Workshop III

Universidad Distrital Francisco José de Caldas

Faculty of Engineering

Authors

Juan David Zárate Moya 20222020184
Jesus Mateo Munevar Mendez 2023202042
Kevin David Rincon Valencia 20232020356
Juan David Romero Morales 20222020102

Teacher
Carlos Andrés Sierra Virguez

November 2025
Bogotá D.C

Contents

1 Team Roles and Responsibilities

1.1 Roles Based on the Scrum Methodology

The project will be developed following the **agile Scrum methodology**, which promotes collaborative work, continuous delivery of results, and constant improvement. Each member of the team assumes a specific role that ensures organization, responsibility, and efficiency during the system's development.

1.1.1 Product Owner — Juan David Zárate Moya

Responsibilities:

- Defines the project vision and ensures that it aligns with both academic and technical objectives.
- Manages the *product backlog*, prioritizing system functionalities (data ingestion, preprocessing, modeling, monitoring, etc.).
- Ensures that each sprint produces valuable results and that the final system meets quality requirements.
- Coordinates communication between the technical and management teams to maintain consistency and alignment.

Motivation: Zárate acts as the link between the system's technical objectives and the academic deliverables, ensuring that the project maintains a clear and well-defined course.

1.1.2 Scrum Master — Kevin David Rincón Valencia

Responsibilities:

- Facilitates communication among all team members.
- Ensures compliance with Scrum principles: transparency, inspection, and adaptation.
- Identifies and removes obstacles that may hinder progress.
- Oversees time management, sprint coordination, and synchronization between sub-teams.

Motivation: Kevin's role is to maintain an efficient workflow, ensuring that sprint deliverables (e.g., diagrams, risk analyses, or documentation) are completed on time.

1.1.3 Technical Team — Development Group

Members: Jesús Mateo Múnevar Méndez and Juan David Romero Morales
Responsibilities:

- Design and implement the system's technical components.
- Build and test the data pipeline: ingestion, preprocessing, model training, and evaluation.
- Document technical decisions and update the architecture diagram.
- Ensure the quality, robustness, and reproducibility of the system.

1.1.4 Role Summary Table

Role	Team Member	Main Focus
Product Owner	Juan David Zárate Moya	Defines the vision, priorities, and scope of the project.
Scrum Master	Kevin David Rincón Valencia	Coordinates the team, removes blockers, and manages sprints.
Developer 1	Jesús Mateo Múnevar Méndez	Designs and implements the technical architecture.
Developer 2	Juan David Romero Morales	Works on modeling, quality assurance, and documentation.

Table 1: Roles and responsibilities of the project team.

2 Robust Architecture Overview

The updated system architecture was refined to ensure robustness, scalability, and maintainability following the principles of modular design and fault tolerance. The architecture is organized into six interconnected subsystems: *Data Ingestion*, *Processing*, *Modeling*, *Serving*, *Monitoring*, and *Orchestration*. Each subsystem performs an independent but coordinated function within the overall machine learning workflow, guaranteeing flexibility and reliability throughout the data pipeline.

- **Modularity:** Each subsystem (e.g., ingestion, preprocessing, modeling) operates as an independent module with well-defined inputs and outputs, allowing updates or replacements without affecting the rest of the system.
- **Scalability:** The architecture supports horizontal scaling through distributed execution of data processing and model inference tasks.
- **Fault Tolerance:** Checkpoints and recovery mechanisms are embedded in the pipeline to minimize data loss or system failure during training and validation.
- **Maintainability:** Source code is organized into modular folders (e.g., preprocessing, training, evaluation), with clear configuration files to ensure readability, reusability, and version control through GitHub.
- **Traceability and Monitoring:** Continuous logging and drift detection enable traceability of model behavior and automated retraining when performance degradation is detected.

This architecture builds upon the foundations from Workshop 2 but integrates more robust controls, ensuring reliability, fault isolation, and adaptability to data or environmental changes. The design aligns with ISO 9000 principles of quality assurance and CMMI standards for process maturity in software systems.

Furthermore, the architectural design aligns with ISO 9000 principles by emphasizing process documentation and quality assurance, while CMMI maturity level 2 practices are reflected through version control, peer review, and continuous improvement cycles.

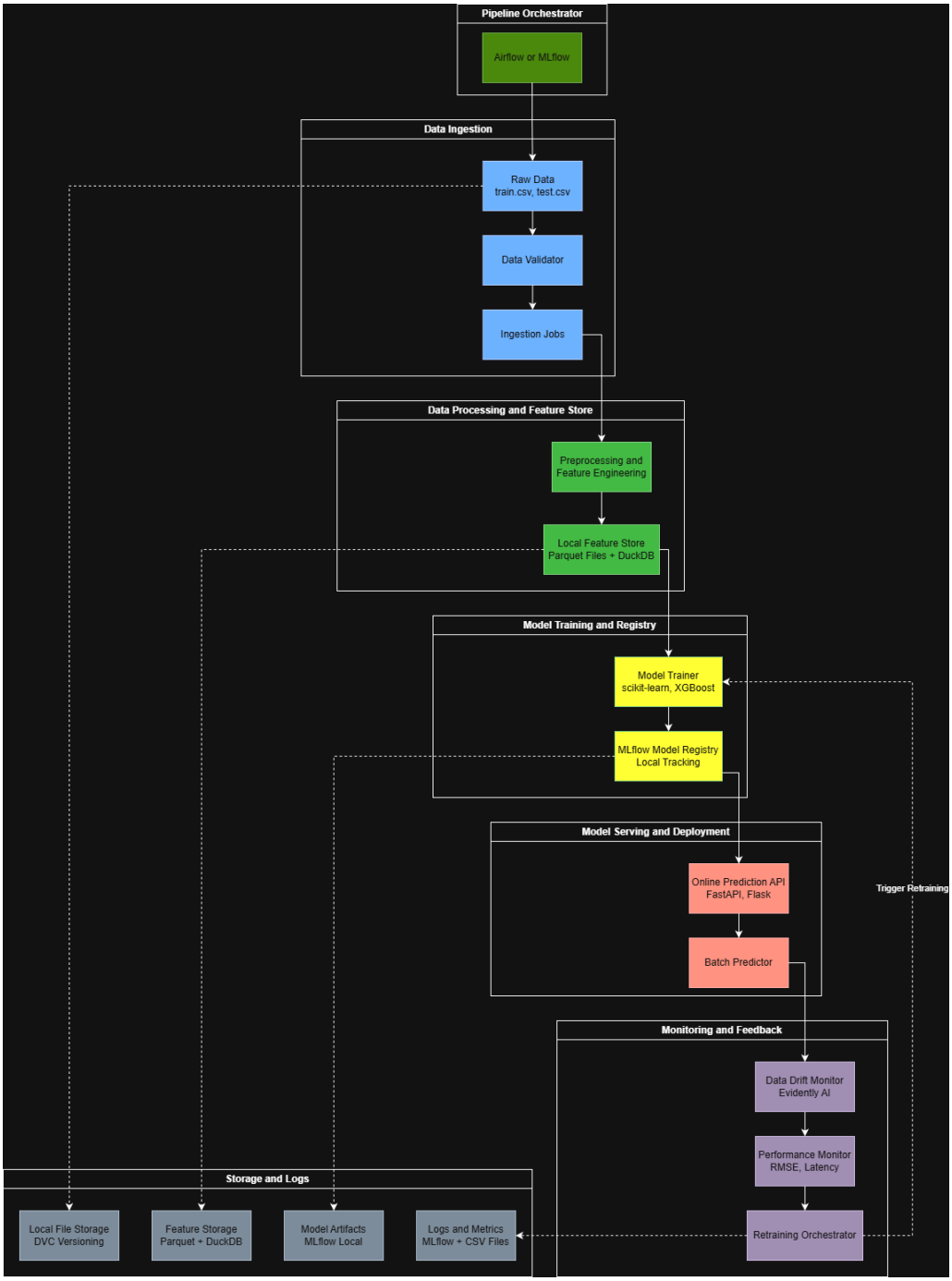

Figure 1: Updated Robust Architecture of the Housing Price Prediction System.

3 Risk and Quality Management

To maintain high reliability and minimize disruptions, a systematic risk and quality management plan was developed. The table below summarizes the main identified risks, their impact on the system, and the mitigation strategies adopted by the team.

Table 2: Risk Identification and Mitigation Strategies

Risk	Description	Impact	Mitigation Strategy	Monitoring Method
Data Loss	Corruption or deletion of the dataset during preprocessing or storage.	High	Implement backup copies in GitHub and maintain version control of all datasets.	Weekly data validation and checksum verification.
Model Drift	Decrease in prediction accuracy due to data distribution changes over time.	Medium	Integrate monitoring scripts to track RMSE variation and automatically trigger retraining.	Evaluate performance metrics after each training cycle.
Security Breach	Unauthorized access to project files or repository.	Medium	Apply least-privilege access policies and enable two-factor authentication in GitHub.	Periodic access log reviews.
Team Coordination Failure	Miscommunication or missed deadlines between sub-teams.	Medium	Maintain weekly Scrum meetings and Trello updates for task visibility.	Track sprint progress via Kanban and Gantt dashboards.
Implementation Error	Bugs or inconsistencies in preprocessing or model code.	Low	Conduct peer code reviews and automated testing using unit test scripts.	Continuous integration checks in GitHub Actions.

This risk management framework ensures that technical, organizational, and security issues are systematically controlled. All identified risks are monitored during each sprint, and corrective actions are logged as part of the team's continuous improvement cycle.

4 Project Schedule and Workflow

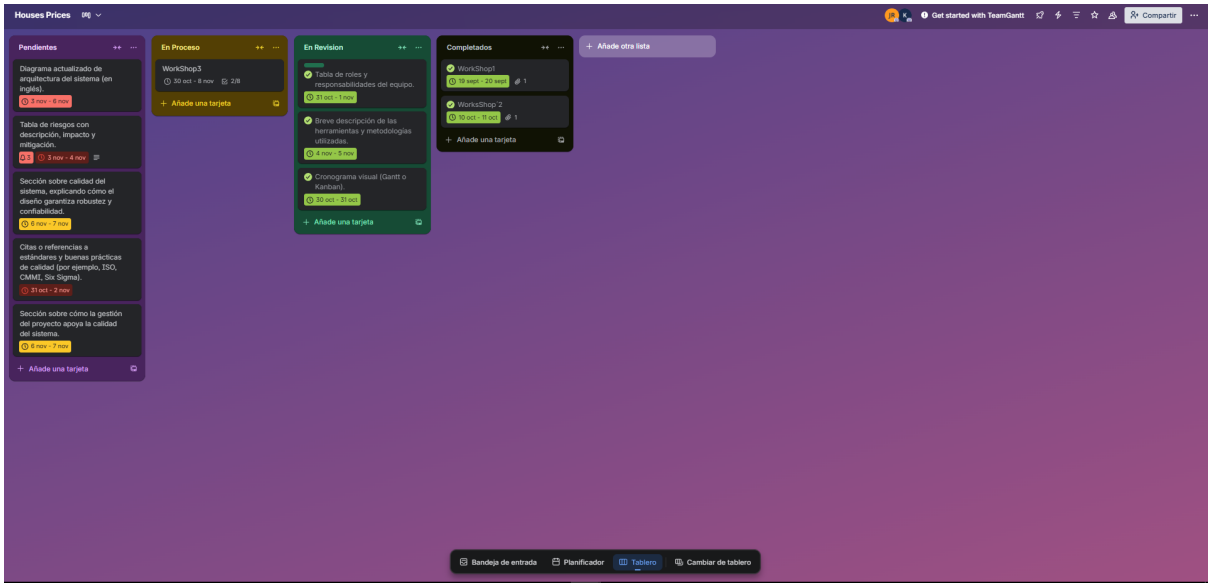

Figure 2: Kanban board used for organizing and tracking project tasks.

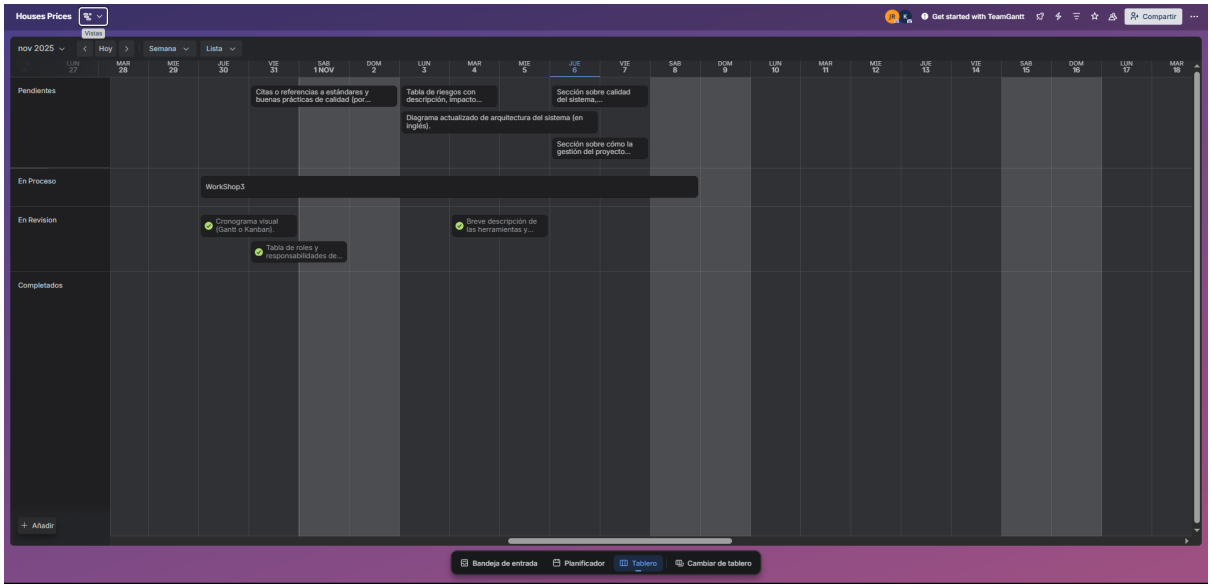

Figure 3: Gantt-style schedule showing task timelines and dependencies.

The project's progress is managed through a combination of **Kanban** and **Gantt** visualizations, both developed using Trello and TeamGantt tools.

The **Kanban board** (Figure 2) provides a clear overview of the status of each activity — from *Pending*, to *In Progress*, *Under Review*, and finally *Completed*. This structure allows real-time collaboration, easy tracking of team contributions, and effective prioritization of deliverables.

Meanwhile, the **Gantt schedule** (Figure 3) details the chronological distribution of tasks throughout November 2025. It defines clear deadlines and overlaps among deliverables such as:

- Updating the system architecture diagram.
- Preparing the risk table and quality section.
- Developing Workshop 3 and reviewing roles documentation.

This dual visualization approach combines the flexibility of agile methods with the clarity of timeline planning. It ensures continuous delivery, balanced workload distribution, and transparent coordination among all team members.

5 Tools, Methodologies, and Project Milestones

5.1 Application of Scrum in the Project

The development process is guided by the **Scrum methodology**, which enables iterative progress, teamwork, and flexibility in response to project evolution. Table 3 summarizes how each Scrum element is applied within this project.

Scrum Element	Application in the Project
Product Backlog	List of system functionalities and deliverables (data ingestion, preprocessing, modeling, monitoring, documentation).
Sprint	Each sprint corresponds to one week of work. Example: Week 1 – architecture and risk analysis; Week 2 – project management and quality assurance.
Daily Scrum	Short meetings or asynchronous updates via WhatsApp or GitHub messages to report progress and identify blockers.
Sprint Review	Presentation of progress by each sub-team (e.g., architecture, simulation, or management).
Sprint Retrospective	Group reflection after each delivery on what worked well and what can be improved in the next iteration.

Table 3: Application of Scrum methodology within the project.

5.2 Summary

This Scrum-based structure allows an efficient distribution of responsibilities while maintaining close collaboration between the technical and management teams. The **Product Owner** defines the direction of the project, the **Scrum Master** ensures coordination and sprint compliance, and the **Development Team** executes the technical tasks, ensuring the quality and reliability of the system. Together, the team achieves a robust, traceable, and adaptable system aligned with the objectives of Workshop 3.

5.3 Project Milestones and Deliverables

5.3.1 Milestone 1 — Workshop 3: System Integration and Quality

Objective: Finalize the system design documentation and ensure the quality of the proposed architecture. **Duration:** 1 week (short sprint)

Deliverables:

- Final IEEE-style document including:
  - General system diagram and modular architecture.
  - Reviewed functional and non-functional requirements.
  - Implementation plan and applied engineering principles.
- Team role assignments (Product Owner, Scrum Master, Developers).
- Initial GitHub repository structure organized by workshops.
- Milestone planning for Workshop 4.

Closing milestone: Version 1.0 of the IEEE document and validated system architecture.

5.3.2 Milestone 2 — Workshop 4: Kaggle System Simulation

Objective: Implement and analyze simulations of key processes within the designed architecture. **Duration:** 2 weeks

Main Activities:

1. **Sprint 1 – Preparation and Design of Simulations**
   - Prepare dataset (cleaning, reduction, exploratory analysis).
   - Define two simulation scenarios:
     a) *Data-driven Simulation:* train and evaluate a traditional ML model (e.g., Random Forest, XGBoost).
     b) *Event-based Simulation:* model a spatial or feedback-driven process using cellular automata or dynamic systems.

- Identify metrics, constraints, and success criteria for each simulation.
- Design the data flow (stock and flow diagram) for the simulation.

Sprint 1 Deliverables:

- Simulation planning document.
- Base code or initial pseudocode.
- Simulation flow diagram added to the repository.

2. **Sprint 2 – Implementation and Results**

- Implement both defined scenarios.
- Run simulations with varying parameters.
- Analyze results and identify chaotic or emergent patterns.
- Compare outputs of both simulations.
- Draft the *Simulation Report (PDF)* with graphs, logs, and discussion.

Sprint 2 Deliverables:

- `Workshop_4_Simulation` folder in GitHub including:
  - Simulation code.
  - `requirements.txt` or `environment.yaml` with dependencies.
  - Final report (PDF in English).
- Updated project `README.md` with summary and links.

Closing milestone: Functional simulation system and documentation delivered before November 29.

5.3.3 Final Milestone — Project Presentation

Objective: Consolidate results, insights, and future projections of the system. **Duration:** 1 week
Deliverables:

- Final system presentation (slides or live demo).
- Updated IEEE document including simulation outcomes.
- Final reflection: impact, sensitivity, and improvement opportunities.

5.4 Tools and Methodologies Used

For project management and development, an **agile approach based on Scrum** was applied, complemented by the collaborative tool **Trello** for task organization and tracking.

- **Scrum** enabled short, iterative sprints that facilitated prioritization, clear role assignment, and adaptive planning as the system evolved.
- The roles of *Product Owner*, *Scrum Master*, and *Development Team* promoted effective communication and fair distribution of responsibilities.
- **Trello** was used as a digital Kanban board to visualize the workflow through columns such as "To Do,In Progress,.ªnd Çompleted."This tool ensured task traceability, documentation of progress, and real-time collaboration between both teams.
- The combination of **Scrum + Trello** fostered organization, transparency, and incremental delivery of results, ensuring that each sprint contributed tangible value to the project and supported continuous system improvement.

6 Quality Assurance through Project Management

The project management structure was designed to ensure the **quality and reliability of the system** at every stage of its development cycle — from planning to implementation and validation of results. To achieve this, several mechanisms were established: version control, collaborative review, and technical verification, all applied continuously throughout the Scrum-defined sprints.

6.1 Version Control

Version control was implemented through **GitHub repositories**, enabling a detailed history of changes, identification of individual contributions, and prevention of code conflicts. Each team member worked on independent branches that were merged into the main repository only after passing peer reviews, ensuring both integrity and accountability.

6.2 Collaborative Review

Periodic reviews were conducted at the end of each sprint. During these sessions, the teams presented technical progress, updated documentation, and preliminary simulation results. These reviews functioned as key quality control checkpoints and provided opportunities for constructive feedback and continuous improvement.

6.3 Verification and Validation

Verification and validation activities were applied to each system module — ingestion, preprocessing, modeling, and monitoring. **Modular testing** ensured that each component performed as intended, while performance evaluation (using metrics such as RMSE) validated the predictive accuracy and system stability. This combination of verification techniques ensured that the final solution was both technically sound and functionally consistent.

6.4 Continuous Improvement and Traceability

By integrating software engineering best practices with continuous improvement principles, the management approach promoted **traceability**, **reproducibility**, and **high-quality results**. Every sprint cycle reinforced the system's reliability through structured testing, collaborative iteration, and transparent documentation, guaranteeing that the final deliverable met academic and technical excellence standards.

Quality assurance and risk management are integrated: metrics such as RMSE and data validation logs are continuously monitored to trigger corrective actions defined in the risk mitigation plan.

7 Incremental Improvements and Lessons Learned

Throughout the progression from Workshops 1 and 2 to this third phase, the project has evolved both technically and organizationally.

- From **Workshop 1**, the team identified key systemic factors and the sensitivity of the real-estate market data. This understanding motivated the design of a modular and adaptive architecture capable of handling uncertainty and chaos.
- During **Workshop 2**, the focus shifted to defining functional and non-functional requirements, resulting in a structured and well-documented data-to-decision pipeline. The experience highlighted the need for continuous monitoring and reproducibility.
- In **Workshop 3**, these insights were consolidated into a robust architecture supported by Agile project management practices. The integration of Scrum methodology, Git-based version control, and quality assurance mechanisms improved traceability, collaboration, and overall system reliability.

These incremental refinements demonstrate the team's commitment to continuous learning and process maturity. By combining robust system design with agile management, the project has achieved higher stability, adaptability, and long-term sustainability.

Finally, during this phase, the main challenge was balancing the technical depth of the system with agile project management requirements. Coordinating roles across development and documentation demanded strict adherence to Scrum principles and version control practices. These challenges strengthened the team's ability to work collaboratively and deliver a robust and verifiable system.