Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Real-Time People Flow Monitoring And Control Software With AI And Computer Vision

Loaiza-Barrios, Alfredo Gerardo[1] , Sañay-Sañay, Segundo Isael[2] , Jiménez-Rivas, Dora Elizabeth[3]

[1]universidad Catolica De Cuenca
Alfredo.Loaiza.46@Est.Ucacue.Edu.Ec
Https://Orcid.Org/0009-0009-8066-7422
[2]universidad Catolica De Cuenca
Ssanay@Ucacue.Edu.Ec
Https://Orcid.Org/0000-0003-4661-318x
[3]universidad Catolica De Cuenca
Djimenezr@Ucacue.Edu.Ec
Https://Orcid.Org/0000-0003-1644-3052

This document presents the development structure of a "real-time monitoring and control system for people flow using artificial intelligence and computer vision." Designed for a furniture company, this system enhances the efficiency, safety, and optimization of commercial spaces, surpassing manual counting methods. The project employs the FURPS+ model to thoroughly evaluate the requirements (Araujo Sandoval, 2020; Yungan Gualli et al., 2019) and follows the Unified Process for iterative development (Sumano, 2012; Hernández González, 2017). Technologies such as Python, OpenCV, PyTorch, and Laravel are essential for its implementation (Itseez, 2018; Paszke et al., 2019; Otwell, 2020). The project results show a minimal difference compared to manual counting, with an error margin between 0.97% and 22.39%. It supports each detection with images and segments by age and gender, providing valuable data for strategic decision-making.

Keywords: Artificial Intelligence, Computer Vision, FURPS+, Unified Process, Real-Time People Flow Monitoring and Control.

INTRODUCTION

The real-time monitoring of the flow of people is crucial for organizations at different levels, such as those dedicated to the manufacture and marketing of furniture. Traditional manual counting methods are inefficient due to their high cost, low accuracy, and slowness (Araujo Sandoval, 2020). Faced with these challenges, the implementation of a system based on artificial intelligence (AI) and computer vision

arises to improve the efficiency and safety of commercial spaces (Gómez Flores, 2020; Redmon & Farhadi, 2018).

The project seeks to develop an automated system that allows people to be detected, tracked and counted in real time using video, overcoming the limitations of manual methods and improving accuracy and scalability. Specific objectives include analyzing the current state of video systems, developing software based on deep neural networks, implementing it in a real-world environment, and validating its effectiveness.

The manual counting method presents issues such as lack of accuracy, high operating costs, and slow turnaround times, making it unfeasible for multiple locations. In contrast, the proposed automated system uses computer vision and neural networks to improve the accuracy of counting people, segmenting them by gender and age. This system reduces costs, provides real-time data and is scalable, allowing it to be deployed across multiple sites without additional complications.

Automating the process through AI and computer vision optimizes data collection, improves crowd management, and contributes to safety and efficiency in commercial spaces. This technology promises to transform operational processes, providing more accurate, efficient and scalable solutions for strategic decision-making in the company.

THEORETICAL FRAMEWORK

UNIFIED PROCESS (PU) development

The Unified Process (PU) is a framework for iteration-based software development, allowing for continuous adjustments and improving the quality of the final product. According to Sumano (2012), this approach reduces the risks associated with changes in requirements, facilitates early identification of problems, and ensures that development remains aligned with customer expectations.

One of the main advantages of PU, as Hernández González (2017) indicates, is its ability to manage complexity and change through short and manageable iterations. This allows for regular evaluations and necessary adjustments, improving the quality of the final product and increasing the satisfaction of both the client and the development team.

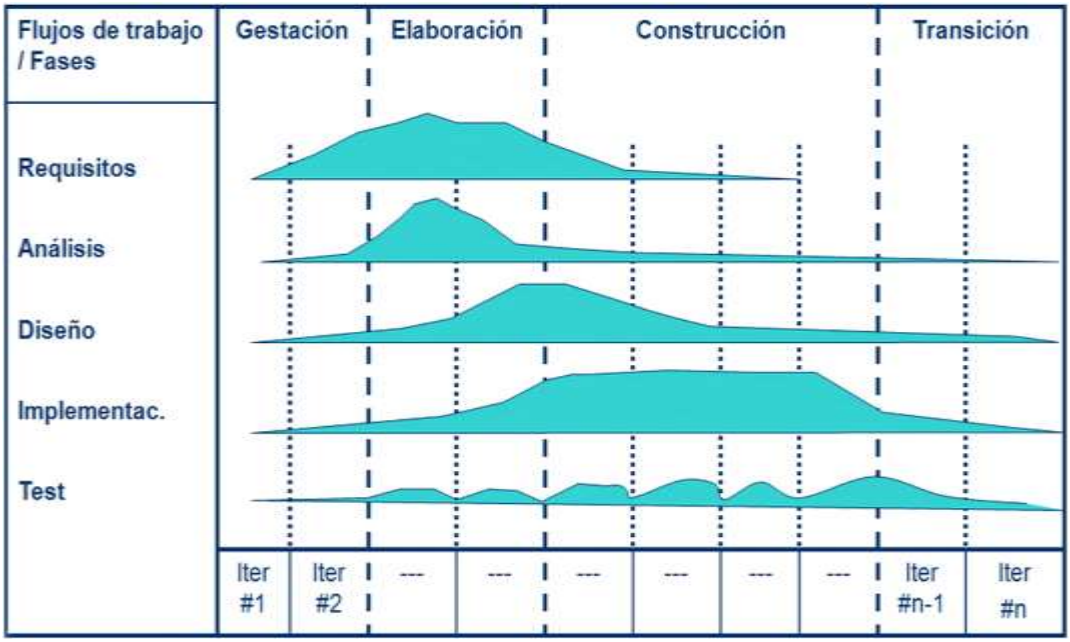

ILLUSTRATION 1. Santana Núñez, J. M. (2015). Unified Process Workflows: The graph provides a detailed visual representation of how the key activities of the Unified Process (PU)—including Requirements, Analysis, Design, Implementation, and Testing—are developed throughout its various phases.

As can be seen in Figure 1, the PU is structured in four phases: Initiation, Elaboration, Construction and Transition. In the Gestation phase, the scope of the project is defined, the main risks are identified and an overview of the system to be developed is drawn up. Sumano (2012) mentions that this phase is critical to establishing a solid foundation, ensuring that everyone involved has a clear understanding of the project's objectives and risks.

The Elaboration phase involves developing a base architecture and refining the project plan, focusing on mitigating the most critical risks previously identified (Hernández González, 2017). During the Construction phase, the entire system is developed through multiple iterations, adding functionality incrementally and performing tests to ensure that each component is working properly.

Finally, the Transition phase focuses on the delivery of the system to the end user, including activities such as installation, initial support, and user training (Sumano, 2012; Hernández González, 2017).

FURPS+

The FURPS+ model is an extension of the original FURPS model, used to classify and evaluate software requirements. As shown in Figure 2, each letter of the acronym represents a specific dimension of quality: Functionality, Usability, Reliability, Performance, and Support. The "+" extension adds additional considerations about maintainability, portability, reusability, and other factors. This model is widely used due to its ability to provide a detailed and structured assessment of the various aspects of a software, allowing developers and project managers to identify critical areas that need attention (Yungan Gualli et al., 2019).

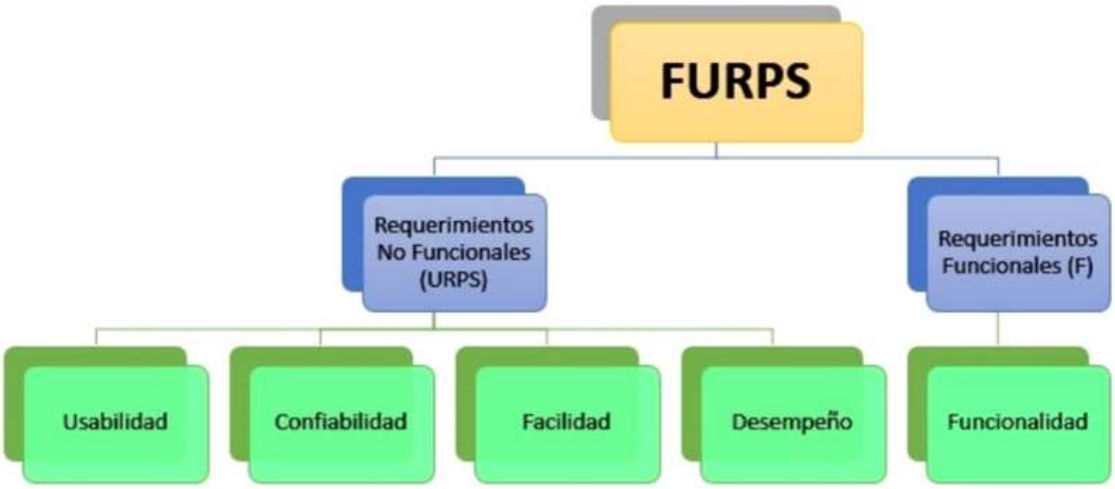

ILLUSTRATION 2. López Meneses, D. A. FURPS Models: The diagram illustrates the FURPS model, which is used to classify and organize software requirements into two main categories: Functional (F) and Non-Functional (URPS).

Implementing the FURPS+ model in software project management provides a clear guide to ensuring product quality. It allows identifying areas for improvement and ensures that the software meets the necessary standards to meet customer expectations (Araujo Sandoval, 2020).

- **Functionality** refers to the ability of software to meet the tasks and requirements for which it was designed, including aspects such as accuracy, interoperability, and security.
- **Usability** focuses on the ease with which users can learn how to use the software and achieve their goals, evaluating the simplicity of the interface, documentation, and technical support.
- **Reliability** measures the ability of the software to operate consistently and without failure, considering the failure rate, recoverability, and availability.

- **Performance** assesses how efficiently the software uses system resources and its ability to handle the workload, by measuring response time, memory usage, and processing capacity (Yungan Gualli et al., 2019).
- **Support** refers to the ease with which software can be maintained, updated, and improved, including documentation, technical assistance, and the availability of updates (Araujo Sandoval, 2020).

In addition to these five main dimensions, the "+" extension of the FURPS model includes other factors critical to software quality, such as maintainability, portability, and security, allowing for a more comprehensive and detailed evaluation of the software (Yungan Gualli et al., 2019).

ARTIFICIAL INTELLIGENCE (AI)

Artificial Intelligence (AI) refers to the ability of machines to mimic human cognitive functions, such as learning and decision-making. According to McCarthy (2021), AI is crucial in the development of applications to analyze real-time data and make decisions based on detected patterns. Machine learning algorithms, especially deep learning, make it possible to process large volumes of data and extract relevant information. AI is applied in various fields, such as health for medical diagnostics, in industry for process optimization, and in security for surveillance and fraud detection. In addition, AI improves user-software interaction through chatbots and virtual assistants, and optimizes business process management (Cordero Guzmán et al., 2020).

CONVOLUTIONAL NEURAL NETWORKS (CNN)

Convolutional Neural Networks (CNNs) are deep networks that are efficient in processing data with a grid structure, such as images. Gómez Flores (2020) highlights that CNNs learn hierarchical characteristics of images, facilitating the detection of patterns and improvements in areas such as facial recognition and image segmentation. Figure 3 shows how CNNs use convolution, pooling, and fully connected layers to process and classify images, enabling machine learning without the need for manual feature extraction.

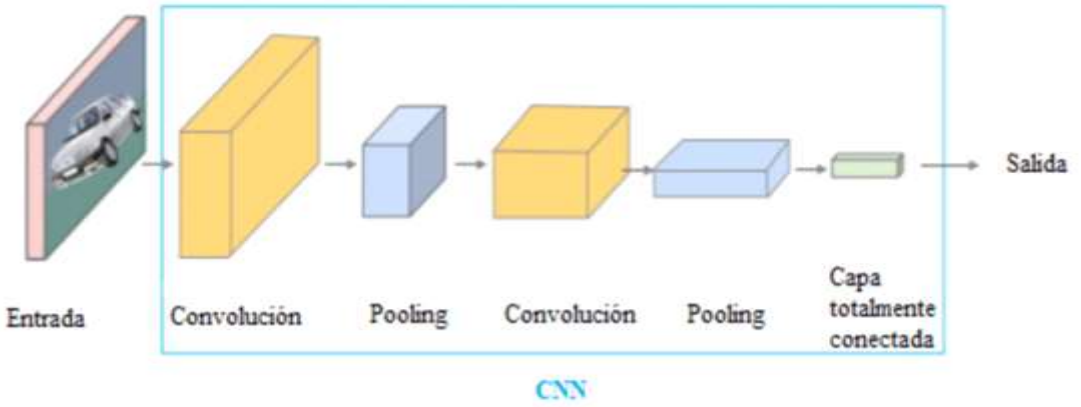

ILLUSTRATION 3. Diagram of the functioning of a convolutional neural network (CNN) Andrade, H. (2021). This illustration provides a detailed view of the flow of data through a convolutional neural network, highlighting the various stages [input, convolution, pooling, and fully connected layers to output] that make up the image analysis process.

YOLO (You Only Look Once)

Within the applications of Convolutional Neural Networks (CNNs), YOLO (You Only Look Once) stands out for its ability to detect objects in real time. According to Redmon and Farhadi (2018), YOLO strikes a balance between precision and speed, making it ideal for applications that require fast processing without sacrificing accuracy.

YOLO splits an image into a grid and simultaneously predicts bounding boxes and class probabilities for each cell in the grid. This methodology allows images to be processed more quickly than other approaches that analyze regions of interest separately. In addition, YOLO is effective at detecting objects at different scales and is robust in detecting small and large objects in the same image.

Thanks to its real-time performance, YOLO is ideal for various applications, such as surveillance systems, autonomous driving, and mobile applications.

COMPUTER VISION

Computer vision is an area of artificial intelligence that allows computers Computer vision is a branch of artificial intelligence that allows computers to interpret and analyze the visual world. De La Escalera Hueso (2020) highlights its use in real-time image analysis to recognize objects, detect movements and extract information from complete scenes.

Techniques such as image segmentation, edge detection, and feature extraction are essential to increase the accuracy of these systems, enabling object identification and classification, pattern recognition, and tracking changes in the environment.

OpenCV, an open-source library mentioned by Itseez (2018), facilitates real-time image and video processing and analysis, with common applications such as facial recognition, motion detection, and augmented reality. The integration of computer vision and artificial intelligence optimizes decision-making based on visual data, creating more robust and efficient systems (De La Escalera Hueso, 2020).

API REST

RESTful (Representational State Transfer) APIs are software architectures designed to create scalable and efficient web services. As shown in Figure 4, these APIs facilitate communication between systems using the protocol.

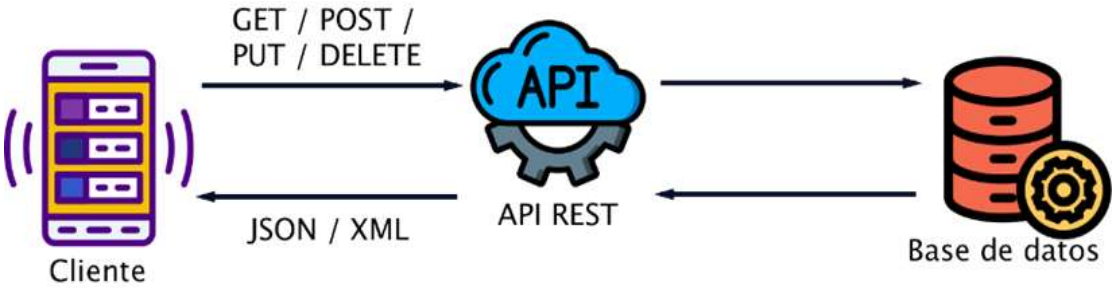

ILLUSTRATION 4. Dossetenta (2024). Description of a data analysis model: The illustration shows a data analysis model where a mobile application communicates with an API, which in turn interacts with a database.

According to Fielding (2000), RESTful APIs are based on CRUD (Create, Read, Update, Delete) operations executed through HTTP verbs (POST, GET, PUT, DELETE), which allows a standardized interaction between applications and facilitates data integration.

In modern application development, RESTful APIs are crucial for interoperability between systems and platforms. Frameworks such as Laravel are frequently used to implement these APIs due to their robustness and advanced capabilities to handle HTTP requests, authentication, and database management (Otwell, 2020).

Laravel offers a clear structure and integrated tools that simplify the development and maintenance of web applications, improving efficiency in application deployment (Campoverde et al., 2024).

RTSP PROTOCOL

The Real-Time Streaming Protocol (RTSP) is a network protocol that controls the transmission of multimedia data in real-time. According to Schulzrinne et al. (1998), RTSP allows clients to issue commands to start, stop, and pause media streaming from a server. It is critical for applications that require live streaming of audio and video, such as video conferencing and security cameras.

RTSP makes it easy to handle interactive media sessions using HTTP-like commands such as DESCRIBE, SETUP, PLAY, PAUSE and TEARDOWN, allowing precise control of the stream. The similarity to HTTP makes it easy to implement and use for developers familiar with this protocol (Fielding, 2000).

RTSP is used in a variety of devices and applications, from video surveillance systems to streaming platforms such as Netflix and YouTube, being especially useful in environments where latency and streaming quality are critical (Wright and Stevens, 2011).

Tools used in the construction of the software

Key tools employed in building the software include:

Laravel: A PHP framework that facilitates the development of robust web applications with an elegant syntax. Laravel manages tasks such as routing, authentication, and databases, and is ideal for building RESTful APIs. In addition, it includes Blade, a template system, and Eloquent ORM, which simplifies interactions with databases (Otwell, 2020).

Python: A versatile language used in software development, data science, and machine learning. Libraries such as NumPy, pandas, and scikit-learn enable rapid development of complex applications.

OpenCV: A library for computer vision used in image and video processing. It facilitates applications such as facial recognition and motion detection, being key in real-time visual analysis systems (Itseez, 2018).

PyTorch: A deep learning library that enables agile development of neural network models, used in computer vision and natural language processing (Paszke et al., 2019).

Software Architecture

Software architecture defines the structure of a system, including its components and their relationships. A good architecture provides a solid foundation for development, allowing for efficient and controlled changes (Bass et al., 2003). It is also essential to meet non-functional requirements such as scalability, security, and performance. Choosing the right architectural style, such as REST for Web services, can improve the flexibility and scalability of the system (Fielding, 2000).

Documenting the architecture is crucial for communicating design decisions to the entire development team (Bass et al., 2003). This includes diagrams, architectural patterns, and design decisions that clarify how system components should interact. Effective documentation ensures a common understanding of the system and facilitates its evolution and maintenance (Sañay et al., 2019).

Software Development Methodology or Process

In the software development, the Unified Process (PU) was used, an iterative approach that guides the team through different phases, allowing for continuous adjustments and improvements. This process was complemented by the FURPS+ model, which classifies and evaluates software requirements in terms of functionality, usability, reliability, performance, and support, ensuring compliance with quality standards and customer expectations.

The PU is divided into four main phases: Initiation, Elaboration, Construction and Transition. In the Initiation phase, the scope of the project is defined, risks are identified, and an overview of the system is developed. Key deliverables from this phase include a Vision Document and Requirements Specifications, based on the FURPS+ model.

In the Development phase, the base architecture of the system is developed and the project plan is refined, focusing on solving the most critical risks. Deliverables include the definition of the System Architecture, Interface Prototypes and a detailed Iteration Plan. During the Construction phase, the complete system is developed through iterations, with deliverables such as the Source Code, Technical Documentation and a User Manual.

Finally, in the Transition phase, the system is delivered to the end user, ensuring its installation, initial support and training. The main deliverables of this phase are the Implemented System, Acceptance Testing, and End-User Training to ensure efficient use of the software.

PROCESS DELIVERABLES

Initiation Phase

Requirements Analysis applying the FURPS+ model

- **Functionality (F):**
    - **CU 001:** The system must count people in real time using IP cameras, with video capture and processing using OpenCV and deep learning models in PyTorch. The data is stored in a PostgreSQL database.

- **CU 002:** Segmentation by gender and age using PyTorch (gender) and Keras (age) models.
- **CU 003:** Identification of employees through a graphical interface, with storage of information in the database.
- **CU 004:** Secure storage of all detections, with integration of the Backend in Laravel and REST APIs.
- **CU 005:** Configuration of IP cameras using a graphical interface to define input and output lines.
- **CU 006:** Real-time display of detections through a graphical interface.
- **CU 007:** Intuitive graphical interface developed in Tkinter for configuration and visualization.

- **Usability (U):** Intuitive interface with clear graphic elements, immediate feedback, and guided configuration of cameras and detection lines.
- **Reliability (R):** Secure storage in PostgreSQL, automatic error recovery mechanisms in IP cameras.
- **Performance (P):** Real-time processing with optimized models in PyTorch and OpenCV, scalability for multiple IP cameras.
- **Support(S):** Complete documentation, technical support, and easy system maintenance.
- **Additional Requirements (+):** Integration with other systems through RESTful APIs.

Use Cases and Sequence Diagram

The "Real-Time People Flow Monitoring and Control" (MCFTR) system uses IP cameras, artificial intelligence, and computer vision to monitor people as shown in Figure 5. The actors include IP cameras, the camera manager, physical security, backend and database.

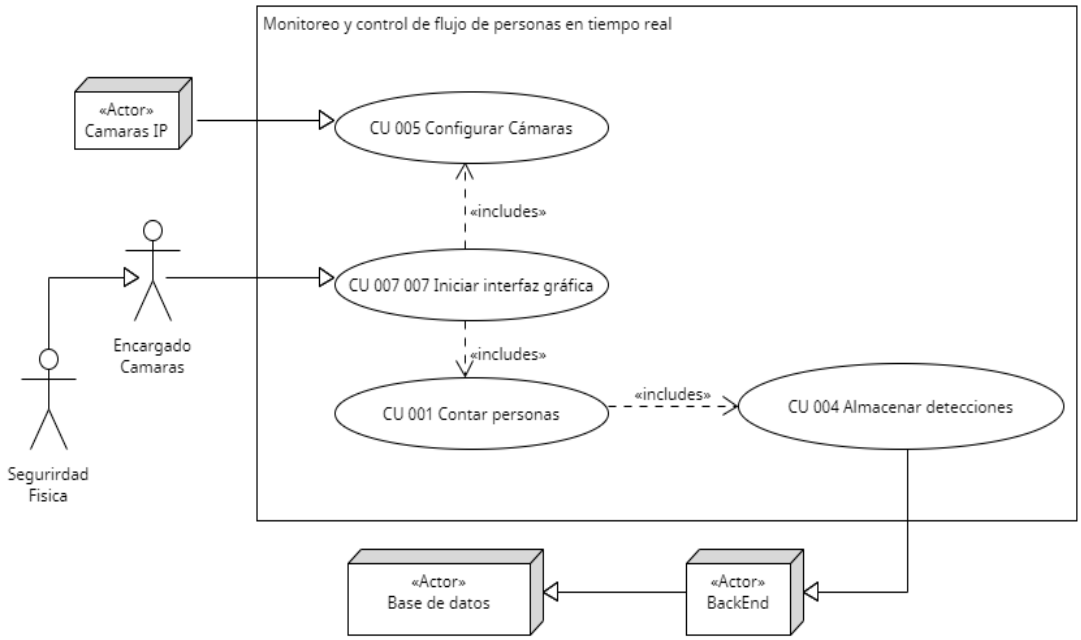

Figure 5. Use Case Diagram: The use case diagram provides an overview of how different actors interact with the functionalities of the real-time people flow monitoring and control system.

- **Camera Configuration (CU 005). -** Add, remove and adjust cameras and detection lines.
- **People Counting (CU 001). -** Video capture and processing to detect and count people.
- **Detection Storage (CU 004). -** Backend centralizes data storage and management in PostgreSQL.
- **Gender and Age Detection (CU 002). -** Segmentation of detected people.
- **Employee Identification (CU 003). -** Manual marking and storage of information.
- **Detection Display (CU 006). -** Graphical interface shows detections and allows real-time analysis.

Sequence Diagram

The sequence diagram shown in Figure 6 describes the flow of interaction between the camera manager, the IP cameras, the main PC, the people counter, the backend, and the database. It highlights the stages of setup, video capture, processing, storage,

and visualization, showing how each component interacts to ensure accurate, real-time people counting.

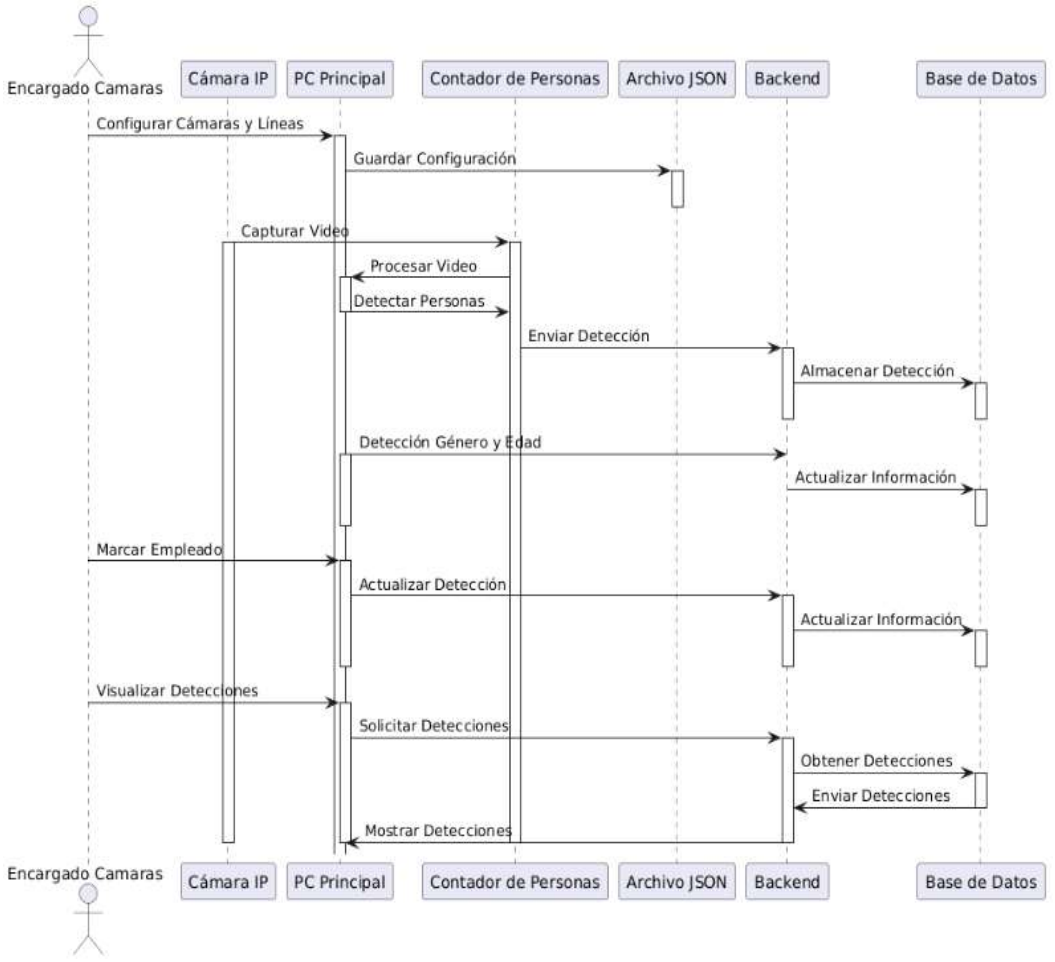

Figure 6. Sequence Diagram: The sequence diagram shows the interaction between the Camera Manager, IP Camera, Main PC, People Counter, JSON File, Backend, and Database during camera setup, video capture, people processing and detection, detecting and updating detections, and viewing detections.

PRODUCTION PHASE

The MCFTR uses IP cameras distributed in multiple locations that connect to a network using the RTSP protocol, sending real-time video to a central server. This server hosts the counting and analysis applications, processing user requests and managing the PostgreSQL database, which stores detections and configurations. Users

interact with the system through a graphical interface, where they set up cameras, view real-time statistics, and analyze stored detections.

System Architecture

During this phase, a base architecture was developed and project plans were refined, focusing on mitigating the most critical risks identified above. The system architecture was designed both logically and physically to ensure an understanding of how the components interact and how they are implemented in the infrastructure.

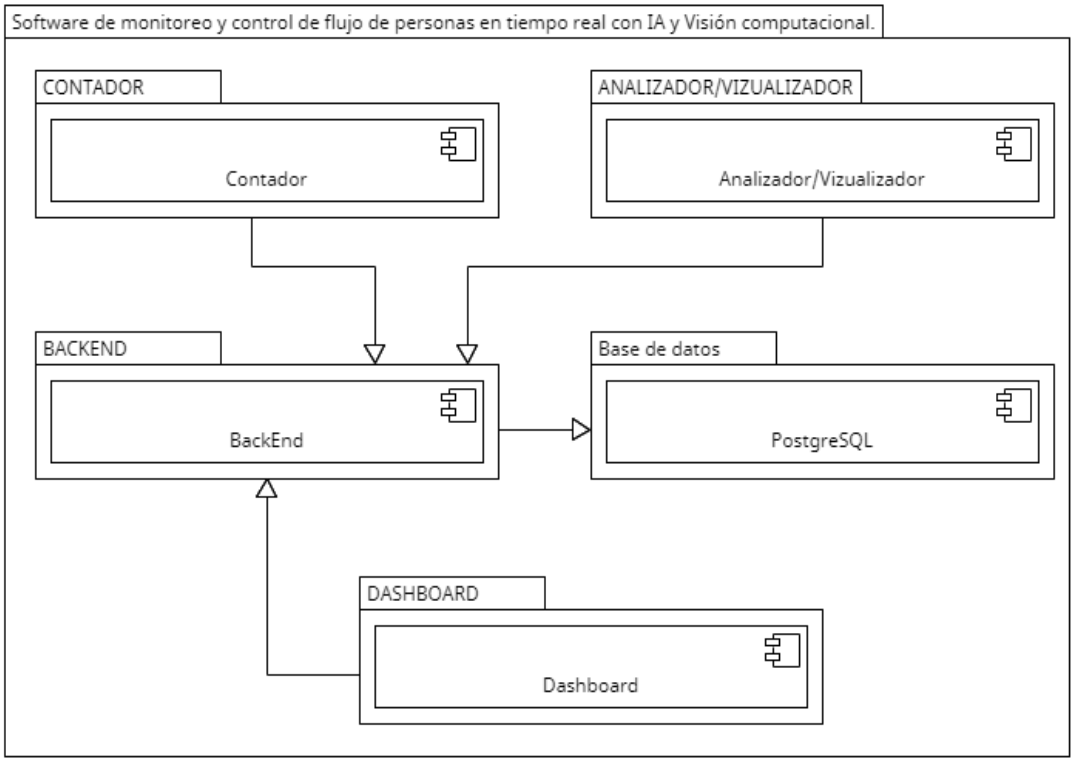

ILLUSTRATION 7. Real-time people flow monitoring and control software with AI and computer vision. The integration diagram shows the interactions between the system components: Counter, Analyzer/Visualizer, BackEnd, Dashboard and PostgreSQL Database. The Counter, Analyzer/Visualizer and Dashboard send and receive data through the BackEnd, which in turn communicates with the Database to store and retrieve information.

The logical architecture (Figure 7) details the main modules: the Backend, the People Counter, the Detection Analyzer and Viewer, and the Dashboard. Each module has specific and clearly defined roles to ensure efficient data flow. The backend acts as the core of the system, handling HTTP requests, providing RESTful endpoints, handling

business logic, and facilitating interaction with the PostgreSQL database. The Counter captures and processes video in real-time, using computer vision and deep learning technologies (OpenCV and PyTorch) to detect and count people. The Analyzer and Viewer predicts the gender and age of detected individuals using PyTorch and Keras models, and allows visualization and adjustment of detections. The Dashboard allows users to visualize statistics and make detailed queries about the flow of people, using data provided by the backend.

The physical architecture (Figure 11) shows how the system is implemented in terms of infrastructure: servers, databases, and network connections. This architecture is critical to ensure that the system is scalable and capable of handling the expected load of data and requests.

Key Features

- **Backend -** Handles HTTP requests and provides RESTful endpoints for CRUD operations, implementing business logic and facilitating interaction with the database.
- **People Counter. -** Captures and processes video to detect people in real time, using OpenCV and PyTorch.
- **Analyzer and Visualizer. -** Analyzes detections to predict gender and age, providing a graphical interface for data visualization and adjustment.
- **Dashboard -** View aggregated statistics and generate reports based on stored data, interacting with the backend for up-to-date information.

This integration ensures that the system operates efficiently and consistently, providing real-time monitoring and analysis of the flow of people in the facility.

CONSTRUCTION PHASE

System Module Development

People Counter. - The People Counter module is responsible for capturing and processing real-time video to detect and count people using advanced computer vision and deep learning techniques. Figure 8 presents its class diagram, which includes the main classes: MultiVideoPersonCounter, PersonCounter, VideoStreamProcessor, and GUIUpdateThread.

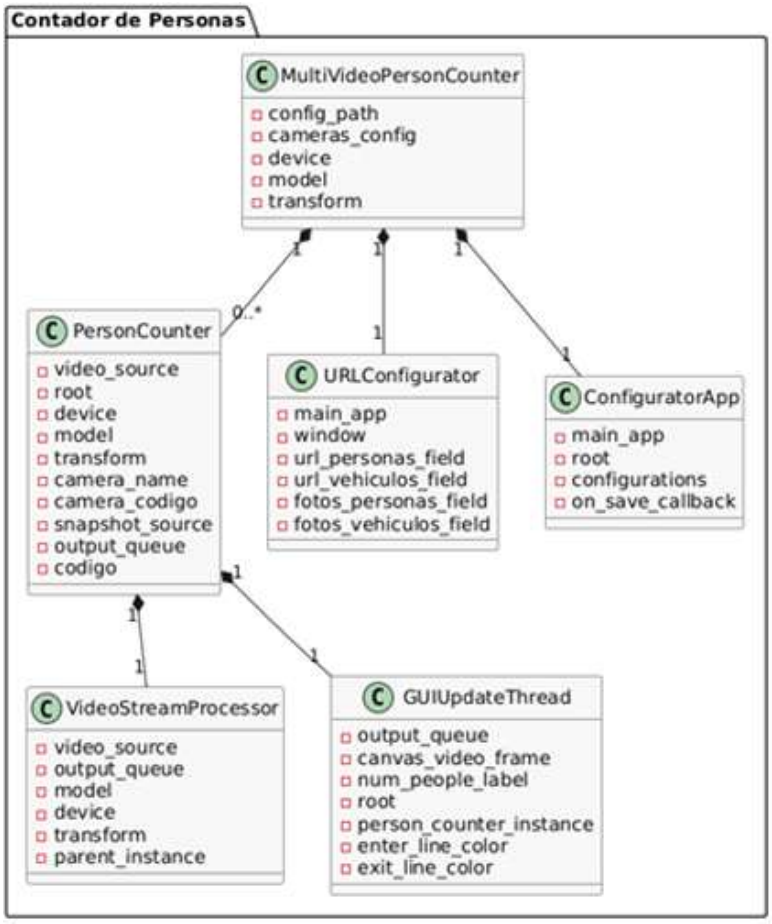

Figure 8. Counter Class Diagram: The class diagram shows the relationships between the main classes of the people counting system, including MultiVideoPersonCounter, PersonCounter, URLConfigurator, ConfiguratorApp, VideoStreamProcessor, and GUIUpdateThread, along with their key attributes.

The modules of the Person Counter subsystem are described below:

- **MultiVideoPersonCounter. -** Manages multiple IP cameras, initializing the YOLO model for person detection. The model is loaded with your specific settings and weights to ensure fast and accurate detections.
- **PersonCounter. -** Captures real-time video and processes each frame using YOLO to detect people, applying non-maximum suppression to eliminate redundant detections.

- **VideoStreamProcessor. -** Processes the video stream in real-time on a separate thread, uses YOLO for object detection, and sends the detections to an output queue for further processing.
- **GUIUpdateThread. -** Updates the graphical interface with real-time detections, displaying the results in the GUI.

Detection Analyzer and Viewer

The Detection Analyzer and Viewer module analyzes detections to predict gender and age, using deep learning models. Figure 9 shows its class diagram, highlighting the MainApp, VisualizerApp, and AnalyzerApp classes.

- **MainApp:** Manages the graphical interface, allowing users to switch between display and analysis modes.
- **AnalyzerApp:** Manage analysis logic, predicting gender and age with a pre-trained image classification pipeline.
- **VisualizerApp:** Manages the display of detections, allowing filters based on criteria such as date, category, age, and location.

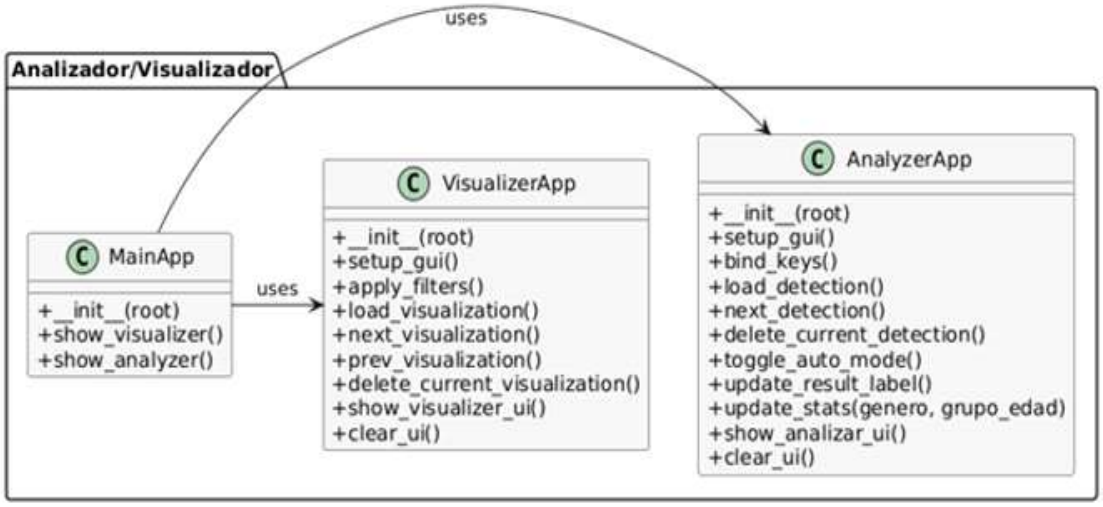

Figure 9. Analyzer/Viewer Class Diagram: The class diagram shows the relationships between MainApp, VisualizerApp, and AnalyzerApp, including the methods used for data visualization and analysis.

Backend and Dashboard

The backend handles HTTP requests, provides RESTful endpoints, and handles business logic. Figure 10 details the class diagram, which includes WebRoutes, DetectionController, Detection, and DashController.

- **Detection. -** Data model for detections, including attributes such as ID, name, date, gender, and age.
- **DetectionController. -** Handles HTTP requests related to detections, allowing CRUD operations.
- **WebRoutes. -** Defines the web routes to handle detection CRUD operations.
- **DashController. -** Manages dashboard logic, providing methods for visualizing statistics and loading specific data.

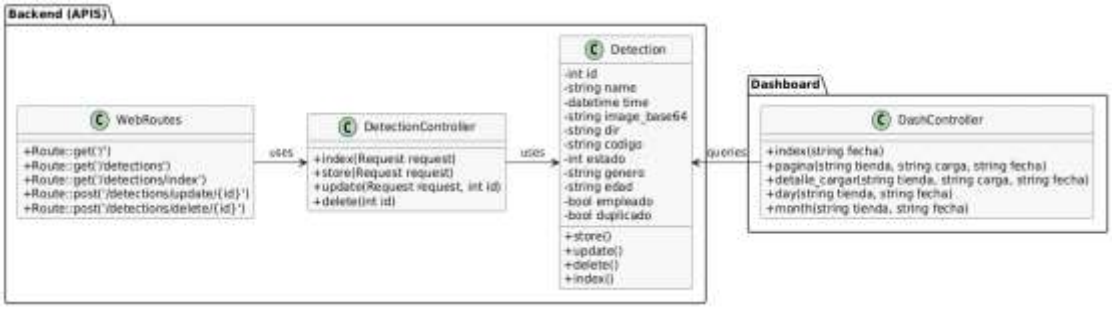

Figure 10. Backend Class Diagram (APIs): The class diagram shows the structure of the backend with the WebRoutes, DetectionController, Detection, and DashController classes, highlighting the relationships and methods used to manage routes, detections, and queries to the dashboard.

TRANSITION PHASE

System Architecture Implementation

The system architecture has been implemented with several key components that interact to deliver people counting and analytics capabilities:

- **IP cameras:** Connected to the local network using the SRTP protocol, they capture video in real time in different locations and send the data to the Main PC.
- **Main PC:** Houses the counting, analysis and visualization programs. It processes the video received from the cameras, performing detections and analysis in real time, and generating the necessary data for its visualization.

- **Central Server:** Hosted on a dedicated infrastructure, it contains the backend developed in Laravel and the PostgreSQL database. It handles HTTP requests, stores detections, and provides RESTful endpoints for interaction with other components.
- **PC User**: Allows users to set up cameras, use counting and analysis applications, and connect to the central server to view and analyze stored detections.
- **Network:** Authorized users can access the dashboard from any allowed network to view statistics and perform additional analysis.

This infrastructure ensures efficient coordination between components to process, store, and analyze data in real time.

How the System Works

The system consists of two main modules: **Counter** and **Analyzer/Display**.

- **Counter -** Captures real-time video, detects people, and updates data on the server. The URLs of the server and the cameras that will capture the flow of people are configured, defining the entry and exit lines.
- **Analyzer/Viewer. -** Loads unanalyzed images, predicts gender and age, and allows manual adjustments. Provides an interface for reviewing and modifying detection data.

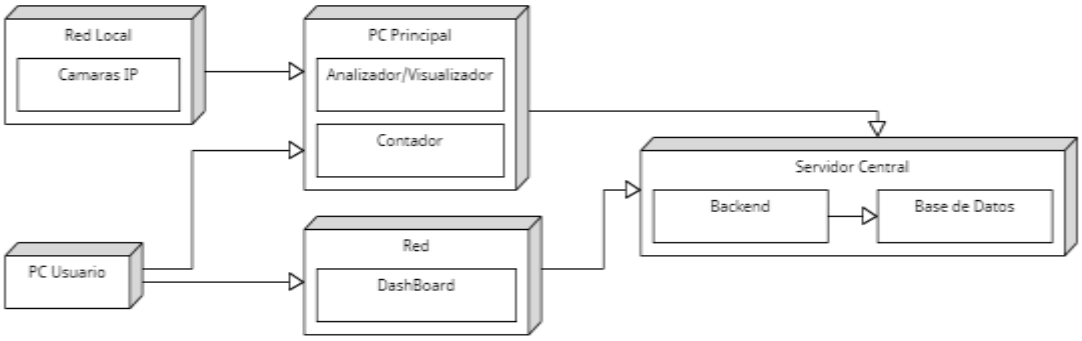

Figure 11. Physical level architecture: The physical diagram shows the structure of the system, highlighting the interconnection between the IP Cameras, the Main PC (with analysis and counting programs), the Central Server (with backend server and database), the User's PC and the Dashboard in the network.

Figure 11 shows the structure of the system, from video capture to data visualization and analysis. Figure 12 details the integration of the modules, showing how they relate

to each other to maintain an efficient flow of information and real-time data processing.

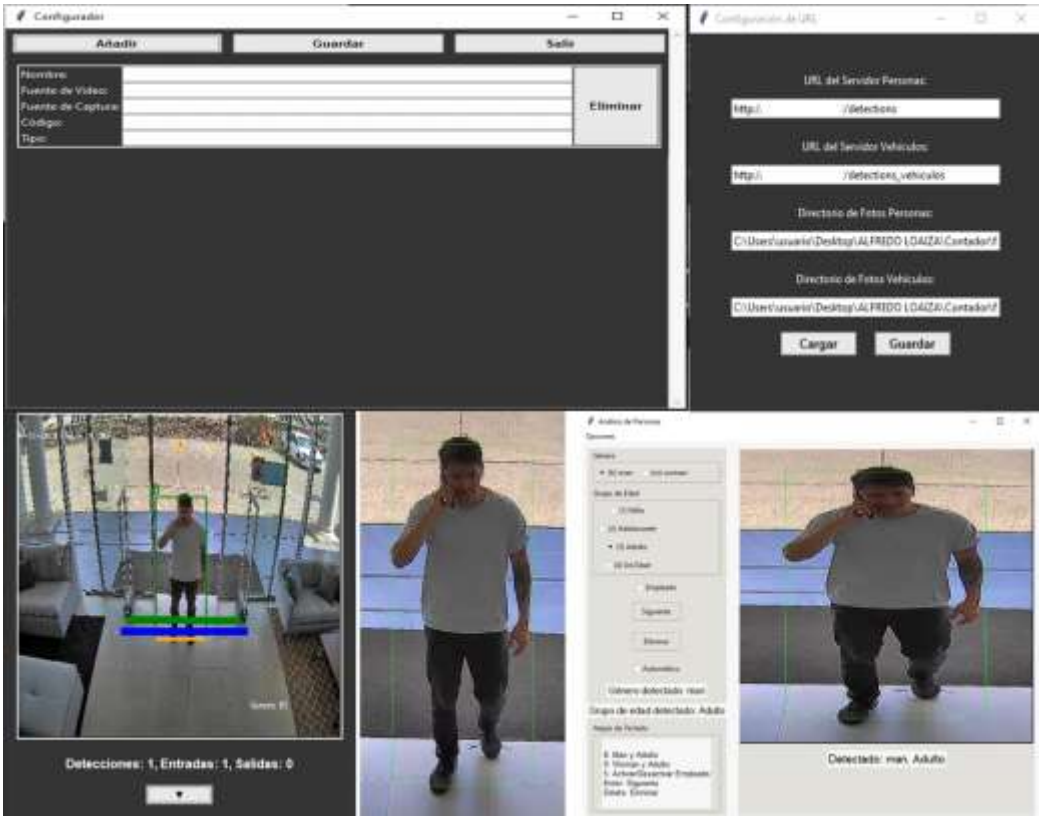

Figure 12. System Operation: Shows the integration and relationship between the main modules of the people counting system.

The Central Server stores the detections in the database and provides APIs for interaction with other modules. The Analyzer/Visualizer uses this information to predict demographic data, ensuring accurate monitoring of the flow of people.

Validation and Testing

To validate the operation of the MCFTR, a final test was carried out in a real production environment of 5 days, during the liquidation of the company's products. Cameras were installed in strategic locations to assess the accuracy and effectiveness of the system:

- **Main Door. -** Captured the flow of people in and out.
- **Rear Door. -** Monitored the flow of people through the secondary entrance.

- **Parking. -** Counted the number of vehicles entering the parking lot.

Comparison Methods

Several methods were used to validate the data from the automated system, including:

- **Manual Tailgate Counting. -** Performed by employees for direct reference, although it only provided a grand total without segmenting by gender, age, or including employees.
- **Previous System on the Main and Rear Doors. -** Comparison to the previous system, which differentiated between men and women, and employees, but did not provide age ranges and only counted adults.
- **Infrared Counter at the Front Door. -** Used as an additional measure, without differentiation between entrances and exits, gender, age, or employee identification.

Comparison Results

The results showed that Project Contador's automated system offered greater accuracy in counting people and vehicles, thanks to advanced artificial intelligence and computer vision technologies.

Specific Comparisons

- **Contador Project vs. Previous System**
    - The previous system presented a significant margin of error, with up to 100 people unaccounted for compared to Project Contador.
    - The differences were due to incomplete segmentation and lack of image backup in the previous system, while Project Contador uses advanced algorithms and each detection is backed by an image.
- **Contador vs. Contador Project Infrared Counter:**
    - The infrared counter showed a significant margin of error due to the lack of differentiation between inputs, outputs, gender, and age.
    - The differences were attributed to errors in detection and lack of specificity in the total count.
- **Project Counter vs. Manual Counting:**
    - The manual count provided a direct reference to assess the accuracy of the system.

- By excluding employees and segmenting by age, a more accurate comparison was achieved, highlighting the reliability of Project Contador.

Analysis of Results

The detailed analysis focused on comparison with the data from the manual count as it was considered the most reliable method:

- **Reliability. -** Manual counting is considered a standard method for validating new automated counting systems.
- **Limitations of Other Methods. -**
  - The above system does not segment by age and is not as accurate as Project Accountant.
  - The infrared counter does not differentiate between inputs and outputs, gender or age.

TITLE	**CONTADOR PROJECT**			**MANUAL COUNTING**
DAY	**Men**	**Women**	**Total**	**Total**
WEDNESDAY	145	176	321	294
THURSDAY	163	154	317	259
FRIDAY	132	140	272	299
SATURDAY	236	231	467	502
SUNDAY	259	260	519	514

Table 1. Results of the Accountant and Personal Counting Project

Day	**Total, Project Accountant**	**Total, Manual**	**Error Rate (%)**
Wednesday	321	294	9.18
Thursday	317	259	22.39
Friday	272	299	9.03
Saturday	467	502	6.98
Sunday	519	514	0.97

Table 2. Percentage of Error Between Project Counter and Personal Count

- **Margin of difference with Manual Counting. -** The margin of difference with Manual Counting is almost minimal, which demonstrates the high degree of certainty of the Counter Project. By excluding employees and segmenting

by age, the data obtained is comparable to manual counting, thus validating the accuracy of the system.

Day	Absolute Difference (Total)	Error Rate (%)
Wednesday	27	9.18
Thursday	58	22.39
Friday	27	9.03
Saturday	35	6.98
Sunday	5	0.97

- **Table 3.** Difference Between Project Counter and Manual Counting

- **Data Backup with Images**: One of the biggest advantages of Project Contador is that each detection is backed up with an image. This makes it possible to verify and corroborate that no errors have been made in the detections, offering greater confidence in the data obtained. It's not just about numbers; Each piece of data is visually documented, ensuring that no detection is a false piece of data.

Day	Total, Detections	Imaging Detections (%)
Wednesday	321	100
Thursday	317	100
Friday	272	100
Saturday	467	100
Sunday	519	100

- **Table 4.** Data Backup with Images

- **Segmentation and Differentiation**: Unlike the previous system and the infrared meter, the Meter Project provides detailed segmentation by gender and age. Not only does this improve the accuracy of the count, but it also offers more valuable data for strategic decision-making.

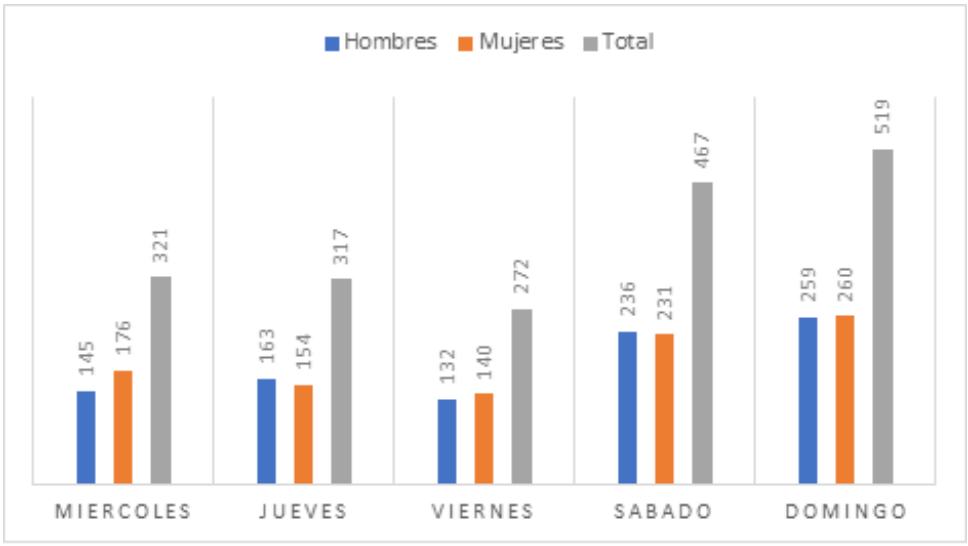

Figure 14. System Results: Graph of detection data segmented by gender.

The MCFTR is established as the best option for monitoring and controlling the flow of people in real time. Its high accuracy, backed by images, and its ability to segment and differentiate between different demographic groups make it superior to previous methods and other automated systems. This ensures that decisions based on Project Contador data are informed and reliable, improving efficiency and safety in commercial spaces.

CONCLUSIONS

Initial Assessment: Analysis of real-time video systems based on neural networks revealed shortcomings in traditional methods of counting people, identifying key areas for improvement. The implementation of advanced deep learning models, such as YOLO, was instrumental in increasing accuracy and efficiency in detecting and tracking people in real time.

Software Development: A robust system was developed that integrates computer vision and deep learning technologies for the detection and counting of people. The software consists of specialized modules such as the Counter, Backend, Dashboard and Analyzer/Visualizer, which operate in a coordinated manner to offer a comprehensive and efficient solution. Each module was designed and optimized for its specific function, ensuring a continuous flow of information and constant system operability.

Implementation in Real Environments: Implementation in a real operating environment allowed validating the functionality and effectiveness of the system. Tests conducted in furniture stores showed that the system can operate in real-world

conditions, capturing and processing real-time data with a high level of accuracy. The system's ability to handle multiple video sources and adapt to various configurations highlighted its scalability and flexibility.

Validation and Results: Extensive testing confirmed that the developed software meets the set objectives. Accuracy in detecting and counting people and in data analysis and visualization were positively evaluated. The feedback obtained allowed adjustments and optimizations to be made, ensuring optimal performance and high system reliability.

GRATITUDE

Sincere thanks are expressed to the company for providing the opportunity to develop and test the real-time people counting system in a real operating environment. Their constant support and collaboration were essential to the success of this project.

Special thanks to the IT team, whose commitment, experience and willingness during all phases of implementation and testing proved invaluable. Their dedication and effort contributed significantly to achieving the objectives set, ensuring the development of a robust and effective solution.

Thank you for your trust in this project and for your valuable contribution to the advancement of technology in the company.

REFERENCES

1. Andrade, H. (2021). Description of the operation of a convolutional neural network (CNN) [Diagram]. ResearchGate. https://www.researchgate.net/profile/Hugo-Andrade-14/publication/348825166/figure/fig1/AS:984814450667520@1611809470042/Figura-1-Descripcion-del-funcionamiento-de-una-red-neuronal-convolucional-CNN-10.png
2. Araujo Sandoval, O. I. (2020). FURPS model applied to the quality analysis of a software developed with Sencha Ext JS. REDDI Unlam. Retrieved from http://reddi.unlam.edu.ar
3. Bass, L., Clements, P., & Kazman, R. (2003). Software architecture in practice. Addison-Wesley.
4. Baugher, M., McGrew, D., Naslund, M., Carrara, E., & Norrman, K. (2004). The Secure Real-time Transport Protocol (SRTP). IETF. Retrieved from https://datatracker.ietf.org/doc/html/rfc3711
5. Campoverde-Calle, M. S., Sañay-Sañay, S. I., & Cabrera-Duffaut, A. E. (2024). Development of PPP validation methodology based on the technological acceptance model: "BodyUC" case study. MQR Inquire, 8(1), 3742–3770. https://doi.org/10.56048/MQR20225.8.1.2024.3742-3770
6. Cordero Guzmán, D. M., & Sañay, I. S. (2020). Business Process Management (BPM) Framework: Case of a Service Company. UPSE Scientific and Technological Journal (RCTU), 7(1), 43-53. https://doi.org/10.26423/rctu.v7i1.509
7. De la escalera Hueso, A. (2020). Computer vision. University of Malaga.

8. Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures (Doctoral dissertation, University of California, Irvine). Retrieved from https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
9. Gómez Flores, W. (2020). Convolutional Neural Network. National Polytechnic Institute.
10. Grinberg, M. (2018). Flask web development: Developing web applications with Python. O'Reilly Media.
11. Hernández González, A. (2017). Application of the unified development process to software projects. ResearchGate. Retrieved from https://www.researchgate.net/profile/AnaisaHernandezGonzalez/publication/312656269_Aplicacion_del_Proceso_Unificado_de_Desarrollo_a_proyectos_de_software/links/58878a87a6fdcc6b791ec281/Aplicacion_del_Proceso_Unificado_de_Desarrollo_a_proyectos_de_software.pdf
12. Itseez. (2018). Learning OpenCV 3: Computer vision with Python. Retrieved from https://repository.unikom.ac.id/67052/1/Learning%20OpenCV%203%20Computer%20Vision%20with%20Python%20%28%20PDFDrive.com%20%29.pdf
13. McCarthy, J. (2021). What is artificial intelligence? Open University of Catalonia. Retrieved from https://openaccess.uoc.edu/bitstream/10609/148039/3/QueEsLaInteligenciaArtificial.pdf
14. Otwell, T. (2020). Laravel: Up & Running. Retrieved from https://profmatiasgarcia.com.ar/uploads/tutoriales/Laravel-Clase1.pdf
15. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. Retrieved from https://www.cs.princeton.edu/courses/archive/fall19/cos484/lectures/pytorch.pdf
16. Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. University of Washington. Retrieved from https://arxiv.org/abs/1804.02767
17. Santana Núñez, J. M. (2015). Unified Process Workflows [Diagram]. ResearchGate. https://www.researchgate.net/profile/Jose-Miguel-Santana-Nunez/publication/279751761/figure/fig1/AS:648590296903681@1531647385501/Flujos-de-trabajo-del-proceso-unificado.png
18. Sañay, I., Becerra Molina, E., & Calle Masache, O. (2019). Software engineering for industrial development in zone 6 of the Austro. RECIMUNDO, 3(1), 1625-1643. https://doi.org/10.26820/recimundo/3. (1).January.2019.1625-1643
19. Schulzrinne, H., Rao, A., & Lanphier, R. (1998). Real-Time Streaming Protocol (RTSP). RFC 2326. Internet Engineering Task Force (IETF).
20. Sumano, A. (2012). Unified Process. Universidad Veracruzana.
21. Wright, G. R., & Stevens, W. R. (2011). TCP/IP illustrated, Volume 2: The implementation. Addison-Wesley.
22. Yungan Gualli, A. F., Morales Alarcón, C. H., Delgado Altamirano, J. E., & Espinoza Tinoco, L. M. (2019). FURPS model for the performance analysis of JSF frameworks. 3C ICT. Development Notebooks Applied to ICTs, 8(4), 6583. https://doi.org/10.17993/3ctic.2019.84.6583