PKG cannot represent with enough granularity. These cases concretely demonstrate the value of perceptual information, which can provide essential elements impossible to reconstruct through purely symbolic reasoning.

Neutral cases, in which multimodal performance remains identical to GraphRAG’s, make up half the dataset (50%). This high percentage indicates that the system can integrate vision without negatively altering an already correct reasoning process. In other words, redundancy does not introduce instability.

Finally, the worsening cases are only 2 out of 14 (14.3%). Although a small number, they are important to interpret: in both instances the image led the system off track, introducing incorrect visual interpretations or elements not coherent with what the graph already knew correctly. This depends in part on the intrinsic “strength” of the visual analyzer’s model: the lower ratings are due to the model failing to detect relatively small details.

Conclusion

This work arises from an observation that is both simple and profound: the personal assistants available today, despite their widespread use and technological sophistication, still operate within limited paradigms. Systems such as Siri, Google Assistant, or Alexa excel at executing specific commands and retrieving information from vast global knowledge bases, but they lack an essential component needed to become true digital collaborators: an authentic and persistent understanding of the user's individual world. Their memory is ephemeral, their knowledge impersonal, and their reasoning ability is decoupled from the unique context that shapes our daily lives. They cannot answer questions such as "What color was the jacket I wore at Marco's birthday last year?" or "What were the main points of the document I was reading yesterday afternoon?", not because their underlying technology is insufficient, but because their architecture is not designed to build, manage, and query a repository of personal knowledge. The central ambition of this project was therefore to design and implement a prototype that goes beyond this limitation, developing an intelligent personal assistant equipped with structured memory and visual reasoning capabilities—one able to draw quickly and naturally from a person's universe of personal data to provide answers that are not only correct but deeply contextualized and meaningful.

To realize this vision, an architecture was designed around three synergistic technological pillars, each addressing a specific challenge in building a system that leverages artificial intelligence in a genuinely personalized way. The first and most fundamental of these pillars is the Personal Knowledge Graph, which acts as the cognitive backbone of the entire system. Unlike a simple database or a disorganized collection of files, the PKG models the user's knowledge as an interconnected network of entities and relationships, transforming the chaotic stream of personal data—documents, calendar events, photographs, contacts—into a coherent, navigable, and semantically searchable structure. This graph is not just a static memory, but a dynamic model of the user's world, a true "digital twin" of their knowledge that allows the system to understand the implicit connections between an event, a person, a place, and a document.

The second pillar is represented by the RAG methodology and its specialized evolution, GraphRAG. This is the reasoning mechanism that enables intelligent access to memory. Instead of relying on similarity-based search over isolated text fragments, this approach exploits the topology of the graph itself, retrieving not only individual nodes but contextual subgraphs, multi-hop reasoning paths, and thematic communities. By grounding answer generation in structured and verifiable evidence, GraphRAG drastically mitigates the risk of hallucinations—a typical issue in large language models—and ensures that every statement produced by the assistant is anchored in the user's real data.

Finally, the third essential pillar is the integration of a large vision-language model (VLM) within an agentic architecture. This component gives the assistant "eyes," allowing it to go beyond textual content and interpret the rich informational depth present in images. Thanks to this module, the assistant can answer questions whose solution is not encoded in any text but lies exclusively in the visual details of a photograph, filling a crucial gap in a system's ability to fully understand human experience.

The construction and use of this ecosystem followed carefully orchestrated methodologies. For example, the PKG population phase was not a simple data ingestion but a sophisticated process of knowledge extraction and structuring. Unstructured content, such as text documents or images, was processed through dedicated pipelines that, relying on language and vision models, identified entities, relations, and key concepts, translating them into semantic triples to be inserted into the graph. In parallel, a post-processing phase enriched this foundational structure with vector embeddings and thematic community detection for downstream retrieval tasks.

The use of the PKG for question answering was instead governed by an agent based on the ReAct (Reasoning and Acting) paradigm. This agent does not simply pass the user's question to a single model, but acts as a coordinator that, depending on the detected intent, selects and invokes the most appropriate tool from its toolkit: GraphRAG for textual retrieval, a recommendation module, and—crucially—the visual analysis module. The latter was specifically designed as a fallback strategy: it is activated only when graph-based reasoning proves insufficient, ensuring targeted and efficient use of computational resources.

Validation of this complex architecture was conducted through a rigorous testing protocol, built around a simulated but realistic use scenario fixed at a single global timestamp to guarantee reproducibility. Two distinct datasets were developed to evaluate different dimensions of the system. The first, the Primary Vision Dataset (PVD), was designed to isolate and measure the added value of the visual analysis module in cases where GraphRAG alone was guaranteed to fail. The 29 questions in this set required identifying purely visual details—colors, patterns, object counts—that were not present in the textual PKG. Results showed solid overall performance, with more than 82% of answers rated "Excellent" or "Good", demonstrating that the visual module can effectively fill informational gaps in the graph. However, a 14% failure rate also highlighted the current limitations of the vision model, especially in perceiving very small details or ambiguous scenes.

The second dataset, the Redundant Vision Dataset (RVD), was instead designed to test the robustness of multimodal integration in scenarios where GraphRAG already possessed the necessary information. The goal was to observe whether the visual component improved, worsened, or left unchanged an answer that was already potentially correct. Results were highly encouraging: in 50% of cases, the visual contribution was neutral, confirming consistency between textual and perceptual data without adding noise; in 36% of cases, it produced a clear improvement, correcting conceptual errors from the graph or adding relevant details; and only in 14% of cases did it degrade the output. This balance suggests that the chosen architecture enables safe and often beneficial multimodal integration, turning informational redundancy into a source of robustness rather than conflict.

Looking forward, the work carried out in this thesis opens up numerous promising avenues that could further enhance the capabilities and usefulness of a next-generation personal assistant. A first direct evolution concerns strengthening the visual analysis module itself. Tests revealed difficulty in detecting small details; adopting more recent and powerful vision models, or alternatively more complex visual pipelines that specialize different models on specific tasks (such as scene-text recognition or human pose estimation), could overcome these limitations.

A deeper architectural innovation would consist in connecting the PKG more tightly to the visual reasoning engine, allowing the system to "guide" the VLM's attention using graph context. Instead of analyzing an image generically, the assistant could instruct the model to exploit entities and relations from the graph to make visual analysis more targeted and efficient. Another crucial development axis is the expansion of data modalities integrated into the PKG. The system currently handles documents, photos, and calendar events, but a holistic understanding of the user would require incorporating other context-rich sources such as emails and text messages. Integrating these conversations would enable the assistant to reconstruct interaction timelines, understand social relationships, and answer complex questions about past communication, exponentially enriching the depth of the personal graph.

In parallel, interaction with the assistant could evolve from a purely consultative model to a proactive and collaborative one. A fundamental step in this direction would be enabling the agent to modify the PKG directly upon user request. Commands such as "Add a dentist appointment for next Tuesday at 10" or "Create a note summarizing this conversation and link it to project X" would transform the assistant from a simple retriever of information into an actual manager of the user's digital knowledge assets. Finally, to complete the transition toward truly natural and fluid interaction, the text-based interface could be replaced or complemented by a voice-based one. Integrating speech-to-text and text-to-speech technologies would allow users to converse with their assistant using their voice, making access to personal knowledge immediate and frictionless, similar to interacting with a human collaborator.

However, it is essential to conclude with a reflection on a key aspect for the feasibility and acceptance of systems of this kind: data privacy and security. A personal assistant with access to documents, photos, conversations, and calendars holds an extremely intimate and sensitive archive of an individual's life. Centralizing such data on remote servers, however protected, introduces significant risks. Any real-world implementation of this technology must address this challenge head-on, exploring architectures that prioritize user data sovereignty. Solutions such as on-device processing for sensitive operations, end-to-end encryption for all communications, and the development of decentralized or federated storage models in which the user maintains full physical and legal control over their PKG are not optional luxuries but essential requirements to build trust between humans and such systems. Future research must therefore balance innovation in cognitive capabilities with equal innovation in security guarantees, ensuring that personalization never comes at the expense of privacy.

Bibliography

[1] J. Baek, A. F. Aji, and A. Saffari. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering, jun 2023.

[2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025.

[3] Krisztian Balog and Tom Kenter. Personal Knowledge Graphs: A Research Agenda. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '19, pages 217–220, New York, NY, USA, September 2019. Association for Computing Machinery.

[4] Nolwenn Bernard, Ivica Kostric, Weronika Łajewska, Krisztian Balog, Petra Galuščáková, Vinay Setty, and Martin G. Skjæveland. PKG API: A Tool for Personal Knowledge Graph Management. In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, pages 1051–1054, New York, NY, USA, May 2024. Association for Computing Machinery.

[5] Prantika Chakraborty, Sudakshina Dutta, and Debarshi Kumar Sanyal. Personal Research Knowledge Graphs. In *Companion Proceedings of the Web Conference 2022*, WWW '22, pages 763–768, New York, NY, USA, August 2022. Association for Computing Machinery.

[6] Prantika Chakraborty and Debarshi Sanyal. A comprehensive survey of personal knowledge graphs. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13, August 2023.

[7] Xinyu Geng, Peng Xia, Zhen Zhang, Xinyu Wang, Qiuchen Wang, Ruixue Ding, Chenxi Wang, Jialong Wu, Yida Zhao, Kuan Li, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Webwatcher: Breaking new frontier of vision-language deep research agent, 2025.

[8] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A. Rossi, Subhabrata Mukherjee, Xianfeng Tang, Qi He, Zhigang Hua, Bo Long, Tong Zhao, Neil Shah, Amin Javari, Yinglong Xia, and Jiliang Tang. Retrieval-augmented generation with graphs (graphrag), 2025.

[9] Rasha Hendawi and Juan Li. Comprehensive Personal Health Knowledge Graph for Effective Management and Utilization of Personal Health Data. In *2024 IEEE First International Conference on Artificial Intelligence for Medicine, Health and Care (AIMHC)*, pages 92–100, February 2024.

[10] Sai Chander Janagama. *THE ROLE AI PLAYED IN OFFERING VIRTUAL ASSISTANCE TO USERS*. PhD thesis, 05 2025.

[11] Rajeev Kumar, Harishankar Kumar, and Kumari Shalini. Leveraging knowledge graphs and llms for context-aware messaging, 2025.

[12] Jung X. Lee and Yeong-Tae Song. Digital Twin Using Clinical Personal Knowledge Graphs: Toward Precision Medicine. In *2025 19th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pages 1–7, January 2025.

[13] Linjie Li, Zhe Gan, Yu Cheng, and Jingjing Liu. Relation-aware graph attention network for visual question answering, 2019.

[14] Xiang Li, Gokhan Tur, Dilek Hakkani-Tür, and Qi Li. Personal knowledge graph population from user utterances in conversational understanding. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 224–229, December 2014.

[15] Lingyuan Liu, Huifang Du, Xiaolian Zhang, Mengying Guo, Haofen Wang, and Meng Wang. A Question-Answering Assistant over Personal Knowledge Graph. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pages 2708–2712, New York, NY, USA, July 2024. Association for Computing Machinery.

[16] L. Luo, Y.-F. Li, G. Haffari, and S. Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning, feb 2024.

[17] Abiram Mohanaraj and Elisabeth Niemeyer Laursen. Entity Linking to Dynamically-Evolving Personal Knowledge Graphs in Conversations.

[18] Manisha Mukherjee, Sungchul Kim, Xiang Chen, Dan Luo, Tong Yu, and Tung Mai. From documents to dialogue: Building kg-rag enhanced ai assistants, 2025.

[19] Lin Ning, Luyang Liu, Jiaxing Wu, Neo Wu, Devora Berlowitz, Sushant Prakash, Bradley Green, Shawn O'Banion, and Jun Xie. User-llm: Efficient llm contextualization with user embeddings, 2024.

[20] Zihan Niu, Zheyong Xie, Shaosheng Cao, Chonggang Lu, Zheyu Ye, Tong Xu, Zuozhu Liu, Yan Gao, Jia Chen, Zhe Xu, Yi Wu, and Yao Hu. Part: Enhancing proactive social chatbots with personalized real-time retrieval, 2025.

[21] R. Omar, O. Mangukiya, P. Kalnis, and E. Mansour. ChatGPT versus traditional question answering for knowledge graphs: Current status and future directions towards knowledge graph chatbots, feb 2023.

[22] Yilun Qiu, Xiaoyan Zhao, Yang Zhang, Yimeng Bai, Wenjie Wang, Hong Cheng, Fuli Feng, and Tat-Seng Chua. Measuring what makes you unique: Difference-aware user modeling for enhancing llm personalization, 2025.

[23] Hugues Sansen, Gérard Chollet, Kristiina Jokinen, Maria Ines Torres, Jérôme Boudy, and Mossab Hariz. Fuzzy Dates in Personal Knowledge Graphs and Dialogue, the Example of "LifeLine". In *2024 16th International Conference on Human System Interaction (HSI)*, pages 1–5, July 2024. ISSN: 2158-2254.

[24] Markus Schröder, Christian Jilek, and Andreas Dengel. A Human-in-the-Loop Approach for Personal Knowledge Graph Construction from File Names. March 2022.

[25] P. Sen, S. Mavadia, and A. Saffari. Knowledge graph-augmented language models for complex question answering. In B. Dalvi Mishra, G. Durrett, P. Jansen, D. Neves Ribeiro, and J. Wei, editors, *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 1–8, Toronto, Canada, jun 2023. Association for Computational Linguistics.

[26] Martin G. Skjæveland, Krisztian Balog, Nolwenn Bernard, Weronika Łajewska, and Trond Linjordet. An Ecosystem for Personal Knowledge Graphs: A Survey and Research Roadmap. *AI Open*, 5:55–69, 2024. arXiv:2304.09572 [cs].

[27] Y. Sui, Y. He, N. Liu, X. He, K. Wang, and B. Hooi. FiDeLiS: Faithful reasoning in large language model for knowledge graph question answering, may 2025.

[28] J. Sun et al. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph, mar 2024.

[29] A. Toroghi, W. Guo, M. M. A. Pour, and S. Sanner. Right for right reasons: Large language models for verifiable commonsense knowledge graph question answering, mar 2025.

[30] Lingraj S. Vannur, Balaji Ganesan, Lokesh Nagalapatti, Hima Patel, and M. N. Thippeswamy. Data Augmentation for Personal Knowledge Base Population, February 2020.

[31] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory, 2023.

[32] Xin Wang, Wordh Ul Hasan, Juan Li, Rasha Hendawi, Yang Du, Bo Xie, and Cui Tao. Empowering Personalized Health Data Queries with Knowledge Graph and GPT-Enhanced Voice Assistant. In *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, pages 1035–1042, December 2023.

[33] Y. Wu et al. CoTKR: Chain-of-thought enhanced knowledge rewriting for complex knowledge graph question answering, mar 2025.

[34] Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, Zhenda Xie, Yu Wu, Kai Hu, Jiawei Wang, Yaofeng Sun, Yukun Li, Yishi Piao, Kang Guan, Aixin Liu, Xin Xie, Yuxiang You, Kai Dong, Xingkai Yu, Haowei Zhang, Liang Zhao,

Yisong Wang, and Chong Ruan. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding, 2024.

[35] Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. Deep modular co-attention networks for visual question answering, 2019.

[36] Ruifeng Yuan, Shichao Sun, Yongqi Li, Zili Wang, Ziqiang Cao, and Wenjie Li. Personalized large language model assistant with evolving conditional memory, 2024.

[37] Yufei Zhan, Yousong Zhu, Shurong Zheng, Hongyin Zhao, Fan Yang, Ming Tang, and Jinqiao Wang. Vision-r1: Evolving human-free alignment in large vision-language models via vision-guided reinforcement learning, 2025.

[38] Xinjie Zhao, Moritz Blum, Rui Yang, Boming Yang, Luis Márquez Carpintero, Mónica Pina-Navarro, Tony Wang, Xin Li, Huitao Li, Yanran Fu, Rongrong Wang, Juntao Zhang, and Irene Li. Agentigraph: An interactive knowledge graph platform for llm-based chatbots utilizing private data, 2024.