People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

M'Hamed Bougara University – Boumerdès



Faculty of Sciences

Department of Computer Science

Field: Mathematics and Computer Science

Sector: Computer Science

Specialty: Computer Systems (Systèmes Informatiqes SI)

Academic Year: 2025 / 2026

Thesis for obtaining the degree of Bachelor's Degree

Theme:

" Creation of a dashboard for monitoring key account operations "

Presented by:

- Baouni Amna.
- Cheklat Sara.
- Zeggane Imene.

Defended on: 02/06/ 2026, Before the jury composed of:

- **Examiner:** Imam Abderrahmane.
- **Supervisor:** Imache Rabah.

Acknowledgements:

First and formest, we would like to express our deepest gratitude to الله تعالى (Allah the Almighty) for granting us the strength, patience, perseverance to successfully complete this work.

We would like to sincerely thank our supervisor, ***Mr.IMACHE Rabah***, for their continuous guidance, valuable advice, and support throughout the development of this project.

We also extend our appreciation to the members of ***Algérie Poste*** for welcoming us during our internship and for providing a professional and supportive environment that greatly contributed to the success of this project.

Our sincere thanks also go to all the professors and teachers of the ***Computer Science Department, Faculty of Science at M'Hamed Bougara University*** for the knowledge, encouragement, and academic training they provided during our years of study.

Finally, we would like to express our heartfelt gratitude to our families, friends, and everyone who supported and motivated us throughout this journey.

Dedications:

I dedicate this work to my beloved parents and siblings, whose endless love, sacrifices, and encouragement have always been my source of strength.

To my mother, sister and brother, who always stayed by my side and checked on me during my endless late nights. Your love, care, and support gave me the strength to keep going. I love you, and would not be where I am today without you.

Zeggane Imene

With great love and gratitude, I dedicate this work to my family and friends, who stood by my side through every challenge and supported me with patience, prayers, and unconditional love.

To my brother who helped me from another continent while living in France, Despite the distance, your guidance and constant presence meant so much to me throughout this journey.

Cheklat Sara

First and always, praise belongs to Allah — who was with me when no one else was, who opened every door I stood helplessly in front of, and whose mercy carried me to this moment.

To my parents — for being my safe place. Thank you for holding me up on the days I could barely hold myself. You never stopped believing in me, even when I forgot to believe in myself. Thank you for reminding me, over and over again, that it is all worth it.

To myself — for staying. Even when it was hard. Especially then. *Baouni Amna*

Table of contents :

I. Chapter1 : Project Context and Motivation 5
I.1 Introduction : 5
I.2 FIELD OF STUDY : 5
I.2.1 About the Host Organization : 5
I.2.2 Information Systems in Organizations : 5
I.2.3 Operational Monitoring and Performance Indicators : 6
I.2.4 KPIs Defined for This Project : 6
I.2.5 Relation of the Project to the Organization : 7
I.3 Study of the existing system : 8
I.3.1 Actors involved : 8
Internal Actors : 8
External Actors : 8
I.3.2 Current functioning of the system : 8
I.3.3 Data Security : 8
I.3.4 Existing data and databases : 9
I.3.5 Limitations of the current system : 9
I.4 Management Rules : 10
I.5 Organizational chart : 10
I.6 Data flow diagram : 11
I.7 The issue to be fixed (what to do?) : 11
I.7.1 Why? : 11
I.7.2 How? : 11
I.8 Dashboard : 12
I.8.1 Definition : 12
I.8.2 Advantages : 12
I.8.3 Types : 12
I.8.4 Dashboard Components : 13
I.8.5 Development process : 13
I.8.5.1 Requirement Analysis : 13
I.8.5.2 Data Collection and Integration 14
I.8.5.3 Data Processing and Storage 14
I.8.5.4 Dashboard Design : 14
I.8.5.5 Development and Implementation 14
I.8.5.6 Testing and Validation : 14

I.8.5.7 Deployment and Maintenance : 14
I.9 Conclusion : 14
II. Chapter 2 : Analysis and Design 15
II.1 Introduction : 15
II.2 Project Objective : 15
II.3 System Description : 15
II.4 Identification of Actors and Their Tasks : 15
II.5 Representative Diagrams : 16
II.5.1 Use Case Diagram : 16
II.5.2 Sequence Diagrams : 19
II.5.3 Class Diagram : 20
II.6 Conclusion : 21
III. Chapter 3 : Application Implementation 22
III.1 Introduction : 22
III.2 Development Environment : 22
III.2.1 Hardware Environment : 22
III.2.2 Software Environment : 22
III.2.3 Rationale for the Laravel Framework : 23
III.3 Application Architecture : 24
III.3.1 MVC Architectural Pattern : 24
III.4 Implementation of Interfaces and Functionalities : 25
III.4.1 Authentication Interface : 25
III.4.2 Main Dashboard : 26
III.4.3 Dynamic Filtering System : 27
III.4.4 Regional Analysis : 28
III.4.5 Performance Optimisation : 28
III.5 Application Security : 29
III.5.1 Data Protection : 29
III.5.2 Access Management : 29
III.6 Challenges Encountered and Solutions Applied : 30
III.6.1 Handling Large Volumes of Operational Data : 30
III.6.2 Chart Synchronisation Across Multiple Panels : 30
III.6.3 Complex KPI Calculations with PostgreSQL : 31
III.6.4 Dynamic Filter Interaction with SQL Aggregations : 31
III.6.5 Frontend Responsiveness and Chart Reflow : 31
III.7 Conclusion : 32

I. Chapter1 : Project Context and Motivation

I.1 Introduction :

In the era of digital transformation, information systems play a crucial role in improving organizational performance and supporting decision-making. Algérie Poste, as a key provider of postal and financial services in Algeria, relies on these systems to manage its daily operations.

However, the existing system may present limitations such as lack of data visibility and difficulty in monitoring activities efficiently. This report aims to analyze the current system, identify its weaknesses, and highlight the main issues affecting its performance.

Based on this analysis, a dashboard-based solution is proposed to enhance data monitoring, improve analysis, and support better decision-making.

I.2 FIELD OF STUDY :

Nowadays, most organizations rely heavily on information technologies to manage their daily activities and services. Every day, institutions generate a large amount of data from transactions, customer services, and administrative tasks. Managing this data efficiently is essential for understanding how the organization is performing and for helping managers make good decisions.(Laudon &Laudon,2018 ;O'Brien, J., & Marakas,2011)

Institutions that provide financial and postal services, like Algérie Poste, have a particularly high volume of operations. Monitoring these activities is not easy because of the number of transactions and the many offices involved. This is why information systems are very important—they help organize the data and provide clear information about what is happening in the organization(Turban et al.,2014).

I.2.1 About the Host Organization :

The internship takes place at Algérie Poste, one of Algeria's major public institutions. The company provides postal, financial, and digital services to millions of users nationwide. It has a wide network of post offices that handle postal deliveries, account management, financial transactions, and payment services.

Given the large number of daily operations, the institution needs effective tools to monitor activities, track performance, and make sure services run smoothly. Reliable information systems allow managers to have an overall view of operations and help detect problems before they affect customers (Laudon &Laudon,2018).

I.2.2 Information Systems in Organizations :

An Information System (IS) is a combination of hardware, software, data, users, and procedures that together collect, process, store, and distribute information. IS help organizations improve efficiency, coordinate activities between departments, and give managers the information they need to make decisions(Laudon &Laudon,2018).

At Algérie Poste, information systems are essential for following the operations in different offices. They allow managers to see what is happening in each branch and to compare performance across offices. This makes supervision easier and supports better management of daily tasks.

I.2.3 Operational Monitoring and Performance Indicators :

Operational monitoring is the process of tracking the activities of an organization to ensure everything works as planned. In large institutions with multiple offices, monitoring is crucial to maintain quality and efficiency (Turban et al.,2014).

One of the main tools used in monitoring is performance indicators (KPIs). KPIs are measurable values that help managers see how well activities are performed. For example, in post offices, KPIs might include the number of transactions processed daily, the speed of service, or the accuracy of operations.

By looking at these indicators, managers can quickly identify areas that need improvement and make better decisions. Operational monitoring combined with KPIs ensures that services are reliable and efficient for customers (Turban et al.,2014).

I.2.4 KPIs Defined for This Project :

For this project, the following KPIs have been defined to monitor Key Account operations:

No.	KPI Name	Description	JS Function	Backend Function	Calculation
1	Total Opérations	Total count of all transactions processed in the selected period and filters.	`fetchTotalOps()`	`totalOperations()`	`COUNT(operations.id)`
2	Montant Total (DA)	Sum of monetary amounts for all filtered operations, displayed in Algerian Dinar.	`fetchTotalOps()`	`totalOperations()`	`SUM(operations.amount)`
3	Taux de Succès	Percentage of operations that completed with status SUCCESS out of all operations.	`fetchTotalOps()`	`totalOperations()` × 2 calls	`(COUNT WHERE status='SUCCESS' / COUNT(*)) × 100`
4	Agent du Mois	Name of the top-performing agent ranked by highest operation volume in the period.	`fetchAgents()`	`agentRanking()`	`COUNT(DISTINCT ops.id) ORDER BY volume DESC LIMIT 1`
5	Volume par Service	Number of operations per service category — rendered as a pie/doughnut chart.	`fetchVolumeByService()`	`volumeByService()`	`COUNT(*) GROUP BY services.name`
6	Montant par Service	Total financial amount per service, ordered descending — rendered as a bar chart.	`fetchAmountByService()`	`amountByService()`	`SUM(operations.amount) GROUP BY services.name`
7	Produit Dominant	The single product generating the highest total amount; shown as a highlighted label.	`fetchProduct()`	`amountByProduct()`	`SUM(amount) GROUP BY products.name ORDER BY total DESC LIMIT 1`
8	Montant Total Produits	Combined total amount across all products linked to filtered operations.	`fetchProduct()`	`amountByProduct()`	`SUM(amount) — all products combined`
9	Classement Agents	Full agent ranking table: volume, amount, and success rate per agent for the period.	`fetchAgentPerformance()`	`agentRanking()`	`COUNT(DISTINCT ops.id), SUM(amount), ROUND(100×SUM(SUCCESS)/COUNT(*),1)`
10	Rentabilité par Service	Gross profit per service applying the active margin rate at each operation's date.	`fetchProfitabilityByService()`	`profitabilityByService()`	`SUM(amount × COALESCE(pm.margin%, svc.margin%) / 100)`

No.	KPI Name	Description	JS Function	Backend Function	Calculation
11	Rentabilité par Wilaya	Aggregated profit per wilaya, joining operations through établissements → UPW → wilaya.	`fetchProfitabilityByWilaya()`	`profitabilityByWilaya()`	`SUM(amount × COALESCE(pm.margin%, svc.margin%) / 100) GROUP BY wilayas.nom`
12	Volume par Bureau	Top-20 postal bureaux ranked by operation count — rendered as a horizontal bar chart.	`fetchVolumeByPostOffice()`	`volumeByPostOffice()`	`COUNT(*) GROUP BY etablissements.nom ORDER BY total DESC LIMIT 20`
13	Volume par UPW	Top-10 UPW entities ranked by operation count — used in a bar chart.	`fetchVolumeByUpw()`	`volumeByUpw()`	`COUNT(*) GROUP BY upws.upw ORDER BY total DESC LIMIT 10`
14	Évolution des Opérations	Time-series of operation counts, switchable between hourly, monthly and yearly views.	`fetchEvolution()`	`evolution()` hourly/monthly/yearly	`COUNT(*) GROUP BY EXTRACT(HOUR) / DATE_TRUNC('month') / EXTRACT(YEAR)`
15	Dernières Opérations	Paginated table of the most recent transactions with agent, bureau, amount and status.	`fetchLatestOperations()`	`latestOperations()`	`SELECT … ORDER BY creation_date DESC LIMIT n OFFSET o`
16	Évolution YoY par UPW	Year-over-year percentage change in total amount per UPW for a chosen year vs. previous.	`fetchYoyByUpw(year)`	`yoyByUpw()`	`((SUM(amount, yr N) − SUM(amount, yr N−1)) / SUM(amount, yr N−1)) × 100`
17	Top 3 Établissements par Service	Three highest-volume bureaux for a selected service, loaded on demand via button click.	`fetchTopEstablishments()`	`topEstablishmentsByService()`	`COUNT(*) GROUP BY etablissements.nom ORDER BY total DESC LIMIT 3`

Table 1.1- KPIs defined for this project.

I.2.5 Relation of the Project to the Organization :

The project focuses on monitoring the operations of large accounts at post offices. Large accounts involve many transactions, and their management is important for the institution's performance. Developing a system to track these operations helps managers understand trends, detect anomalies, and improve decision-making (Davenport & Harris,2007).

This project directly supports Algérie Poste by providing practical tools that help managers supervise operations more efficiently, improve service quality, and make informed decisions based on accurate data.

Monitoring operational activities and using performance indicators are key for managing modern organizations. Institutions like Algérie Poste rely on information systems to supervise daily operations, improve efficiency, and support decision-making (Laudon &Laudon,2018; Turban et al.,2014).

Studying these concepts gives the foundation needed to develop systems that help managers monitor activities, analyze data, and ensure that services are performed at a high standard. This Field of Study directly connects the project to practical organizational needs.

I.3 Study of the existing system :

The purpose of this study is to analyse the current functioning of the system, identify the actors involved, understand how operations are processed, and highlight the limitations of the existing solution.

I.3.1 Actors involved :

Actors are the people, organizations, and systems that participate in Key Account operations. Internal actors work inside Algérie Poste; external actors are outside.

Several actors participate in the functioning of the system:

Internal Actors : represent the human and organizational entities within Algérie Poste:

· Administrateurs systèmes (IT): Maintain system infrastructure, ensure security, and manage reference data including Wilayas, Établissements, and partner service configurations within the dashboard system.

External Actors : are entities outside Algérie Poste's organizational boundaries:

· API Partenaire (Partner API): The technical API endpoints exposed by partner companies (Algérie Télécom, Mobilis, Djezzy, Ooredoo, ADE, SEAAL, Sonelgaz, ONEFD, Cetelem, Société Générale, Al Baraka Bank). These APIs provide the transaction data that the dashboard system ingests and analyzes.

I.3.2 Current functioning of the system :

_When a customer pays a bill at the post office, the agent uses a computer application that talks to the partner company's system via the internet (API). The partner confirms the payment and sends back a response.

The system operates through an integration between the information system of partner companies and the Algérie Poste platform. The process follows these steps:

1.A transaction is initiated (by a client at a guichet, online via E-CCP, or via other remote services).

2.The Algérie Poste platform (URGO) sends a request to the partner company's API.

3.The partner company receives and processes the request.

4.The requested operation is executed by the partner company.

5.The partner company returns a JSON response to Algérie Poste with transaction status and metadata.

6.The operation is recorded in the Algérie Poste database with information such as company, service type, amount, and date.

7.Administrators can consult the stored data to monitor system activity.

I.3.3 Data Security :

Simple Explanation: All transaction data sent to partners is encrypted during transmission. Algérie Poste authenticates itself to partners, and only authorized personnel can access internal logs.

All communication between Algérie Poste and partner companies occurs via API (Application Programming Interface) connections over the internet. To protect transaction data during transmission, the following security measures are implemented:

.**Encryption :** All API communications use HTTPS (Hypertext Transfer Protocol Secure) with TLS 1.2 or higher encryption. This ensures that data is encrypted while traveling between the URGO system and the partner's servers, protecting it from interception or tampering.

.**Authentication** : Each partner API requires authentication before accepting any request. Algérie Poste authenticates itself using Bearer tokens or API keys. This ensures that only the authorized URGO system can send transaction requests to the partner's API endpoint.

· **One-Way Data Flow :** Algérie Poste only sends transaction data to partners. It does not receive or store sensitive data from partner systems. Only the JSON confirmation response (containing the transaction status and ID) is logged for reconciliation purposes.

· **Internal Access Control** : Within Algérie Poste, access to transaction logs and system configuration is restricted to authorized personnel only, specifically the Administrateurs systèmes (IT) within the DISR (Direction de l'Informatique et de la Sécurité Réseau) and the DSIE (Direction de la Sécurité Interne de l'Entreprise) (Algérie Poste, 2024a).

I.3.4 Existing data and databases :

The system relies on several databases containing different types of informations, these databases include:

- **Geographical data:** Wilayas, Dairas, Communes.
- **Postal structure data:** establishments and postal units (UPWs).
- **Partner company data:** companies connected to the platform (services).
- **Service data:** types of services offered (bill payment, recharge, …).
- **Transaction data:** operations performed through the system, including date, amount, and service.

I.3.5 Limitations of the current system :

Although the system processes a large volume of operations, several limitations can be observed.

Currently there is no centralized dashboard that allows administrators to easily monitor the activity of major companies. The available data is stored in different databases and is not presented in a way that facilitates analysis and decision-making(Sharda et al., 2018).

Therefore, it is difficult to quickly obtain informations like:

- The number of operations performed by each company.
- The total transaction amounts.
- The distribution of operations by service type.
- System performance indicators.

These limitations lead us to develop a monitoring solution that enables administrators to visualize key performance indicators and monitor major account activities in real time (Few.2006; Eckerson, 2010).

I.4 Management Rules :

Business rules (also called business rules) define or constrain some aspect of the business. They are intended to assert business structure or to control or influence the behaviour of the business (Business Rules Group, 2000).

Management rules describe what is allowed, what is required, and how processes and data must behave within an organization.

In Algérie Poste, we find:

1. Each wilaya contains several dairas, but each daira belongs to one and only one wilaya.
2. Each daira contains several communes, but a commune belongs to one and only one daira.
3. Each commune contains at least one establishment.
4. Each establishment belongs to one and only one commune.
5. Each wilaya has one and only one UPW (Unite Postale Wilaya), except for Algiers, where we have 3 UPWs: Algiers East, Algiers West and Algiers Center.
6. Each company has its own API connection with Algerie Poste.
7. A company can perform multiple operations.
8. Each operation is performed by one and only one company.
9. Each operation has a date, amount, and service type.
10. A service can generate multiple operations.
11. Each operation must be linked to one and only one specific service.

I.5 Organizational chart :

An organizational chart, or org chart, graphically represents an organization's structure by detailing the roles, responsibilities, and relationships between individuals within the organization (investopedia).

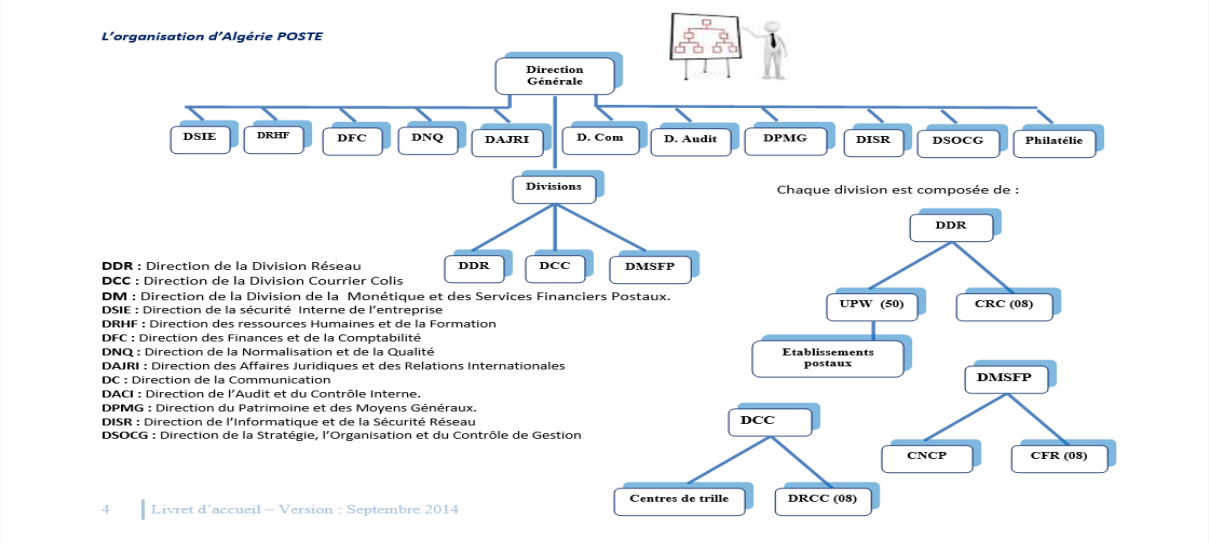
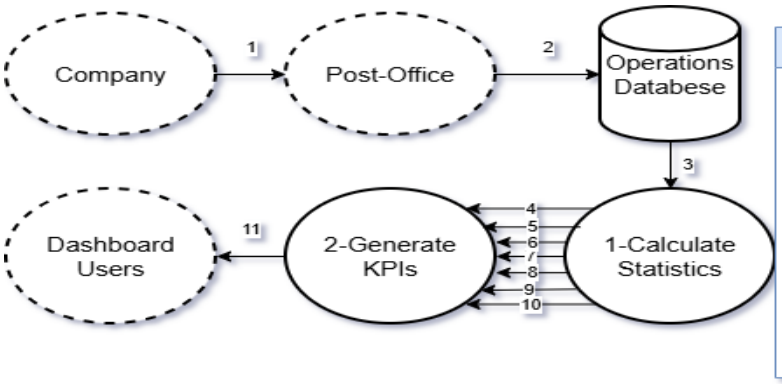


Figure 1.1- Algérie Poste's organizational chart.

The system we are studying is part of the service **DISR**: **Direction de l'Informatique et de la Sécurité Résau** (Information Technology and Network Security Department).

I.6 Data flow diagram :

A data flow diagram (DFD) is a graphical representation of the flow of data through a system, modelling its processes, data stores, and external entities (DeMarco, 1979).



Flux Number	Signification
1	-Transaction.
2	-Operation data.
3	-Informations.
4	-Number and amount of transactions per service
5	-Profitability.
6	-Performance by agents
7	-Agent ranking.
8	-Total achieved by UPW compared to last year.
9	-Top 3 establishments by service.
10	-Breakdown of amount by product.
11	-Dashboard reports.

Figure 1.2- Data flow diagram.

I.7 The issue to be fixed (what to do?) :

Algérie Poste collaborates with several major partners such as Djezzy, AADL,… .These companies preform different types of operations through Algérie Poste services, such as:

- Payment of telecom bills.
- Mobile credit recharge (Flexy).
- Other service payments.

These companies integrate their systems with Algérie Poste using APIs (Application Programming Interface) which means each partner maintains its own database and information system(Laudon &Laudon,2018).

The objective of this project is to **collect and centerlize the data related to these operations in order to build a dashboard that allows monitoring and analysis of major accounts operations**.

I.7.1 Why? :

Since each company has its own API and information system, the data related to operations are distributed across multiple systems. This situation makes it difficult for Algerie Poste to:

- Monitor operations in real time.
- Analyse the volume and evolution of transactions.
- Track performance indicators for major accounts.
- Generate consolidated reports for decision-making.

A centralized monitoring system would therefor allow better **control, analysis, and management of operations**(Sharda et al., 2018; Turban et al.,2017).

I.7.2 How? :

This objective can be achieved by developing a **dashboard application** that collects and centralizes operational data from different systems through APIs.

The application will store the data in a **centralized database** and provide a dashboard that allows users to:

- Visualize key performance indicators (KPIs).
- Monitor operations by partner, post office, and time period.
- Analyse trends and transaction volumes.
- Generate reports that supports decision-making.

The system will be developed using REST APIs, LARAVEL framework, and PostgreSQL database (Sharda et al., 2018; Turban et al.,2017).

I.8 Dashboard :

I.8.1 Definition :

A dashboard is a visual tool that presents key organizational data in a clear and structured format on a single screen. It consolidates Key Performance Indicators (KPIs) and displays them through interactive visualizations, such as charts and tables, enabling quick monitoring of performance and detailed operations. Dashboards integrate data from multiple sources and can be customized for different departments, supporting real-time operational and strategic decision-making (Few.2006; Laudon &Laudon,2016).

I.8.2 Advantages :

I.8.2.1 Data Visualization and Simplification :

A dashboard transforms complex and large volumes of data into visual elements such as charts, graphs, and indicators. This representation simplifies information, allowing users to quickly understand trends, patterns, and relationships between datasets (Few.2006). It helps both technical and non-technical **users interpret data efficiently** (Knaflic,2015).

I.8.2.2 Performance Monitoring through KPIs :

Dashboards integrate Key Performance Indicators (KPIs) and provide continuous monitoring of organizational performance. **Managers can track objectives, detect deviations early, and identify areas that require improvement** (Kaplan & Norton,1996).

I.8.2.3 Faster and More Accurate Decision-Making :

By providing reliable, up-to-date information, dashboards enable decision makers to respond promptly to issues, make informed strategic choices, and optimize operational processes (Eckerson,2010).

I.8.2.4 Time and Effort Saving :

Dashboards automate data collection and reporting, reducing the need for manual compilation. **Employees can focus more on analysis and decision-making rather than gathering data**, which saves both time and effort (Turban et al.,2017).

I.8.2.5 Centralization and Improved Communication :

Dashboards consolidate data from multiple sources into a single interface, enhancing communication, collaboration, and transparency across departments**. Skateholders have access to consistent and actionable information**(Few.2006)**.**

I.8.3 Types :

Dashboards can be categorized into four main types, based on their purpose and the level of decision-making they support:

I.8.3.1 Operational dashboard :

An operational dashboard monitors daily activities and transactional data in real time. It helps employees and supervisors track processes and respond promptly to any issues.

I.8.3.2 Strategic dashboard :

A strategic dashboard is used by top management to track overall organizational performance and long-term objectives. It is updated less frequently and focuses on trends and high-level results.

I.8.3.3 Analytical dashboard :

An analytical dashboard is designed to analyze large volumes of data in order to identify trends and perform deeper analysis. It is mainly used by analysts to explore historical data and support decision-making.

I.8.3.4 Tactical dashboard :

A tactical dashboard is used by middle management to monitor departmental performance and short-term objectives. It focuses on analyzing data related to specific teams or processes in order to improve operational efficiency (Eckerson, 2010; Sharda et al., 2018).

These elements together provide a comprehensive view of organizational performance, enabling stakeholders to explore, monitor, and act on data efficiently (Few.2006; Eckerson, 2010).

- **In this project**, the developed dashboard is mainly **operational and analytical**, as it allows monitoring daily transactions and analyzing performance data to support decision-making.

I.8.4 Dashboard Components :

Dashboards consist of several key components that allow organizations to monitor performance and make informed decisions efficiently. The main components include:

Charts : Used to visually represent data, such as bar, line, or pie charts, to make trends easier to understand.

Tables : Display detailed and organized data, allowing users to read information precisely.

Indicators : Show key values directly, such as the number of operations or transaction volume.

Filters : Allow users to analyze data based on different criteria, such as date, department, or type of operation.

Alerts : Notify users of important changes or performance issues (Few.2006; Eckerson, 2010).

I.8.5 Development process :

I.8.5.1 Requirement Analysis :

Identify the purpose of the dashboard and the needs of its users. Define Key Performance Indicators (KPIs) and decide which data sources are required.

I.8.5.2 Data Collection and Integration :

Gather data from multiple sources such as databases, spreadsheets, or operational systems. Ensure that data from different sources can be combined in a consistent way.

I.8.5.3 Data Processing and Storage :

Clean, transform, and structure the data for analysis. Store it in a reliable database, ready for retrieval by the dashboard.

I.8.5.4 Dashboard Design :

Choose the best visual representations for each type of data (charts, graphs, tables).Organize the layout to make information easy to read and interpret.

I.8.5.5 Development and Implementation :

Build the backend and frontend of the dashboard using suitable technologies (e.g., Laravel for backend, REST API for data communication).Connect the dashboard interface to the database and implement dynamic updates.

I.8.5.6 Testing and Validation :

Check that all data is accurate and that the dashboard functions as intended. Validate KPIs and ensure visualizations correctly represent the underlying data.

I.8.5.7 Deployment and Maintenance :

Launch the dashboard for users. Regularly update data, improve performance, and add new features as needed (Sharda et al., 2018; Turban et al.,2017).

I.9 Conclusion :

In this chapter, we presented the study environment and analysed the existing system of Algérie Poste. We also identified the main problems related to data monitoring and decision-making. Based on this analyzis, a dashboard solution was proposed to improve data visualization, activity monitoring, and operational management.

II. Chapter 2 : Analysis and Design

II.1 Introduction :

After presenting in the previous chapter the different concepts and technologies required for the realization of our application, we now move to the analysis and design phase of the proposed system.

This phase consists of identifying the different actors interacting with the system as well as their functional requirements. Then, we model the behavior and the structure of the system in order to better understand its operation.

To achieve this design, we use the UML (Unified Modeling Language), which allows us to represent the different aspects of the system through several diagrams such as the use case diagram, sequence diagram, and class diagram.

II.2 Project Objective :

The main objective of this project is to design and develop a dashboard system for Algeria Poste in order to centralize data and facilitate its visualization and analysis.

The system allows administrators to manage users, services, and platform data, while users can consult statistics and filter the information displayed on the dashboard.

The dashboard also provides performance indicators (KPIs), statistical charts, and dynamic filtering functionalities to improve decision-making.

II.3 System Description :

Our system represents a web platform for managing and visualizing data through an interactive dashboard.

Users access the application through a secure web interface requiring authentication. After logging in, they can consult the available dashboard data, perform searches, and apply filters to obtain specific information.

The administrator has additional privileges allowing them to manage user accounts, modify information, manage permissions, export data, and add, remove, or block services.

The system is based on a Laravel architecture where requests sent from the user interface are redirected to controllers in order to process data and interact with the database.

Statistical results and KPIs are generated dynamically and displayed through charts and KPI cards without requiring a full page reload.

II.4 Identification of Actors and Their Tasks :

- **Administrator :**

The administrator is responsible for the overall management of the system. The system must allow the administrator to:

- Authenticate to access the private space.
- View the dashboard.
- Manage users.
- Add, modify, and delete user accounts.

- Reset passwords.
- Manage user permissions.
- Manage the database.
- Add, remove, or block services.
- Export data and statistics.

❖ **User :**

The user represents any authorized person allowed to consult dashboard data. The system must allow the user to:

- Authenticate to access the personal space.
- View the dashboard.
- Search data.
- Filter displayed information.
- Consult statistics and indicators.

II.5 Representative Diagrams :

II.5.1 Use Case Diagram :

A use case represents a set of action scenarios performed by the system that produces an observable result of interest for a particular actor. It describes what the system should do without specifying how to do it.

II.5.1.1 Relationships in Use Case Diagrams :

1. **Association Relationship :**
   It is a relationship between an actor and a use case that describes the interaction between the user and the system functionalities.

2. **Include Relationship :**
   It is a relationship in which a use case includes the functionality of another use case. An include relationship between use case A and use case B means that A necessarily contains the behavior described in B. Therefore, use case A cannot execute without use case B.

3. **Extend Relationship :**
   It is a relationship used to specify that a use case extends the behavior of another use case. An extend relationship between use case A and use case B means that the behavior of B can be optionally extended by A under specific conditions.

4. **Generalization Relationship :**
   It is a relationship in which a child element is based on another parent element. In use case diagrams, generalization indicates that the child actor inherits the characteristics and functionalities of the parent actor.

After identifying the actors and their tasks, the modeling of the use case diagrams is described below.

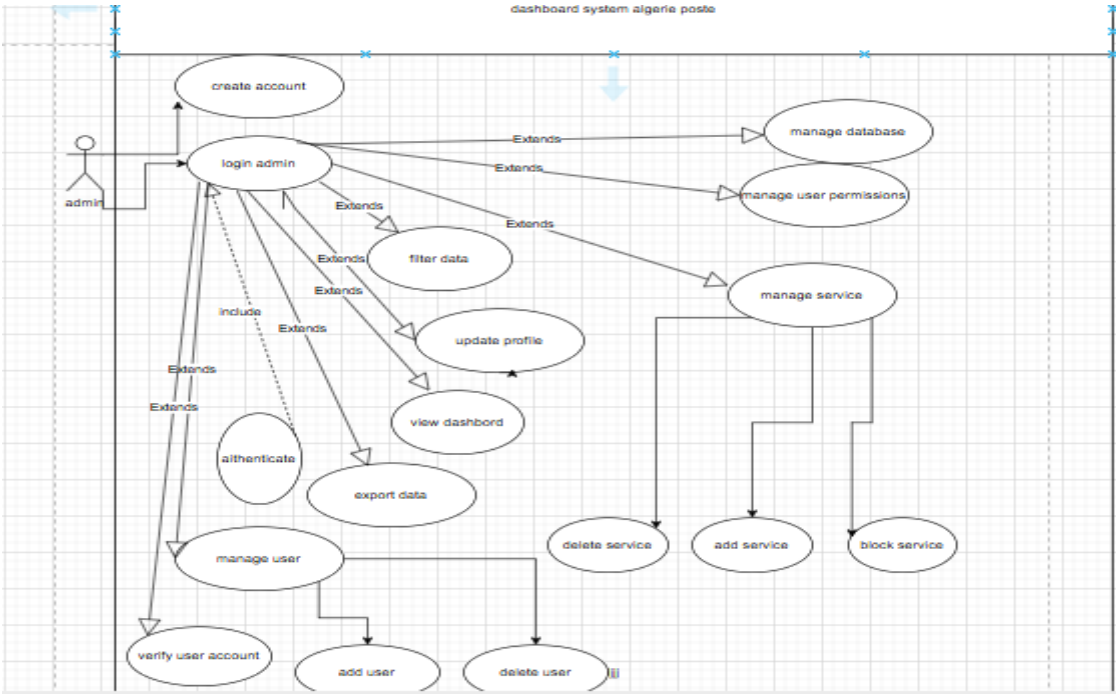


Figure 2.1- Use Case Diagram "Administrator Space".

Textual Description :

Name : Access Dashboard Space

Actor : Administrator:

Description : The use case starts when the administrator accesses the private dashboard space.

Main Scenario :

1. The system asks the administrator to enter the username and password.
2. The administrator enters the authentication information.
3. The system validates the administrator credentials.
4. The administrator accesses the dashboard interface.
5. The administrator consults dashboard statistics and KPIs.
6. The administrator manages users and permissions.
7. The administrator manages database operations and services.
8. The administrator exports statistical data if necessary.
9. Logout.

Alternative Scenario :

10. If the administrator decides to manage users:
    - Triggering the "Manage Users" use case.

11. If the administrator decides to manage services:
    - Triggering the "Manage Services" use case.
12. If the administrator decides to export data:
    - Triggering the "Export Data" use case.

Error Scenario :

13. Incorrect authentication information:
    - The administrator is not recognized by the system.
    - The system displays an error message.
    - Return to step 1.

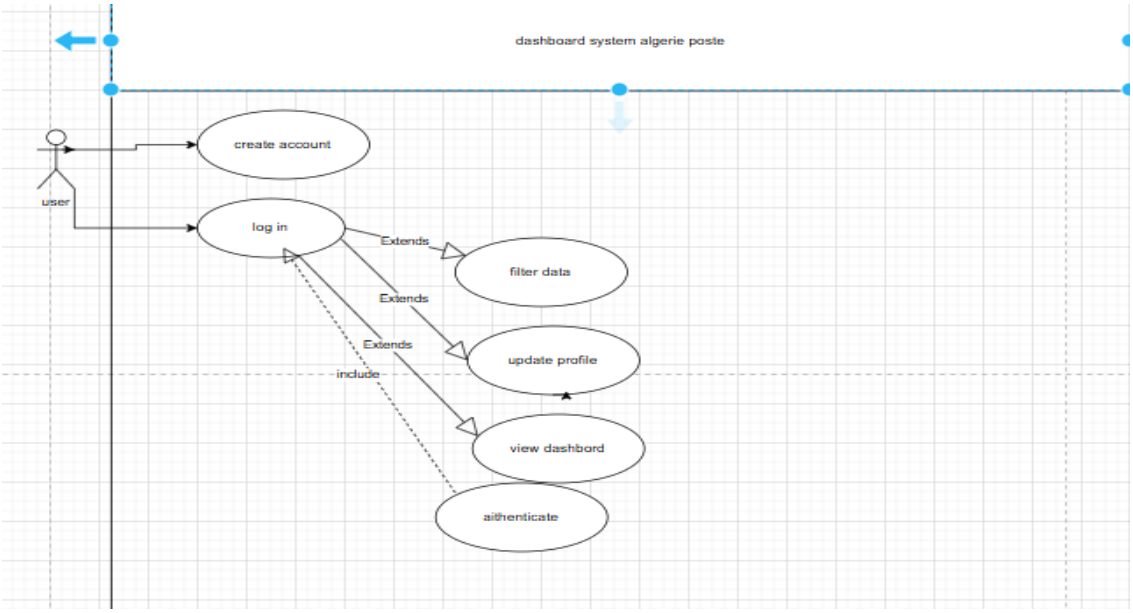


Figure 2.2- Use Case Diagram “User Space”.

Textual Description :

Name : Access User Dashboard Space.

Actor : User.

Description : The use case starts when the user accesses the dashboard application.

Main Scenario :

1. The system asks the user to enter the username and password.
2. The user enters the authentication information.
3. The system validates the user credentials.
4. The user accesses the dashboard interface.
5. The user consults dashboard statistics and KPIs.
6. The user searches for specific information.
7. The user applies filters to the displayed data.
8. The dashboard updates charts and statistical data dynamically.
9. Logout.

Alternative Scenario :

10. If the user decides to search for information:
    - Triggering the “Search Data” use case.
11. If the user decides to filter dashboard data:
    - Triggering the “Filter Data” use case.

Error Scenario :

12. Incorrect authentication information:

- The user is not recognized by the system.
- The system displays an error message.
- Return to step 1.

II.5.2 Sequence Diagrams :

Sequence diagrams are part of behavioral diagrams. They represent the interactions between the different elements of the system in a chronological order. They illustrate how messages are exchanged between the user, the interface, the controllers, and the database in order to achieve a specific functionality.

In our dashboard system, these diagrams describe the dynamic behavior of the main operations of the system, mainly authentication and data filtering.

II.5.2.1 Interaction Operators Used :

In our design, we mainly used the following operators:

1. **Alternative Operator (alt) :** It represents a condition of type if / else. Only one branch is executed depending on the result of the condition.
2. **Reference Operator (ref) :** It allows referencing another sequence diagram already defined, in order to simplify the model and avoid repetition.

The sequence diagrams of the main use cases are presented below.

II.5.2.2 Sequence Diagram « Authentication » :

This diagram describes the login process to the system.

The user enters credentials through the interface, then the request is sent to the server for verification. The server interacts with the database to check user information.

ref : Authentication and Database Model (used for database verification process).

alt :If the credentials are correct, access to the dashboard is granted.

Otherwise, an error message is displayed.

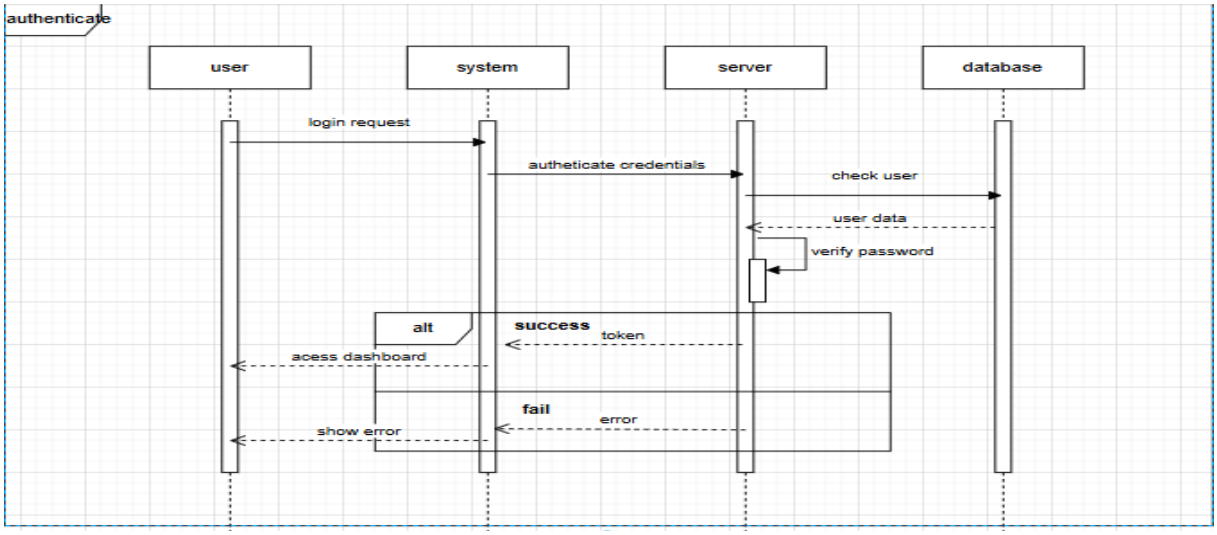


Figure 2.3- Sequence Diagram « Authentication ».

II.5.2.3 Sequence Diagram « API Filters » :

This diagram describes the dynamic filtering process of data.

The user selects filters from the interface, then the request is sent to the server. The server processes the request and queries the database to retrieve filtered data.

alt :If the data is valid, the results are returned and the dashboard is updated.

Otherwise, an error message is displayed.

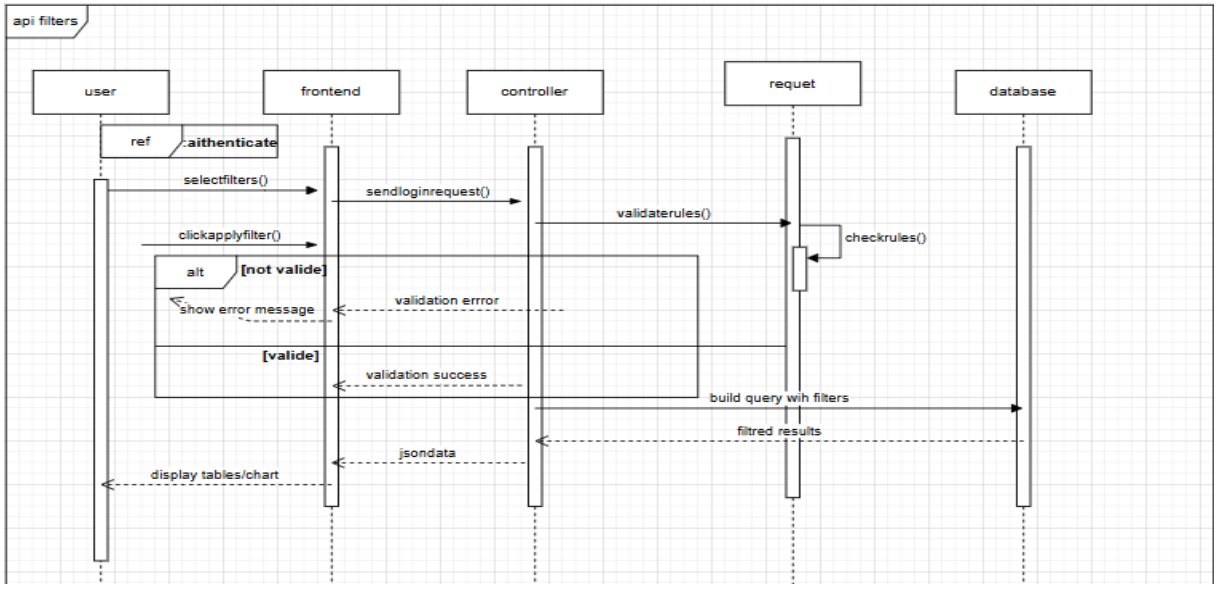


Figure 2.4- Sequence Diagram « API Filters ».

II.5.3 Class Diagram :

The class diagram represents the overall structure of the operations and establishments management system. It illustrates the different classes of the system, their attributes, and the relationships between them.

After analyzing the system requirements, several main classes were identified, including :

- Wilaya.
- Daira.
- Commune.
- Etablissement.
- Agent.
- User.
- Service.
- Product.
- Operation.
- ProfitMargin.
- UPW.

Each class contains a set of attributes that describe the information managed by the system.

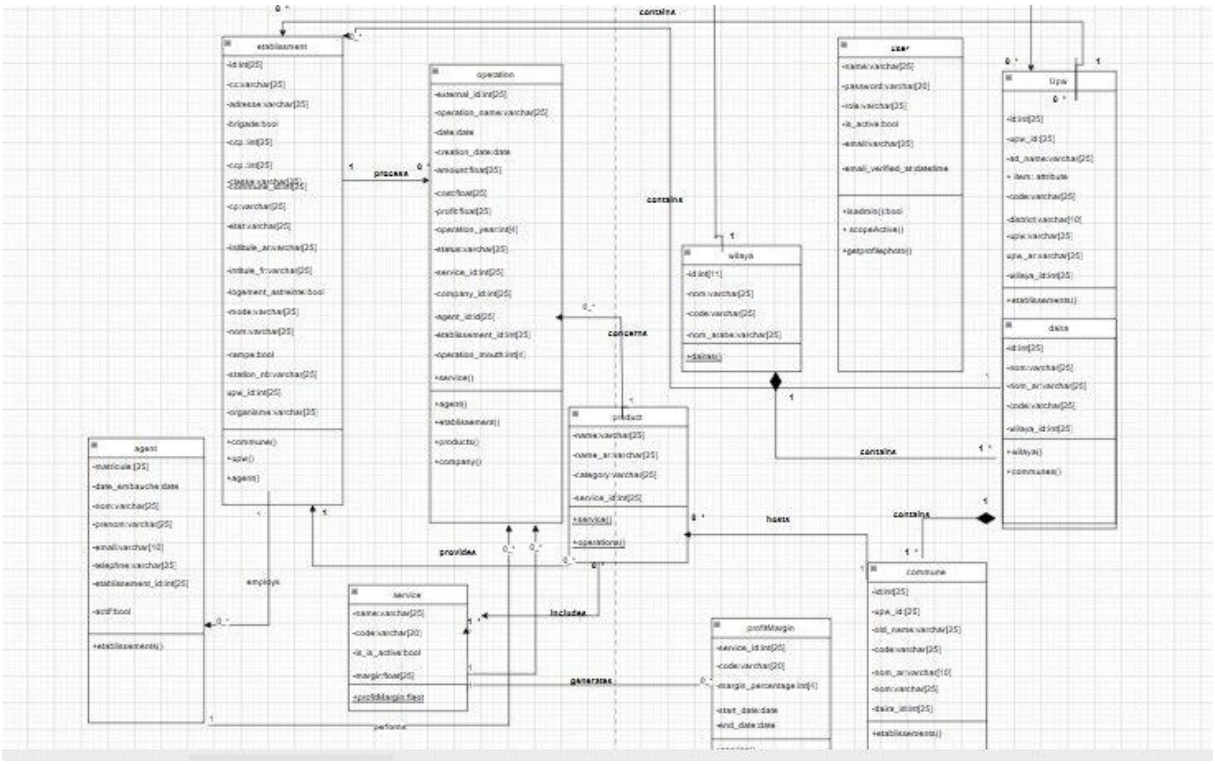


Figure 2.5- Class Diagram Of System.

II.5.3.1 Relationships Between Classes :

The main relationships between the classes are as follows:

- One wilaya contains several dairas.
- One daira contains several communes.
- One commune contains several establishments.
- One establishment has several agents.
- One establishment can perform several operations.
- One agent can perform several operations.
- One service contains several products.
- One service can have several profit margins.
- One operation may involve several products.

The class diagram is an essential step in the system design process since it serves as the foundation for building the relational database model.

II.6 Conclusion :

The design phase allowed us to define the overall structure of the system through the class diagram and the relational model. This step helped identify the different entities, their attributes, and the relationships between them. It also provided a solid foundation for the development and efficient management of the system database.

III. Chapter 3 : Application Implementation

III.1 Introduction :

This chapter presents the implementation phase of the web application developed for Algérie Poste, a platform designed to monitor and analyse operations pertaining to grands comptes clients through an interactive and dynamic dashboard. Having established the theoretical foundations and design specifications in the preceding chapters, the discussion now turns to the concrete realisation of the system: the choices made regarding the development environment, the architectural decisions governing the interaction between layers, the implementation of functional modules, the security mechanisms deployed to protect sensitive operational data, and the performance optimisation strategies adopted to ensure acceptable responsiveness under realistic workloads. The chapter concludes with a candid account of the principal technical difficulties encountered during development and the engineering solutions that were devised to address them.

III.2 Development Environment :

III.2.1 Hardware Environment :

The application was developed and tested on standard workstation hardware representative of the machines available within the organisation. The configuration is summarised in Table 1.

Component	Specification
Processor	Intel Core i5 / i7 (8th generation or later)
RAM	8 GB – 16 GB DDR4
Operating System	Windows 10 / Windows 11 (64-bit)
Storage	SSD 256 GB (NVMe preferred)

Table 3.1-Hardware configuration used during development.

III.2.2 Software Environment :

A carefully selected set of technologies and tools was employed throughout the development lifecycle. The stack was chosen to maximise developer productivity, leverage community-supported libraries for charting and UI components, and ensure that the deployed application would integrate smoothly with Algérie Poste's existing infrastructure. Table 2 provides a comprehensive overview of the software environment.

Technology / Tool	Role in the Project
PHP 8.4	Server-side programming language
Laravel 11	MVC web framework — routing, ORM, middleware, caching
PostgreSQL 18	Relational database management system
HTML5 / CSS3	Markup and styling of web pages
JavaScript (ES2022)	Client-side logic and dynamic interactions
Bootstrap / Tailwind CSS	Responsive UI layout and component design
Chart.js / ApexCharts	Statistical charts and interactive visualisations
Laravel Herd	Local development server for Laravel on Windows
Visual Studio Code	Primary source-code editor
Git	Distributed version control
pgAdmin 4	PostgreSQL schema design and query analysis

Table 3.2-Software technologies and tools.

III.2.3 Rationale for the Laravel Framework :

Laravel was selected as the primary framework after evaluating several PHP alternatives. Its built-in MVC structure enforces a clean separation of concerns from the outset, which is particularly valuable in a project where the data layer (complex aggregate queries), the business logic layer (KPI calculations, filter resolution), and the presentation layer (dashboard interfaces) must evolve independently without introducing regressions across the codebase. Beyond architectural discipline, Laravel provides an integrated security subsystem — CSRF protection, parameterised query binding through Eloquent, session management, and middleware-based authentication — that would otherwise require significant custom implementation. Its Eloquent ORM simplifies the construction of dynamic, filter-driven queries while still permitting raw SQL when aggregate complexity demands it. Finally, Laravel's cache abstraction layer and its queue system offered credible paths toward performance optimisation without requiring a change of framework mid-project.

III.3 Application Architecture :

III.3.1 MVC Architectural Pattern :

The application is structured according to the Model–View–Controller (MVC) paradigm as implemented by Laravel. This pattern divides the application into three collaborating layers, each with a well-defined responsibility, as illustrated in Figure 3.1.

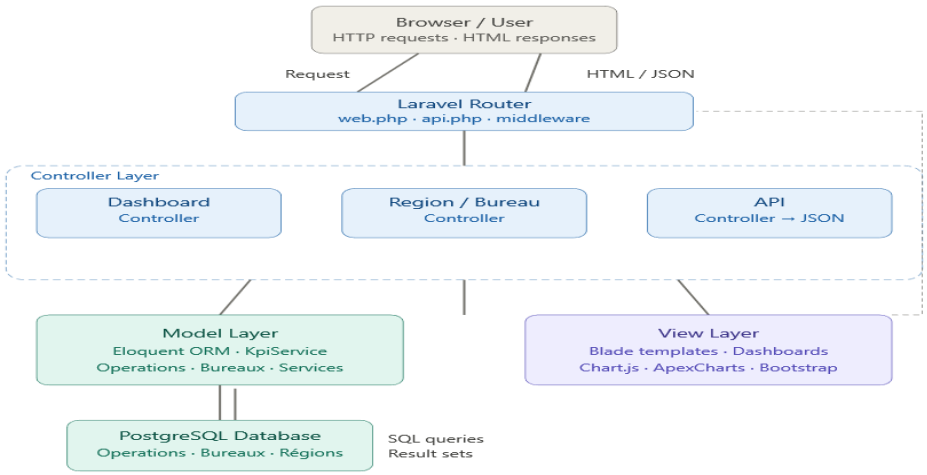


Figure 3.1-MVC architecture of the application.

III.3.1.1 Model Layer :

The model layer encapsulates all interactions with the PostgreSQL database. Each entity — operations, bureaux, services, regions, users — is represented by a dedicated Eloquent model class that defines the table mapping, the fillable fields, and the relationships to other entities. Models also host scope methods that encapsulate frequently reused query conditions (e.g., filtering by active service, scoping to a date interval), keeping the controllers free of repeated query fragments. KPI calculations that involve aggregation across multiple records — total transaction volumes, average profitability, bureau-level rankings — are implemented as model methods or repository classes that expose a clean interface to the controller layer.

III.3.1.2 View Layer :

Views are implemented through Laravel's Blade templating engine. Each dashboard page is composed from a master layout file that defines the persistent elements — the navigation sidebar, the top header bar, and the footer — with content sections injected by child templates. This inheritance model eliminates duplication across pages and guarantees visual consistency. For dynamic components whose data must be refreshed without a full page reload (charts, KPI cards, filter-dependent tables), the view contains JavaScript logic that issues asynchronous requests to JSON API endpoints; the returned data is then used to update the DOM elements in place. Static assets (compiled CSS, JavaScript bundles) are managed through Laravel Mix and served from the public directory.

III.3.1.3 Controller Layer :

Controllers act as orchestrators: they receive validated HTTP requests, coordinate the retrieval and processing of data through model methods and service classes, and return either a fully rendered Blade view or a JSON response for AJAX callers. The application defines several specialised controllers. The DashboardController handles the main overview page, computing global KPI summaries and populating the principal charts. The RegionController and BureauController manage geographic breakdown pages, accepting dynamic filter parameters and constructing the corresponding scoped queries. The ApiController exposes the JSON endpoints consumed by the frontend JavaScript layer when filters change or charts require updated datasets. This decomposition keeps each controller focused and testable in isolation.

III.4 Implementation of Interfaces and Functionalities :

III.4.1 Authentication Interface :

Access to the application is gated by a secure authentication system built upon Laravel's Auth scaffolding. The login interface presents the user with an email address field and a password field; upon submission, the credentials are verified against the utilisateurs table, where passwords are stored as bcrypt hashes with a cost factor calibrated to provide adequate resistance to brute-force attacks while remaining transparent to legitimate users. Failed authentication attempts are rate-limited through Laravel's built-in throttling middleware to mitigate credential-stuffing attacks. Upon successful authentication, a server-side session is established and a CSRF token is embedded in all subsequent form submissions, preventing cross-site request forgery. The authentication interface is shown in Figure 3.2.

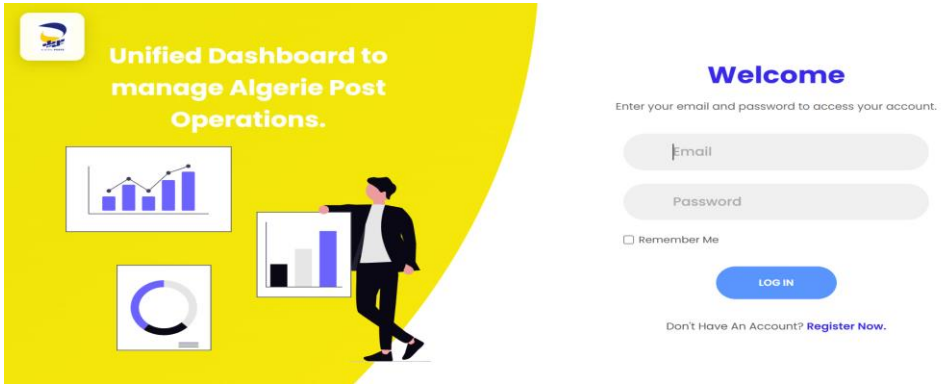


Figure 3.2- User authentication interface.

Once authenticated, the user's role attribute is read from the session on every protected request. The role determines which routes, dashboards, and administrative functions the session holder may access, as described in further detail in Section 5.

III.4.2 Main Dashboard :

The dashboard constitutes the central component of the application and the primary point of interaction for analysts. Its design follows the principle of progressive disclosure: the most critical KPIs are presented at the top of the page as summary cards, with more granular breakdowns available through charts and tables lower on the page and in dedicated sub-pages reachable from the navigation sidebar.

- **KPI Cards :**

A row of KPI cards at the top of the dashboard surface the most operationally significant indicators: the total amount of transactions processed within the selected period, the total number of operations, the success rate, the agent of the month based on transaction volume and the evolution of transactions. They allows a manager to assess performance at a glance without consulting the underlying charts. KPI values are computed server-side and embedded in the initial Blade render; they are refreshed asynchronously whenever the user modifies the active filter set. The main dashboard with KPI cards is illustrated in Figure 3.3.



Figure 3.3-Main dashboard with KPI cards and summary statistics.

- **Statistical Charts :**

Below the KPI cards, the dashboard presents a set of statistical visualisations, each targeting a specific analytical question. The chart types employed and their respective purposes are described in Table 3.3.

Chart Type	Library	Analytical Purpose
Bar chart (grouped)	Chart.js	Comparison of transaction volumes or amounts across bureaux for a selected period

Chart Type	Library	Analytical Purpose
Line chart	Chart.js	Temporal evolution of operations volume and profitability over monthly intervals
Pie / Doughnut chart	Chart.js	Distribution of transaction types across service categories

Table 3.3-Chart types integrated into the dashboard.

All charts are rendered client-side using data supplied by asynchronous JSON requests to the ApiController. When a filter is modified, the JavaScript layer dispatches updated requests and calls each chart library's native data-update method to animate the transition to the new dataset, avoiding the visual disruption of a full chart redraw.

III.4.3 Dynamic Filtering System :

A persistent filter toolbar positioned above the chart area allows users to narrow the scope of all displayed statistics simultaneously. The available filter dimensions are: date range (with calendar pickers for start and end dates), postal bureau, service category, and UPW/region. Filter controls are bound to a shared JavaScript state object; any change triggers a debounced update function (400 ms delay) that collects the current state, encodes it as a query string, and dispatches fetch requests to all active API endpoints on the page. Debouncing prevents a cascade of network requests when a user modifies multiple filters in rapid succession — a common interaction pattern when resetting a date range. The filtering interface is shown in Figure 3.4.



Figure 3.4- Dynamic filter toolbar and filtered dashboard results.

On the backend, filter parameters are received by the relevant controller, validated through Laravel's Form Request classes (checking date format, value membership for enumerated parameters, and logical consistency constraints such as start date preceding end date), and then passed to a QueryBuilder wrapper that applies each non-null filter as an additional WHERE clause on the base query. This architecture ensures that all filter combinations — including the unfiltered case — are handled by a single code path, eliminating duplication and the risk of inconsistent behaviour across filter combinations.

III.4.4 Regional Analysis :

The regional analysis module provides a comparative view of performance across the geographic hierarchy: from individual bureaux, up through UPW entities, to full regions. The page is structured around all regions selector; the controller fetches aggregated statistics for all bureaux belonging to that region, renders them as a ranked table and as an interactive bar chart, and highlights the top and bottom performers. An inter-regional comparison view allows side-by-side benchmarking of regions on a chosen metric, enabling management to identify geographic disparities in service uptake or profitability. The regional analysis interface is shown in Figure 3.5.



Figure 3.5-Regional analysis dashboard — upw-level performance breakdown.

III.4.5 Performance Optimisation :

Several optimisation measures were implemented to keep dashboard response times within acceptable bounds on the production dataset. PostgreSQL indexes were created on the columns most frequently used as filter predicates: the transaction date column, the bureau foreign key, and the

service foreign key. These indexes allow the query planner to perform efficient range scans rather than full-table scans, reducing query execution time from several seconds to well under 300 ms for the most common dashboard requests.

For aggregate queries that are computationally expensive but do not require real-time freshness — such as the monthly evolution time series — the results are cached using Laravel's cache layer with a configurable time-to-live (TTL, set to five minutes in the current deployment). Cache entries are keyed on the filter parameters so that different filter combinations are cached independently. The cache is invalidated whenever a new batch of operational records is imported, ensuring that users always see data no older than one import cycle.

On the frontend, JavaScript bundles and CSS stylesheets are compiled and minified through Laravel Mix, reducing the total initial payload size. Chart rendering is deferred until the DOMContentLoaded event fires, preventing long-running initialisation from blocking the browser's rendering pipeline and degrading the perceived load time of the page.

III.5 Application Security :

Security in an operational dashboard that surfaces sensitive financial and performance data is a non-negotiable requirement. The application addresses security at multiple layers, from the network boundary down to individual database queries.

III.5.1 Data Protection :

All user passwords are stored as bcrypt hashes; no plaintext credential is ever persisted. Laravel's Eloquent ORM uses parameterised queries exclusively, eliminating the possibility of SQL injection regardless of the filter values supplied by the user. All HTML output rendered by Blade templates is escaped by default, preventing stored or reflected cross-site scripting (XSS) attacks. CSRF tokens are validated on every state-changing HTTP request; requests lacking a valid token are rejected with a 419 response before reaching any controller logic. Form inputs are validated through Laravel's Form Request classes, which enforce type, format, and range constraints, preventing malformed data from reaching the business logic layer.

III.5.2 Access Management :

Access control is implemented through a layered middleware architecture. The auth middleware, applied to all dashboard routes, redirects unauthenticated requests to the login page before any controller code executes. A secondary role middleware reads the authenticated user's role attribute — one of administrator, analyst, or read-only — and compares it against a route-level policy map. Administrator-only routes (user management, data import triggers) return a 403 Forbidden response

when accessed by non-administrator sessions. Read-only users can consult all dashboards but cannot trigger data exports or modify configuration parameters. This role separation ensures the principle of least privilege is respected across the user population.

III.6 Challenges Encountered and Solutions Applied :

The development of a real-time analytical dashboard over a large operational dataset inevitably gives rise to technical difficulties that cannot be fully anticipated during the design phase. This section documents the principal challenges encountered and the engineering decisions that resolved them.

III.6.1 Handling Large Volumes of Operational Data :

- **Challenge :**

Early prototypes of the dashboard query logic were performing full-table scans on the opérations table even when the user had applied a narrow date filter. On the development dataset — approximately 800,000 transaction records spanning three years — several dashboard queries were returning results in four to six seconds, which rendered the interface unusable for interactive analysis.

- **Solution :**

Profiling with PostgreSQL's EXPLAIN ANALYZE tool revealed that the query planner was not using the available date index because the filter was being applied after a join that had already expanded the result set. Rewriting the affected queries to apply the date range predicate before any join, and adding composite indexes on (bureau_id, date) and (service_id, date), reduced execution time to under 200 ms for the same dataset. The experience reinforced the importance of consulting the query execution plan rather than relying on the ORM to generate optimal SQL in all circumstances.

III.6.2 Chart Synchronisation Across Multiple Panels :

- **Challenge :**

The regional analysis page displays three charts simultaneously, each consuming data from a separate API endpoint. During initial development, each chart issued its own independent fetch request when a filter changed, leading to a race condition in which a slow response for one endpoint left the page in an inconsistent state — one chart reflecting the new filter while the others still displayed data for the previous filter configuration.

- **Solution :**

A centralised DataManager singleton was introduced in the JavaScript layer. When a filter change is detected, the DataManager dispatches all required fetch requests in parallel using Promise.all and commits the results to the chart instances only once all responses have been received. This guarantees that all charts on the page transition to the new dataset simultaneously, eliminating the inconsistency

window. The additional latency introduced by waiting for the slowest response was measured at under 120 ms on the test environment and was deemed acceptable given the improvement in data integrity.

III.6.3 Complex KPI Calculations with PostgreSQL :

- **Challenge :**

Several KPI metrics — notably the year-on-year variation in transaction volume and the rolling average profitability over a sliding three-month window — require correlated subqueries that compare the current period's aggregate against a corresponding aggregate for a different time window. Initial attempts to express these calculations through the Eloquent ORM produced N+1 query patterns that were both slow and difficult to reason about.

- **Solution :**

The complex KPI queries were rewritten as raw SQL using Common Table Expressions (CTEs) submitted through Laravel's DB::select facade with bound parameters. CTEs allowed each intermediate aggregate to be defined once and referenced multiple times within the same query, eliminating redundant computation. The use of bound parameters ensured that the security guarantees of parameterised queries were preserved despite bypassing the ORM. A dedicated KpiService class encapsulates all such queries, exposing a clean PHP interface to the controller layer and isolating the raw SQL from the rest of the codebase.

III.6.4 Dynamic Filter Interaction with SQL Aggregations :

- **Challenge :**

A regression was observed in which applying both a region filter and a service filter simultaneously produced aggregated totals that were inconsistent with the sum of the bureau-level values visible in the accompanying table. The discrepancy was intermittent and dependent on the specific combination of filter values selected.

- **Solution :**

Investigation revealed that the QueryBuilder wrapper was appending WHERE clauses after the GROUP BY expression in certain filter combinations, causing the region filter to be evaluated as a HAVING condition rather than as a row-level predicate. The fix required enforcing a strict clause ordering within the QueryBuilder: all WHERE conditions are appended before GROUP BY is applied, regardless of the order in which filter parameters are received. A regression test suite covering all two- and three-dimension filter combinations was added to the test harness to prevent recurrence.

III.6.5 Frontend Responsiveness and Chart Reflow :

- **Challenge :**

ApexCharts does not automatically resize its canvas when its container element changes dimensions — a known limitation of the library. On the regional analysis page, toggling the sidebar caused the main content area to expand or contract, but the chart canvases retained their previous pixel dimensions, breaking the visual layout.

- **Solution :**

A ResizeObserver was attached to each chart container element. Whenever the observer detected a dimension change, it invoked the chart's updateOptions method with the new container width. The resize handler was debounced with a 150 ms delay to avoid triggering layout recalculations on every animation frame during the sidebar CSS transition. After this fix, charts reflowed correctly on all tested screen widths, including the edge case of the browser window being resized manually.

III.7 Conclusion :

This chapter has documented the full implementation of the web application designed to monitor and analyse grands comptes operations at Algérie Poste. The development environment was chosen to maximise productivity and compatibility with organisational infrastructure, while the Laravel MVC framework imposed an architectural discipline that kept the codebase maintainable as the scope of the project grew. The implementation of the authentication system, the main dashboard, the dynamic filtering mechanism, the regional analysis module, and the performance optimisation pipeline collectively deliver a platform capable of supporting evidence-based operational decision-making at multiple levels of the organisation.

The security mechanisms integrated at the authentication, access-control, and data-validation layers ensure that sensitive operational data is accessible only to authorised personnel and is never exposed to injection or forgery attacks. The challenges encountered during development — particularly around query performance, chart synchronisation, and filter interaction with complex SQL aggregations — were resolved through methodical diagnosis and targeted engineering interventions whose solutions have been incorporated as permanent fixtures of the codebase.

References :

- Business Rules Group, (2000). *Defining Business Rules: What Are They Really?* Final Report.
- Davenport, T. H., & Harris, J. G. (2007*). Competing on Analytics: The New Science of Winning*. Harvard Business School Press.
- DeMarco, T., 1979. *Structured Analysis and System Specification*. Englewood Cliffs: Prentice-Hall.
- Eckerson, W. W. (2010). Performance dashboards*: Measuring, monitoring, and managing your business* (2nd ed.). Hoboken, NJ: John Wiley & Son.
- Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems (7th ed.*). Pearson.
- Few, S. (2006). *Information Dashboard Design. O'Reilly Media.*
- Knaflic,C. N. (2015). *Storytelling with data: A data visualization guide for business professionals*.Hoboken,NJ: John Wiley & Sons.
- Kaplan,R. S., & Norton, D. P. (1996). *The balanced scorecard: Translating strategy into action*. Boston, MA: Harvard Business School Press.
- Laudon, K. C., & Laudon, J. P. (2016). *Management Information Systems. Pearson.*
- Laudon, K. C., & Laudon, J. P. (2018). *Management Information Systems: Managing the Digital Firm*. Pearson.
- O'Brien, J., & Marakas, G. (2011). *Management Information Systems*. McGraw-Hill Education.
- Sharda, R., Delen, D., & Turban, E. (2018). *Business intelligence, analytics, and data science: A managerial perspective (4th ed.)*. Pearson.
- Turban, E., Sharda, R., & Delen, D. (2017). *Decision support and business intelligence systems* (10th ed.). Pearson.
- Turban, E., Sharda, R., & Delen, D. (2014*). Business Intelligence and Analytics: Systems for Decision Support*. Pearson.
- Yourdon, E. (1989). *Modern Structured Analysis*. Prentice Hall.