TANGO

D4.1 TANGO system specifications

Submission date: 23/12/2024

Due date: 30/09/2024

DOCUMENT SUMMARY INFORMATION

Grant Agreement No	101120763	**Acronym**	TANGO
Full Title	It takes two to tango: a synergistic approach to human-machine decision-making		
Start Date	01/10/2023	**Duration**	48 months
Project type	Research and Innovation Action (RIA)	**Funding programme**	Horizon Europe
Deliverable	D4.1: TANGO system specifications		
Work Package	WP4 – The TANGO software ecosystem		
Type	R	**Dissemination Level**	PU
Lead Beneficiary	UH		
Authors	Daniele Miorandi, Stefano Tavonatti, Marco Angheben (UH)		
Co-authors	Roberto Pellungrini (SNS), Carlo Metta (CNR), Salvatore Rinzivillo (CNR), Francesca Naretto (UNIPI), Thomas Kehrenberg (BCAM), Simone Brentan (UH), Ben Wilson (UoS), Burcu Sayin Günel (UNITN), Rossana Bartolacelli (UH), Nicola Arpino (UH), Samuel Bortolin (UH), Elena Leonelli (UH), Simone Piaggesi (UNIPI), Luca Zardini (UH), Vincenzo Marco De Luca (UNITN), Erich Robbi (UNITN), Tim Tobiasch, Wolfgang Stammer (TUDa), Samuele Bortolotti (UNITN), Claudio Giovannoni (UNIPI), Massimiliano Luca (FBK), Simone Centellegher (FBK), Oliver Thomas (BCAM)		
Reviewers	Ronald Chenu Abente (UniTN)		

Funded by the European Union

DISCLAIMER

COPYRIGHT

DOCUMENT HISTORY

Version	Date	Changes	Contributor(s)
V0.1	12/07/2024	Initial structure and ToC	Daniele Miorandi
V0.2	30/08/2024	Initial architecture design and core specifications.	Stefano Tavonatti & Marco Angheben
V0.3	04/09/2024	Added executive summary, introduction and conclusion	Daniele Miorandi
V0.4	06/09/2024	Revised architecture design and core specifications. Added deployment considerations.	Stefano Tavonatti & Marco Angheben
V0.5	06/09/2024	Added requirements specifications.	Daniele Miorandi & Marco Angheben
V0.6	09/09/2024	Alignment and content; revision; submission to internal quality assessment.	Daniele Miorandi

V0.9	19/09/2024	Revised structure and content following comments from internal quality assessment.	Daniele Miorandi, Marco Angheben, Stefano Tavonatti
V1.0	20/09/2024	Final touches	Daniele Miorandi
v1.1	23/12/2024	Revised according to the feedback received during the M12 review meeting.	Daniele Miorandi, Marco Angheben, Stefano Tavonatti, Ben Wilson, Erich Robbi, Burcu Sayin Günel, Francesca Naretto, Salvatore Rinzivillo, Massimiliano Luca, Simone Centellegher, Wolfgang Stammer, Tim Tobiasch

PROJECT PARTNERS

Partner	Country	Short name
UNIVERSITÀ DEGLI STUDI DI TRENTO	IT	UNITN
TECHNISCHE UNIVERSITÄT DARMSTADT	DE	TUDa
UNIVERSITA DI PISA	IT	UNIPI
UNIVERSITE PARIS CITE	FR	UPC
FONDAZIONE BRUNO KESSLER	IT	FBK
CARR COMMUNICATIONS LIMITED	IE	CARR
ISTRAZIVACKO-RAZVOJNI INSTITUT ZA VESTACKU INTELIGENCIJU SRBIJE	RS	IVI
SURGICAL SCIENCE SWEDEN AB	SE	SuS
UNIVERSITÄTSKLINIKUM HEIDELBERG	DE	UKHD
CENTRE FOR EUROPEAN POLICY STUDIES	BE	CEPS
BCAM - BASQUE CENTER FOR APPLIED MATHEMATICS	ES	BCAM
U-HOPPER SRL	IT	UH
INTESA SANPAOLO SPA	IT	ISP
EIT DIGITAL	BE	EITD
SHARE FONDACIJA	RS	SHARE
A11-INITIATIVE FOR ECONOMIC AND SOCIAL RIGHTS	RS	A11
MINISTARSTVO ZA BRIGU O PORODICI I DEMOGRAFIJU	RS	MFWD
AZIENDA PROVINCIALE PER I SERVIZI SANITARI	IT	APSS
SWANSEA UNIVERSITY	UK	UoS
THE UNIVERSITY OF WARWICK	UK	UoW

Funded by
the European Union

LIST OF ACRONYMS

Acronym	Definition
DoA	Description of Action
EC	European Commission
HaDEA	European Health and Digital Executive Agency
WP	Work Package
API	Application Programming Interface
REST	Representational State Transfer
RFC	Request for Comments
MLOps	Machine Learning Operations
CI/CD	Continuous integration and continuous delivery
UC	Use case
BPMN	Business Process Model and Notation
DAG	Direct Acyclic Graph
MR	Merge Request
LLM	Large Language Model
AR	Algorithmic Recourse

Executive Summary

The TANGO project represents a pioneering step in the evolution of Hybrid Decision Support Systems (HDSS), merging human expertise and AI capabilities to enhance decision-making processes across various sectors. This deliverable, titled "TANGO System Specifications," outlines the architectural framework and operational specifics of the TANGO backend system. Designed to support advanced AI-driven applications, the platform is built to be scalable, secure, and adaptable to real-world challenges.

At the core of this deliverable is the design of the TANGO backend system, created to enable seamless interaction between AI models and human users, fostering collaborative decision-making. The architecture was carefully crafted, drawing from real-world use cases, and guided by principles such as modularity, flexibility, and openness. The system is designed to be robust, with each component—from the TANGO Library and Orchestrator to the API and storage solutions—playing a crucial role in delivering on these promises.

Throughout the development process, a strong emphasis was placed on ensuring that the system can meet the diverse needs of the diverse real-world use cases addressed by the project in WP5. Functional and non-functional requirements were analysed to guarantee that the platform can handle demanding performance, scalability, and security requirements, while also providing ease of integration through open standards. Additionally, the architecture is supported by a comprehensive set of APIs, developed using the OpenAPI standard, to facilitate interaction with external applications. These APIs play a pivotal role in making the platform both developer-friendly and extensible, allowing external teams to build on the platform's capabilities and integrate it into their existing workflows.

With deployment flexibility in mind, the TANGO system is designed to adapt to a variety of environments, from cloud-based infrastructures to specialised third-party applications. This ensures that the platform is not only capable of scaling with demand but also meets stringent security and privacy requirements, particularly in sensitive sectors like healthcare and finance.

Ultimately, this deliverable provides a solid foundation for the ongoing development of the TANGO platform, setting the stage for future enhancements and innovations. By adhering to the architectural principles and design laid out in this document, the project is well-positioned to meet its goal of creating AI-driven systems that amplify human decision-making, transforming the way organisations approach complex challenges.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

Not applicable

1 Introduction

1.1 Background

The TANGO project, funded under the Horizon Europe program by the European Union, signifies a major advancement in the integration of artificial intelligence (AI) with human decision-making processes. With a budget of €7 million, TANGO is dedicated to developing the theoretical and computational frameworks necessary for creating synergistic human-machine decision-making systems. This initiative not only aims to pioneer a new generation of human-centric AI systems but also strives to establish Europe's leadership in this transformative technology domain.

The primary mission of TANGO is to enhance the decision-making capabilities across critical areas by reducing cognitive overload and biases, which are common in high-stakes environments like healthcare, finance, and public policy. Despite the vast potential of AI to improve outcomes in these settings, its adoption has been slow, often hindered by trust and reliability concerns. TANGO addresses these challenges by focusing on creating systems that are not only technologically advanced but also transparent and aligned with human values and needs.

The project brings together 20 partner organizations from nine European countries, pooling a wide array of expertise to push the boundaries of what can be achieved through collaborative human-machine interaction. The real-world impact of TANGO will be measured through various high-impact use cases, including support for women during and after pregnancy, assistance for surgical teams during operations, enhancements in credit lending processes, and optimization of public policy design and fund allocation.

By fostering a deep mutual understanding and symbiotic relationship between humans and AI, TANGO sets the stage for developing AI systems that genuinely extend human capabilities, ensuring that technology serves society in meaningful and sustainable ways. This foundational approach not only underpins the project's goals but also guides its strategy for future technological and social integration.

1.2 Purpose of the Document

This document, "TANGO System Specifications," serves a critical role in the overarching TANGO project by detailing the architectural specifications required to develop and deploy the backend of Hybrid Decision Support Systems (HDSS). It is designed to provide a comprehensive blueprint that outlines the technical and operational framework necessary for integrating advanced artificial intelligence functionalities with human decision-making processes. The purpose of this deliverable is to ensure that all project stakeholders, including developers, project teams, and partner organizations, have a clear and shared understanding of the system architecture, component interactions, API specifications, and deployment strategies. This clarity is essential for the successful implementation, scalability, and maintenance of the systems developed under the TANGO initiative, facilitating a smooth transition from theoretical models to practical, real-world applications.

1.3 Document Structure

This document is structured to provide a comprehensive overview and detailed explanation of the architectural framework for the TANGO backend system, facilitating a clear understanding for all stakeholders involved in the project. The structure is organized as follows to guide the reader through the essential components and specifications of the system:

Section 2: System Requirements Analysis - This section delineates the functional and non-functional requirements that define the scope and expectations of the TANGO backend system. It lays the foundation by specifying what the system is expected to achieve and the standards it must adhere to, ensuring alignment with the project's objectives.

Section 3: Core Specifications - Detailed information about the core components of the TANGO system is provided here, including a breakdown of both internal and external components and their interactions. This section serves as the technical core of the document, describing the component structure and functionality necessary for the system's operation.

Section 4: Architecture Design - A comprehensive description of the architectural principles, guidelines, and the actual architecture of the TANGO system is provided. This includes high-level designs and detailed views, such as sequence diagrams and data flows, which illustrate how the system components interact and operate within the defined architecture.

Section 5: API Specifications - This section outlines the specifications for component-level and platform-level APIs following the OpenAPI standard. It focuses on how these APIs facilitate interaction with the system, enabling functionality for developers and other systems within the TANGO Ecosystem.

Section 6: Deployment Considerations - Details regarding various deployment architectures, scalability, performance optimizations, and security measures are discussed here. This section ensures that the system architecture is not only theoretically sound but also practically deployable across different environments and scalable to meet future demands.

Section 7: Conclusion - Summarizes the key points covered in the document and outlines future work and potential enhancements. This final section reflects on the document's contents and their implications for the ongoing development and success of the TANGO project.

Each section is designed to build upon the information provided in the previous ones, ensuring a logical flow that facilitates both depth and breadth of understanding. Diagrams, flowcharts, and tables are included throughout the document to aid in visualizing concepts and systems discussed, enhancing comprehension and usability of the information provided.

The documentation is maintained live on the project git at https://gitlab.com/tango-ecosystem/platform-documentation/-/tree/main .
Some diagrams, hereby reported for the sake of completeness, may be hard to read in printed version. The level of details therein is required for development purposes. High-resolution versions of all images can be

conveniently retrieved from the project git at: https://gitlab.com/tango-ecosystem/platform-documentation/-/tree/main/diagrams .

2 System Requirements Analysis

The requirements for the TANGO backend system were elicited through a process involving a number of interactions with WP5, the work package focused on the development of case studies and pilot implementations. The need to get actionable feedback from case studies in WP5 triggered a co-design, collaborative process, in which the case study leaders were first asked to provide early versions of the Pilot Handbook (see Deliverable D5.1 for details), while specific follow-ups were carried out with technical supporting partners identified for each case study (said technical supporting partners are by design active in WP4, so they act as a "natural" bridge towards the use case and application scenarios).
In particular, with reference to the case studies, this liaison role was covered by teams having access to end-user groups able to feed back on issues of situated use:

- T5.2 - Supporting women in perinatal health and wellbeing: TUDa
- T5.3 - Supporting surgical teams in making intraoperative decisions: UniTN
- T5.4 - Supporting loan officers and loan applicants in credit lending: FBK
- T5.5 - Supporting policy makers in shaping social welfare policies: FBK

This collaborative approach ensures that the system's core functionalities align closely with the real-world needs of the users and applications envisaged in the project. As work on the case study and pilot implementation is still in early stage, we do expect that some of the requirements highlighted in the following subsections may change over the next months; the adoption of an agile approach ensures that WP4 is ready to iteratively revise and update requirements (and the consequent technological choices) over the remainder of the project lifetime.

2.1 Methodological Process for Requirements Collection and Evolution

The requirements collection process for the TANGO system is designed to ensure alignment with contextually situated end-user needs and project objectives, particularly focusing on the real-world applications embodied in the pilot use cases in WP5. This iterative and adaptive process involved the following key steps:

1. *Initial input from trial handbooks* - As a starting point, we consulted early versions of the "Trial Handbooks" (referenced in Deliverable D5.1) as contributed by the pilot leaders. These documents provided an initial understanding of the functional and non-functional requirements based on the specific contexts of each pilot. It also led WP4 to ask for a number of clarifications on the actual application scenarios, clarifications that were deemed necessary to select a consistent and coherent set of requirements.
2. *Stakeholder consultations* - To refine our understanding, structured discussions were conducted with the stakeholders and developers of each use case, supported and facilitated by the aforementioned

technical supporting partners. These interactions aimed to contextualize the requirements within the operational environments of the pilots, identifying constraints, dependencies, and additional needs. These consultations proved mutually beneficial and inevitably helped develop a shared appreciation of both the human and technical development needs of the project.

3. *Iterative refinement and validation* - The feedback obtained was reviewed collaboratively within the technical team in WP4 to align the requirements with the system's overarching design principles, such as modularity and flexibility. Special attention was given to ensuring the architecture could accommodate not only the current pilots but also potential future use cases. This involved balancing specific requirements with the need for a generic, extensible system design.
4. *Mechanisms for accommodating evolving requirements* - Recognizing that pilot use cases may evolve or introduce new inputs during the project lifecycle, the following mechanisms have been put in place to ensure the system remains adaptable:
    - *Agile development approach*: the project follows an agile methodology, allowing periodic (~every 6 months) revisits of requirements. This ensures that any new insights or changes in pilot needs can be integrated into the system's specifications and architecture in a controlled manner.
    - *Support for formative evaluation*: the project encourages and supports early evaluation of how case study formulation can make use of system outputs and interactions. Formative, situated evaluation is promoted in order to obtain the maximum benefit from human-machine synergies.
    - *Continuous feedback loops*: regular check-ins with stakeholders and pilot developers will be conducted as part of ongoing WP5 activities. These feedback loops allow for early identification of changes or emerging needs, minimizing disruption to the overall development timeline and ensuring WP4/WP5 alignment.
    - *Versioning and backlog management*: a dedicated requirements backlog is maintained, where new or modified requirements are logged, prioritized, and incorporated into the development roadmap. This process ensures traceability and transparency in how changes are handled.
    - *Design for flexibility*: the system architecture has been intentionally designed to support modularity and scalability. This is meant to enable the smooth incorporation of new features, use cases, or technological advancements without requiring major architectural overhauls.
5. *Integration of feedback into specifications* - As new inputs are received by pilot use cases, they will be systematically integrated into the system specifications, ensuring traceability to the original input and alignment with the TANGO project goals.

The overall process is schematically represented in the following image, following the BPMN 2.0 standard.

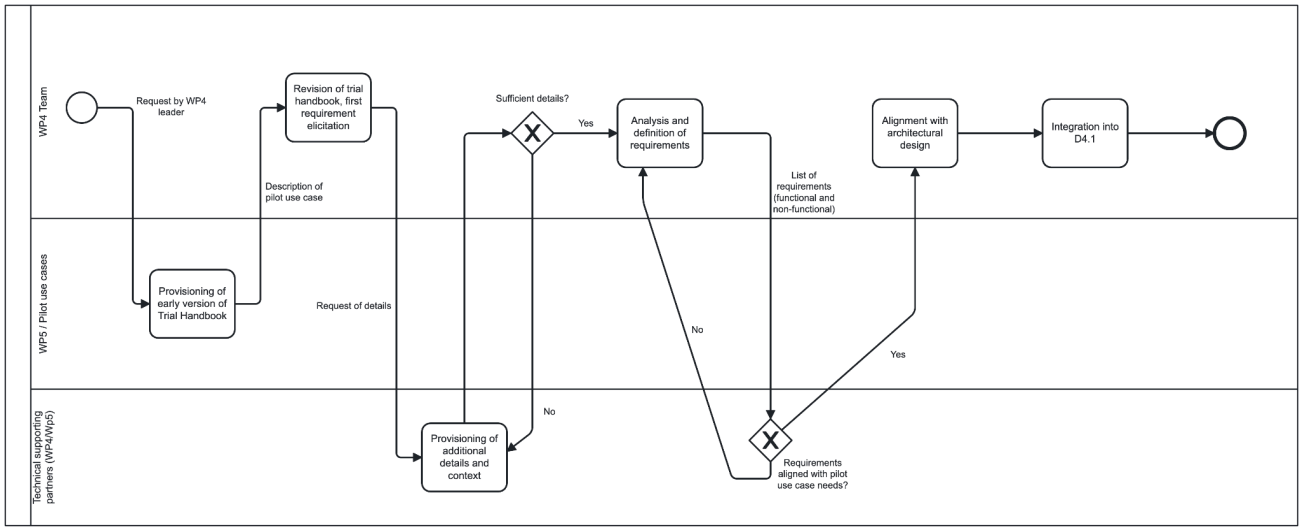

Figure 1: BPMN representation of the requirements collection process.

2.2 Functional Requirements

In alignment with RFC 2119, the following functional requirements outline the critical behaviors that the TANGO backend system **MUST**, **SHOULD**, and **MAY** support to fulfill the goals of the Hybrid Decision Support Systems (HDSS). These requirements ensure that the system can facilitate human-AI collaboration effectively and transparently across diverse decision-making scenarios.

- AI model management: the system MUST provide capabilities for managing the full lifecycle of AI models, including training, deployment, monitoring, and versioning. It MUST support various types of models, such as predictive models and generative models, enabling users to create, modify, and roll back to previous model versions as needed. Additionally, model reusability and sharing SHOULD be encouraged by supporting standardized model formats, ensuring easy integration with external systems.
- Mixed-initiative interaction: the system MUST enable both humans and AI to initiate interactions and actions during the decision-making process, following a mixed-initiative model. Said interactions and actions are always mediated by a purposeful application, which interacts with the TANGO backend system. This functionality SHOULD allow for both synchronous and asynchronous interactions, ensuring flexibility in how users engage with AI models. The system MUST also allow users to interrupt, modify, or guide the decision process based on AI recommendations in real time.
- Data ingestion and preprocessing: the TANGO backend system MUST ingest data from multiple, heterogeneous sources and MUST support flexible preprocessing pipelines that clean, normalize, and transform the data into formats suitable for model training or inference. It SHOULD ensure compatibility with both structured and unstructured data to maximize the system's adaptability to various use cases.

- Prediction and recommendation services: the system MUST provide reliable APIs for invoking models to generate predictions or recommendations in real-time. These services MUST be designed to handle high-frequency requests and return results that are comprehensible and actionable for end-users. Additionally, batch predictions SHOULD be supported for large-scale data processing.
- Explainability and transparency: the system MUST offer explainability features that enable users to understand the reasoning behind AI-driven decisions. It MUST be capable of generating explanations in a manner aligned with the EU AI Act's requirement for transparency in AI systems. These explanations SHOULD be contextualized based on the specific use case, ensuring that users receive meaningful and actionable insights into how decisions were made.
- User and role management: the platform MUST implement robust user and role management systems, ensuring that different stakeholders have appropriate levels of access to models, data, and services. It MUST support authentication and authorization mechanisms, including multi-factor authentication, to safeguard access.

2.3 Non-functional Requirements

In addition to functional requirements, non-functional requirements ensure that the TANGO platform is reliable, scalable, secure, and compliant with key regulations. The requirements outlined below adhere to the language of RFC 2119, specifying what the system **MUST**, **SHOULD**, and **MAY** provide to meet these non-functional criteria.

- Scalability: the system MUST be designed for horizontal scalability, allowing it to handle growing amounts of data, increasing user loads, and expanding model deployments. Components such as the TANGO Orchestrator and Model Server MUST scale automatically to maintain performance under high-demand scenarios. This scalability SHOULD be achieved with cloud-native architectures, leveraging containers or serverless functions to dynamically allocate resources.
- Performance: the TANGO backend MUST ensure high-performance standards, particularly for real-time predictions and decision-making tasks. Response times SHOULD be minimized, ensuring that users experience low latency during interactions. For training and batch processing tasks, the system SHOULD optimize resource usage to balance speed and cost efficiency.
- Security: security MUST be embedded throughout the system, with strong encryption for both data at rest and in transit. The system MUST support attribute-based access control (ABAC) and SHOULD include audit logs to track user actions. To protect sensitive data, compliance with GDPR and encryption protocols MUST be ensured. The platform MUST support multi-factor authentication for all user accounts to safeguard access.
- Compliance: the TANGO system MUST adhere to the European Union’s AI Act, ensuring that all AI models meet the transparency, robustness, and accountability standards outlined in the regulation. Additionally, the platform MUST comply with the Digital Services Act (DSA) when interacting with third-party data providers, particularly regarding data governance, transparency, and content moderation responsibilities. Given that the TANGO system will be used in the pilot studies in WP5, and that this will include processing of personal data, the TANGO system MUST comply with existing privacy and data protection regulations, including GDPR.

- Reliability and availability: the system MUST guarantee high availability (99.9% uptime or higher) through redundancy and failover mechanisms. Key components, such as the Orchestrator and API services, SHOULD be designed with redundancy in mind to ensure continuous operation, even in the case of hardware failures or network outages.
- Interoperability: the platform MUST support interoperability with external AI systems, MLOps platforms, and data sources. Standardized APIs (using OpenAPI) SHOULD be implemented to allow seamless interaction between the TANGO system and third-party systems, ensuring that the platform can easily integrate with new tools and technologies.
- Openness, transparency, and reusability: the TANGO platform SHOULD prioritise openness and transparency by leveraging open-source technologies wherever feasible. This approach supports auditability, security, and community-driven contributions while fostering reuse and adaptability. The system's codebases MUST be made publicly available and licensed in a manner that promotes collaboration, ensuring integrity and compliance with security standards. Additionally, the platform MUST employ open standards and APIs to enhance ease of integration, encouraging third-party development and building a collaborative ecosystem around the platform.

By adhering to these non-functional requirements, the TANGO system will be well-positioned to meet the operational demands of hybrid decision-making environments, ensuring that it is scalable, reliable, secure, and compliant with the latest European regulations.

3 Architecture Design

3.1 Architectural Guidelines

To ensure a robust and flexible architecture, the following guidelines have been followed.

3.1.1 Modularity and Separation of Concerns

Each component in the diagram should have a single responsibility and be loosely coupled with other components.

The TANGO Library manages available utility models; the TANGO Model interface defines model specifications; the TANGO Model focuses on packaging models weights, support data and prediction features. Defining clear, versioned interfaces for each component, particularly the TANGO API, will allow individual components to evolve without breaking the system's overall functionality.

TANGO MLOps manages machine learning operations; the TANGO Orchestrator coordinates the overall workflow.

3.1.2 Interoperability and Reusability

Components should be designed to be interoperable and reusable across different parts of the system.

The TANGO API exposes well-defined interfaces that allow seamless interaction with the other TANGO components and with external systems: this ensures that the system can easily integrate with new MLOps tools or data providers without significant changes.

The TANGO Library gathers components that can be reused, ensuring consistency and reducing redundancy.

The TANGO Model and TANGO Model Proxy interfaces standardize the way those components behave in order to be seamlessly integrated in the TANGO Ecosystem.

3.1.3 Scalability and Performance Optimization

The architecture should be able to scale horizontally or vertically to handle increased loads while maintaining performance.

The TANGO Storage is designed to scale horizontally with the volume of data, ensuring that it can handle an increasing number of models and associated data without bottlenecks.

The TANGO Orchestrator is optimized to handle concurrent operations efficiently, enabling it to manage multiple workflows simultaneously without degradation in performance.

3.1.4 Flexibility and Extensibility

The system should be designed to accommodate future changes with minimal impact on existing components.

The TANGO Library, TANGO Model Code and TANGO Model Proxy interfaces allow for the easy addition of new models or extensions to the ecosystem.

External Integration Points ensure that the system can easily integrate with external MLOps tools or data providers by abstracting the integration points and using standardized communication protocols.

3.1.5 Security and Compliance

Security should be embedded in every component of the system, with compliance to industry standards.

The TANGO API and Storage implement robust authentication and authorization mechanisms to secure access to the API and storage. Data in transit is encrypted using TLS communications.

3.1.6 Observability and Monitoring

Ensure that the system's health and performance can be easily monitored and debugged.

The TANGO MLOps and Orchestrator integrate detailed logging, monitoring and alerting mechanisms that track the performance and health of the system; this will help in identifying issues early and scaling the system based on demand. Moreover, implementing logging across all components will allow TANGO to track data flow and identify bottlenecks or failures.

3.1.7 Documentation and Best Practices

Maintain comprehensive documentation and enforce coding best practices across the architecture.

Each component (e.g., TANGO Library, TANGO Model, etc.) presents a detailed documentation covering its purpose, interfaces, dependencies, and expected behavior. Implementing and enforcing coding standards, particularly for components that are likely to be reused or extended, ensure long-term maintainability.

3.2 Overview of architecture components

The following is a high-level component diagram representing the architecture of the TANGO Ecosystem.

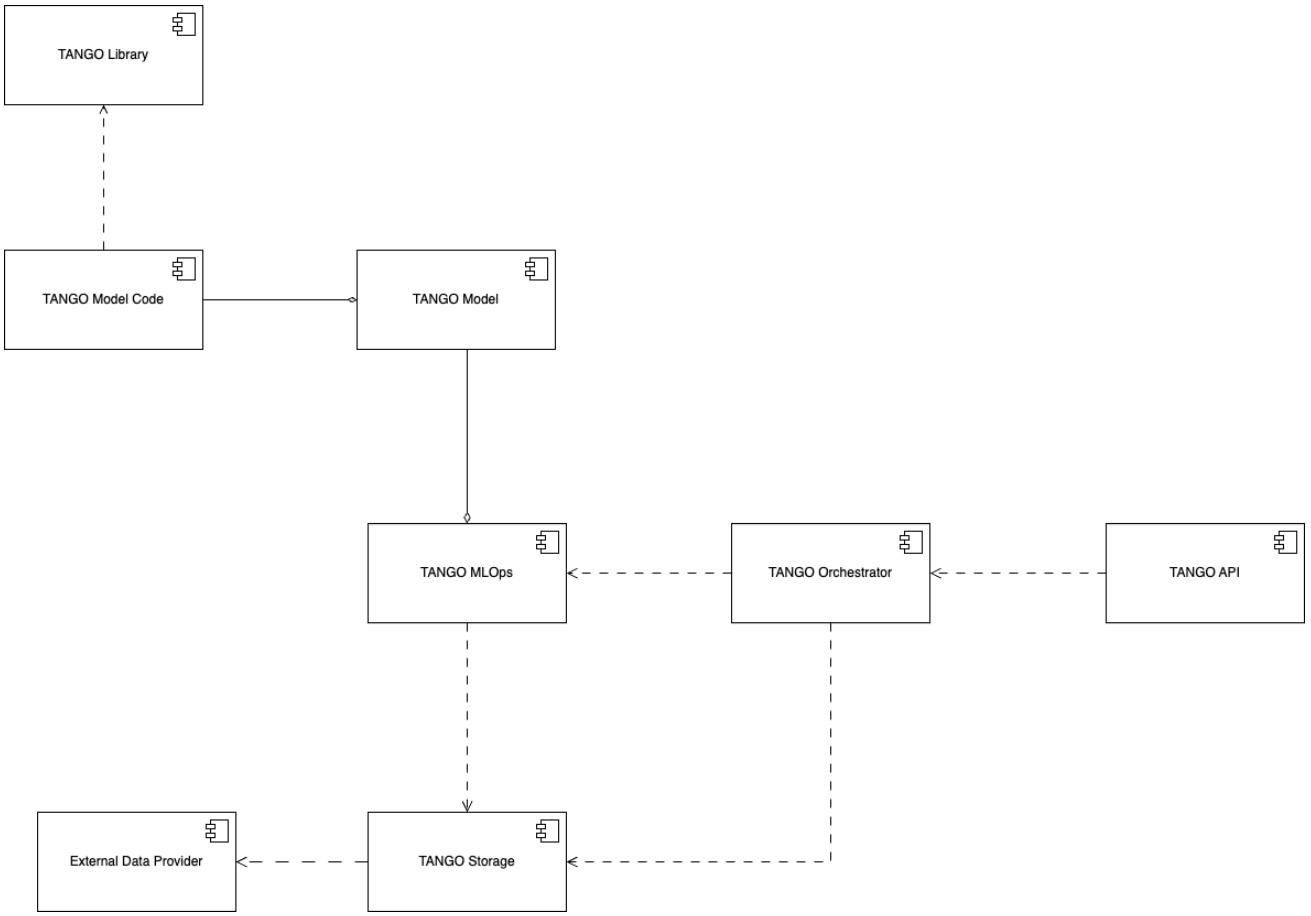

Figure 2: High-level TANGO Ecosystem architecture

TANGO is based upon a modular architecture, where different components interact with each other to manage machine learning models and their lifecycle. The TANGO system is designed to be extensible, allowing for modular integration with external MLOps platforms and data providers, thus enabling a flexible and scalable machine learning operation environment.

3.3 Detailed architecture components

3.3.1 Core components and their roles

TANGO Library

This component represents a shared library containing functional interfaces and utilities for TANGO models development. In particular, models developed within the scope of WP2 and WP3 will depend on the TANGO library content, implementing its behaviours or using its modules (see Annex E for more details).

TANGO Model Code

The TANGO Model Code component represents the machine learning codebase used within the TANGO system. This includes various types of ML models, such as predictive models, that once trained become TANGO Model instances (artifacts) available for different tasks. The TANGO Model Code must implement TANGO Library interfaces and can use one or more components or utilities coming from the TANGO Library.

A TANGO Model Code should expose the following features:

- pre/post processing data,
- provide both training and invocation behaviour,
- local storage management.

TANGO Model

This component wraps:

- the model generated by the TANGO Model Code, such as trained models,
- model parameters and/or
- any output produced during the model's lifecycle.

A TANGO Model typically includes code, weights (pickle or similar) and dataset useful for support (prediction, explanation, ...).

TANGO Storage

The TANGO Storage is a distributed component that manages the storage of data, models and other relevant information within the TANGO system. It includes databases, file systems and/or cloud storage services.

TANGO MLOps

The TANGO MLOps (Machine Learning Operations) is the component responsible for managing the operational aspects of the machine learning models lifecycle, including deployment, monitoring and maintaining models in production.

TANGO Orchestrator

The TANGO Orchestrator is responsible for managing and coordinating the various tasks and processes across the TANGO system. It manages the execution of workflows, task scheduling and interaction between different components.

TANGO API

This component provides an interface for external systems or applications to interact with the TANGO platform. The TANGO API may expose endpoints for accessing and retrieving models, running predictions, and more.

3.3.2 External components

External Data Provider

The External Data Provider component represents external sources of data that TANGO might consume. This could include data feeds, databases or APIs that supply the necessary data for model training, predictions or other operations.

3.3.3 Interactions between components

The TANGO Library is utilized:

- by the TANGO Model Code as a blueprint to expose features and as a repository of utility functions;
- by the TANGO MLOps as a blueprint to expose features;
- by the TANGO Orchestrator as a blueprint to wrap all requests and responses in a way they are correctly read by the MLOps component.

The TANGO Model stores the TANGO Model Code, weights, datasets and results.

The TANGO MLOps works closely with the TANGO Model and with the TANGO Orchestrator to ensure that the models are correctly deployed, managed and maintained.

The TANGO Orchestrator coordinates the TANGO API, the TANGO Storage and other components to manage workflows and tasks.

The TANGO API serves as the single access point for external entities to interact with the TANGO system, potentially involving External MLOps and External Data Provider.

The TANGO Storage serves as the backbone for storing all essential data and models that are used or generated within the TANGO system.

External MLOps and External Data Provider are integrated into the TANGO system, allowing external data and operations to be part of the overall workflow managed by TANGO.

4 TANGO Core specifications

4.1 Components

The TANGO architecture, structured as depicted in Figure 2, is designed for flexibility, allowing easy integration of new models and components. The use of external providers and MLOps ensures that the system can scale and adapt to different use cases.

4.1.1 TANGO Library

The TANGO Library is the core of the TANGO Ecosystem, providing reusable components such as interfaces, classes, algorithms and utilities. Its purpose is to:

- support model development with shared artifacts,
- enable integration of models into the TANGO Ecosystem.

The TANGO Interfaces reside in the TANGO Library and allow contributors to add new models/extensions easily, defining functional constraints for

- models (interface TangoModel)
- prediction explainability (interface ExplainableTangoModel)
- standardized metadata for model registration (TangoModelMetadata and ExplainableTangoModelMetadata).

WP2 and WP3 will enhance the Library by:

- implementing interfaces to meet ecosystem constraints.
- adding AI methods to repositories, meeting the requirements detailed in Annex E.

Refer to Annex E for a more detailed description.

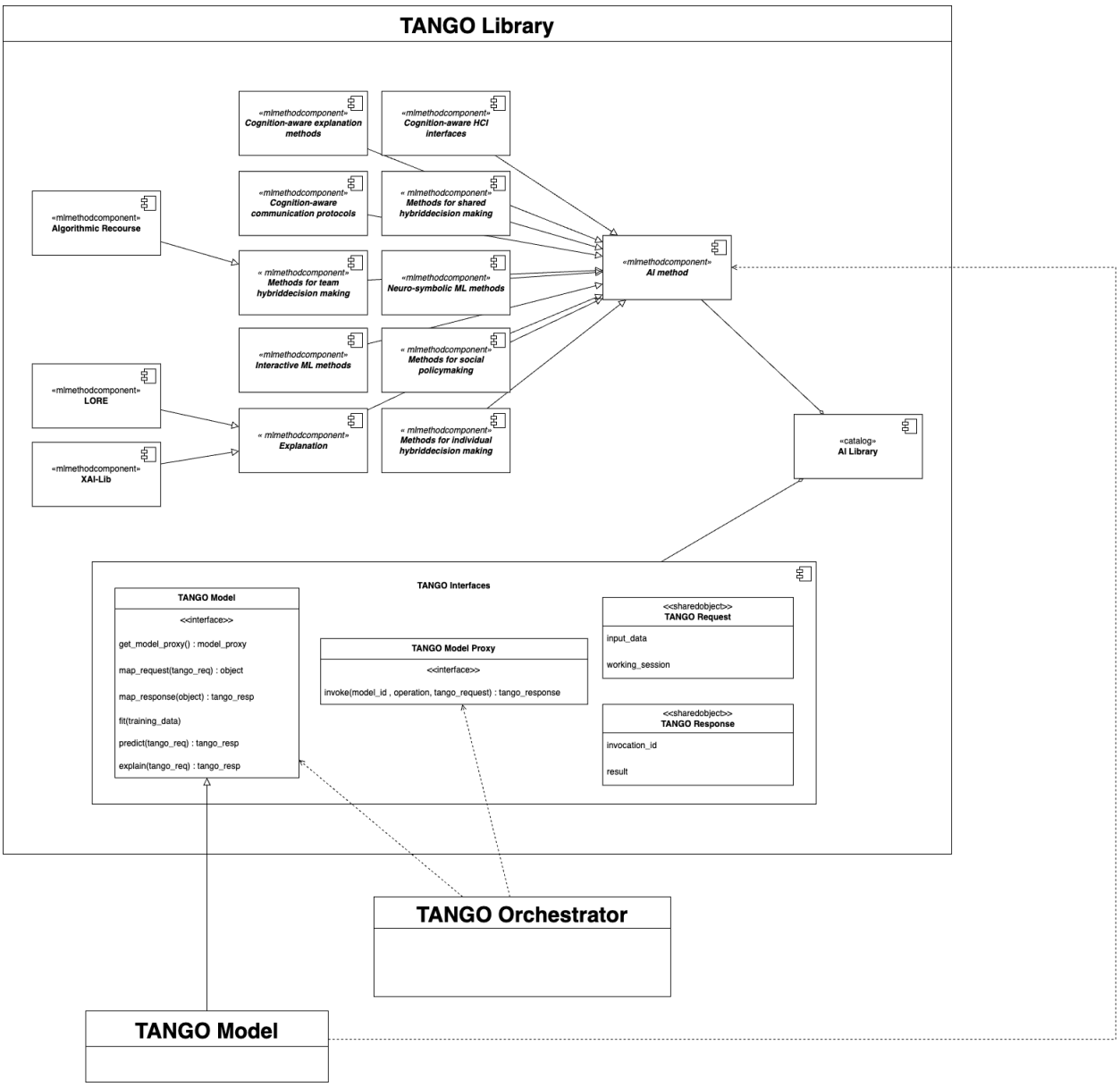

Figure 3: The TANGO Library

4.1.2 TANGO Model Code

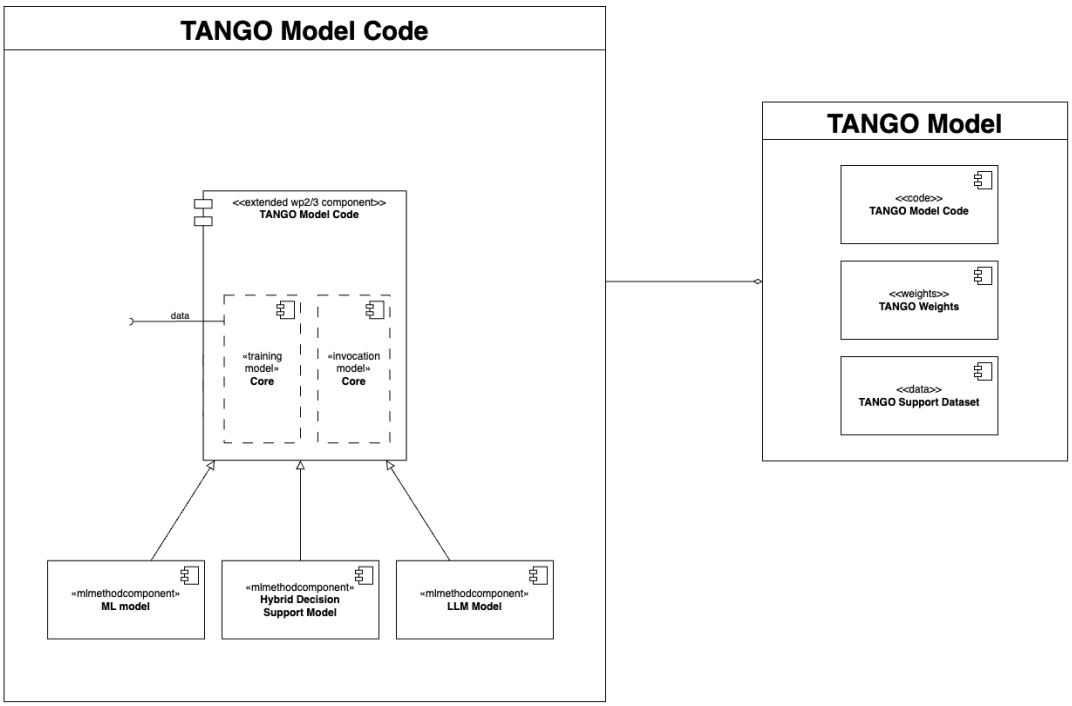

Figure 4: The TANGO Model

The TANGO Model Code represents the core predictive model component within the system. It encompasses source code originating machine learning models designed to perform specific tasks, such as forecast, classification, regression or conversation. In particular,

- It contains
  - A training core that has to be fed with training data during the corresponding phase of the MLOps pipeline, processing input data coming to train the model;
  - An invocation core, responsible for providing forecasts, classifications and conversations when published;
- It includes methods and algorithms for preprocessing and postprocessing;
- It implements the functional interfaces exposed by the TANGO Library.

Besides interacting with the TANGO Library, the TANGO Model Code is part of the TANGO Model that emerges from it as a bundle, ready for deployment, and it is used by the MLOps component, which manages the whole Model lifecycle.

4.1.3 TANGO Model

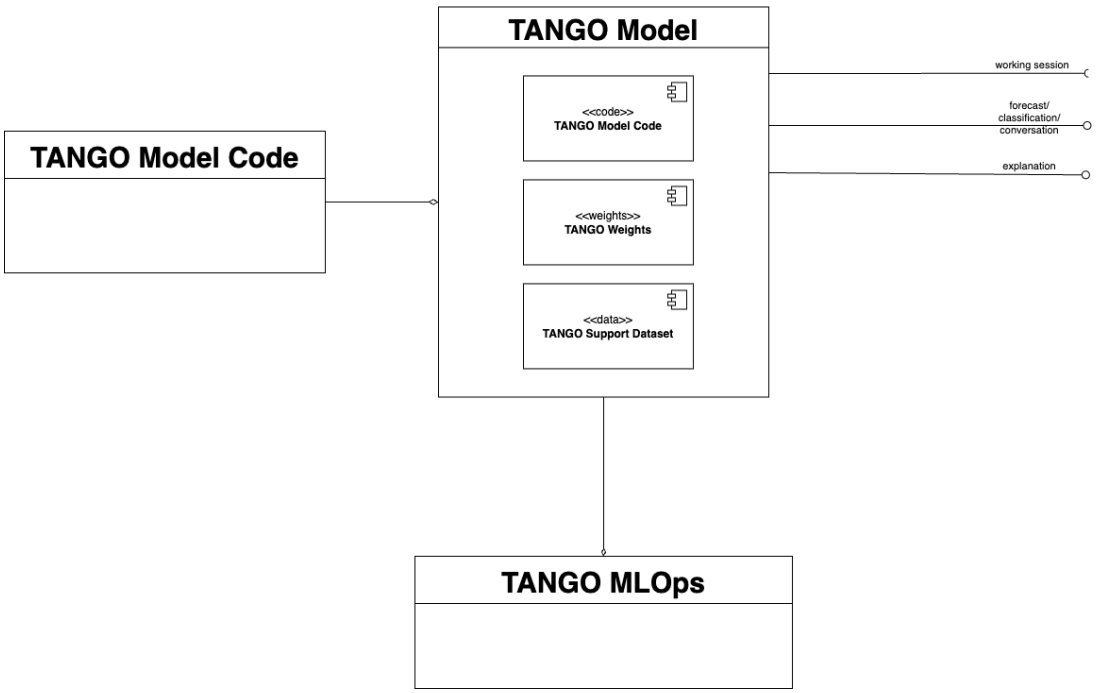

Figure 5: The TANGO Model & Model Code

This component is originated by a specific TANGO Model Code in conjunction with serialized model weights (from model training/tuning process) and any other supporting files or data needed for model deployment and execution (for example, data useful for explanation analysis). These elements compose a ML model artifact ready to be deployed on a MLOps environment and accessed by APIs to provide predictions.

Refer to Annex E for a more detailed description.

4.1.4 TANGO Storage

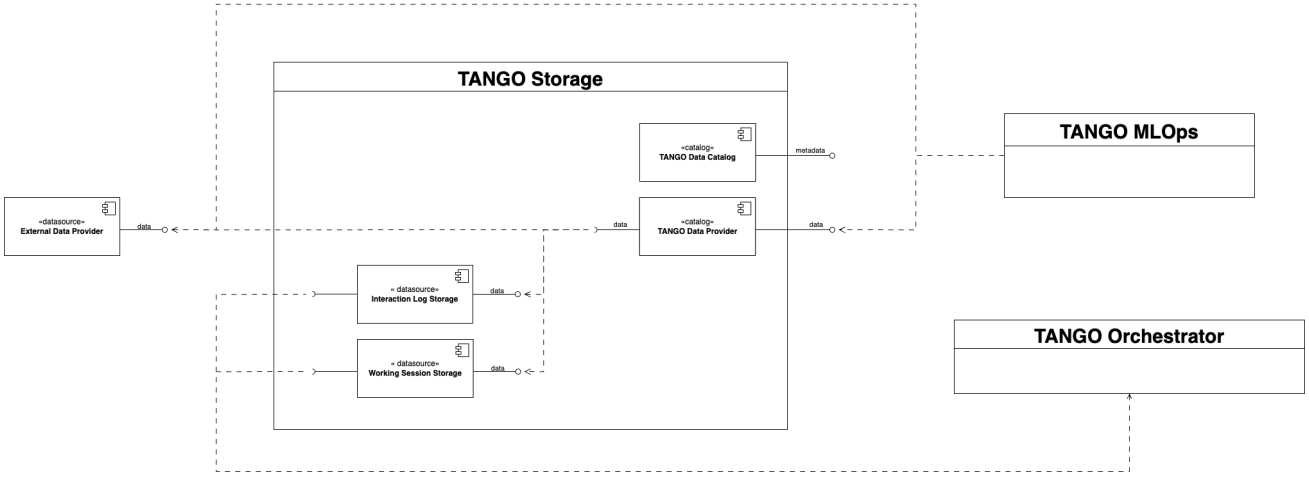

Figure 6: The TANGO Storage

The TANGO Storage serves as the distributed repository for storing data and other critical information for ensuring correct system execution:

- It provides secure and scalable storage for data used by the TANGO Model;
- It ensures data integrity and availability;
- It facilitates data retrieval and management for the other components.

Not all the data available in the project can be persisted on the storage, as some sensitive data must reside in different storages (due to GDPR compliance, for example). Regarding this, the role of the Data Catalog component will be crucial as it will contain the metadata of the data involved in the project, with appropriate indication of availability and (internal or external) storage location information.

The TANGO Storage mainly interacts with the TANGO MLOps, exposing data and - if needed - offering storage capabilities for models; it exposes features to the Orchestrator providing:

- Session history for analysis, monitoring, re-training and
- An endpoint to store interaction logs.

4.1.5 TANGO MLOps

The TANGO MLOps component manages the whole lifecycle of machine learning models in the TANGO platform. This includes aspects like training, evaluation, integration, deployment and serving. In particular:

- It automates deployment processes for the TANGO Models;
- It manages the CI/CD pipeline for continuous integration and deployment of models.

The flow typically starts with data being processed through the training pipeline; the obtained trained model is then registered in the Model Registry, evaluated via the Evaluation Pipeline and has to pass Integration Tests. Successful models are then deployed using the Deployer and served for inference by the Model Server.

In terms of interaction with other components, the MLOps needs the TANGO Models and the TANGO Storage as input to start the process, that completes with the deployment of the eligible Models on the Model Server

component, which in turn exposes them to the Orchestrator for further interaction with other components (in particular the TANGO API).

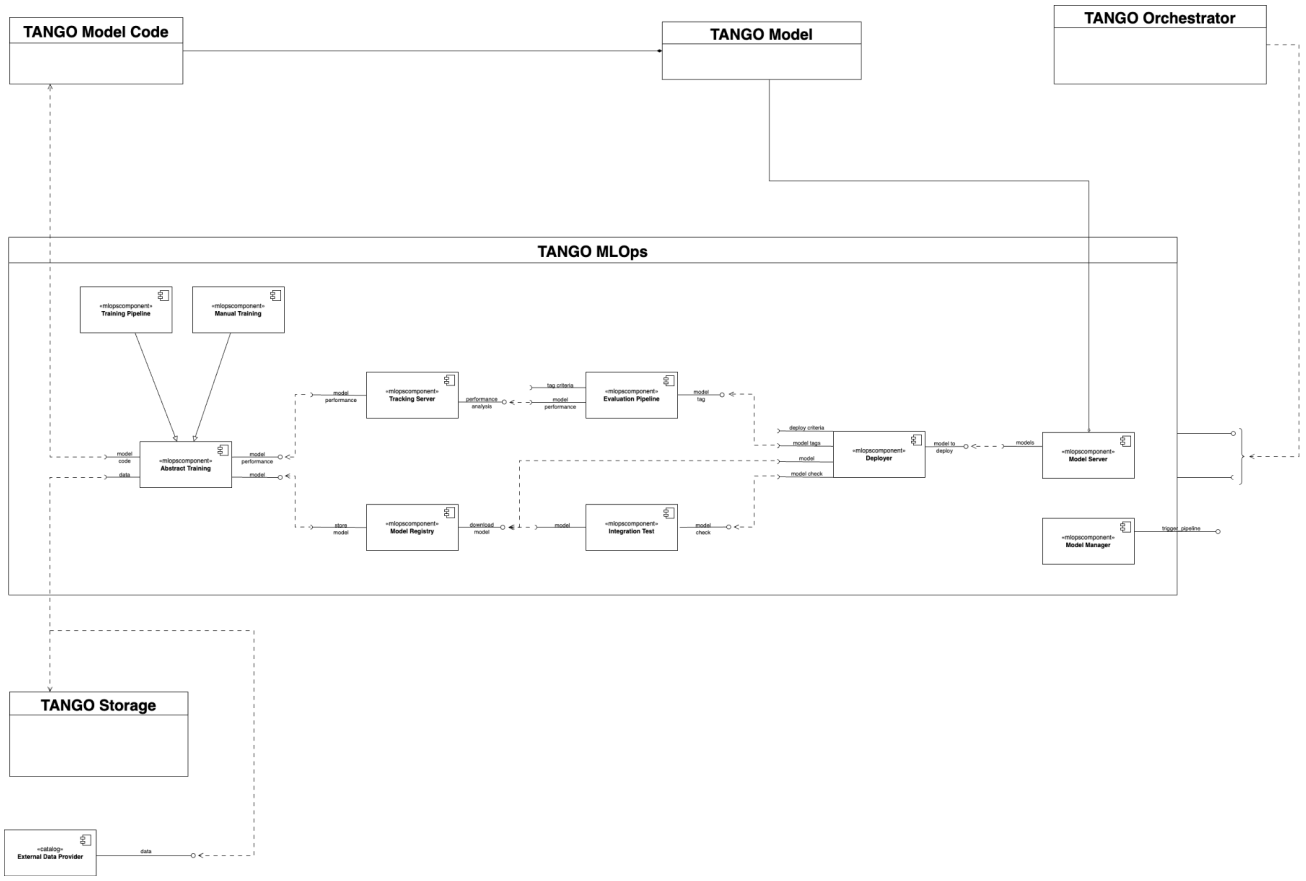

Figure 7: The TANGO MLOps

The design of the TANGO MLOps has been driven by the willingness to maintain the platform as open, modular and flexible as possible. It is worth noticing that, differently from most similar platforms, TANGO supports a plurality of MLOps instances, which can differ in terms of both their deployment as well as enabling technology stack. This is particularly relevant for two aspects.

The first one relates to those use cases where, for confidentiality or compliance reasons, the use case owner wants to keep full control on its data and models; in this case it may host its own MLOps instance on its own datacenter. This may be the case of T5.3 (Supporting surgical teams in making intraoperative decisions). The second one relates to those situations in which trained models on third-party platforms need to be integrated (think, e.g., of LLMs hosted on HuggingFace, a common case for some of the tasks carried out in WP3, e.g., in Task T3.1). In this case the hosted model can be abstracted as a MLOps instance, while still retaining the common TANGO workflow.

To let Model Server Gateway communicate with different instances of MLOps, the TANGO Model and the TANGO Model Proxy interfaces play a fundamental role, providing a specific implementation with a common interface used by the Gateway.

The TANGO MLOps layer consists of the following components:

- **Training**: this component is responsible for data, training and validating models, tuning hyperparameters, ensuring that the training process is consistent, repeatable, and scalable. The training can be done both automatically using a pipeline (that automates the process of training the ML model) or launched manually, allowing the execution of custom code to manually control the training process, useful for experimentation, debugging, or custom training scenarios that may not fit within the automated pipeline.
- **Tracking Server:** the racking server tracks and logs metrics and other relevant data after the training process; this component is crucial for model auditing and comparison.
- **Evaluation Pipeline**: it assesses and validates the performance of trained models, using performances stored in the previously described Tracking Server and default or custom criteria to determine the models to be deployed (default criteria are coded in the component itself, but custom criteria can be defined externally and passed to pipeline as parameters). Tagging a model means adding a label to the model: the label can be more complex than challenger/champion (hard decision), as some models will be evaluated with different metrics (soft decision) leading to more than one candidate going to production; even A/B testing could be implemented using custom logics in the evaluation pipeline.
- **Model Registry:** the Model Registry serves as a centralized repository for storing and versioning machine learning models, facilitating the management of model versions and metadata and allowing for consistent and organized deployment of models across different environments.
- **Integration Test:** this component tests the integration of trained models with other system components, ensuring that models work as expected when integrated into the larger system or application, preventing deployment issues and ensuring smooth operation. The model check consists in a binary check, used by the deployer to decide whether the model has to be deployed or not.
- **Deployer**: the deployer handles the automated deployment of models to production environments ensuring that deployment is reliable and reproducible. The deployer acts on the basis of the tags applied by the Evaluation Pipeline; default criteria for interpreting tags are defined in the component itself, but custom criteria can be defined externally and passed to pipeline as a parameter.
- **Model Server**: it hosts the deployed models and serves them for inference in production providing API endpoints, through which the Orchestrator can access the models for predictions or other machine learning tasks.
- **Model Manager:** the manager triggers MLOps pipelines on the basis of new model code or new model availability, performance monitoring, etc. The trigger can refer both to an automatic pipeline for adding a new model code and to the addition of an already trained model.

4.1.6 TANGO Orchestrator

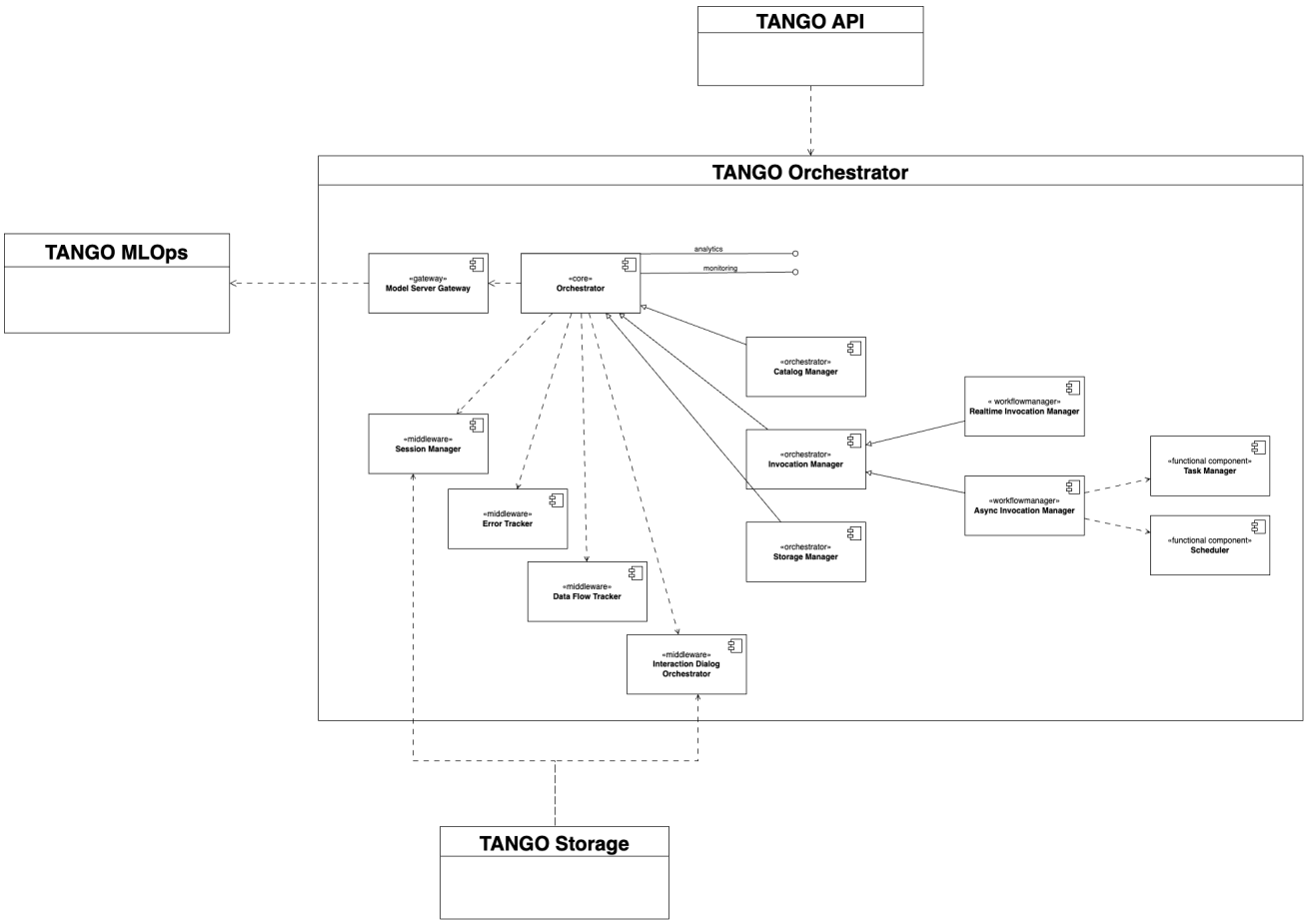

Figure 8: The TANGO Orchestrator

The TANGO Orchestrator coordinates the different components within the system. It manages workflows, ensuring that data and commands flow seamlessly between components. In particular:

- it acts as a middleware between models and services: gateway to models, load balancer, …;
- it distinguishes between realtime invocation (orchestrator acts as proxy to model server storing session) and async invocation (DAG execution);
- it orchestrates the interactions between the main TANGO other components;
- it manages task scheduling and execution order;
- it ensures efficient resource utilization by dynamically allocating tasks based on priority and load.

The **Model Gateway**, a module part of the orchestrator, allows requests to be dispatched to different MLOps instances. When forwarding an invocation, the TANGO Model Gateway:

- retrieves information and location of the model requested by querying the TANGO Model Gateway, which stores and lists information about MLOps and Models exposed;
- downloads the model from catalog manager (acting as a proxy of the model registry);
- uses the model to retrieve Model Proxy implementation and to invoke prediction methods, using TANGO Request and TANGO Response objects as standard for communication to Model Server via Model Gateway;

- once the Gateway receives the response, it uses the model to convert the custom response to TANGO Response.

The **Catalog Manager** stores and lists information about the MLOps instances and Models exposed in order to let Model Server Gateway retrieve everything needed to forward API requests to the correct Model Server.

The **Storage Manager** manages interactions with the TANGO Storage in terms of exposing catalog information and processing ETL pipelines.

The **Invocation Manager** discriminates between realtime invocation (orchestrator acts as proxy to model server storing session) and async invocation (DAG execution).

4.1.7 TANGO API

The TANGO API acts as the interface through which external systems interact with the TANGO Ecosystem. In particular, the TANGO API:

- exposes endpoints for model predictions, monitoring, analysis, user management and other operations;
- handles authentication and authorization for secure access;
- manages request-response cycles, ensuring that inputs are processed correctly by the TANGO Model.

The main modules of the TANGO API components are:

- **Private API**, which includes:
  - **Data Catalog API**: responsible for managing and providing access to a data catalog; the data catalog contains metadata about datasets, such as their location, structure and access permissions;
  - **Model API**: used for register a new model to be added to the Catalog Manager and made available to Model Gateway;
  - **Monitor API**: used for monitoring purposes, it reports metrics related to the system's performance, usage and health;
  - **Analytics API**: used for analytical operations, processing data, and generating insights;
- **Public API**, which includes:
  - **Model Catalog API**: allows users to access a catalog of models hiding information about MLOps component containing it and endpoints for direct invocation (these are used only internally by the orchestrator);
  - **Invocation API**: responsible for triggering specific actions or processes within the system, wraps requests to the followings:
    - **Real-Time API**: handles real-time data processing and interactions, being responsible for managing live data processing requests;
    - **Async API**: designed for asynchronous operations, where requests are processed in the background and responses are delivered later; it is useful for operations that possibly take a long time to complete;

- **Authentication/Authorization Server**: it handles the authentication of users and the authorization of access to various system resources/endpoints; it ensures that only authenticated and authorized users can interact with the APIs and other components.

The TANGO API Admin Dashboard is a user interface that allows system administrators to manage and monitor the TANGO APIs, providing functionalities such as API usage statistics, error reporting, access control and system health monitoring. Third Parties App and Researchers Monitor (dashed) are applications outside the scope of WP4.

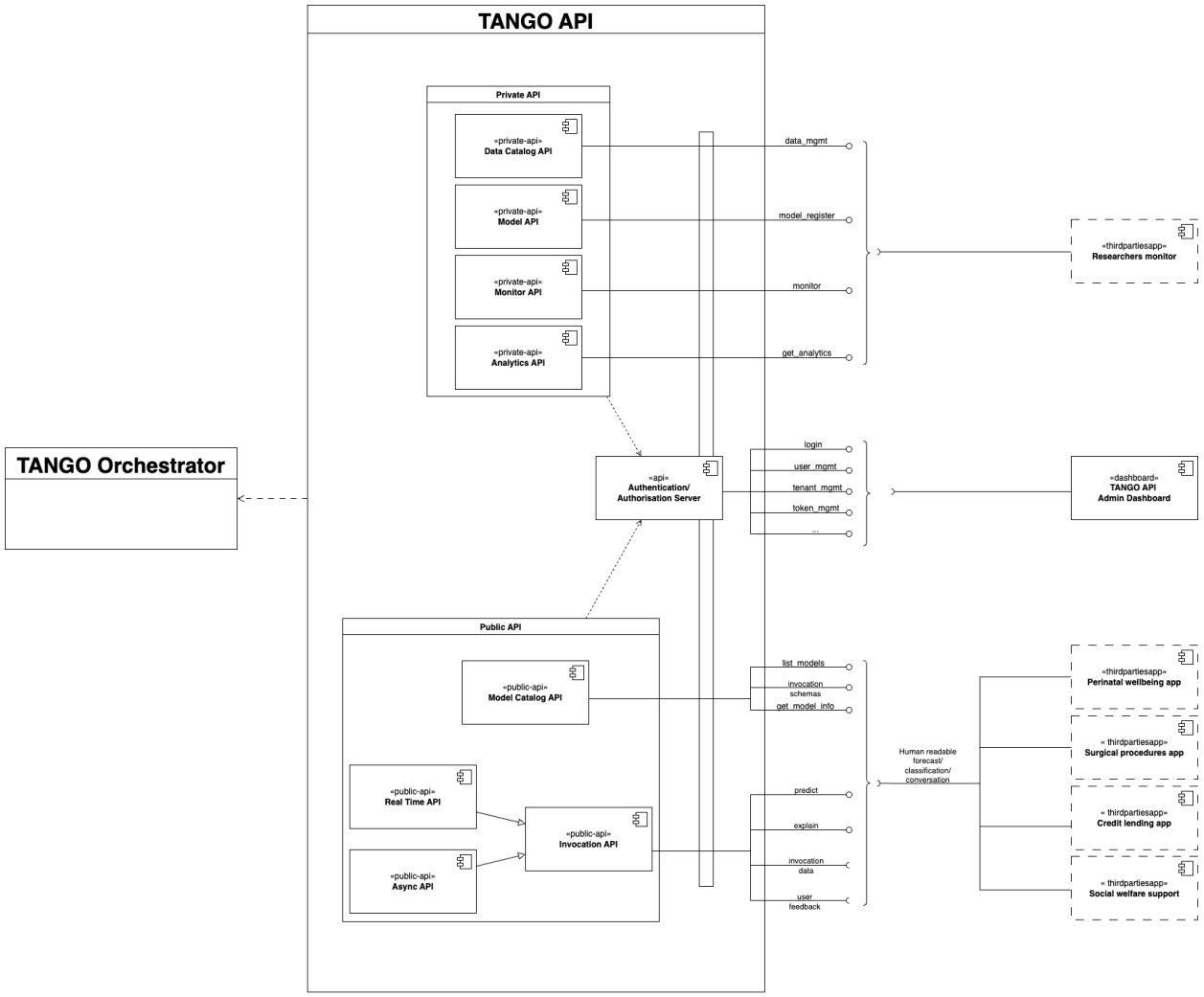

Figure 9: The TANGO API

4.2 Sequence diagrams

Sequence diagrams represent interactions among components. By providing a detailed view on some specific aspects of the interactions within the system, they are a useful tool to enable developers to better understand the role each component plays in the TANGO Ecosystem.

Sequence diagrams have been developed to detail interactions among the following entities within the TANGO ecosystem:

- **Model Manager** (MLOpsManager) is responsible for overseeing the complete lifecycle of machine learning models; this includes tasks such as initiating the training process.
- **Trainer** (MLOpsComponent) handles the actual training of machine learning models: it consumes data and applies algorithms to create a model that can be ready to be deployed for predictions and it is responsible for ensuring that the model training process is automated and can be re-executed whenever needed.
- **Tracking Server** (MLOpsComponent) monitors and tracks the performance of machine learning models; it stores metadata related to model runs and other relevant information to help evaluate and compare model performance.
- **Model Registry** (MLOpsComponent) is a component for managing and storing trained models; it holds various versions of models and ensures that they are available for deployment.
- **Evaluator** (MLOpsComponent) is responsible for assessing the performance of trained models against a given set of criteria; it determines whether the model is fit for deployment by comparing its performance with benchmarks or past models and assigning a performance score.
- **Integration Tester** (MLOpsComponent) ensures that the trained model integrates well with the overall system; it performs checks to see if the model interacts correctly with other components such as APIs, data pipelines and downstream systems.
- **Deployer** (MLOpsComponent) is responsible for deploying the trained machine learning models to the production environment; this includes managing the release process, setting up the deployment pipeline and ensuring that the models are correctly placed in the appropriate environment.
- **Model Server** (MLOpsComponent) is where the trained models are deployed and made accessible for inference; it provides an endpoint that can be accessed by external systems or APIs to request predictions or other outputs from the model.
- **App** (ThirdPartiesApp) represents any third-party application or client that interacts with the TANGO backend system via public APIs; the app initiates requests, such as synchronous or asynchronous model invocations, and receives results or status updates in return.
- **API** (PublicAPI) acts as the gateway through which external applications (such as third-party apps) can interact with the internal components of the system; it handles requests such as invoking models, listing available models or retrieving execution statuses.
- **Auth** (AuthServer) is responsible for authenticating requests coming into the system; it verifies the validity of the tokens or credentials provided by the client (e.g., third-party app) to ensure that only authorized users can access the system's resources and services.
- **Orchestrator** coordinates the entire flow of operations once a request has been authenticated; it manages the sequence of actions, such as retrieving models, forwarding requests to other components and tracking progress, ensuring that all parts work together harmoniously.
- **Session Manager** manages the session-related data and state for individual API requests; it is responsible for tracking the progress of long-running operations, maintaining session information across multiple requests and ensuring that results can be retrieved later if needed.

- **Catalog Manager** oversees the retrieval and management of available models; it interacts with the model registry and other components to provide information about which models are available for invocation and which models should be used for a given task.
- **Model Server Gateway** (ModelServerGateway) acts as an intermediary between the orchestrator and the model server; it handles forwarding requests to the correct model, retrieving predictions or other results and ensuring that the request is properly processed by the Model Server.
- **Model** (TANGO Model) is the specific machine learning model being invoked or used for predictions; it performs the core machine learning tasks, such as making predictions based on input data, and returns the results to the orchestrator for further processing.
- **Model Catalog** (ModelCatalogAPI) manages and provides access to the different machine learning models stored in the system; it allows third-party apps or internal components to list, retrieve or access model metadata and details.
- **MLOps Implementation** refers to the broader implementation of the MLOps framework that governs how models are managed, tracked and deployed; it involves multiple components working together to ensure the successful management of the model lifecycle.

The following diagram illustrates detailed components interaction during asynchronous invocation of the prediction method. More diagrams are available in the Annex A: Sequence diagrams section.

Full resolution images of the following detailed diagrams are available at this link.

4.2.1 Asynchronous invocation of the predict method

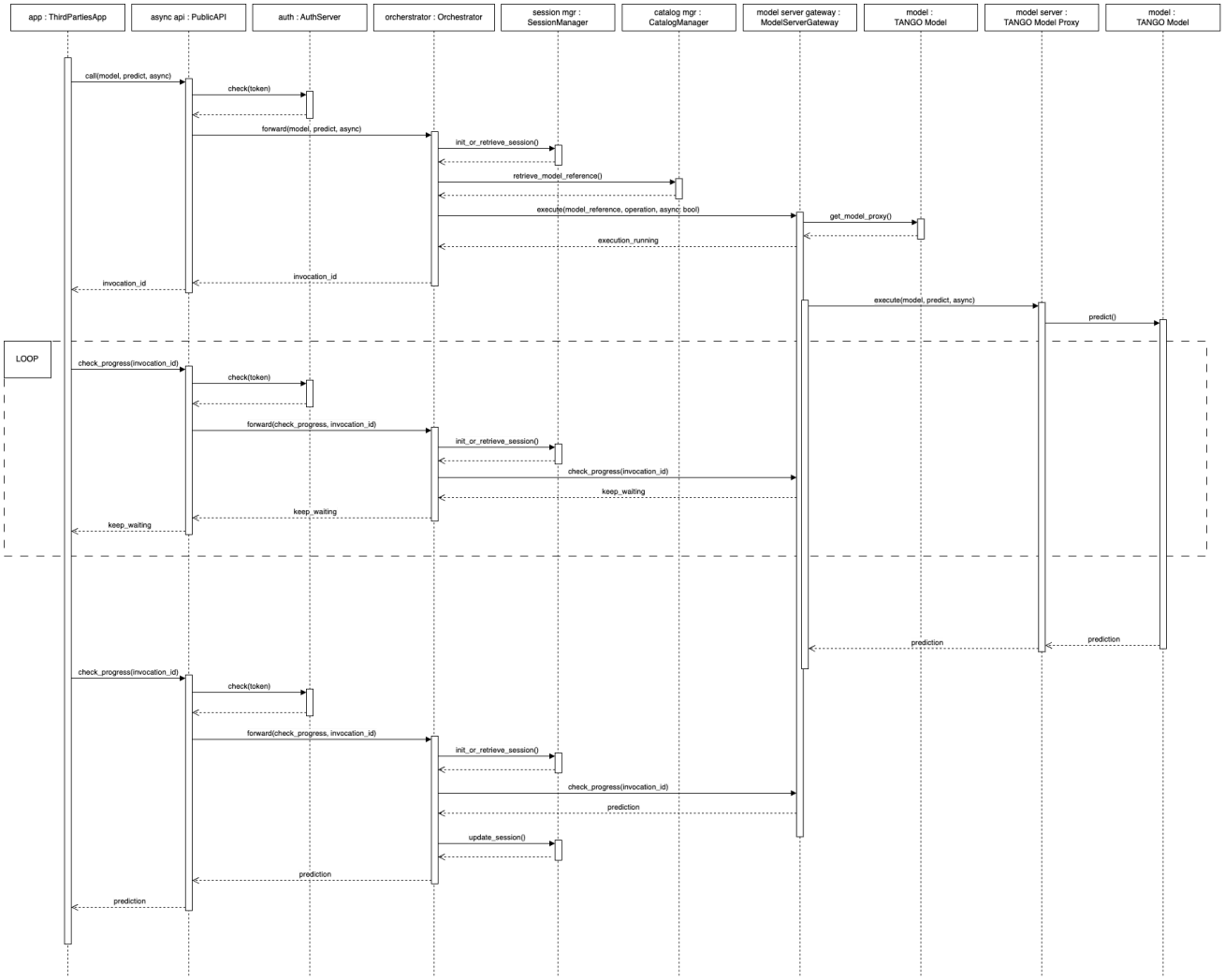

Figure 10: Sequence diagram: async invocation of the predict method

The diagram in Fig.10 depicts an asynchronous invocation process where the client initiates a long-running prediction operation. The sequence includes authentication, session management and the handling of an asynchronous process that can be monitored through periodic progress checks. The process continues until the final prediction result is returned to the client, making it suitable for operations that require significant processing time, such as complex model predictions.

Key points in this diagram are:

- asynchronous operation: the sequence diagram emphasizes the asynchronous nature of the operation, where the prediction request is made and the result is retrieved later;
- session management: the session manager ensures that the state is preserved across multiple interactions, critical in asynchronous processes;
- authentication: the authentication server verifies every interaction, ensuring secure access;

- orchestration: the orchestrator coordinates the complex sequence of operations, interacting with multiple components to manage the asynchronous prediction process.

4.3 Data Flow

A data flow diagram (DFD) illustrates how data moves between different components of the system, as shown, e.g., in the Figure below.

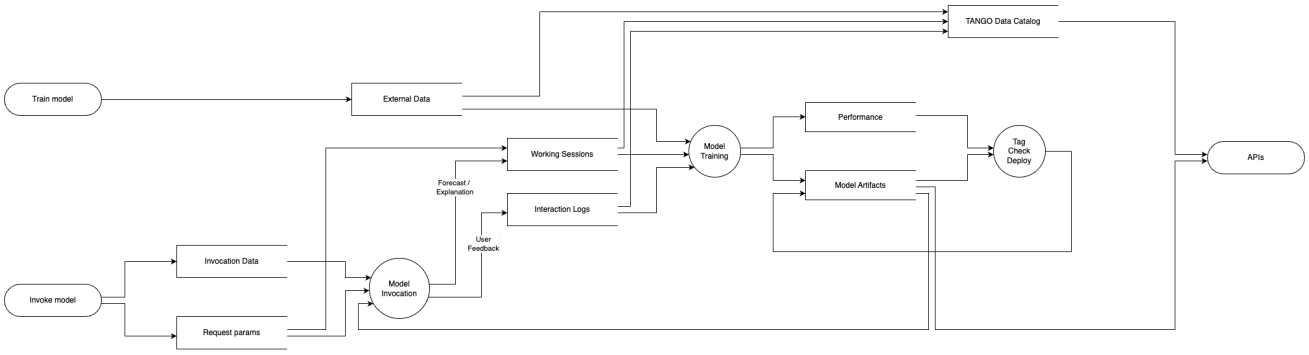

Figure 11: Data flow diagram

Full resolution images of detailed diagrams (the figure above and the followings) are available at this link.

In TANGO, the data flow is driven by the action of the following **processes**:

- **Model Training** involves the training of models using various inputs like external data, working sessions and interaction logs. The output of this process includes performance metrics and models to be stored in dedicated storage units;
- **Model Invocation** handles the invocation of the trained model, utilizing invocation data and request parameters;
- **Tag Check Deploy** gathers operations of tagging, checking and deploying models, involving evaluation and integration testing before the model is made available through APIs.

The main **External Entities** involved in the data flow are the followings:

- **Train Model** represents the initiation of the model training process, driven by a user's request or a system trigger;
- **Invoke Model** initiates the invocation of the trained model, triggered by a user's request coming from the APIs.

The **end node** of the diagram is the **APIs** component, responsible for both publishing the catalog of the data available in the system and the list of the available models.

4.4 Use Case

This Use Case Diagram provides an overview of the system's primary functionalities and how different actors interact with them. The diagram focuses on managing, training and invoking machine learning models, with additional capabilities for system monitoring, user management and accessing analytics.

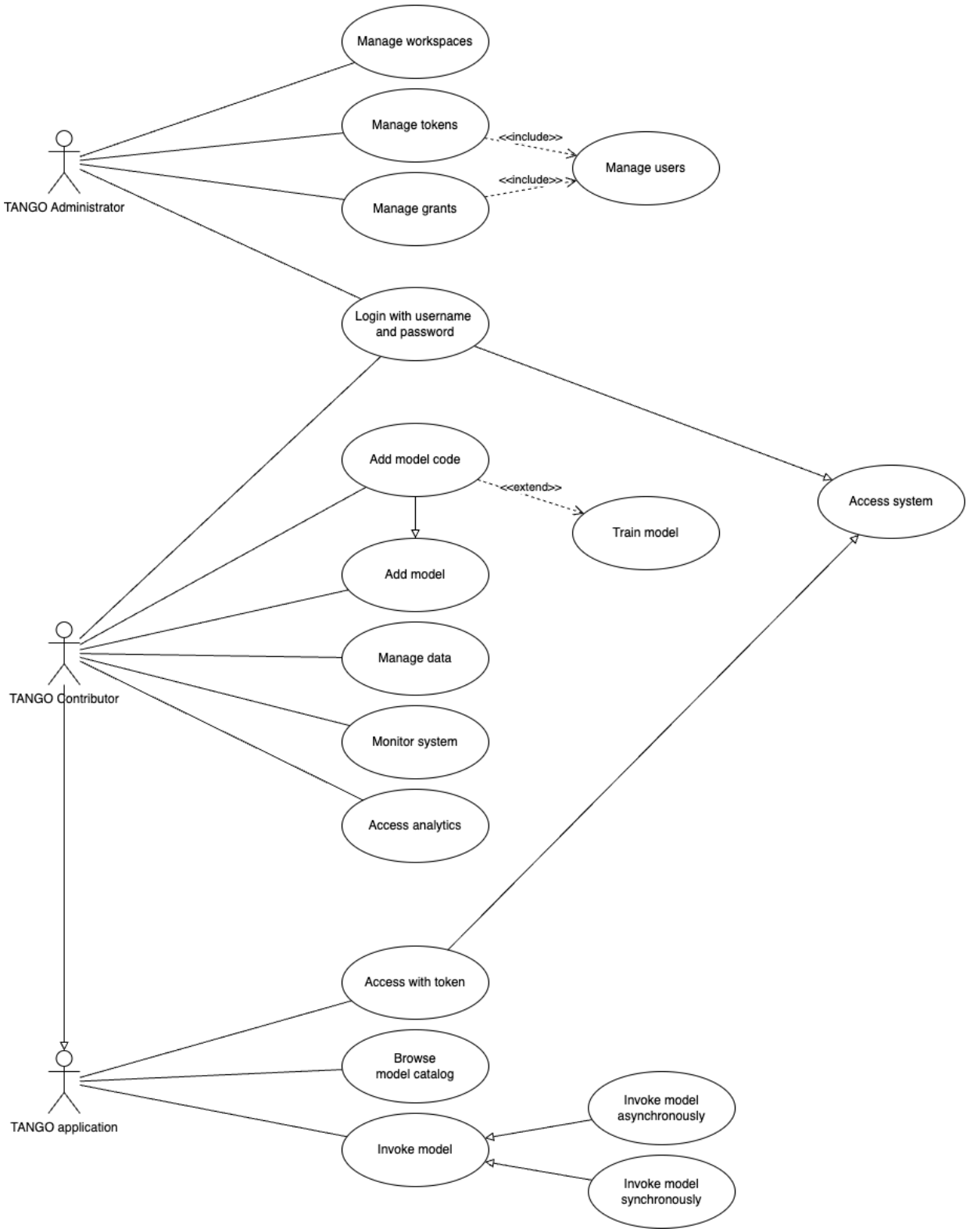

Figure 12: Use case diagram

Full resolution images of detailed diagrams (the figure above and the followings) are available at this link.

Actors of the TANGO Ecosystem are represented by

- **Administrator**, which interacts with use cases related to managing workspaces, tokens, grants, users and general system operations;
- **Contributor**, which interacts with use cases related to model management, including adding models, managing data, monitoring the system and accessing analytics;
- **Application**, which interacts with use cases related to invoking models, including browsing the model catalog and invoking models either asynchronously or synchronously.

This categorisation maps to the structure of the TANGO project. The Administrator role includes WP4 partners, which will manage the platform in operation. The Contributor role covers partners from WP2 and WP3, as well as WP4, which will provide models and datasets to the platform. The Application corresponds to the pilot use case applications that will be developed within WP5.

4.4.1 Administrator UCs

UC **Manage Workspaces** allows the actor to create, modify or delete workspaces within the system.

UC **Manage Tokens** allows the actor to manage authentication or access tokens. This use case includes the functionality of UC **Manage Users**.

UC **Manage Grants** allows the actor to manage permissions or grants within the system. This also includes the functionality of UC **Manage Users**.

UC **Manage Users** is a shared functionality between UC **Manage Tokens** and UC **Manage Grants** that allows managing user accounts and access levels.

UC **Login** represents a generic use case that allows actors to access the TANGO system; it is specialized by UC **Login with username and password** (for actors of type Administrator and Contributor) and by UC **Access with token** (for Application actor).

4.4.2 Contributor UCs

UC **Login** - see above section.

UC **Add Model Code** allows the actor to add new model code to the system. This use case extends to the UC **Train Model** that covers the functionality of training the newly added model.

UC **Add Model** allows the actor to add models.

UC **Manage Data** provides the actor with the ability to manage datasets or other data resources within the system.

UC **Monitor System** allows the actor to monitor system operations, likely involving real-time metrics, logs or system health checks.

UC **Access Analytic** provides access to analytical reports or dashboards within the system.

4.4.3 Application UCs

UC **Login** - see above section.

UC **Browse Model Catalog** allows the actor to browse through a catalog of available models.

UC **Invoke Model** is the main use case for executing or invoking a model, implemented by the UC **Invoke Model Asynchronously** that allows the actor to invoke a model in an asynchronous manner and by the UC **Invoke Model Synchronously** that allows the actor to invoke a model and immediately receive the result.

5 API Specifications

5.1 Component-Level APIs

The concept of a Component-Level API refers to an application programming interface (API) that provides access to and control over individual components of a system, typically within a larger architecture or ecosystem. These components can be microservices, modules or individual parts of a software platform, each handling a specific responsibility.

Within the scope of the TANGO platform, component-level APIs are typically not exposed to external users/applications. The only exception is the MLOps component that, as already mentioned in the previous chapters, is treated as a black-box, as different implementations (one for each specific need based on technologies used, data segregation, performance requirements and so on, even based on already existing solutions, e.g. Databricks or HuggingFace) can be integrated within the TANGO Ecosystem.

TANGO interfaces and classes are available in the dedicated public GitLab repository.

TANGO interfaces and their use for adding a new MLOps component to TANGO are detailed in the dedicated Annex B: Component-Level APIs.

5.2 Platform-Level APIs

Platform-level APIs offer access to the TANGO Ecosystem core functionality, exposing to third parties specific endpoints for interacting with the system.

Platform-level APIs are detailed in the dedicated Annex C: Platform Level APIs.

The **TANGO Public API** provides various REST endpoints allowing external applications and users to interact with the platform's models and is structured into distinct groups, each focusing on a specific functionality like managing models, sessions, and invocations. The OpenAPI specifications are available at the following url.

The **TANGO Private API** is dedicated to internal users of the platform, including research groups responsible for managing data and models, as well as WP4 developers. It serves as the primary interface for internal operations related to data management, model monitoring, and analytics. The full documentation in OpenAPI format is available at the following url.

The TANGO platform implements a robust authentication and access management system through the **TANGO Auth API**, designed to manage user sessions, tokens, grants, and workspace access. This system ensures secure, role-based access to the platform's resources, providing flexibility for users who belong to

multiple workspaces. While the API provides the technical means for access management, the platform's dashboard offers a more user-friendly interface for managing workspaces, grants, and tokens. The OpenAPI descriptor is available at the following url.

6 Deployment Considerations

6.1 Deployment Architecture

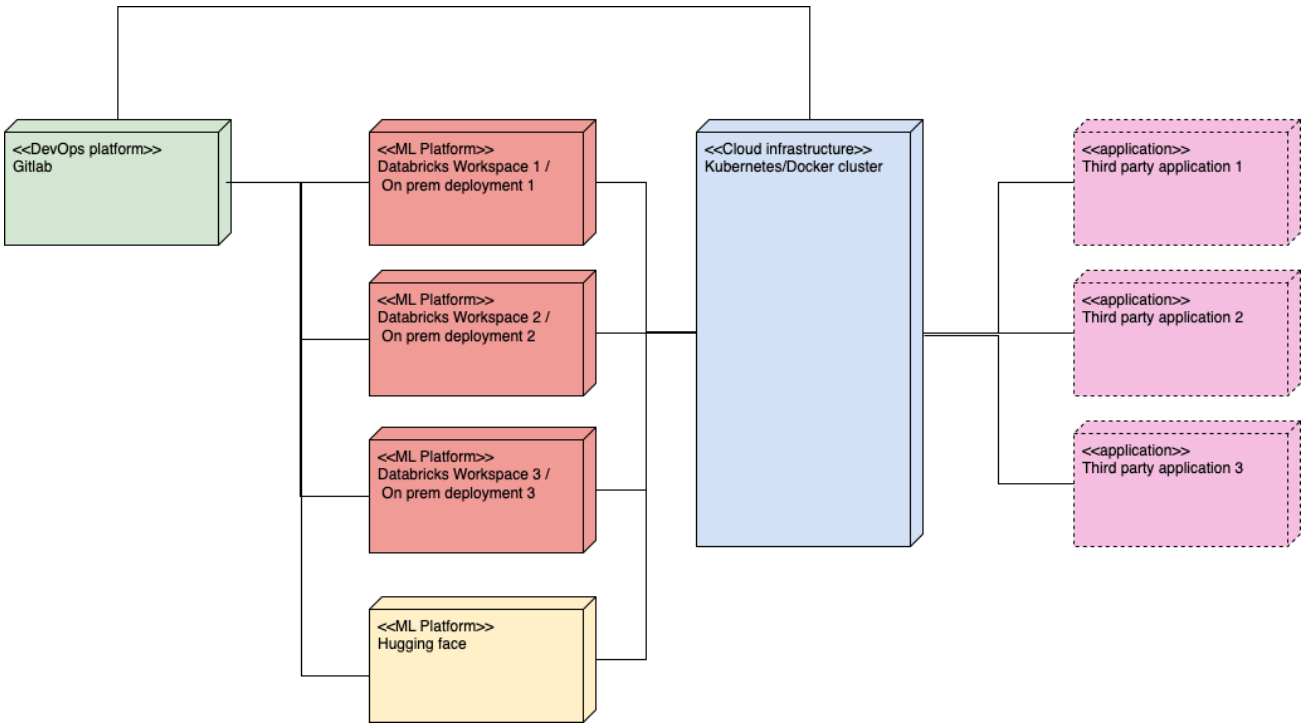

Figure 13: The TANGO deployment architecture

The term "deployment architecture" refers to the framework used to implement and manage software applications within a physical environment. It encompasses various components and considerations that ensure efficient, reliable and scalable deployment of systems. The deployment architecture reflects the system architecture as detailed in the previous chapters, detailing platform and technologies to be fully functional in serving TANGO features in different contexts/environments (development, staging, production).

This diagram provides a clear picture of the layered structure of TANGO's deployment, showing how each component—DevOps, ML platforms, Cloud Infrastructure and Third-party applications—interacts with one another.

The **DevOps Platform** is a suite of tools, practices and processes that facilitates the integration of software development (Dev) and IT operations (Ops) to enable faster and more reliable software delivery. Implemented in TANGO by GitLab, used for implementing version control, CI/CD pipelines and automation workflows. GitLab interfaces with multiple ML platforms and triggers pipeline automation for model training, deployment and other operations.

An **MLOps platform** is a specialized framework designed to streamline the deployment, monitoring and management of machine learning (ML) models in production environments, combining practices from DevOps (Development and Operations) with the requirements of machine learning to ensure that models are efficiently developed, tested, deployed and maintained. In the TANGO Ecosystem, self-hosted and on-premise solutions can co-exist, their choice depending on factors such as specific technologies, data segregation or performance requirements. The default MLOps implementation in the TANGO project will be based upon the Databricks stack: MLFlow has been chosen for its flexibility and open-source nature, Unity Catalog for data management and Jupyter for manual model training. The platform supports other open-source or commercial solutions to be seamlessly integrated (e.g., HuggingFace for deploying large language models).

In the TANGO platform, all components of the **TANGO Orchestrator** and API will be encapsulated within Docker containers and deployed on either a Docker or Kubernetes cluster, to leverage the power of containerization and to ensure a scalable, reliable and consistent deployment environment for the platform. This acts as the primary cloud infrastructure, managing containerized deployments for the ML platforms and orchestrating the workloads.

Further details on deployment are available in the dedicated Annex D: Deployment detail.

6.2 Scalability and Performance

Scalability and performance are critical aspects of the TANGO platform, which should be able to handle varying workloads while maintaining high levels of responsiveness. The platform's architecture has been designed with flexibility and scalability in mind, leveraging containerization and cloud technologies to offer high performance.

6.2.1 Scalability of the Orchestrator and API Components

All components of the TANGO platform's orchestrator and API are encapsulated in Docker containers, providing a high degree of flexibility in terms of deployment and scaling. These components can scale horizontally, by creating additional instances as needed. This horizontal scaling is achieved by adding more servers to the Docker or Kubernetes cluster, effectively distributing the workload across multiple computing instances. This approach ensures that both the orchestrator and API can handle increased demand without any degradation in performance.

By leveraging Kubernetes' advanced orchestration features, such as automatic scaling and load balancing, the platform can dynamically adjust the number of running instances in response to real-time demand. This capability ensures that the orchestrator and API remain responsive, efficient, and reliable, even under heavy workloads.

6.2.2 Scalability of the Model Server

The model server, a crucial element of the TANGO platform's machine learning pipeline, is designed to scale both horizontally and vertically, depending on the deployment strategy chosen.

- **Vertical Scaling on Databricks**: When using Databricks as the model server, vertical scaling is achieved by increasing the size and resources of the Databricks cluster. This allows the platform to handle more complex models and larger datasets by allocating additional computing power and memory to the model server
- **Horizontal Scaling with Docker/Kubernetes**: If the self-hosted option is selected, the MLFlow model server can be scaled horizontally by deploying multiple instances within a Docker or Kubernetes cluster. Similar to the orchestrator and API components, this horizontal scaling distributes the load across multiple instances, ensuring that the model server can efficiently handle increased traffic and model inference requests.

6.2.3 Scalability of the DevOps Platform

As previously mentioned, GitLab serves as the DevOps platform for the TANGO project, providing essential tools for source code management, continuous integration, and continuous deployment (CI/CD). While scalability is not a critical concern for the DevOps platform, it is important to ensure that the system remains responsive, even under heavy workloads.

In scenarios where multiple training, deployment, and testing pipelines are executed simultaneously, leading to a potentially long queue of tasks, the platform offers a solution to maintain efficiency. By deploying additional GitLab runners, the task queue can be served faster, reducing waiting times and accelerating the overall CI/CD process. This ability to scale the execution capacity of the pipeline runners ensures that the DevOps platform can adapt to increased demand when necessary, maintaining smooth and efficient operations across the TANGO platform.

6.3 Security and Privacy

Security and privacy are of utmost importance in the TANGO platform, particularly when dealing with sensitive user data and research outputs. To ensure robust protection of user information, the platform is designed with several key security measures that safeguard data access, storage, and sharing.

6.3.1 Isolated MLOps Platform Instances

To enhance both security and privacy, the TANGO platform allows the integration of separate MLOps instances. By creating isolated instances, each research group or model developer has exclusive access to its own data catalog, ensuring that sensitive information is securely contained within its dedicated environment. This isolation minimizes the risk of unauthorized access, as only the designated users of each research group can access and manage their data.

6.3.2 Shared Model Registry and Tracking Server

While data catalogs are kept isolated, the model registry and tracking server may be shared across multiple MLOps instances to facilitate collaboration and resource efficiency. In cases where a shared instance of the model registry and tracking server is created, a robust Attribute-Based Access Control (ABAC) system will be implemented. This ABAC system ensures that access to models and tracking data is strictly controlled by assigning specific roles (or other relevant user attributes) with defined permissions. Only authorized users will be able to view or modify specific entries based on their assigned roles. This approach ensures that

privacy is maintained even in a shared infrastructure, with clear boundaries between different research groups' data.

6.3.3 Access Control for Shared Storage Instances

In scenarios where a storage instance is shared between multiple MLOps platform instances, a rigorous Attribute-Based Access Control (ABAC) system will be implemented. This ABAC system ensures that only authorized users, based on their assigned roles and other relevant attributes, can access specific datasets, maintaining the security and privacy of data across shared environments. This layered security approach guarantees that even within shared storage environments, each research group's data remains isolated and protected.

6.3.4 Controlled Access to User Sessions

User sessions in the TANGO platform include interactions and requests made by users, often containing sensitive information such as user details, prompts, and queries submitted to the platform. Given the confidential nature of these sessions, it is crucial to ensure that access to this information is tightly controlled.

To maintain privacy and security, user sessions are accessible only to users who have access to the workspace where the session resides. Each workspace acts as a boundary for session access. However, to accommodate specific needs, researchers may need to have access to all sessions connected to a particular model they are managing, even across multiple workspaces, as long as privacy concerns are correctly and consistently addressed.

An ABAC (Attribute-Based Access Control) system will be implemented to enforce these access restrictions. This ensures that only authorized personnel can view or interact with specific sessions. By limiting access to only those who need it, the risk of unauthorized access is minimized, protecting sensitive user information and ensuring that all interactions within the platform are handled securely and privately.

Lastly, for third-party applications integrated into the platform, the privacy of users within a workspace must be handled by the application developers. Developers are responsible for ensuring that their applications comply with privacy standards and adequately protect user data within the boundaries of the workspaces they operate in. This is crucial to maintaining trust and safeguarding sensitive information across the entire TANGO platform ecosystem.

6.3.5 Access to platform models

Most of the TANGO models will be available for use across all workspaces, allowing a wide range of users and application developers to benefit from the platform's capabilities. This open accessibility fosters collaboration and ensures that users can leverage the platform's full potential. However, certain models may be restricted to a specific set of workspaces for privacy or confidentiality reasons. These restrictions are necessary when models handle sensitive or proprietary data that should not be shared broadly. In such cases, access to these models will be confined to designated workspaces to safeguard the privacy and confidentiality of the data involved, ensuring that only authorized users can interact with these models. This approach strikes a balance between promoting widespread access and maintaining strict data privacy protections where needed.

6.3.6 Encrypted Communications

In the TANGO platform, all communications between the various components, including the orchestrator, APIs, MLOps platform, and user interfaces, will be secured through encrypted connections using the HTTPS protocol. HTTPS (HyperText Transfer Protocol Secure) ensures that data transmitted between clients and servers is encrypted, protecting it from interception and unauthorized access. This encryption is essential for maintaining the confidentiality and integrity of sensitive information, including user data, model parameters, and API requests. By enforcing HTTPS across the platform, TANGO ensures that all data exchanges are secure, reducing the risk of data breaches and enhancing overall platform security.

7 Conclusion

7.1 Summary

This document has successfully set forth the architectural framework for the TANGO backend system, a pivotal result in advancing the capabilities of Hybrid Decision Support Systems as envisioned by the project and in supporting the project along its impact pathway. The document starts with a definition of both functional and non-functional requirements, ensuring the architecture not only supports the immediate needs of the project but is also robust enough to handle future challenges and expansions.

In the document at hand, we have articulated a detailed account of the core and ancillary components that constitute the system's infrastructure, highlighting their individual roles and the synergy between them. This insight into the component architecture underscores the system's readiness for integration and scalability within diverse operational contexts.

Furthermore, the detailed exposition on API specifications, crafted in accordance with the OpenAPI standard, establishes a clear pathway for system interaction and integration. These specifications are designed to facilitate developer engagement and enhance system accessibility for various stakeholders involved in the project.

In discussing deployment, we have underscored strategies that cater to a variety of environments, thereby enhancing the system's adaptability and operational longevity. These considerations are crucial for maintaining performance and reliability as the project scales.

In sum, the document has meticulously laid out the necessary architecture for TANGO, providing a robust foundation for future developments. This ensures the project is well-prepared to meet its objectives, driving forward the integration of AI in decision support systems in a manner that is both innovative and user-centric.

7.2 Future Work

As the TANGO project progresses towards its goal of refining and deploying Hybrid Decision Support Systems, there are several areas where future work and enhancements could further optimize the architecture:

1. *Enhanced AI Model Integration:* Continuous research into AI advancements can be incorporated to improve decision-making accuracy and speed. Future iterations could focus on deeper integration of cutting-edge AI models that offer greater adaptiveness to varying data inputs and user interactions.
2. *Scalability Enhancements:* While the current architecture supports scalability, ongoing assessments and updates will be crucial as the system scales. Future work could explore more dynamic scaling solutions that adjust more fluidly to the changing load and data volume.
3. *Security Upgrades***:** As threats evolve, so too should our security measures. Future enhancements could include the implementation of advanced techniques to ensure data integrity and privacy.
4. *Real-time Data Processing*: To enhance the system's responsiveness and efficiency in decision-making, future developments could focus on advancing real-time data processing capabilities. This would enable faster turnaround times for decision support and increase the system's utility in time-sensitive scenarios.
5. *Cross-domain Adaptability***:** Given the diverse applications envisioned for the TANGO system, exploring adaptability across different domains could be beneficial. We do expect that, as work within WP5 progresses, we will get insightful feedback on how the envisioned TANGO architecture fits the specific case study needs.
6. *Environmental Impact Considerations*: Future updates could also look at optimizing the energy efficiency of the system to minimize its environmental impact, particularly in terms of data center operations and overall carbon footprint.

We do expect in particular that, as activities in WP5 on the case studies and pilots progress, we will be able to refine the architecture to ensure full compliance with the needs and requirements coming from experimental HDSS deployments. These suggestions aim to not only address potential limitations of the current architecture but also to pave the way for innovations that keep the TANGO project at the forefront of science and technology in AI.

7.3 Community building

The TANGO consortium is committed to ensuring the accessibility and adoption of the TANGO Ecosystem by a wider community of stakeholders, including researchers, developers, and industry practitioners. This commitment is rooted in a philosophy of transparency and collaboration, as reflected in our "building in public" approach.

7.3.1 Accessibility and Availability

The TANGO Ecosystem and its components are being developed openly, with all source code, documentation, and related materials made publicly available through accessible platforms. Specifically:

- Source Code and Documentation: Hosted on the official project GitLab, ensuring that any interested party can access, review, and contribute to the project. Links and updates are regularly shared through the project's official communication channels.

- Early Access: Interested users can engage with and test preliminary versions of the TANGO Ecosystem during its development phases, enabling feedback collection and iterative improvements.
- Open Licensing: The ecosystem components are released under open-source licenses (e.g., Apache 2.0) to encourage adoption and integration into other projects.

7.3.2 Dissemination and Exploitation Plans

At present, the TANGO consortium employs two main mechanisms to promote awareness and engagement. The first is the project website, which acts as a hub for project updates, deliverables, and public-facing content. The second is social media (in particular Linkedin), which is used to reach a professional audience and foster discussions around the project's advancements and potential applications.

To maximize the impact and adoption of the TANGO Ecosystem, the consortium will outline a detailed strategy in the forthcoming D6.4 (First exploitation strategy), which will include:

- Expanded Online Presence: Leveraging additional social media platforms, blogs, and forums to engage a broader audience.
- Workshops and Webinars: Hosting events to demonstrate the ecosystem's capabilities and gather feedback from diverse stakeholders.
- Presentations and Publications: Presenting TANGO at relevant conferences and events to showcase its technical achievements and potential use cases.
- Pilot Case Studies: Publishing detailed results and lessons learned from pilot deployments in WP5 to validate the system and inspire further applications.

As the long-term goal is to ensure sustainability of the TANGO software ecosystem beyond the project end, we are planning to adopt best practices from successful open-source software projects :

- Clear and Comprehensive Documentation: Ensuring that developers and users can quickly understand how to use and contribute to the project.
- Engaging Tutorials and Examples: Providing practical, easy-to-follow guides for onboarding new users and contributors.
- Dedicated Community Support Channels: Establishing forums, mailing lists, and Discord channels to facilitate discussions and create an open environment.
- Regular Updates and Milestones: Maintaining momentum by sharing progress updates, celebrating milestones, and demonstrating the project's ongoing value.

By fostering an inclusive and collaborative environment, the TANGO consortium aims to ensure that the ecosystem not only meets its immediate project objectives but also establishes itself as a widely adopted and continuously evolving tool for hybrid decision-making support.

Annexes

Annex A: Sequence diagrams

Full resolution images of the following detailed diagrams are available at this link.

A.1 List models

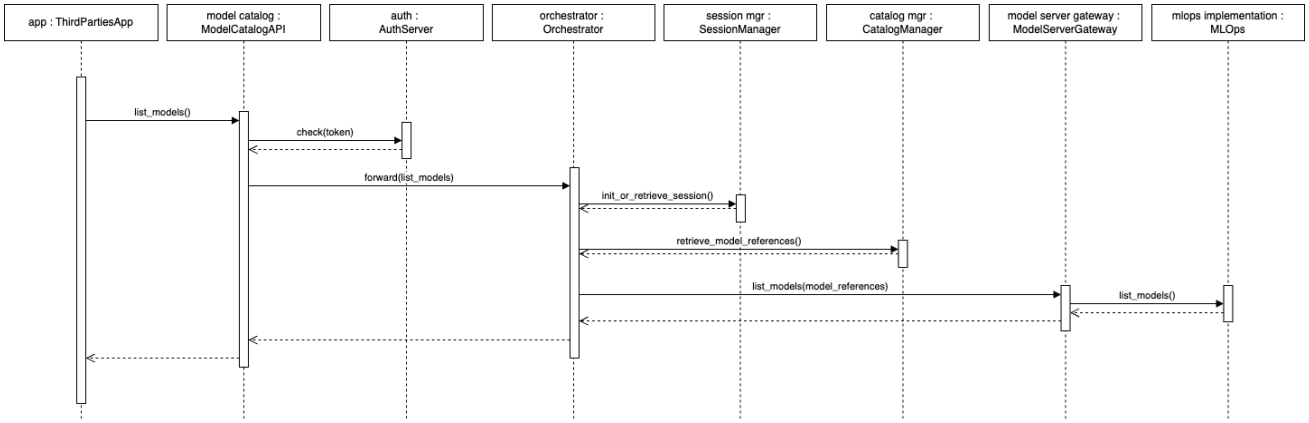

Figure 14: Sequence diagram: request to list models

The diagram illustrates the interactions occurring when a request from a third-party application to list models triggers a series of calls across different components, including API handling, authentication, orchestration and model operations. Each component plays a specific role in ensuring the request is processed correctly, with authentication checks and session management being crucial steps in the sequence.

A.2 Synchronous invocation of the predict method

The synchronous invocation of a model method is similar to the previously described asynchronous one, except for the fact that the calling participants wait for the prediction coming from the model until this returns back the prediction.

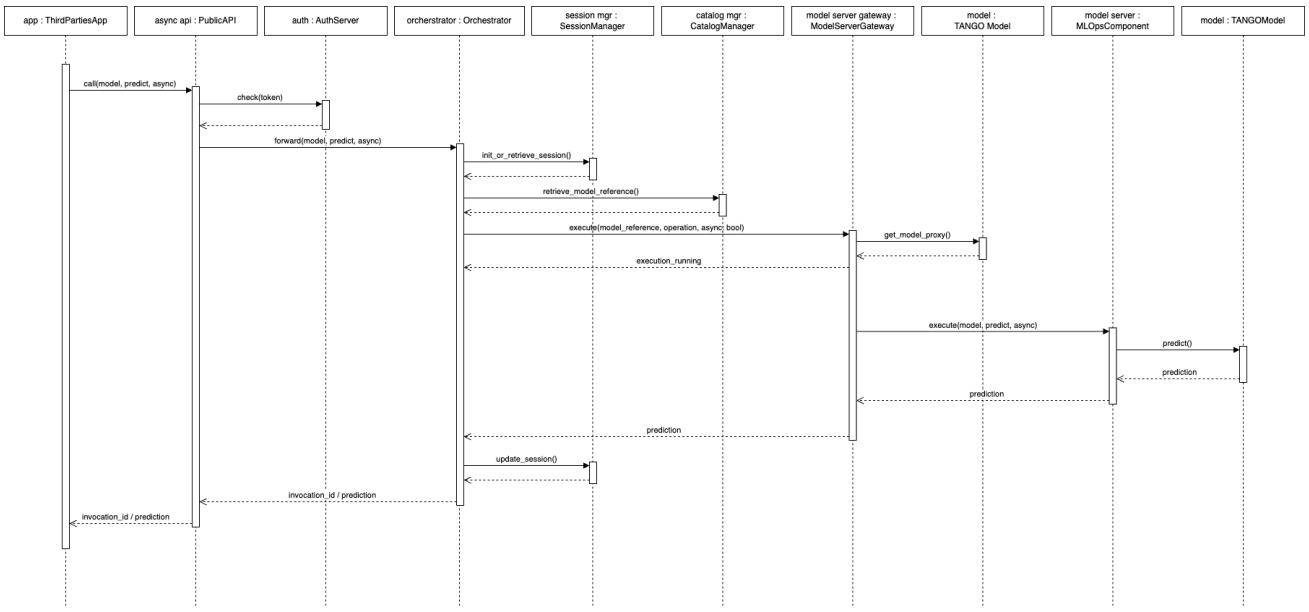

Figure 15: Sequence diagram: sync invocation of the predict method

A.3 Model deploy

This workflow illustrates a complete lifecycle from model training to deployment in a structured MLOps environment.

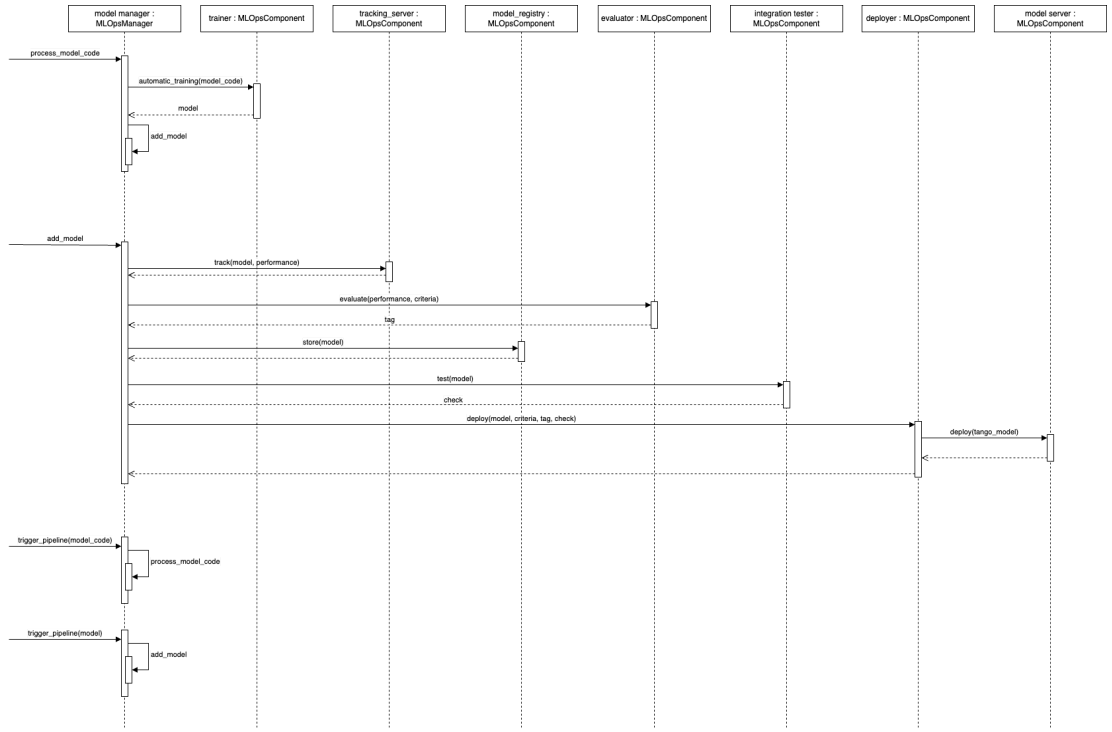

Figure 16: Sequence diagram: model deploy

Annex B: Component-Level APIs

B.1 TANGO Model Proxy

```
class TangoModelProxy:

   def invoke(self, model_identifier: str,

              invocation_type: InvocationType, payload: TangoRequest) -> TangoResponse:

       raise NotImplementedError
```

The TANGO Model Proxy interface defines a single method, which implementation must contain the logic the execute the call to the custom model server, given

- the identifier of the target model on the model registry,
- the invocation type (fit, predict, explain, ...) and
- the request payload as TantoRequest.

It returns a TangoResponse to the caller.

B.2 TANGO Model

```
class TangoModel:

   def get_model_proxy(self) -> TangoModelProxy:

       raise NotImplementedError

   def map_request(self, tango_request: TangoRequest) -> object:

       raise NotImplementedError

   def map_response(self, response: object) -> TangoResponse:

       raise NotImplementedError

   def fit(self, training_data: TangoTrainingData):

       raise NotImplementedError

   def predict(self, tango_request: TangoRequest) -> TangoResponse:

       raise NotImplementedError

   def explain(self, tango_request: TangoRequest) -> TangoResponse:

       raise NotImplementedError
```

The TANGO Model interface must to be implemented by every Model deployed in the TANGO Ecosystem, providing implementation for the following methods:

- `get_model_proxy`: returns the TANGO Model Proxy interface implementation pointing to the Model Server where the current model is deployed;

- `map_request`: maps a TangoRequest to the input data structure to fit the Model Server invocation APIs (e.g. array or DataFrame in case of MLFlow);
- `map_response`: maps the response coming from the Model Server to the standardized TantoResponse class;
- `fit`: proxy to the fit method on the Model;
- `predict`: proxy to the predict method on the Model;
- `explain`: proxy to the explain method on the Model.

Typically, predict and explain methods on model will be invoked passing through the corresponding Model Proxy, in order to let the specific Model Server manage execution details (e.g., usage of GPU); these methods remain anyway available on the model itself.

B.3 Adding a new MLOps component to TANGO

Adding a new MLOps component to TANGO, given the interfaces detailed above, is straightforward: by calling the endpoint register_model on the Private API with the TANGO Model implementation and its metadata, the Orchestrator adds the model to the Catalog Manager. Once registered, the model is made available to the Model Gateway, which retrieves the model from the Catalog Manager and uses the Proxy Model stored in it to establish the connection to the custom MLOps.

Annex C: Platform Level APIs

C.1 Public APIs

The TANGO Public API provides various REST endpoints allowing external applications and users to interact with the platform's models. The OpenAPI file containing a detailed description of the API is available here: https://gitlab.com/tango-ecosystem/tango-api/-/blob/main/documentation/public-api-openapi.yml

The API is structured into distinct groups, each focusing on specific functionalities like managing models, sessions, and invocations. Here is an overview of the API groups.

Model APIs

The "Model" group of the API is used to manage and retrieve information about the models available on the platform. This group allows users to:

- List all available TANGO models;
- Retrieve details about specific models, such as human-readable metadata and model specifications;
- Access model signatures and versions, which provide further insight into the capabilities and updates of a model.

This API is crucial for users who want to explore the models available on the platform before invoking them for specific tasks.

Session APIs

The "Session Management" group of the API handles user sessions within the platform. A session represents a collection of user interactions with a model, including queries, feedback, and the model's responses and explanations. The session provides an overarching context for tracking interactions over time and consists of a list of invocations (individual queries).

This group of API supports:

- Listing all sessions associated with a specific model;
- Creating new sessions;
- Retrieving metadata for a specific session, including its status and details;
- Deleting a session if needed.

This helps in managing different user interactions with models and provides a structured view of their activity and results.

Invocation APIs

The "Invocation Management" group of the API handles individual user queries, called invocations, within a session. An invocation represents a single query made by the user to the model and contains the following:

- **request**: information about the user's query, such as prompt, feedback, or other data;
- **response**: the model's reply to the query;
- **explanation**: a detailed explanation of the model's response, offering transparency into how the result was derived.

By using the invocation API, the user can:

- List all invocations within a specific session;
- Add a new invocation to a session;
- Retrieve the details of a specific invocation, including its response and explanation;
- Delete an invocation if needed.

C.2 Private APIs

This section provides an overview of the various API groups within the TANGO Private API, which is dedicated to internal users of the platform, including research groups responsible for managing data and models (WP2/WP3), as well as WP4 developers. The TANGO Private API serves as the primary interface for internal operations related to data management, model monitoring, and analytics. The full documentation in OpenAPI format is available at the following URL:

https://gitlab.com/tango-ecosystem/tango-api/-/blob/main/documentation/private-api-openapi.yml

Data Catalog Management

The Data Catalog Management group allows researchers to organize, manage, and interact with their data catalogs. A data catalog is a collection of datasets that are relevant to specific research projects or workflows. These catalogs are critical for maintaining a structured data environment, enabling internal users to efficiently organize and control their datasets.

The key functionalities in this group include:

- List Data Catalogs: Researchers can retrieve a list of all available data catalogs they have access to within the platform.
- Create a Data Catalog: Researchers can add new data catalogs to manage their datasets.
- View Data Catalog Details: Retrieve detailed information about a specific data catalog using its unique ID.
- Update a Data Catalog: Modify the metadata or structure of an existing data catalog.
- Delete a Data Catalog: Remove data catalogs that are no longer in use.

Dataset Management

The Dataset Management group provides internal users with the ability to manage individual datasets within their data catalogs. Datasets are the fundamental units of data within catalogs, and they are used for various research activities, including model training and evaluation.

APIs in this group include:

- List Datasets: View all datasets within a specific data catalog.
- Create a Dataset: Add new datasets to a specific catalog, facilitating data management for research purposes.
- View Dataset Details: Retrieve information about a particular dataset, including its metadata, by its unique ID.
- Update a Dataset: Modify the content or metadata of an existing dataset.
- Delete a Dataset: Remove datasets from the catalog when they are no longer needed.

Model Monitoring and Analytics

The Model Monitoring API group allows researchers and WP4 developers to monitor the status of their deployed models. These APIs provide real-time insights into model health, helping internal users identify and address any issues that may arise.

The Model Analytics API group provides access to detailed performance metrics and analytics for deployed models. This allows internal users to evaluate the effectiveness of their models and make data-driven improvements where necessary.

C.3 Authentication

The TANGO platform implements a robust authentication and access management system through its TANGO Auth API, designed to manage user sessions, tokens, grants, and workspace access. This system ensures secure, role-based access to the platform's resources, with flexibility for users who belong to multiple workspaces. While the API provides the technical means for access management, the platform’s dashboard offers a more user-friendly interface for managing workspaces, grants, and tokens.

The OpenAPI descriptor is available at following url:

https://gitlab.com/tango-ecosystem/tango-api/-/blob/main/documentation/auth-api-openapi.yml

Access Management

The Access Management APIs are central to handling user authentication, session management, and access control on the TANGO platform. These APIs are meant to be used mostly via a graphical user interface (dashboard).

Workspaces and Workspace Management

In the TANGO platform, a workspace functions as a tenant, isolating user sessions and resources. Only users with valid tokens specific to a workspace can access its associated data and features. A single user may belong to multiple workspaces, each with distinct permissions.

The Workspace Management APIs handle the technical side of creating, updating, and deleting workspaces.

Grant Management

A grant defines a user's role and permissions within a workspace. The Grant Management APIs allow administrators to manage these permissions, ensuring that users only access resources they are authorized to use.

Token Management

The **Token Management** APIs are responsible for handling the creation, management, and revocation of tokens within the TANGO platform. Each token is created specifically for a **workspace**, which acts as a secure tenant that isolates user sessions and resources.

Tokens are essential for authenticating external applications in the public API and provide access to key endpoints, including session management, invocation, and model metadata.

Annex D: Deployment detail

D.1 DevOps platform

A DevOps platform is a comprehensive suite of tools, practices, and processes that facilitates the integration of software development (Dev) and IT operations (Ops). The primary goal of a DevOps platform is to enable faster, more reliable software delivery by fostering collaboration between development and operations teams.

For the TANGO project, GitLab has been chosen as the DevOps platform due to its comprehensive suite of tools that seamlessly integrate with the software development lifecycle. GitLab will serve as the central repository for storing the source code of the various models and components that make up the platform. This centralized version control system will ensure that all team members have access to the latest code, fostering collaboration and maintaining a consistent development environment.

The DevOps platform will serve the following purposes in the project:

- **Source Code Management**: GitLab's robust Git repository features allow for efficient version control, branch management, and collaboration. Each component and model within the platform will have its

own repository, making it easier to track changes, manage different versions, and ensure code integrity across the project.

- **CI/CD Capabilities**: GitLab provides powerful Continuous Integration/Continuous Deployment (CI/CD) tools that automate the testing and deployment of platform components. By leveraging these tools, the development team can ensure that each change to the codebase is automatically tested and validated, reducing the risk of introducing errors into the production environment. The automated deployment process also ensures that new features and updates can be delivered quickly and reliably.
- **Automation of Machine Learning Workflows**: Where possible, GitLab will also be utilized to automate the training and deployment of machine learning models. By integrating model training into the CI/CD pipeline, the platform can automatically retrain models when new data becomes available, ensuring that the deployed models are always up-to-date. This automation reduces manual intervention, accelerates the deployment of machine learning models, and ensures consistent and reproducible results.

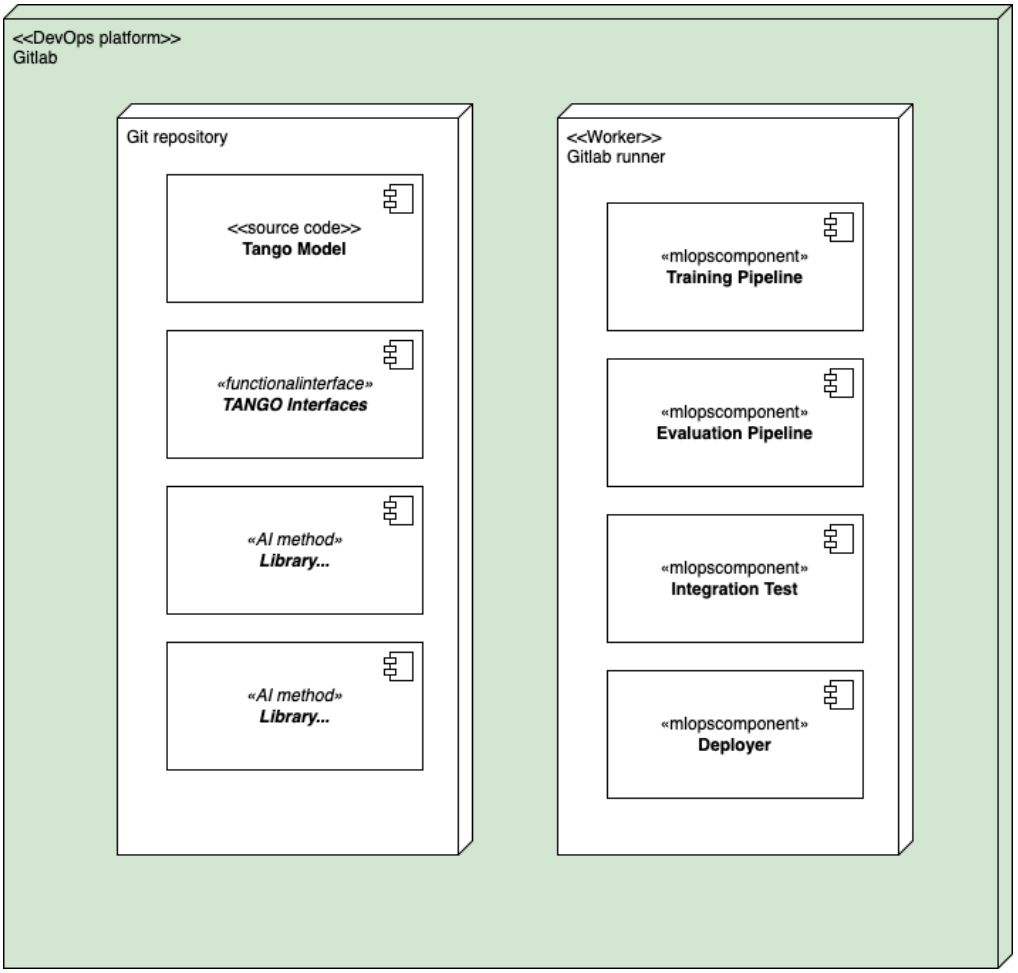

Figure 17: Deployment architecture: DevOps platform

D.2 MLOpsPlatform

An MLOps platform is a specialized framework designed to streamline the deployment, monitoring, and management of machine learning (ML) models in production environments. MLOps, which stands for Machine Learning Operations, combines practices from DevOps (Development and Operations) with the unique requirements of machine learning to ensure that ML models are efficiently developed, tested, deployed, and maintained.

Unlike traditional software, machine learning models require continuous monitoring and retraining as new data becomes available. An MLOps platform provides tools and processes that automate and optimize these tasks, ensuring that models remain accurate and effective over time.

The MLOps platform will serve the following purposes in the project:

- **Model Lifecycle Management**: MLOps platforms facilitate the entire lifecycle of ML models, from initial development and experimentation to deployment and ongoing maintenance. This includes version control for models, automated training pipelines, and easy deployment mechanisms.
- **Automated Pipelines**: Just as CI/CD pipelines are used in software development, MLOps platforms provide automated pipelines for training, validating, and deploying ML models. This automation reduces the time and effort required to move models from development to production.
- **Monitoring and Governance**: MLOps platforms include tools for monitoring model performance in real-time, detecting issues such as model drift, and triggering retraining when necessary. They also provide governance features to ensure compliance with regulatory requirements and organizational standards.
- **Collaboration and Reproducibility**: MLOps platforms enable collaboration between data scientists, engineers, and operations teams, ensuring that models can be reproduced, audited, and updated by different team members.

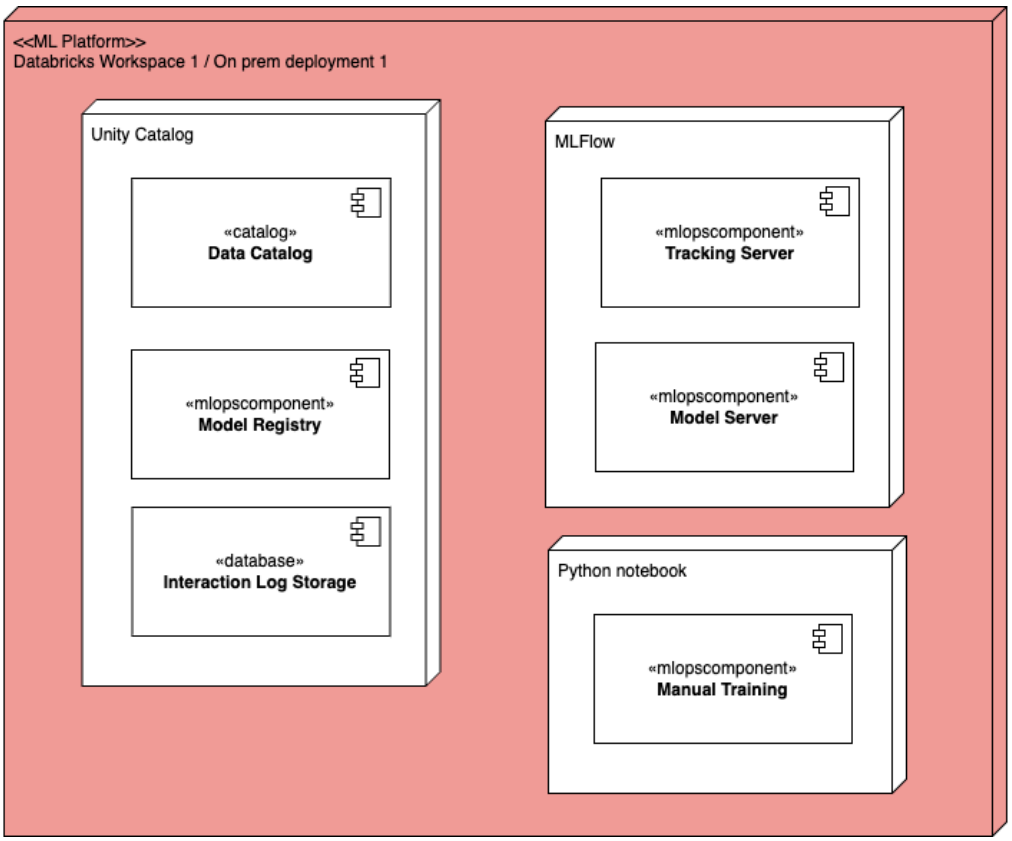

Figure 18: Deployment architecture: ML platform

In the TANGO project, a robust and scalable MLOps platform is essential for managing the entire lifecycle of machine learning models, from development and training to deployment and monitoring. After evaluating various options, MLFlow has been selected as the core of the MLOps platform due to its open-source nature and ability to meet the specific project needs. As an open-source framework, MLFlow offers flexibility, transparency, and community support, making it a highly adaptable and cost-effective solution. Its comprehensive suite of tools and services for effective machine learning operations further solidifies MLFlow as the ideal choice for the TANGO project.

The MLFlow framework provides the **Model Registry, Tracking Server** and **Model Serving** required by TANGO.

To complement MLFlow's capabilities, the TANGO project has incorporated additional components into the MLOps platform:

- **Data Catalog with Unity Catalog**: Managing and cataloging the training data is a critical aspect of any MLOps platform. For the TANGO project, the Unity Catalog has been selected as the data catalog enabling technology. The Unity Catalog provides a unified and organized repository of all training datasets, ensuring easy access, versioning, and governance of data used in model training. This integration enhances data management and supports reproducibility and compliance with data governance policies.

- **Manual Training with Jupyter Environment**: The TANGO platform is designed to support automated training processes, streamlining the development and deployment of machine learning models. However, there are situations where manual training and experimentation are necessary to refine models or explore new approaches. To accommodate these data science needs, a Python Jupyter environment has been integrated into the MLOps platform. This environment allows data scientists to interactively develop, fine-tune models, explore data, and conduct ad-hoc analyses. Jupyter notebooks provide the flexibility required for these tasks, enabling a hands-on approach to model development when automation alone is insufficient.

D.2.1 Commercial Managed Solution: Databricks

In addition to the open-source and self-hosted options, Databricks has been identified as a commercial solution that offers all the aforementioned components in a managed environment. Databricks provides a fully integrated platform that includes MLFlow, Unity Catalog, and Jupyter notebooks, all managed and maintained by Databricks. This managed solution offers scalability, ease of use, and enterprise-grade support, making it a suitable option for the TANGO project.

D.2.2 HuggingFace as an Alternative Option for the TANGO Platform

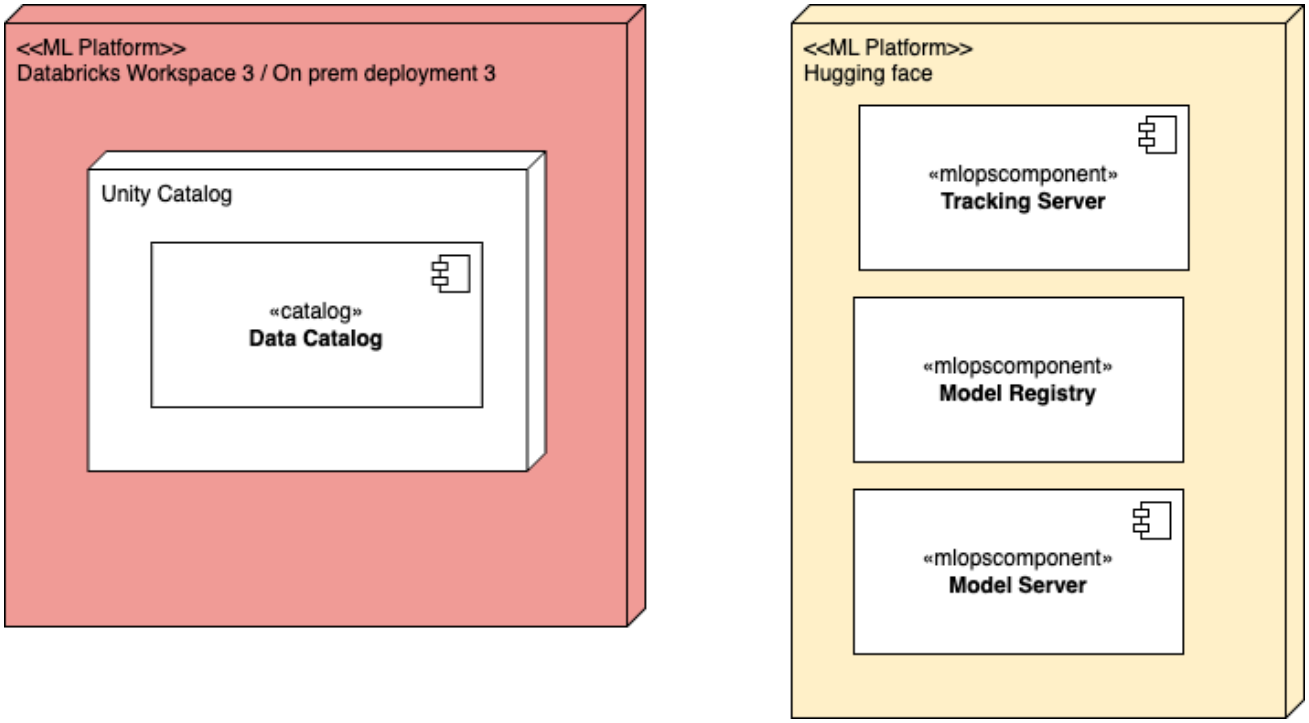

Figure 19: Deployment architecture: HuggingFace

HuggingFace can be considered as an alternative to MLFlow on Databricks as a commercial solution for managing machine learning workflows within the TANGO platform. Particularly well-suited for deploying and integrating large language models (LLMs), HuggingFace is renowned for its powerful tools and user-friendly services. It simplifies the deployment of advanced models and offers strong support for LLMs, making it a compelling choice for specific machine learning tasks where specialized capabilities are needed.

However, despite its advantages, HuggingFace is only partially based on open-source technologies, which is a key consideration for the TANGO platform's core infrastructure. The TANGO platform is committed to using open-source solutions to ensure flexibility, transparency, and minimising vendor lock-in. Accordingly, HuggingFace will not serve as the platform's primary MLOps solution.

That said, HuggingFace may still be used within the TANGO platform for specific scenarios. Some models, particularly those that require the advanced capabilities of HuggingFace's LLM support, could indeed be deployed on HuggingFace and integrated into the TANGO platform via the orchestrator. To enable this integration, a dedicated implementation of the TANGO Model and TANGO Model Proxy interfaces must be provided, as detailed in Chapter 4.1.2. This implementation will allow the seamless communication between HuggingFace and the TANGO Ecosystem, ensuring that models can be orchestrated and managed efficiently within the platform. This selective integration allows the TANGO platform to leverage the strengths of HuggingFace where it makes the most sense, while continuing to prioritize open-source technologies for the majority of its MLOps pipeline.

This approach provides the best of both worlds, ensuring that the TANGO platform remains open and flexible while also benefiting from HuggingFace's cutting-edge tools for certain specialized tasks.

D.2.3 Automation in the MLOps platform

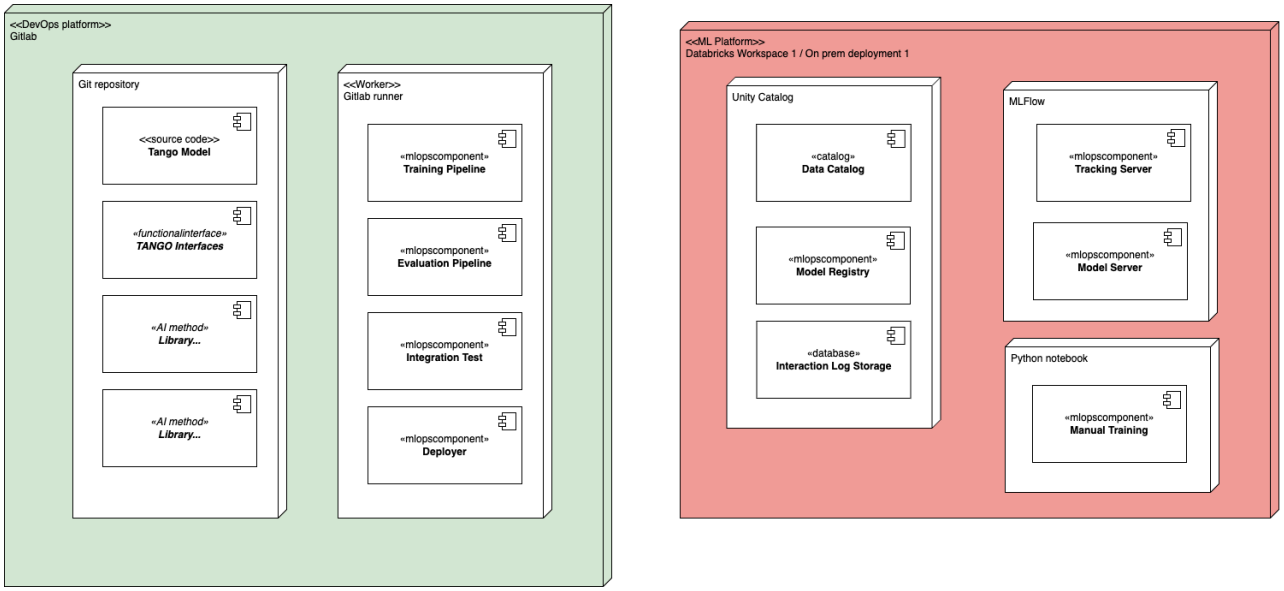

Figure 20: Deployment architecture: Automation in the MLOps platform

Automation is a key component of the machine learning lifecycle in the TANGO platform, enabling efficient, scalable, and reproducible model development. To facilitate automated training, the platform leverages the powerful capabilities of the GitLab DevOps pipeline in conjunction with the MLFlow-based MLOps framework.

The GitLab DevOps pipeline is at the heart of the TANGO platform's automation strategy. It provides a streamlined process for managing the continuous integration and continuous deployment (CI/CD) of machine learning models. By integrating the GitLab pipeline with the MLFlow platform, the TANGO project enables

the automation of model training, testing, and deployment, ensuring that models are consistently updated and optimized as new data becomes available.

The following is an example of a possible automated training workflow:

1. **Triggering Training Pipelines**: In the TANGO platform, the training process can be automatically triggered through the GitLab pipeline whenever new code or data is pushed to the repository. This automated trigger initiates a series of predefined steps within the CI/CD pipeline, including data preprocessing, model training, evaluation, and versioning.
2. **Seamless Integration with MLFlow**: The GitLab pipeline works in tandem with MLFlow to manage the training lifecycle. As models are trained, their parameters, metrics, and models are logged in the MLFlow tracking server. Successful models are then registered in the MLFlow model registry, making them available for deployment.
3. **Continuous Improvement**: The combination of GitLab and MLFlow allows for continuous improvement of machine learning models. As new data is integrated into the platform, the automated pipeline retrains models, ensuring that they remain accurate and effective over time. This approach reduces the need for manual intervention, accelerates development cycles, and enhances the overall reliability of the platform.

D.3 Cloud infrastructure

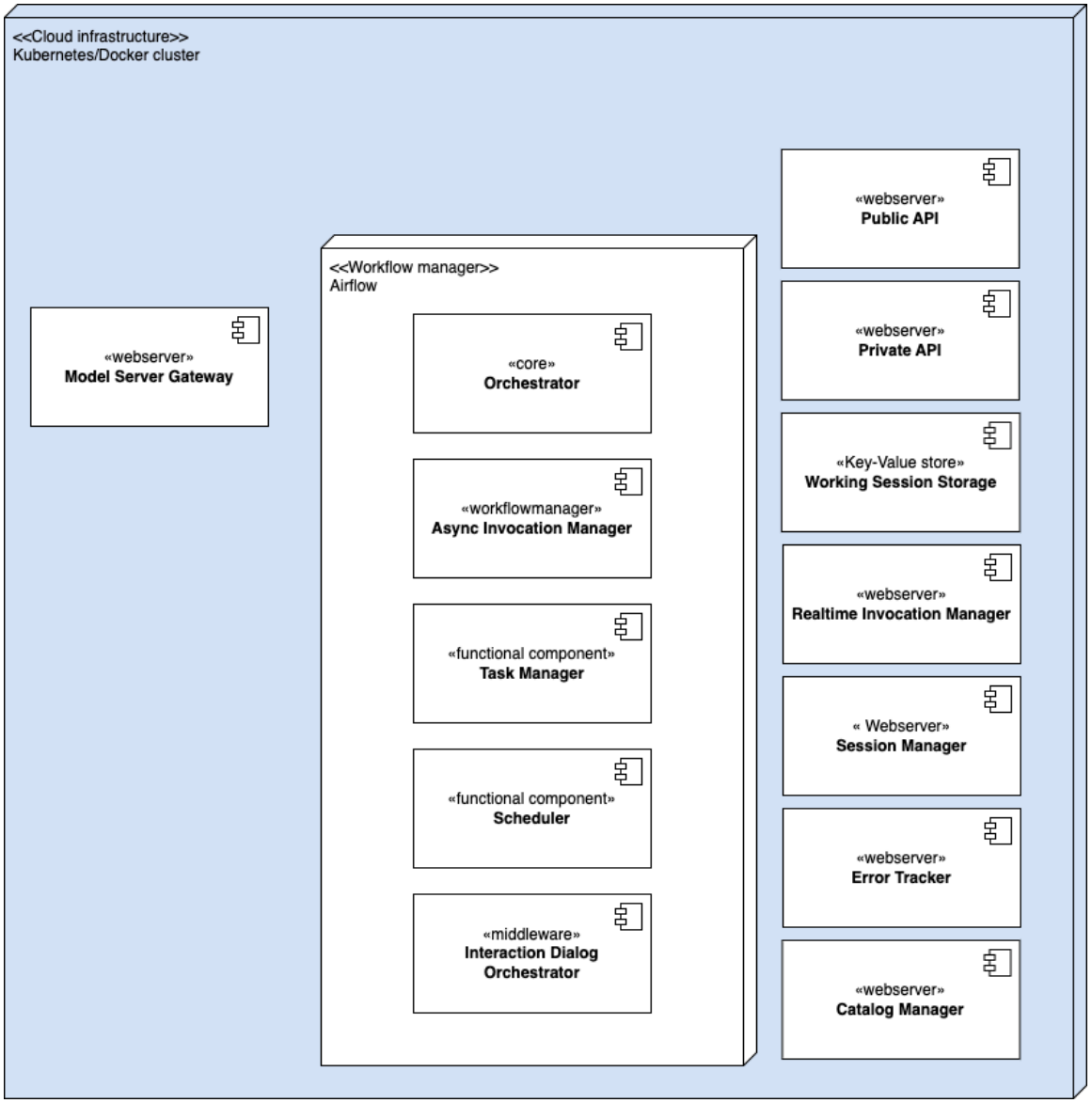

Figure 21: Deployment architecture: Cloud infrastructure

In the TANGO platform, all components of the orchestrator and API will be encapsulated within Docker containers and deployed on either a Docker or Kubernetes cluster. This approach leverages the power of containerization to ensure a scalable, reliable, and consistent deployment environment for the platform.

Docker is a platform that enables developers to automate the deployment of applications inside lightweight, portable containers. These containers package the application code along with its dependencies, libraries, and runtime environment, ensuring that the application runs consistently across different computing environments. Docker containers are highly efficient, allowing multiple containers to run on the same host system while using minimal resources.

Kubernetes is an open-source container orchestration platform designed to manage containerized applications at scale. It automates the deployment, scaling, and operation of application containers across a cluster of machines. Kubernetes provides advanced features like load balancing, self-healing, and rolling updates, making it a powerful tool for managing complex, distributed applications in production environments.

Advantages of Docker

1. **Portability**: Docker containers encapsulate everything the application needs to run, making them portable across different environments—whether it's a developer's local machine, an on-premises data center, or a cloud environment. This ensures that applications run consistently, regardless of where they are deployed.
2. **Consistency Across Environments**: Since Docker containers include the application along with all its dependencies, they ensure that the application behaves the same way in development, testing, and production environments, eliminating the well-known "But it works on my machine!" issue.
3. **Lightweight and Efficient**: Docker containers are lightweight compared to traditional virtual machines, allowing multiple containers to run on the same host with minimal overhead, making efficient use of system resources.
4. **Simplified Deployment**: Docker simplifies the process of deploying applications by packaging them and their dependencies into a single container. This reduces the complexity of setting up and configuring environments, speeding up the deployment process.

Advantages of Kubernetes

1. **Scalability**: Kubernetes excels at managing applications at scale. It can automatically adjust the number of running containers based on the current load, ensuring that the application remains responsive under varying workloads. This scalability is crucial for the TANGO platform, which may need to handle varying levels of demand.
2. **Reliability and Resilience**: Kubernetes provides self-healing capabilities, automatically restarting containers that fail, replacing containers when nodes die, and rescheduling containers across the cluster to ensure high availability. This ensures that the TANGO platform remains reliable and resilient in the face of failures.
3. **Automated Load Balancing**: Kubernetes automatically distributes network traffic across multiple containers, ensuring that no single container is overwhelmed, which enhances the platform's performance and reliability.
4. **Rolling Updates and Rollbacks**: Kubernetes allows for seamless rolling updates, ensuring that updates can be deployed without downtime. If an update causes issues, Kubernetes also supports easy rollbacks to a previous version.
5. **Efficient Resource Utilization**: Kubernetes optimizes resource usage across the cluster, ensuring that containers are running on the most appropriate nodes based on available resources, which is particularly beneficial in large-scale deployments.

For the TANGO platform, each component of the orchestrator and API will be encapsulated in its own Docker container. These containers will include all necessary dependencies, libraries, and configuration settings, ensuring that each component operates consistently across different environments.

Once encapsulated in Docker containers, these components will be deployed on a Docker or Kubernetes cluster. Using Kubernetes, the TANGO platform can manage these containerized components at scale, ensuring that the orchestrator and API are always available, responsive, and capable of handling the demands of the platform.

The deployment process will take advantage of Kubernetes' advanced orchestration features, such as automatic scaling, load balancing, and rolling updates, to maintain high availability and performance. This containerized approach not only simplifies the deployment and management of the TANGO platform but also enhances its scalability, reliability, and efficiency.

The specific technologies and tools to be used for the various components within the orchestrator and API will be determined during the overall development of the platform (Task 4.2).

D.4 Third party applications

The third-party applications integrated with the platform are not part of the work performed under Work Package 4 (WP4). This includes the applications that will be used within the framework of WP5 to run the pilot use cases. Each of these third-party applications will be maintained and deployed independently by their respective maintainers. These applications will interface with the platform exclusively through its public API, allowing them to access the necessary services and data without requiring direct involvement in the platform's core development or deployment processes. This approach ensures that the platform remains modular and flexible, while also allowing third-party developers to manage and update their applications as needed, independently of the main project timeline.

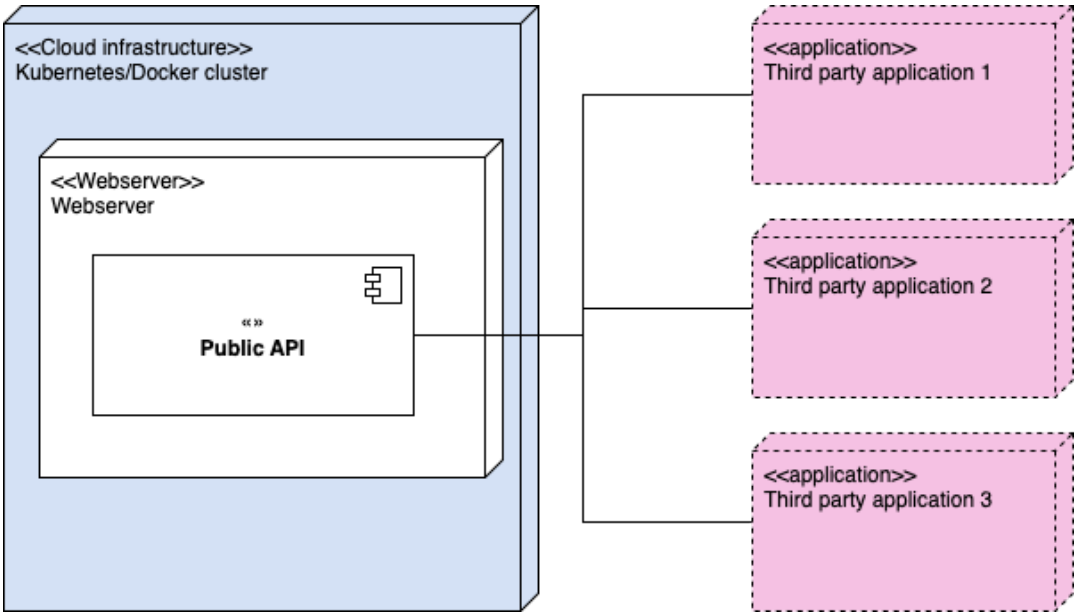

Figure 22: Deployment architecture: Third party apps

Annex E - Interactions with WP2/WP3

The collaboration between WP4 and WP2/WP3 serves as a cornerstone of the TANGO project. This partnership enables the models developed in WP2 and WP3 to be easily used within the WP5 case studies through a structured, reliable, and scalable platform.

In general, there are three ways in which outputs from WP2/WP3 can be integrated into the TANGO ecosystem:

1. as part of the TANGO library (applies to, e.g., explainers or AI methods for which the creators decide to make available the complete source code required to train and run the system);
2. as a TANGO model (applies to pre-trained models that can be used for tackling specific use cases, as well as - potentially - foundation models);
3. as a TANGO MLOps (applies to those situations where the creator wants to keep control of her artifacts, making it available only through APIs without sharing model weights or similar).

The three options are not mutually exclusive. We do expect indeed that most ML/DL models developed within WP2/WP3 will be made available through both option 1 and 2 (for the actual source code and the trained model, respectively).

The following sections provide detailed insights into these options, providing operational details and processes to be followed.

E.1 Integration in the TANGO library

The TANGO Library is the backbone of the TANGO Ecosystem, providing:

- a set of common interfaces to guide code development and
- utilities to enhance models with the features targeted by the TANGO project.

The structure of the TANGO library is represented in the following diagram (including some potential candidates for inclusion).

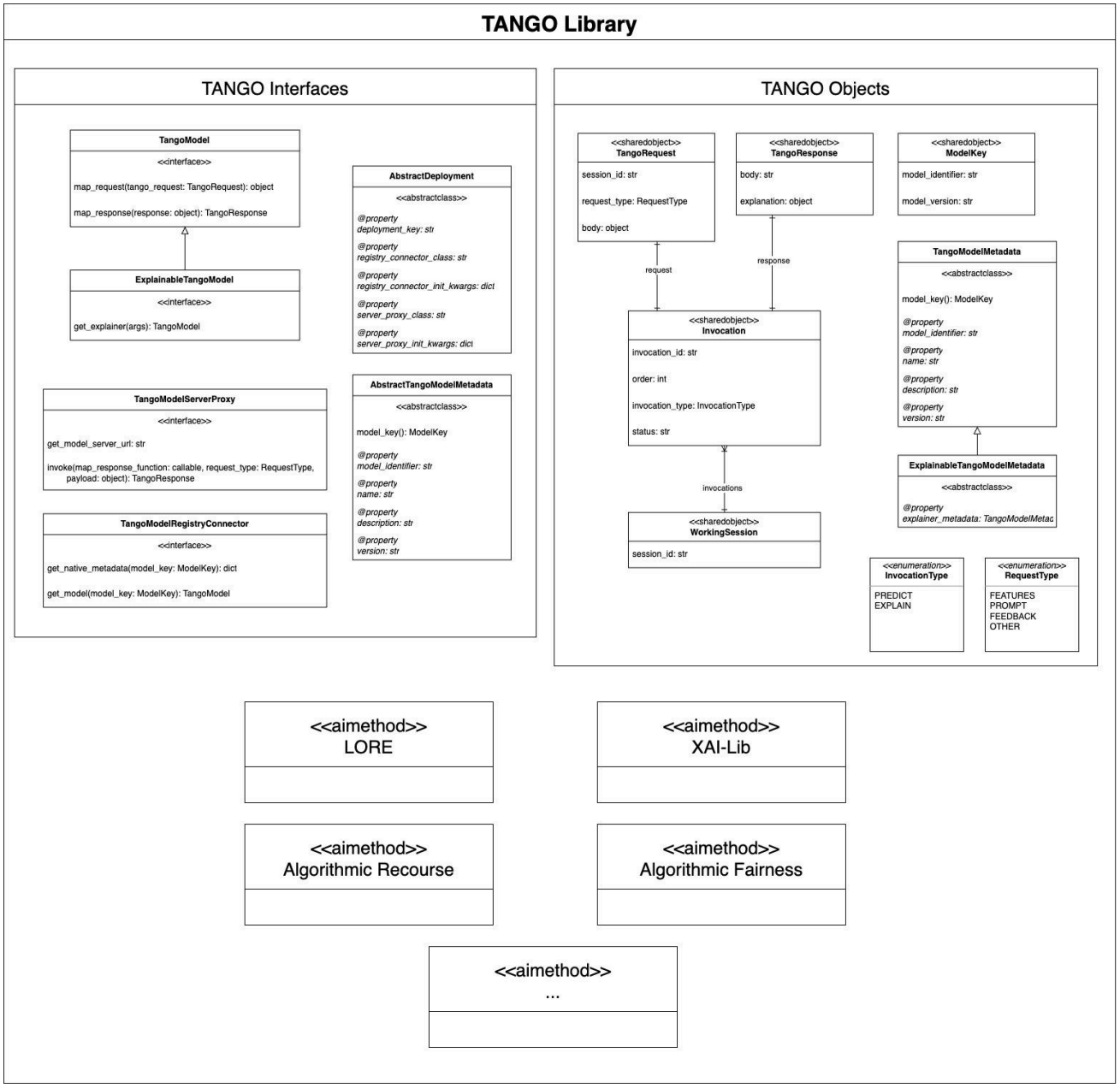

Figure 23: The TANGO Library structure

The TANGO Library contains reusable components (interfaces, classes, algorithms, packages) every TANGO contributor can have access to in order to empower its model development using shared artifacts and to implement models ready to be integrated into the TANGO Ecosystem.

Regarding the integration in the TANGO Ecosystem, TANGO Interfaces and TANGO Objects contain functional interfaces and useful objects allowing contributors to easily add new models or extensions to the ecosystem. One of the core interfaces coming from the Library is the `TangoModel` interface, responsible for defining functional constraints every model must fulfill in the form of a bidirectional way to map request and responses to the Tango-well-known format of TangoRequest and TangoResponse:

- The method `map_request(self, tango_request: TangoRequest) -> object` converts a TANGO request (the default way to access prediction passing by the Public API) to the input needed by the model server publishing the model;
- The method `map_response(self, response: object) -> TangoResponse` converts a custom response (output from the model server publishing the model) to a TANGO response (the default way to get back predictions from the Public API).

Once this interface is implemented, the TANGO Ecosystem is able to forward custom requests to models published on the specific MLOps and to map back the response to the client invoking prediction on TANGO APIs.

When releasing an explainable model to the TANGO Ecosystem, the `ExplainableTangoModel` interface (child of the `TangoModel` itself) must be implemented defining the method `get_explainer(self, **kwargs) -> TangoModel` that returns a model responding to the 'predict' method providing explanation for related prediction.

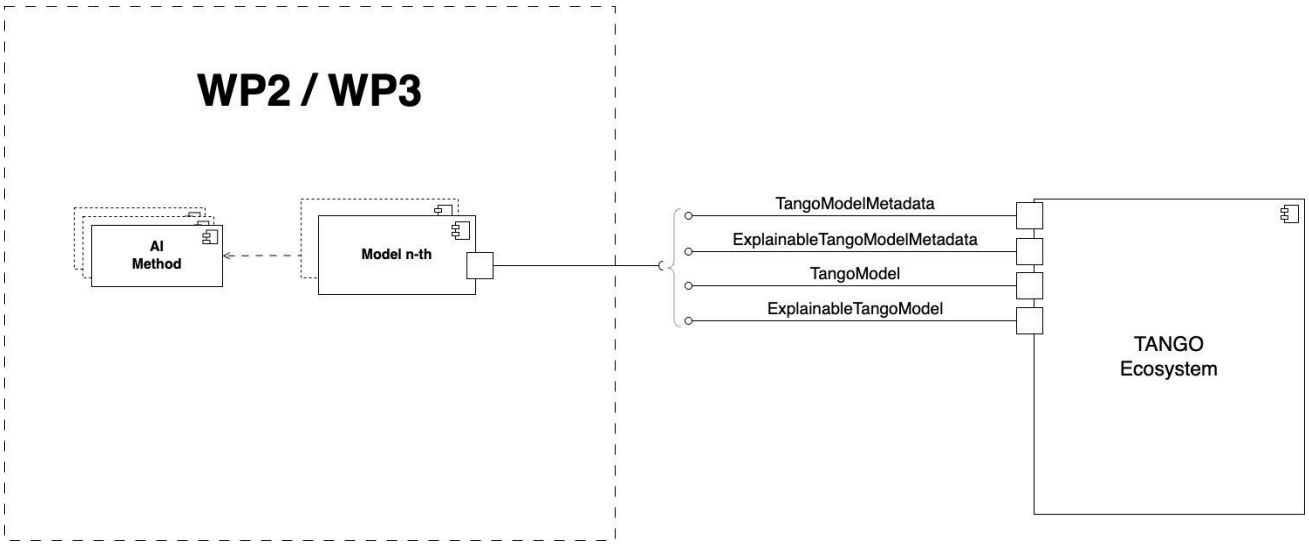

Figure 24: Conceptual model for the integration of WP2/WP3 outputs in the TANGO Library

Among interfaces related to the model definition phase, the `TangoModelMetadata` and `ExplainableTangoModelMetadata` are worth a mention as they provide standardized ways to format the descriptive metadata of a model during the registration process (more details can be found in Adding new model in the TANGO Model Catalog section).

More on other TANGO interfaces (and their implementations) can be found on Adding new MLOps deployment in the TANGO Model Catalog section.

To maintain the quality and relevance of the TANGO Library, a qualification and acceptance process has been established. This process ensures that any addition to the library meets the technical and usability standards set forward for the TANGO Ecosystem.

The pre-requisites for inclusion in the TANGO library are the following ones:

- **Publication**: The candidate library/tool for inclusion must be published in a public Git repository. This repository must be accessible to the TANGO community and other potential users, ensuring transparency and ease of access. The library should be released under Apache v2 licence.
- **Documentation requirements**: Comprehensive and detailed documentation must accompany the library. This documentation should cover the library's purpose, functionality, installation instructions and usage examples. Proper documentation is critical to enable users to understand and utilize the library effectively.

The process will include the following steps:

1. **Submitting a Merge Request**: The requester (who must be a maintainer of the library/tool for which the inclusion is requested) must submit a merge request (MR) to the public documentation repository of the TANGO Ecosystem. This MR should include:
    a. The addition of the library to the list of approved libraries.
    b. A concise yet detailed description of the library's purpose, features, and potential benefits for TANGO users.
    c. A link to the library's Git repository.
    d. A link to the library's documentation to ensure users can access detailed information about the library's usage and capabilities.
2. **Community Discussion and Review**: The submitted merge request serves as the starting point for a public community discussion. During this phase, the library/tool maintainer commits to engage with the TANGO community, providing reasons for the library's inclusion and addressing any feedback or concerns raised. The goal is to convince the TANGO community of the library's utility and relevance to the ecosystem.
3. **Final Review and Decision**: After 30 days from the MR, a TANGO platform maintainer will conduct a final review of the library after the community discussion. If the maintainer determines that the library aligns with the TANGO Ecosystem's goals and meets the established standards, the merge request will be accepted, officially validating the library. If the library does not meet the criteria, the merge request will be rejected, and the maintainer will receive feedback on the reasons for rejection.

This qualification and acceptance process ensures that only high-quality and relevant tools become part of the TANGO Library, maintaining its value as a resource for the TANGO platform and its users.

E.1.1 Candidate WP2/WP3 methods to be integrated within the TANGO library

A considerable number of methods and tools, proposed for inclusion in the TANGO Library, are being developed by partners in WP2 and WP3. These partners focus on advancing AI methodologies and creating solutions that address the diverse needs of the TANGO ecosystem. Hereby we describe some candidate methods for integration in the TANGO library, detailing how they shall be integrated.

- LORE is a post-hoc, local explanation method designed specifically for tabular data. Given a single input record, formatted as an array of fixed length, and a black-box classifier with a predict function, it outputs a rule and a set of counterfactuals. Additionally, to enhance the clarity of the explanation, it can also return feature importances, examplars, and counter-exemplars (in the form of synthetic

records) as part of the output. LORE generates local explanations through four main steps. First, synthetic records are created based on the input record we aim at explaining, using a genetic-based generation, which can also incorporate statistical properties of the data. Next, a simple and interpretable decision tree is trained on the synthetic data as a local surrogate model, which aims at describing the vicinity of the record to explain. From this surrogate, rules and counterfactuals are extracted to explain the black-box model's decision. Technically, a rule corresponds to the path in the tree traversed by the input record, while counterfactuals are the closest paths in the tree with a different outcome label. Lastly, additional insights, such as feature importances, exemplars, and counter-exemplars, are provided to enhance the explanation's clarity. Feature importances are extracted from the tree, exploiting its training procedure. For the exemplars, instead, we select the synthetic records closer to the input record that traverse the same path in the tree. Similarly, the counter-exemplars are the closest records which have a different outcome label and traverse the paths of the counterfactuals. Developed by TANGO Consortium partners, LORE is part of the XAI-Lib, and will be used within the project (e.g., as an enabler of TRACE). In Annex F we present an experiment where we integrated LORE as a TangoExplainableModel.

- **TRACE** - TRACE is a visual interface that exploits interactivity and progressive disclosure to present to the user the outcome of an explanator. TRACE combines both data and model exploration to create a context for local explanations. In the current implementation it exploits the LORE algorithm[1] to derive rules and counter rules from a given instance. It also exploits feature importance algorithms to assist the user to identify the relevant attributes. TRACE could be conveniently integrated to the TANGO framework by interacting directly with the endpoints exposed by the library. TRACE is designed as a web-based visualization so it can be used and integrated in more complex interfaces or it can be used as the prominent access point to classification and explanation results. A detailed description of TRACE is presented in D3.1. TRACE shall be integrated with the TANGO Library as a TangoExplainableModel.
- **Algorithmic recourse** - Algorithmic Recourse (AR) is the problem of computing a sequence of actions that once performed by a user overturns an undesirable machine decision. In AR, given an undesirable machine-generated decision, the goal is to identify a sequence of actions that once implemented by the user overturns said decisions (e.g. rejected loan application). Several approaches have been recently proposed for computing AR, but it is critical that the suggested recourse plans are not too difficult or expensive to carry out. This entails that recourse should be personalized, because different users in the same situation may need substantially different recourse plans. The AR library integrates AR and ideas from interactive Preference Elicitation (PE) in a fully Bayesian setup, which uses a human-in-the-loop approach for generating personalized recourse tailored for a target end-user. The library will also provide a visual interface built on top of an interaction paradigm based on a guided interaction pattern aimed at both eliciting the users' preferences and heading them toward effective recourse interventions. AR shall be integrated with the TANGO library as a TangoModel.

[1] Guidotti, R., Monreale, A., Ruggieri, S. et al. Stable and actionable explanations of black-box models through factual and counterfactual rules. Data Min Knowl Disc 38, 2825–2862 (2024). https://doi.org/10.1007/s10618-022-00878-5

- **NeSy-CL** - The Neuro-Symbolic Concept Learner (NeSy-CL) is a framework that integrates neural networks and symbolic reasoning to enable better interpretability and control in machine learning systems. It decomposes visual scenes into symbolic representations, mapping features into human-understandable concepts. This allows the system to reason about these representations and perform tasks such as classification or decision-making at a semantic level, rather than relying solely on low-level visual features.

  Key features include:

  - Conceptual Decomposition: It uses methods like Slot Attention to identify and encode objects or components of a scene as symbolic entities.
  - Symbolic Reasoning: A reasoning module processes the symbolic representations to derive decisions or predictions, providing a logical layer that enhances transparency.
  - Explanatory Interactive Learning (XIL): The system supports user feedback to revise and improve the model's reasoning by intervening at the symbolic level. For example, users can correct erroneous associations by imposing rules such as "color should not influence decisions."
  - Clever-Hans Behavior Correction: By leveraging neuro-symbolic explanations, the framework identifies and mitigates "Clever-Hans" behaviors, where models rely on spurious correlations.

  The NeSy-CL framework could be conveniently wrapped as an ExplainableTangoModel. NeSy-CL is described in detail in D2.1.

E.2 Integration in the TANGO Model Catalog

The second option for integrating outputs from WP2/WP3 is to publish the model (code plus model weights) in the TANGO model catalog. This is adequate, e.g., for pre-trained models ready to be consumed by third party applications, taking advantage of the TANGO features while abstracting clients from complexities related to hardware allocation, deployment issues and/or inner model complexity.

Every model belonging to the TANGO Ecosystem needs to fulfill specific requirements in terms of implementing functional interfaces; the Model Catalog collects a set of said AI models.

The pre-requisites for inclusion in the TANGO Model Catalog are the following ones:

- The model implements the TangoModel interface provided by the TANGO Library;
- The model (code to run the model plus model weights) should be released under an Apache v2 license.

The model to be added to the catalog should provide the following:

- Metadata for the model in the form of a TangoModelMetadata (or ExplainableTangoModelMetadata if related to explainer) with detailed description, signature, usage examples, enriched by the adoption of descriptive markup languages (e.g., Markdown);
- Explainer model: as defined by the ExplainableTangoModelMetadata interface, if the model comes with its explanation counterpart, this can be released with the corresponding reference to the

explained model; note that every explainer is a model itself, and it could occur that a meta-explainer (explainer providing explanation for explainer outcome) will be made available to publishing (whilst this meta-explanation nested structure could be limitless, the platform will impose an upper threshold to avoid performance issues when accessing cascading explanations).

To add the model to the catalog, the model contributor will be provided access to the TANGO Private API by a platform maintainer. As access policies are determined by the platform maintainers, this means that the ability to add a model to the catalog is tightly controlled, and the access is provided only to trusted parties (limited only to project beneficiaries during the lifetime of the project); whose access policies are determined by the TANGO Community and the grants are assigned by the WP4 maintainers.

Technically, the registration takes place by invoking the endpoint `add_models_metadata`, passing as payload a list of pairs containing the deployment key for the target MLOps and the metadata to be linked to the model itself.

The payload is structured as a list in order to allow registering at once the model and its explainer (and, if any, meta-explainers): the first element in the list is the last explainer to be invoked during prediction, and the last element in the list will be the predictive model itself.

For example, the following is the JSON representation of a model metadata to be deployed on the MLOps identified by the deployment key `uh-mlflow-model`:

```
{
   "model_identifier": "proto_model",
   "name": "proto-model",
   "description": "proto-model is the best ever!",
   "version": "88"
}
```

The request can be successful only if a MLOps deployment exists among the registered ones in the TANGO Ecosystem with the corresponding key.

As for application developers (in particular those developing applications powering the case study pilots in WP5), they can access models via the public APIs. In particular, they can access the full list of models, use a specific model and, whenever possible, download a model locally. Let us detail these functionalities.

Obtain the list of available TANGO models.

In order to let TANGO Ecosystem users access the list of models available, the invocation of the `list_model_metadata` endpoint on the Public API will return the metadata of the models, each one containing the following information:

- `model_identifier`: the identifier of the model as defined by the contributor;
- `version`: the version of the model as defined by the contributor;
- `name`: the name of the model as defined by the contributor;
- `description`: the description of the model as defined by the contributor;

- `explainer_metadata`: if any, the metadata of the explainer of the model, as defined by the contributor;
- `model_key`: the unique identifier of the model in the TANGO Ecosystem, defined as the pair model_identifier and version
- `other`: a dictionary containing native metadata (means the custom metadata as managed by the specific MLOps implementation hosting the model), useful to store additional information about the model.

Among these, the `model_key` is crucial to let users invoke predictions on the chosen model.

Use a TANGO model

Using published TANGO Models on the Public API endpoints is the typical example of interaction between WP4 work and WP5 case studies, the latter aiming to take advantage of the predictive features of the models and the scalability and performance optimization offered by the ecosystem.

This can be done by invoking on the Public API the `predict` endpoint, passing as input arguments the followings:

- The model key, identifying the model in the ecosystem (see previous section)
- The TangoRequest object, containing the session id, the type of the request and the data to be processed:
    - The session id is useful when multiple interactions with the ecosystem are in place, allowing requests to be gathered to provide the model, by accessing the session history, the sequence of invocations, to be used to give a better context-aware response;
    - The type of request can distinguish between 'features', 'prompt', 'feedback' and 'other' options, to be appropriately managed by the model;
    - Data to be processed must be passed in the model in the raw format, remembering that the model itself, as implementation of the TangoModel interface, has the code needed to convert this raw data into a standard TangoRequest object.

The response to a predict invocation is always of type `Invocation` (see TANGO Objects in the sections above), but they can differ on the basis of the type of invocation: typically:

- a real-time request returns to the invoker the `Invocation` instance with a status `READY` or `ERROR` together with the body of the response itself, wrapped in a TangoResponse object,
- an asynchronous request returns to the invoker the Invocation instance with one of the following possible status: `QUEUED`, `IN_PROGRESS` and `ERROR`, without any further detail on the body response. This kind of response informs the invoker that the server is processing (or is going to process) the request and the invoker should make use of the invocation id to poll the dedicated endpoint to get updates on the status of the request: only once the status is equal to `READY` (or `ERROR`) the result will be available (or not available in case of `ERROR`).

Download a TANGO model

Downloading TANGO Models allows users to get a trained model artifact in order to execute it in a custom environment outside TANGO Ecosystem (i.e., locally on the user machine or on premise). This operation is allowed only on models coming with the `allow_download` flag set to true in their metadata.

E.2.1 Candidate WP2/WP3 models to be integrated in the TANGO Catalog

This section presents one WP3 model as a candidate for inclusion in the TANGO Model Catalog. (Besides the ones presented in Sec. E.1.1.)

MedSyn: Enhancing Diagnostics with Human-AI Collaboration

One line of research carried out in WP3 (in particular in T3.1) explores the potential of Large Language Models (LLMs) to assist medical professionals through multi-step, interactive dialogues. Unlike static diagnostic tools, LLMs in this context serve as dynamic collaborators, enhancing physicians' reasoning and judgment during clinical decision-making.

In medical practice, cognitive biases, incomplete information, and case complexity often influence decisions. Through bidirectional interaction, physicians can engage LLMs to:

- Uncover overlooked symptoms.
- Reevaluate treatment options.
- Address potential oversights in clinical reasoning.

This iterative dialogue allows physicians to challenge the model's suggestions, prompting refinements, while the LLM highlights alternative approaches or insights, fostering a collaborative decision-making process. This model is particularly impactful in complex medical cases, where such interactions improve diagnostic accuracy and therapeutic outcomes. The focus of this work lies in designing and evaluating these dialogues, ensuring that LLMs enhance human expertise while preserving the physician's critical role in patient care.

We foresee MedSyn to be published on the TANGO model catalog.

E.3 Integration via custom MLOps deployment

In order to manage the needs of WP2/WP3 models in terms of hardware resources, privacy issues and multi-platform model deployment, the TANGO Ecosystem provides a default MLOps implementation as target for model hosting as well as a well-defined procedure to add new MLOps deployment, making them available as endpoint to serve models. This option has been included in the design requirements to cover situations where, due to privacy or confidentiality issues, the creator of the model does not want to publish it, but only to make it available through an appropriate endpoint.

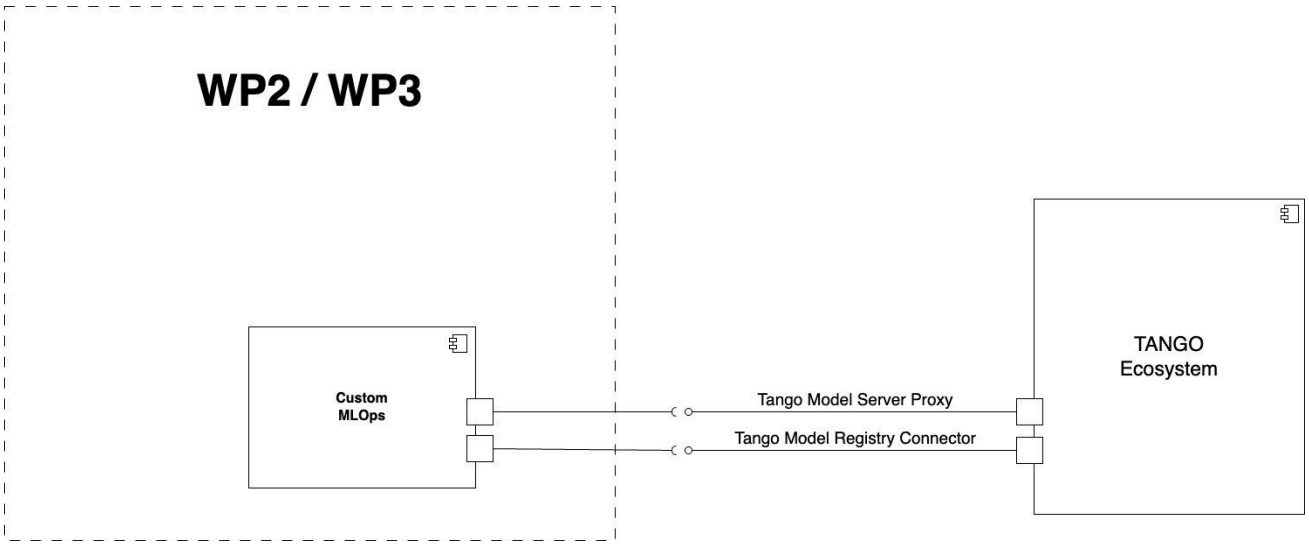

Figure 25: Conceptual model for the integration of WP2/WP3 outputs via custom MLOps deployment

The way to accomplish this includes providing a Python implementation for the following interfaces, responsible for designing the features of the connectors to the custom MLOps to be integrated.

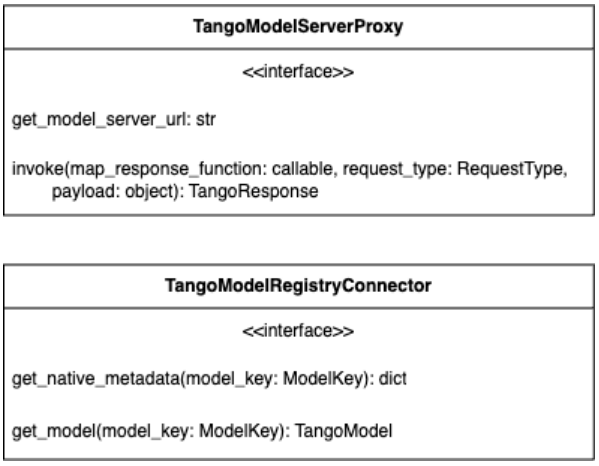

Figure 26: Details on the interfaces to be implemented for integration via custom MLOps deployment

TangoModelRegistryConnector

The TangoModelRegistryConnector is the interface for classes responsible to retrieve models and their native metadata from the MLOps deployment; these operations can be carried out by passing the identifier key of the model used when the model is added to the ecosystem (see Adding new model). Note that native metadata provided from TangoModelRegistryConnector implementation are the custom ones managed by the MLOps implementation and serialized as a Python dictionary; this differs from the TANGO metadata, implementation of the interface AbstractTangoModelMetadata, provided to the ecosystem during the add model operation (see Adding new model).

TangoModelServerProxy

The TangoModelServerProxy is the interface for classes responsible to forward invocations to specific MLOps by implementing a method that returns the model server url and another one invoking the model served by passing the following arguments:

- `map_response_function`: function that is responsible to convert di outcome if the prediction to a well known response form of type `TangoResponse`,
- `request_type`: request type, to choose among `FEATURES`, `PROMPT`, `FEEDBACK`, `OTHER`
- `payload`: payload of the prediction invocation, for example a dataframe.

Once these classes have been created, WP2/3 contributing developers must create a Merge Request to the Components public GitLab repository: the TANGO repository maintainers will then evaluate the MR content in order to check its TANGO Ecosystem specifications compliance and to run integration tests, promoting then the merge of the code in the main branch making it available for deployment in production.

The next step needed to map the MLOps on the ecosystem consists in adding the deployment to the one already registered by invoking on the Private API component the endpoint `add_deployment`, passing the following as argument:

- `deployment_key`: unique identifier for the deployment
- `registry_connector_class`: fully qualified name of the provided implementation of the registry connector class, e.g. default MLOps implementation uses
  `"components.connectors.mlflow_registry_connector.MlflowModelRegistryConnector"`
- `server_proxy_class`: fully qualified name of the provided implementation of the server proxy class, e.g. default MLOps implementation uses
  `"components.connectors.mlfow_server_proxy.MlflowModelServerProxy"`
- `registry_connector_init_kwargs`: keyword arguments for correctly initialize provided implementation of the registry connector class, e.g. for a typical MLFlow development environment, default MLOps implementation uses
  ```
  {"model_registry_url": "http://127.0.0.1:5000",
  "tracking_server_url": "http://127.0.0.1:5000"}
  ```
- `model_server_proxy_init_kwargs`: keyword arguments for correctly initialize provided implementation of the registry connector class, e.g. for a typical MLFlow development environment, default MLOps implementation uses
  `{"server_proxy_url": "http://127.0.0.1:5001"}`

The MLOps deployment is now ready to be used to add models (see Adding new model) deployed in the TANGO Ecosystem, making them available to use and download using the Public API (see Using TANGO Model and Downloading TANGO Model).

E.2.1 Candidate WP2/WP3 models to be integrated via custom MLOps

Within WP5, one of the pilot use cases (T5.3) deals with investigating the usefulness of an AI-based Decision Support Tool in the context of Abdominal Aortic Aneurysms. The typical clinical workflow is composed of three peri-operative phases: Pre-operative phase, Intra-operative phase and Post-operative phase.

During the pre-operative phase, surgeons do planning by taking various measurements from the patient's CT scan. The system is designed to take CT scans as input and provide a report with all pertinent measurements and suggested decisions for surgeons. These morphological features, besides saving surgeons' time and improving intraoperative performance, may be utilized to anticipate post-operative complications. This requires the development and training of AI models. In the training phase, as well as during operations, the model will handle clinical data which, according to GDPR, is to be considered personal data of special type. It is therefore foreseen, for such a use case, to deploy a custom MLOps, to ensure maximum protection of said data.

Annex F - Example: creating a TANGO Explainable Model (with LORE)

As part of the ongoing development of the TANGO platform (T4.2), a prototype for a TANGO explainable model has been created. This prototype serves as an example implementation for other developers and project partners, demonstrating how to integrate the TANGO interfaces into a machine learning model.

For this example, we took a standard and well-understood ML model (RandomForest from sklearn), wrapped it as an ExplainableTangoModel by integrating it (via a set_explainer() method) with LORE (as described in Section E.1.1).

The structure of the class is shown in Figure 27. The RandomForestWrapperModel acts as the core implementation, supported by the RandomForestExplainer for interpretability and leveraging the LORE method for explanations. The structured use of common interfaces ensures extensibility and consistent integration across different models and frameworks.

Let us now explain the diagram in detail.

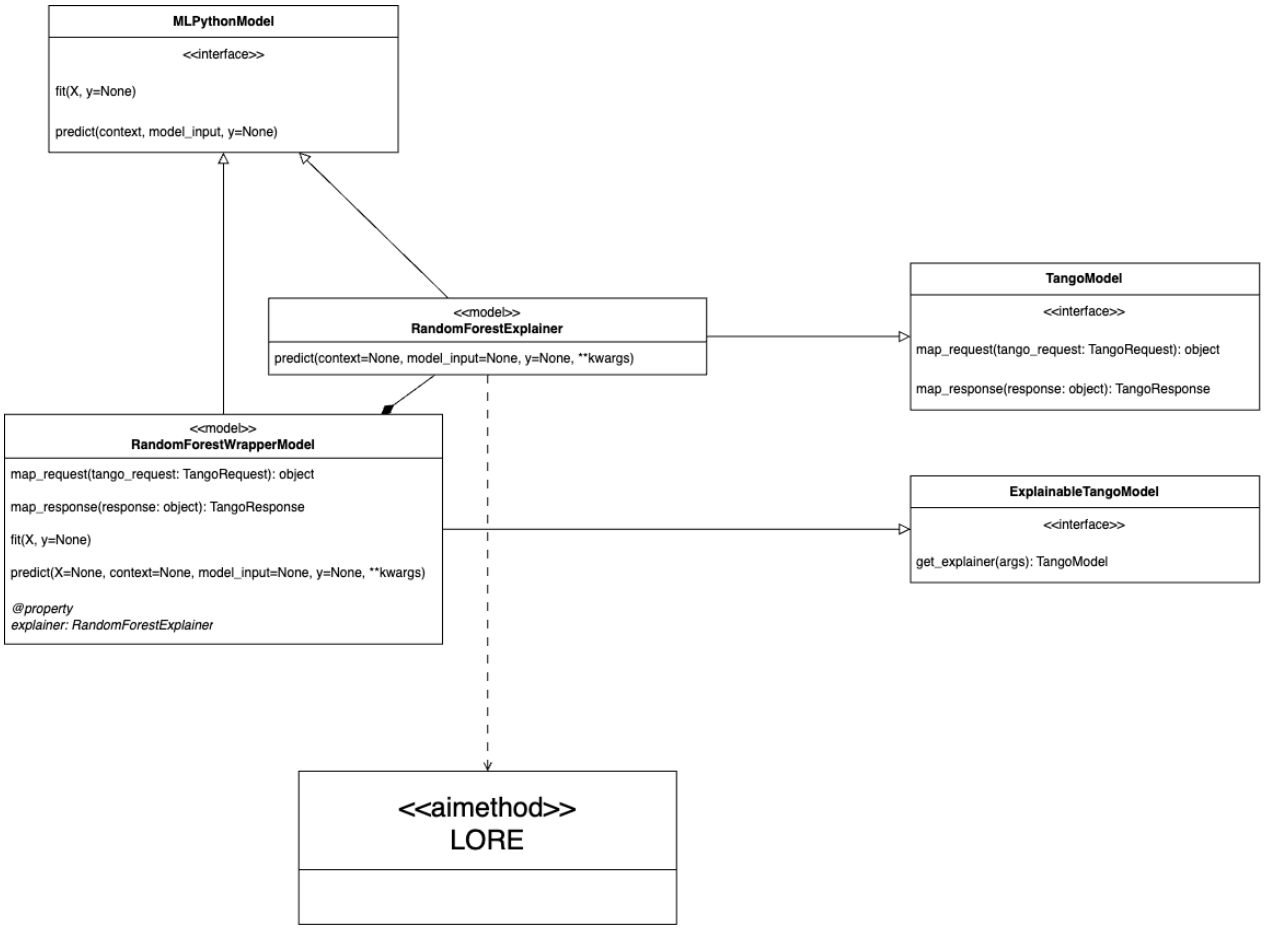

Figure 27: Structure of the example prototype

The *MLPythonModel* is an interface class that provides two key methods:

- *fit(X, y=None)*: A method to train the model on the input data X and target labels y.
- *predict(context, model_input, y=None)*: A method to make predictions given a context and input data. y is optional and may represent additional data for prediction.

The *RandomForestWrapperModel* is a concrete class that implements the MLPythonModel interface and serves as a wrapper around the sklearn RandomForest model. It exposes the following methods:

- *fit(X, y=None)*: Implements model training for Random Forest.
- *predict(X=None, context=None, model_input=None, y=None, **kwargs)*: Predicts using the Random Forest model. Accepts various inputs like X, context, and model_input, providing flexibility for different use cases.
- *map_request(tango_request: TangoRequest) -> object*: Maps a TangoRequest object to a format that the model can process.
- *map_response(response: object) -> TangoResponse*: Converts the model's output into a TangoResponse object.

The *RandomForestWrapperModel* exposes the *explainer* property, which returns an instance of RandomForestExplainer, enabling the model to provide explanations for its predictions.

The *RandomForestExplainer* is a class specifically designed to provide explainability for predictions made by the Random Forest model. It implements the following method:

- *predict(context=None, model_input=None, y=None, **kwargs)*: A method to generate explanations for a given prediction, possibly using auxiliary inputs such as context.

The *TangoModel* is an interface that defines methods for mapping requests and responses in the context of a Tango workflow:

- *map_request(tango_request: TangoRequest) -> object*: Maps Tango requests to the model's input format.
- *map_response(response: object) -> TangoResponse*: Converts the model's predictions into Tango-compatible responses.

This interface ensures interoperability with the TANGO framework.

The *ExplainableTangoModel* is an interface that extends *TangoModel* by adding an additional method:

- *get_explainer(args) -> TangoModel*: Retrieves an explainer instance for the model, ensuring that models implementing this interface can provide explanations for their outputs.

LORE (Local Rule-based Explanations) is an algorithm used to generate interpretable explanations for individual predictions (see Section E.1.1). In Figure 27, *LORE (AIMethod)* represents an explainability method that integrates with the *RandomForestWrapperModel* and *RandomForestExplainer*.

In terms of the relationships among the entities represented in the Figure,

- *RandomForestWrapperModel* inherits from *MLPythonModel* and implements the *TangoModel* interface, integrating machine learning functionality with the TANGO workflow.
- *RandomForestWrapperModel* also utilizes *RandomForestExplainer* for explainability.
- The *ExplainableTangoModel* interface ensures that any model implementing it can provide explanations via an explainer, aligning with the Random Forest model and LORE method.
- Interfaces like *MLPythonModel*, *TangoModel*, and *ExplainableTangoModel* define the core structure, while specific classes like *RandomForestWrapperModel* and *RandomForestExplainer* provide concrete implementations.

The codebase for the example is published in the project open-source repository and serves as a practical example on how outputs from WP2/WP3 can be integrated within the TANGO Library. Looking ahead, this prototype will not only serve as a practical example of implementing the TANGO interfaces but also lays the foundation for future enhancements in model development and deployment within the TANGO platform.