A Proposed Model for Developing a Monitoring Solution for e-Business Enhancement

Alexandru COCIORVA
Bucharest University of Economic Studies, Romania
cociorvaalexandru@yahoo.co.uk

In the context of a growing, digitally developed society, new measures for improving e-business solutions are considered mandatory. One of these measures is monitoring enhancement, as an IT operation that has the purpose of optimizing resource utilization and processes from an e-business application. Thus, it is important not only to monitor e-business application infrastructure but also to create correction plans and solutions that can be, in some cases, applied as an immediate response to a system error. This article proposes a model for developing a monitoring application, detailing all its implications in the e-business world, in the context of the information society. The model is sustained by an experimental framework, along with theoretical basis, which will be developed in the pages of the present paper.
Keywords: *Monitoring, ETL, E-business, Cloud, System, IoT*
DOI: 10.24818/issn14531305/26.2.2022.01

1 Introduction

The demands of today's digital world are in conjunction with entrepreneurial activity in the e-business environment, and cloud solutions are increasingly being promoted for a multitude of online businesses. Thus, the premises are created to build an ecosystem of cloud solutions that would allow access to the penetration of as many businesses as possible in the online environment, the marketing activity being strongly concentrated in the digital area, of the information society [1]. Cloud computing solutions often require continuous improvement depending on the scale of the e-business activity, and in recent years, a multitude of solutions have been found to help complex online business activities. One of these solutions is the flexible database system, to better address today's e-business environment, which includes different types of end-users (users) who perform a wide variety of ad-hoc queries and use dozens of integrated applications. At the moment, the use of a different system for creating databases is being discussed, having as characteristics: application-specific structures, read-only images and materialized views. There is a query-based method that uses smart design and selects read-only components to provide quick access to a variety of queries, including unforeseen queries [2]. Basically, this method of using flexible databases has as a premise the speed of queries, which makes it easier to perform transactions in an e-business / e-commerce solution. This aspect also offers an important premise for facilitating the monitoring activity, so that the processes can be supervised much more efficiently and in a more controlled way. Of course, in addition to this aspect, there are also details regarding the architecture of the IT system used in the development of e-business / e-commerce solutions (CPU, memory, types of integrations, technologies used, etc.). It is important to highlight that benefits of using a monitoring system with ETL (Extract, Transform, Load) enhancement goes beyond the area of an e-business solution, the concept can have implications in standalone applications from various fields like: retail industry, medicine, biology and even manufacturing. The extended concept of such a monitoring system could easily incorporate the idea of IoT development, where real-world activities and operations from various domains can be controlled through an internet platform. Although the idea of using such a mechanism is very convenient for a lot of growing businesses and already existent robust market areas, still, very few business entities choose to make such an enhancement, based on IoT solutions. The main reason is represented by

the high cost that comes with such an implementation. It is important to mention that all this structure can be developed only through the implementation of latest technologies, including cloud solutions, especially due to the distributed applications system and large volume of processed data.

Coupled with the applicability of cloud computing solutions, the concept of Machine Learning is becoming increasingly useful and popular in making various applications with high complexity, thus becoming an important lever in performing activities that require neural analysis and predictive adjustments. Thus, great emphasis is placed on proactive activity in detecting an error or a problem, ML algorithms becoming increasingly useful and usable in this regard. As a result, monitoring activity can be substantially aided by the use of these types of algorithms, both in the structures of the monitoring application and as an integral part (through various functionalities) of the monitored application. The usefulness of ML algorithms is reflected in the fast response time to various common user queries. Although the system created by ML cannot understand the applicable or material part as a human, its ability to make connections between different information frames and interactions, allows the user to obtain an expected result, if the information used by ML algorithms has been consulted in advance, in the past, many times. Thus, ML offers a much more powerful research tool than simple classical methodologies [3]. These aspects can be applied, to a certain extent, in the monitoring activity as well. Returning to the aspect of monitoring in cloud solutions, they can provide a scalable IT infrastructure framework, QoS services, as well as a customizable computing environment. Although considerable progress has been made in researching how to implement cloud solutions in e-business solutions, most of these studies and results have been in the category of single-cloud provider or primary cloud service provider, so that all efforts to efficiently allocate resources in a cloud-based framework were focused on a single provider, not on the interaction between different providers [4]. This aspect could also be a limitation in the activity of monitoring e-business solutions, especially in the case of complex systems for monitoring distributed / complex applications.

Together with cloud and ML solutions, mining processes play an important role in extracting information relevant to the business environment (from related transactions or processes), stored in activity logs or application events [5]. This aspect also plays an important role in an ETL data processing mechanism that can be used in a monitoring system, in order to return any corrections applied to the supervised e-business system. As part of the process of adapting to IoT and ML, the process mining technique successfully serves to improve e-business processes, while highlighting non-compliant event patterns based on the analysis of data logs extracted from processes [6]. Such a practice can also help the monitoring activity in setting thresholds and, in response to the violation of certain compliance criteria, in defining alerts, non-compliant events can be the basis for triggering monitoring systems processes. The development of all these analysis tools for e-business, in the context of the information society, led to the definition of a new concept based on the activity of organizations and in response to their way of adapting to the new social and technological reality. Thus, the concept of distributed organization has become more widespread, in the context in which a multitude of business analysis tools are becoming more accessible, facilitating the collection, management and processing of data, both from customers and suppliers or various parties involved in business framework [7]. In order to achieve the necessary monitoring framework, it is important that the structures involved in this process are as well defined as possible and that the monitoring application is as flexible and easy to customize as possible, considering the latest technologies used to develop IoT.

In the process of adapting e-business solutions to the active and changing entrepreneurial environment in the context of digital-

ization and the information society, it is important not only to monitor aspects related to business and associated infrastructure, but also to anticipate, predict different scenarios in the activity area of a business, based on certain results of monitoring / supervision. Thus, in order to predict certain stages in the development of an online business, we can even use ML (Machine Learning) models and compare the performance of algorithms such as: logistic regression, SVM and XGBoost, using representative data sets, extracted from event logs of monitored e-business solutions [8].

All the presented aspects constitute important premises for developing optimized monitoring solutions for e-business applications, implementing ETL mechanisms, based on metrics from the IPCC framework. Thus, various integrations can be made with the monitoring solution, in order to efficiently customize the final monitoring product, adaptable or suitable for each e-business solution.

2 Descriptive elements related to the creation and implementation of the model

Due to its characteristics (versatility and efficiency), PHP technology is very often used in the realization of various e-business applications. Following the necessary configurations at the application level, according to the adaptations to the market requirements, the solution chosen for processing transactions can suffer various blockages both at the level of structure and, automatically, and at the level of performance. To meet such situations and to resolve, as far as possible, the problems encountered (considering the complexity and level of integration of an ERP with other information systems), there are a multitude of performance criteria that a developer must consider, especially in the context of the development of the cloud-computing area, which has grown considerably in recent years. There are a wide range of errors that can occur in the development of a PHP application, especially when it comes to an e-business application. Some of the common problems encountered relate to:

1. Creating too many queries / transactions in the database (creating a high flow of queries / transactions can significantly increase the power of request processing, causing the database to crash and the application to become unavailable, or in the best case, to register performance problems);
2. Incorrect web server configuration (certain parameters in the php.ini configuration file, and not only, can cause problems with the operation of the application; it is necessary to first check the availability of a port with the netstat command and to ensure that all database connection parameters are correct - database connection string, credentials, ports, etc.);
3. Uploading too many files (depending on the space available on the server, file uploads can be performed, but in general, developed algorithms need to be able to perform this upload operation in the best possible way);
4. Insert the sleep method into a call function;
5. External API requests (integration is generally a delicate issue, in this case due to the layers of connection between different technologies, which, through specific configurations, at code level, can create non-functional structures or complicated problems in customization).

In order to be able to verify all this described framework, in this case, evaluating the performance of the application in its various stages of development but also later, as various integrations appear, a multitude of monitoring support applications have been developed, thus the following performance indicators (KPIs) are scored with preference:

1. *Response time* - it is necessary to capture the response time of requests in an application, because requests that take too long can affect the stability of the application, especially when we have a predefined number of competing users that can log into an

application. Evaluating this KPI can give us important clues about the functionality of the application.

2. *Apdex score* - the evaluation of user satisfaction is an important indicator of application performance, because it can highlight the following aspects: stability and efficiency of the application, design, how to market the promoted products, and more.
3. *Exception rate* - the number of problems (exceptions) recorded at the structure level in the PHP application may be a valid reason for code-level optimization for certain functionalities, this indicator being relevant.
4. *Time to perform an SQL query or transaction* - it is important to know how long a transaction takes at the database level to optimally set up the database so that no latency occurs, including at the performance level of the application.
5. *SLA Report* - Service Level Agreement is a central synthetic indicator in evaluating an application because it provides details about its functionality - end-user validity, downtime and uptime.
6. *HTTP failure rates* - The evaluation of HTTP errors related to the application provides clues about its structural aspect, especially for developers, in order to optimize certain functionalities, improving some lines of code in the various compatibility issues.

Monitoring activity is also particularly important at the database level, so a number of metrics may be relevant not only in the context of the monitored object but also in relation to other metrics or influence in the calculation of other indicators. Thus, from the analysis undertaken on the activity of databases and how to supervise them in relation to business requirements, the following metrics can be highlighted in the context of monitoring an e-business application:

1. Memory allocation distribution, usage and KPI status.
2. Allocation of CPU resources by database or process and related counters.
3. Tempdb performance and usage measures, along with conflict points and costly operations.
4. Highly detailed collection of storage usage statistics (volume-specific database file).
5. Use the database and file transaction log with points of dispute and costly transactions.
6. Query performance for the most interesting server and database operations (depending on business specifications).
7. Blocking operations on the server (including its victims).
8. Statistics on the use and configuration of the resource governor, with details about the workload group.
9. Modify the server configuration and all its logs, such as: server, agent, SMT alerts, maintenance and automation of work.

After clearly defining the indicators that will be involved in the monitoring process and implicitly validating them, it is necessary to build a general scheme of the processing system within the monitoring, by highlighting the main elements, considering the business context. It is important to note that in the process of applying system corrections (evaluation and modification of transactions according to certain predefined criteria) the ETL mechanism can be composed of RPA (Robotic Process Automation) bots to act in the meaning of an automation implemented to reduce the execution time of some steps (such as data processing or system uploads). Thus, the entire ETL mechanism can be composed of separate subprocesses of RPA with specific operations in a logical, structured sequence or of a single mechanism (unmodularized) to perform the specific correction operations, this being also a process associated with an RPA type mechanism. The last proposed variant has the disadvantage that, at process level, it may contain sequences of redundant structures, due to the repetition of the elements related to defining

input variables and calling another mechanism (API application) to invoke the specific correction function depending on system needs. The ETL mechanism itself will be invoked through an API, thus being an integration to the monitoring application, in order to avoid overloading the basic structure of the application and to ensure a higher security of the whole mechanism, the integration with the API of ETL is secured by SOAP or REST settings via an SSL security key. Thus, the entire ETL subsystem is an API that encompasses another API (RPA process), or more (depending on possible connections to external applications or various needs of the monitoring application).

It is necessary that the monitoring mechanism has integrations with other API structures, depending on the complexity of the area of supervised e-business applications. One such example is integration with SAP, for a complex process of monitoring financial transactions and evaluating specific metrics. The ETL mechanism thus integrated will have as a component part, a database, in which there is input data, of the monitored application or system of applications, which will undergo transformations according to the specific criteria defined within the ETL functions. Following the application of corrections (data transformations, recalculations, process route changes, administration operations, etc.) the input data will become output data within the general system, the monitored application undergoing changes in the processes. At the next analysis of the monitoring system, new input data taken from the e-business application (or application system) will become analysis data, including for the ETL mechanism, the process being repeated automatically (with small interventions of administration, depending on the stability of the whole system). This process will be activated whenever a correction to the monitored application is required.

Taking into consideration the presented aspects, the ETL mechanism database will contain the following elements, in general:

- Table for input data:

1. Input data field as identification series (ID),
2. Process identification field,
3. Process description field,
4. Other fields related to the identification and description of the input data associated with the monitored infrastructure and application (server ID, CPU, memory, I / O, refresh rates, application-related transactions / requests, session IDs, transaction IDs SQL, database load level, other system processes).

- Table for applying the correction function (association with the required correction function):

1. Process identification field via a unique series (this series will link to the input table, this field being the primary key - having the same value as the process identification field in the input data table),
2. The field of the applied correction function, within the ETL API,
3. Correction field, recorded as a correction series,
4. Field for an additional RPA function,
5. Field for identifying an external process, realized through an API associated with an external application,
6. Fields related to the identification and description of all the above elements.

- Table for recording output data, following the application of the correction:

1. Field for recording a new series corresponding to the output,
2. Field for recording the code / solution applied as a correction method (this field will contain a text / string sequence that will be parsed / analyzed in a process related to the API, applying a JSON type transformation),
3. Field for recording possible process errors (this field will be an important record in an audit log, required in automation performance reports),
4. Other fields related to the description of the mentioned elements as well as other process elements (status of the completion of the process).

- Table for logging, following the execution of all processes related to collection, storage and modification / transformation, as well as data rewriting:

1. Log registration number field,
2. Field related to the type of log collected / recorded (process information, runtime error, administration),
3. Fields related to all processes performed under the ETL mechanism (sequence records, operation statuses, identifiers).

After writing the output data in the corresponding table, a process will be activated that will bring the necessary corrections to the monitoring application. After this operation, a process of revalidation of the data within the monitored application will be activated, according to the principle of the feedback and regulation mechanism within the system. In view of all the above, we propose the following scheme for a monitoring system, as presented in Figures 1 and 2:

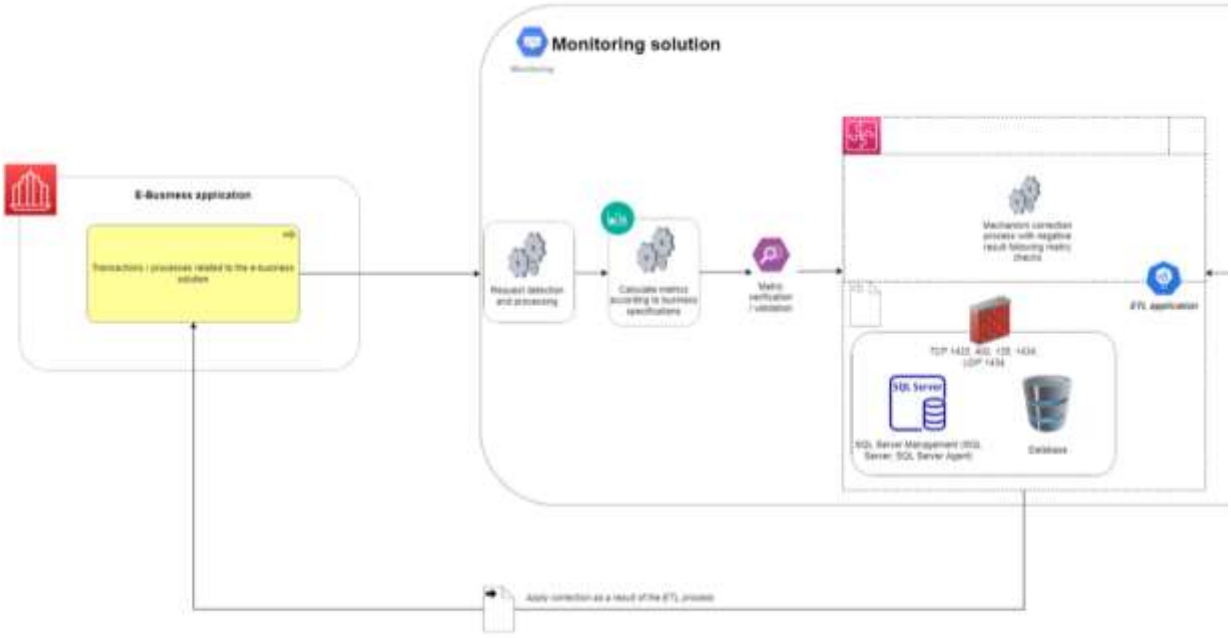

Fig. 1. Monitoring solution, with the related integrations (first part of the model)

As an additional note to the context of this research, there is the possibility that an e-business application may have, even within its structure, small components for monitoring and automatic correction of processes, but this aspect is not a viable scenario, in this case due to the fact that those inserted components will not be able to replace the activity of a complex monitoring function, especially within a complex system with multiple integrations. For this reason, for a complete and efficient monitoring, especially when it is necessary to apply some corrections on the processes, it is indicated to include a separate monitoring system, with the corresponding ETL mechanism. The process of applying the corrections within the ETL mechanism is very similar to a Machine Learning algorithm within AI (Artificial Intelligence). Moreover, this aspect can be interspersed with the effective application of an ML type algorithm, taking into account the fact that the whole system presented can reach a very high level of complexity, requiring special technical requirements related to cloud computing. Thus, depending on the degree of customization and adaptability of the developed system (depending on business requirements and implicitly the monitored application) we can discuss the implementation of the following elements of cloud computing technology: Google Cloud Fusion (Google Cloud Platform) - data storage in databases with a very high processing capacity, corresponding to a volume of data up to millions and tens of millions of records; data management through Big Query - related to Google Cloud Fusion;

data processing through pipelines developed within the Google Cloud Platform; implementing an infrastructure with multiple virtual machines (a viable solution could be Microsoft - Azure). Algorithms implemented for data processing in the cloud can be developed in Python, as a versatile method and adaptable to any system, especially in terms of automation. Thus, the calculation of relevant KPIs can also be done through libraries specific to working with data and statistical transformations (Pandas, NumPy, SciPy libraries within the Python language). Representation or visualization of data, including for processing related to ML algorithms, can be done through libraries and specific functions (Matplotlib, Seaborn, Plotly Express - used regularly). For Machine Learning the following libraries can be used as standard: TensorFlow, Keras, PyTorch, Theano, Scikit-Learn, XGBoost, LightGBM, Catboost, Dlib, Annoy, StatsModels, Pattern, Prophet, and view specifics are: Apache Superset, Matplotlib , Plotly, Seaborn, folium, Bqplot, VisPy, PyQtgraph, Bokeh, Altair.

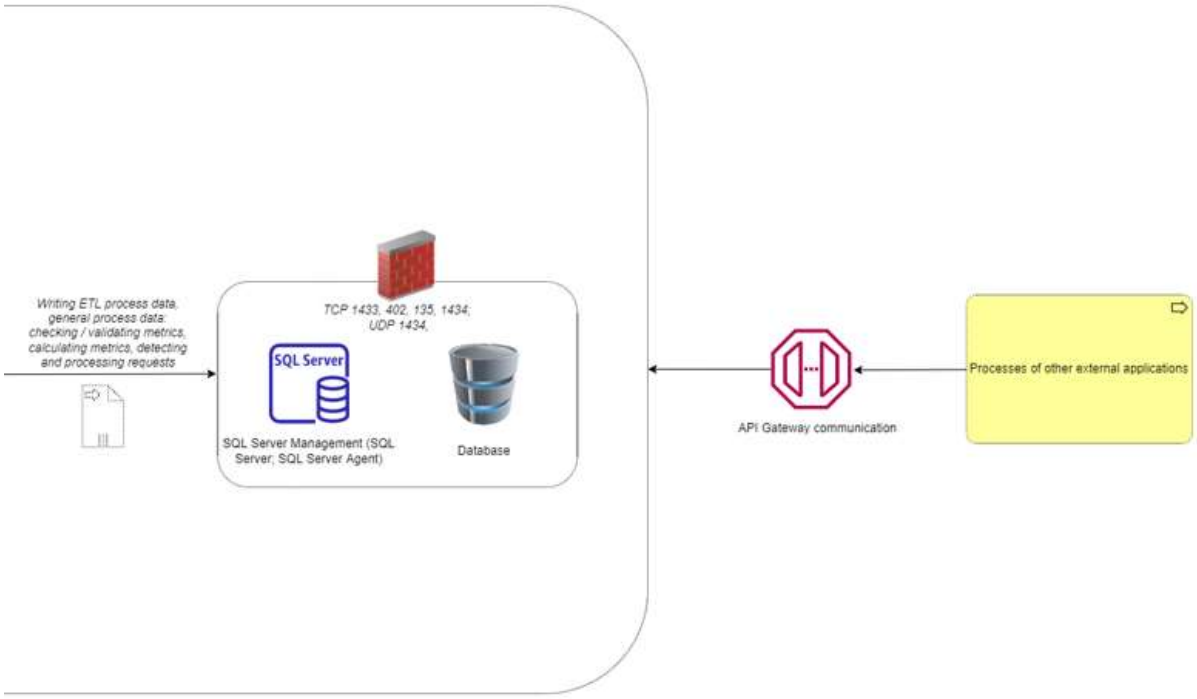

Fig. 2. Monitoring solution, with the related integrations (second part of the model)

It is important to note that, depending on the monitoring and transformations brought to the system (implicitly the business requirements) it will be determined if it is necessary to implement a technology related to ML and cloud computing, this variant being quite expensive both in terms of development and implementation. Another option, quite common for an ETL, is to use transformation algorithms that may involve standard frameworks for Python, PHP, SQL and even VBA. A whole series of automations can be achieved through optimized algorithms, using VBA, thus creating, from a simple process / macro in an Excel file, a true standalone application; the automation can be called by: Batch, Python and even PHP or ASP .NET. Depending on the context and the data introduced in the system, as well as the related processes, one can even use an ETL mechanism created in VBA Excel, as long as the entire monitored system is not complex. Also, at a low or medium level of system complexity, a standalone RPA application such as Blue Prism can be invoked. It should be noted that these ETL transformation processes may themselves have processing errors for the corrections to be applied, all data being collected both in the historical table and, in particular, in the audit log table (with more detailed information on actions / opera-

tions). Thus, errors can occur within the ETL system, at the process level, affecting all sections related to operations: input data recording, data processing (correction / transformation), output data creation to be applied to the monitored system. In this situation, the intervention of a system engineer is necessary, as there is no other application or metastructure that can bring additional corrections to regulate the whole ecosystem. Depending on the complexity of the context, the ecosystem may include the following elements: the monitored application or set of applications, the ETL subsystem within the monitoring system, converged external systems (SAP, integrations with various cloud solutions).

The entire monitoring system database will communicate directly with the ETL mechanism database and will contain important link elements (primary keys - identification fields) for the operation of the entire system. It is important that the two databases communicate in order to support the entire monitoring mechanism, through an optimal exchange of data, input and output. It is important to note that in a monitoring system (starting from the simple and reaching the complex, even more so within a complex system) there are many interconnected elements, on the principle of neural networks, the connections thus created can be considered as synapses between neurons, able to form a simulated cognitive process through mechanisms / algorithms associated with ML / AI (Machine Learning / Artificial Intelligence). The applied ML techniques can be: supervised, unsupervised and deep learning or advanced learning (applied to complex systems in which predictions are difficult to make). From a technical-functional point of view, at a physical level, this system has quite well interconnected elements, according to the principle stated above, but from a logical point of view, situations may arise in which the recorded data simulate a difficult model to anticipate and for which predictions cannot be made by ordinary statistical methods (linear and nonlinear regressions). Thus, there is a need to develop and implement machine learning algorithms.

In view of the above, we consider that a database for a monitoring system should contain the following structures:

1. Table for input data (with own elements as well as specific ones, coming from the ETL mechanism) - basically, this table contains, in general, the same data as the homonymous table under the ETL mechanism, with the addition of at least one field for the description of the elements of the system, plus a field for identifying and demarcating processes, so that it can be delimited the usual processes in the monitoring system, from those related to the ETL mechanism. The structure will have the following composition:

a) Input data field as identification series (ID),
b) Field for identifying the process within the monitoring system,
c) Field for describing the process within the monitoring system,
d) Other fields related to identifying and describing the input data associated with the monitored infrastructure and application (server ID, CPU, memory, I / O, refresh rates, application transactions / requests, session IDs, SQL transaction IDs, database loading level, other system processes),
e) Field detailing the specific elements of the monitoring system (infrastructure, application),
f) Field for demarcating processes within the general system (specific to the monitoring system and, separately, the ETL mechanism).

2. Table for recording output data, having the following:

a) Field ID of the process ID applied to correct erroneous functionality in the monitored system,
b) Fields for recording output data, each output record will have a status identifier, whether it is a correction process or not.
c) Fields related to the registration of all processes and how to apply / execute them.

3. Table for logging, following the execution of all processes related to the collection, storage and modification / transformation, as well as data rewriting - the structure is largely preserved as in the table of the same name within the ETL mechanism. The structure is composed of:

a) Log registration number field,
b) Field related to the type of log collected / recorded (process information, runtime error, administration),
c) Fields related to all processes performed under the ETL mechanism (sequence records, operation statuses, identifiers),
d) Application-specific process log fields (alert activation mechanisms - associated processes, page renderings, users, profiles, administration pages / sections with related operations, number of hits per page, refresh rates, HTTPS error status, string records connection, etc.).

Considering all the mentioned aspects, the two databases will have the following approximate structure / representation (database specific to the ETL mechanism / application; general database, specific to the monitoring application, with all the constituent elements), as presented in figure 3.

It is important to note that within the database server associated with the ETL mechanism, there are other databases associated with external processes, which could help streamline / optimize the processes within the general ETL mechanism. These databases associated with external processes correspond to integrations (API type) and represent intermediate structures for the main ETL mechanism. It is important that, in certain situations, there are separate databases for recording the complex processes of highly developed integrations (in terms of logic and technological requirements). As a rule, this measure is also taken if the volume of data involved is very large and the usual mechanisms for management / administration are not sufficient, as well as the related capacities. However, it is advisable, as much as possible, in order to not unnecessarily / redundantly load the database server with processes, to keep a single structure, a single database, in which all the data that enters the system can be stored and managed. The load level of the database server depends mainly on three aspects: the level of integration of the ETL mechanism with other elements necessary for the correction processes, the technological possibility to configure all the interdependencies at process level, within a single database, as well as the volume of data entering the system. As a principle, it is important that the main database associated with the ETL mechanism to be able to communicate, through specific ports and external mechanism firewalls, with the databases associated with the integrations and / or applications of all parties involved in the overall process. Here you can include: Oracle type DBMS, MariaDB, MongoDB, MSSQL DB, etc. It is also important to follow the basic principles of creating and defining database-type structures, tabular elements as well as linking elements, interconnections both within the main database and in relation to external databases.

Depending on the type of database used, we can determine which ports should be opened in the system, requiring the application / intervention of a database-specific firewall to secure data and transactions. For example, if MSSQL Server or any other type of SQL Server is used, there is a default mechanism for encrypting data and related transactions, called Transparent Data Encryption (TDE). You can also check the encryption status of the database system you are using, through a check in the sys table. dm_database_encryption_keys, the encryption status column will specify whether the system is encrypted or not (usually the system is encrypted). You can also use data encryption procedures at the column level so that the data set used is additionally protected (usually by performing the following actions: creating a master key for the database, creating a self-certificate -signed for SQL server, configure a symmetric key for encryption, encrypt data in columns, perform a query and verify encryption).

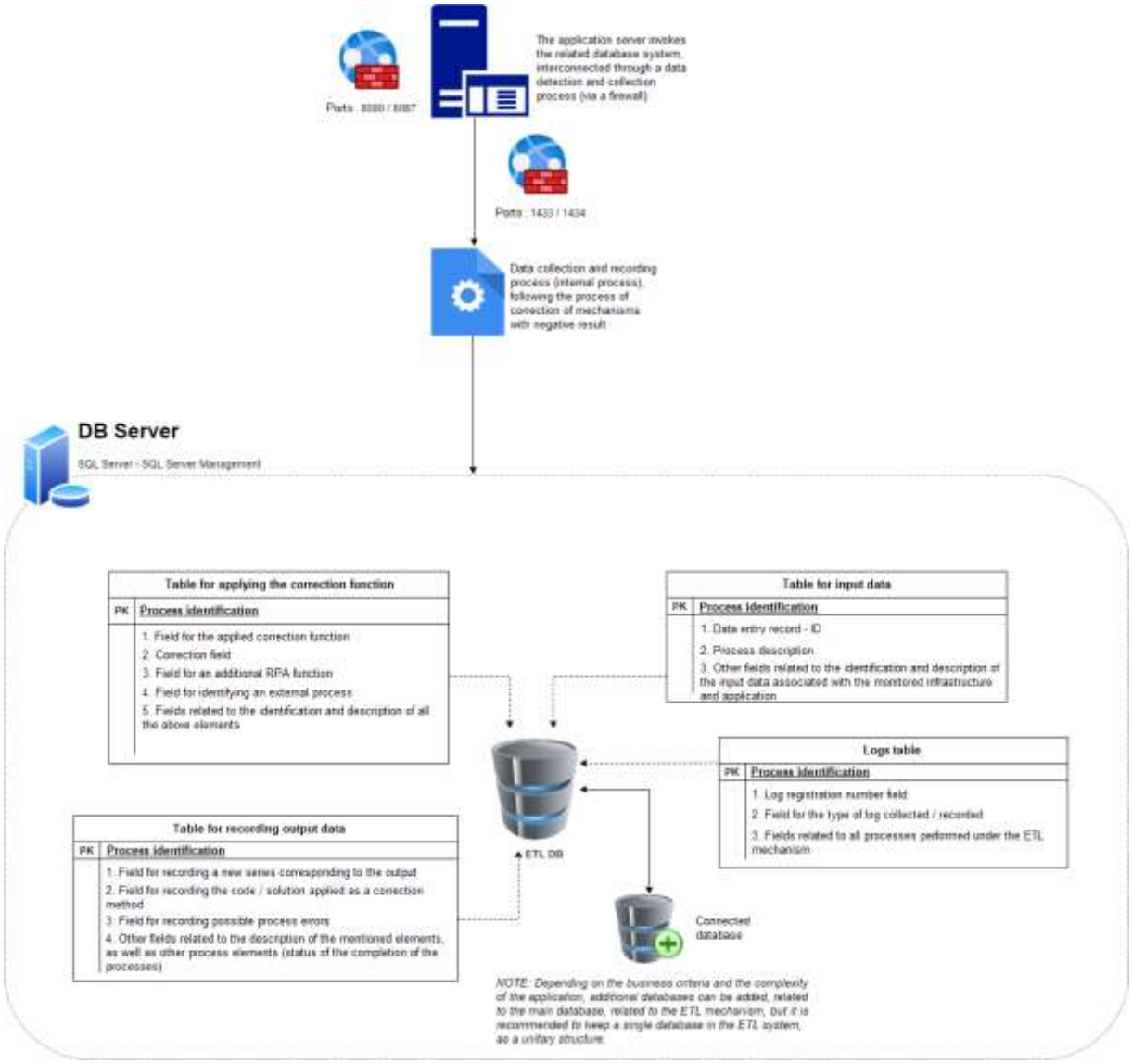

Fig. 3. ETL database server composition, main database description

A database-specific firewall monitors suspicious activity and intervenes in the following cases: detection of SQL injection attacks, detection of buffer overflow attacks and denial of service attacks. At the level of database ports, there is, by definition, a mechanism for securing data connection / transfers, via SSL (Secure Socket Layer), but it is recommended to use other ports besides the default ones, in order not to allow external intrusions by detecting the most used ports. As a rule, for the security of a database server and implicitly of a database system, it is recommended to install an agent to monitor the activity within them and to intervene through the protection mechanisms in case of detection of a threat (of a suspicious transaction, of an unidentifiable process or with a potential security risk). For example, MSSQL Server has an SQL Agent installed in SQL Server Management that performs the functions mentioned above.

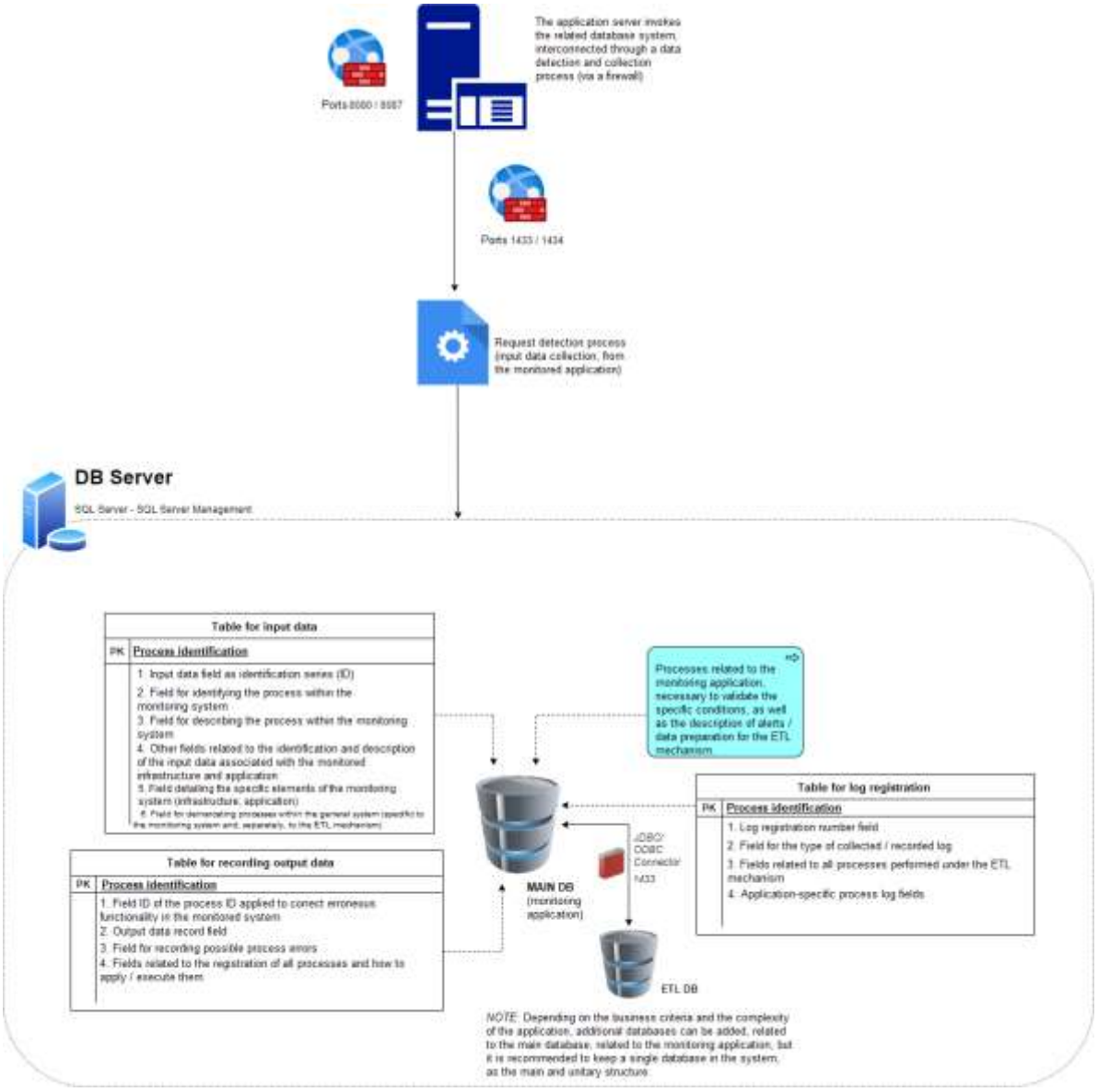

Fig. 4. Monitoring application structure diagram, with details of the connection between the application server and the two databases

Communication of the application server (Web Application Server) with the database server (SQL DB Server) can be done securely by applying well-defined firewall settings so that traffic can only be logged in certain directions or only through certain ports, it is preferable to set the traffic through less used static ports (1434) or by using dynamic ports (51688), so that its detection is more difficult to achieve. All these operations can be performed by a system engineer/ administrator and/ or a network engineer. It is important that these operations are done with the utmost care and thoroughness, as they can decisively influence the following: data transfer and processing, application stability and security, uncontrolled data flow, in this case, affecting the monitoring activity of the system. Please note that between standalone applications or APIs it is recommended to use security Gateways, which, through SSL keys, can create a default protected framework between databases as well. The two databases, the main one, the monitoring application, and the one specific to the ETL mechanism must be very well interconnected, through a precise and established communication following carefully defined criteria (both at the level of structure and in terms of security view: primary keys, data types, classification, re-

spectively firewall rules and security protocols). The general representation scheme of the monitoring system includes the following main elements: the application server related to the monitoring mechanism, the base server related to the main monitoring application as well as the database server related to the ETL mechanism. The application server invokes the related, interconnected database server through a process of detecting and collecting data through a firewall. Below (in figure 4) is part of the monitoring application structure diagram, with details on the connection between the application server and the two databases.

Within the general monitoring system there will be four servers: application server for the general monitoring application, database server (SQL Server) related to the monitoring application (main database of the monitoring application), application server for the mechanism ETL, as well as the database server (SQL Server) related to the ETL mechanism. It can also be mentioned that the monitoring application as a whole is the result of three processes: technological, business and the monitored application.

3 Experimental methods used for testing the model

In order to carry out a series of evaluations necessary for the complex monitoring operation, it is necessary to take into account a whole series of aspects related to the application development (in this case, using PHP technology), so that we still have an optimal procedural framework during the development of the monitoring application as well as the implementation. In view of these considerations, it is necessary to list the following aspects related to the development of the application in relation to the evaluation of metrics, which we consider valid:

1. Choosing a PHP server that can create a local host and provide all the running functionality of the technology, so you can start developing the application smoothly (recommended is the latest version of XAMPP - for Windows, LAMPP - for Linux or the following: MAMP, WAMP, AMPSS, Easy PHP).
2. Choosing technology-related library packages (in this case PHP, as it offers a wide range of possibilities for the monitoring part) as optimal / efficient as possible in terms of implementation and the possibilities it offers for various monitoring functions (request tracking, URL validity, SQL transactions, refresh rate, HTTP errors, etc.). Thus, a conclusive example for PHP could be OpenTelemetry. This library provides a collection of monitoring tools designed to help generate and collect web application telemetry data (metrics, logs, and traces).
3. Build or choose a framework that is as easy to use as possible for application development - this requires working tools such as: PHP compilers (among the most recommended are: PHPStorm, Netbeans, Eclipse, Visual Studio - with Xamarin, ZendStudio); a dependency manager for PHP (the Composer utility is recommended for installing the library); code testing utilities (recommended would be Ghost Inspector or SonarQube) as well as HTML compilers and editors (Coffee Cup HTML, Notepad ++, Visual Studio Code, Adobe Dreamweaver CC, etc.).
4. Modularized testing and revision of the code as well as the entire application (CoderByte, Frologic). For the PHP framework, the following are recommended: test management tools (TestRail, TestPad); automatic testing tools (Kobiton, Selenium); API testing tools (Soap UI, SOAP Sonar, WebInject); security testing tools (Simple Test, JUnit 5, HtmlUnit), cross-browser testing tools (Browsera, GhostLab); application load testing tools (Webload, Loadrunner); application testing tools on mobile phones (Espresso, Perfecto); CSS validation tools (W3C CSS validator, Telerik studio).
5. The choice of tools for testing the security of the framework and the final application is another important point, in this

case due to the general increase in the number of cyber-attacks and the multiple possibilities of fraud, corruption of the system and interruption of transactions / processes, with serious effects from the GDPR point of view and not only. Thus, the following useful tools for testing the security of a PHP application are valid (recommended): Simple Test, JUnit 5, HtmlUnit, DataDog, SonarQube, PHP Malware Finder, RIPS, SonarPHP, Exakat, PHPStan, Pslam, Progpilot, PHP Vulnerability Hunter, Grabber, Symfony, etc.

6. After each unit testing and modules development, it is important to revise the business plan and adapt, if needed, the business case (KPIs, procedures, etc.). For alerting mechanisms, tests can be performed using the following platform integrations: Grafana, Graphite and Monday. Grafana could provide relevant metrics about the infrastructure, while Monday could integrate IPCC framework related elements, rules creation being intuitive into this platform. Also, Monday as well as Grafana platforms, could integrate other elements. Thus, Monday could enhance the monitoring platform with integrations such as: Jira (for Agile ways of working), OneDrive, GitLab (platform enhancements through code developments repositories), Zendesk and Eventbrite (for events and ticket priority rules management). Integration with approved and secure platforms, such as Zendesk, could also represent a premise for ticket status event log details backup, in case of some major disruptions in the main monitoring system. On the other hand, Grafana could integrate elements from: Apollo Server, CloudWatch Metrics, Confluent Cloud, CoreDNS, MongoDB, MySQL, PostgreSQL, Custom Logs, Google Cloud Monitoring, for enhancing the data management part of the main monitoring system.

4 Conclusions

In the creation of a monitoring system for e-business solutions it is important to highlight not only domain related aspects (metrics, alerts, rules, conditions) but also to take into consideration the interconnections that can be realized with other applications/ systems from other areas of expertise (marketing, retail industry), for the creation of a monitoring ecosystem.

An efficient monitoring system could enhance productivity in an IT system (e-business application or set of applications), through the evaluation and re-evaluation of the IPCC framework indicators (Incident, Problem, Change, Configuration). Thus, monitoring could help reducing the number of User Service Restoration tickets and the creation of reactive work orders. Efficient Problem Management tracking, through proactive PIRs (Problem Investigation Records) could significantly raise the availability of an e-business application, reducing the number of interruptions in the system, thus creating an increase in the customer satisfaction.

AI/ ML solutions represent a prerequisite for the implementation of an effective ETL mechanism into a monitoring solution for e-business applications. The ETL mechanism must be adapted to suit the specific business case of an e-business application, thus creating a personalized framework for proactive detection of errors and eventual correction into the system.

In the creation of a model for monitoring solutions, applied on e-business applications, it is important to take into consideration every kind of technological and business prerequisite, as well as possible limitation, in order to create a functional supervision framework. Since e-business solutions are increasingly competitive and thus growing in complexity, it is important to take into consideration specific business criteria for the development of the monitoring system. Thus, adaptable and custom specific functions must be created for a standard and functional monitoring system. It is very difficult to create a general monitoring solution that could incorporate functionalities and characteristics relevant for a

large area of e-business applications, thus a customizable and adaptable platform could represent the key. Through the latest developments in AI/ ML field, an auto-scalable and adaptable platform could be created to meet, as much as possible, the requirements of a vast majority of e-business applications, but this is, still, the subject of research.

Since, in general, metrics are very difficult to analyze in complex e-business applications, due to the large set of demands and rapidly changing internet environments, some relevant estimations can be provided through the collection of live metrics (managed at event logs level), synthetic monitoring and periodic measurements with trend analysis and forecast. These cumulative measurements, along with business specific indicators, could prevent serious system interruptions in e-business platforms/ applications, but could also help in predicting user behavior and even business disruptions, in some cases.

As a prerequisite for creating a monitoring system, relevant integrations must be suited/ adapted to the main monitoring application (SAP, Data Analytics platforms, Trading platforms, Ticketing, etc.) in order to enhance the metrics of the platform, through API usage, if possible. An in-depth analysis must be conducted for choosing and validating the best suitable technologies for platform integrations, since, especially PHP framework offers a variety of monitoring and integration possibilities.

A stable monitoring system would require an enhanced SSL protection, since security vulnerabilities could represent a high risk or threat, especially in the case of a complex system, using large volumes of data. In this case, traditional security layers and data encryption algorithms, must be covered by enhanced security solutions like enhanced DES (Data Encryption Standard) [9].

In a monitoring system it is important not to rely on integrations to accomplish main functionalities but to consider them as an enhancement in the distributed applications world, meant to interconnect multiple parties, in order to receive more relevant and comprehensive monitoring results.

References

[1] A. Menychtas, J. Vogel, A. Giessmann, A. Gatzioura, S.G. Gomez, V. Moulos, F. Junker, M. Müller, D. Kyriazis, K. Stanoevska-Slabeva and T. Vargariou. (2014). 4Caast marketplace: An advanced business environment for trading cloud services. *Future Generation Computer Systems* [Online]. 41, 104-120.

[2] A. Chen, P.B. Goes and J.R. Marsden. (2002/2003) A Query-Driven Approach to the Design and Management of Flexible Database Systems. *Journal of Management Information Systems* [Online]. 19(3), 121-154.

[3] W.E. Agin.. (2017). A Simple Guide to Machine Learning. *Business Law Today* [Online]. pp. 1-5.

[4] X. Yang, B. Nasser, M. Surridge and S. Middleton. (2012). A business-oriented Cloud federation model for real-time applications. *Future Generation Computer Systems* [Online]. 28, 1158-1167.

[5] J. Kim and M. Comuzzi. (2021). A diagnostic framework for imbalanced classification in business process predictive monitoring. *Expert Systems with Applications* [Online]. 184, 115536.

[6] D. Loreti, F. Chesani, A. Ciampolini and P. Mello. (2008). A distributed approach to compliance monitoring of business process event streams, *Future Generation Computer Systems* [Online]. 82, 104-118.

[7] T. Bayrak. (2021). A framework for decision makers to design a business analytics platform for distributed organizations. *Technology in Society* [Online]. 67, 101747.

[8] K. Zbikowski and P. Antosiuk. (2021). A machine learning, bias-free approach for predicting business success using Crunchbase data. *Information Processing and Management* [Online]. 58, 102555.

[9] A.V. Ryndel, S.M. Ariel and M.P. Ruji, "Enhanced Data Encryption Standard (DES) Algorithm based on Filtering and Striding Techniques," *in Proc. of the 2nd International Conference on Information Science and Systems*. Association for Computing Machinery, New York, USA, 2019, pp. 252–256.

Alexandru COCIORVA has graduated the Faculty of Economic Cybernetics, Statistics and Informatics in 2010. He is currently PhD student, in the third year of studies, at the Economic Informatics Doctoral School, from Bucharest Academy of Economic Studies. He has an overall experience in the IT field of 8 years on roles such as: IT System Engineer, Business Analyst, MS Service Engineer, ETL Deployment Engineer and ICT DevOps Engineer. He is currently working as a Data Scientist in a multinational company. He has published scientific articles on topics related to monitoring systems for e-business solutions. His research interests are focused on, but not limited to monitoring systems and metrics, IPCC (ITIL) framework applied in IT solutions.