INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Intelligent Batch Processing System in a Complex Distributed Computer Environment

MARCO ANDRÉ MARINHO PINHEIRO
Outubro de 2023

POLITÉCNICO
DO PORTO

Intelligent Batch Processing System in a Complex Distributed Computer Environment

Marco Pinheiro

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Informatics Engineering

Supervisor: Jorge Santos

Evaluation Committee:
President:
Dr. Professor, DEI/ISEP

Members:
Dr. Professor, DEI/ISEP

Porto, October 13, 2023

Abstract

In the rapidly evolving digital era, the ability to manage and process vast amounts of data has become paramount. This challenge has promoted the use of batch processing in distributed systems, an approach integral to activities like in-depth data analysis, warehousing, and harnessing business intelligence. Within this domain, Retail Consult stands as a notable entity, finding numerous retail solutions via Oracle Retail technology. These solutions are grounded in rigorous daily and nightly batch processing schedules, where precise timing is critical to ensure smooth business operations.

However, the current landscape of batch processing, although populated with a variety of solutions, struggles with the magnified scale and intricacy of today's data. Challenges emerge in the form of configuration nuances, real-time execution demands, consistent updates, and the need for vigilant monitoring. Retail Consult's diverse client base and unique technological specifications only amplify these challenges.

To navigate this intricate landscape, Retail Consult, in partnership with the Laboratório de Inteligência Artificial e Ciência de Computadores (LIACC) of Faculdade de Engenharia da Universidade do Porto (FEUP), aspires to innovate an intelligent process scheduler — a solution designed for adaptability. This scheduler, in spite of the multifaceted demands of the Complex Distributed Computer Environments, aims to harmonize it and promises compatibility across a diversity of platforms, the capability to manage geographically disparate operations, and the sophistication to oversee a gamut of automated tasks. Set to be a reference in batch processing, the ultimate goal of this system is to orchestrate these critical business processes with unparalleled efficiency and effectiveness.

Keywords: Batch processing, Distributed Systems, Job Scheduler, Oracle Retail Technology, Data Management, Intelligent Systems, Retail Solutions

Resumo

Na rápida evolução da era digital, a capacidade de processar e gerir a crescente quantidade de dados é fundamental. Este desafio levou ao desenvolvimento e uso do processamento em batch em sistemas distribuídos. Uma abordagem integral em atividades como análise de dados aprofundada, armazenando e beneficiando de inteligencia empresarial. Neste contexto a Retail Consult apresenta-se como entidade de renome, encontrando inumeras soluções na área de venda a retalho através da Tecnologia Oracle Retail, em conjunto com os seus clientes. Clientes estes que têm necessidade de um processamento em *batch* de forma diária e noturna. Este conjunto de processos são críticos para o negócio, e os seus tempos de execução/termino são fatores determinantes para assegurar o melhor fluxo do mesmo.

No entanto, o panorama atual de processamento em *batch*, apesar de populada por uma variedade de soluções, confronta-se com o grande volume e complexidade dos dados gerados atualmente. Inumeros desafios surgem na forma de falhas na configuração, exigências de execução em tempo real, *updates* contínuos e a necessidade de monotorização. A base diversa de clientes da Retail Consult e as suas especificações tecnológicas apenas amplificam estes desafios.

De forma a culmatar e agregar valor para os seus clientes, a Retail Consult, no ambito de um projeto de Investigação e Desenvolvimento Tecnológico (IDT) e em parceria com o LIACC da FEUP, tem como objetivo desenvolver um escalonador de processos inteligente e adaptável às mudanças da estrutura do negócio. Este sistema, apesar das inumeras demandas do *Complex Distributed Computer Environments*, visa harmoniza-lo e garante compatibilidade através das diferentes plataformas, a capacidade de organizar geograficamente diferentes operações e de supervisionar grandes quantidades de tarefas automatizadas. Tido como uma referência no processamento de *batch*, o objetivo principal do sistema é orquestrar estes processos de negócio de caráter crítico com efeciência e eficácia incomparável.

Acknowledgement

This work has been partially supported by the project “IBPS - Intelligent Batch Processing System” , with the reference POCI-01-0247-FEDER-069998, co-financed by the European Regional Development Fund (ERDF), through the Operational Programme for Competitiveness and Internationalization (COMPETE 2020), under the PORTUGAL 2020 Partnership Agreement.

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligent.
ALLOC	Retail Allocation.
BIP	Business Intelligence Publisher.
CODINE	Computing in Distributed Networked Environments.
DQS	Distributed Queueing System.
FEUP	Faculdade de Engenharia da Universidade do Porto.
HPC	High-Performance Computing.
IBPS	Intelligent Batch Processing System.
IDT	Investigação e Desenvolvimento Tecnológico.
IPE	Intelligent Process Engine.
JMS	Job Management System.
LIACC	Laboratório de Inteligência Artificial e Ciência de Computadores.
LL	LoadLeveler.
LSF	Load Sharing Facility.
NAS	Numerical Aerodynamic Simulation.
NFR	Non-Functional Requirements.
NQE	Network Queueing Env.
NQS	Network Queueing System.
ODI	Oracle Data Integrator.
PBS	Portable Batch System.
QFD	Quality Function Deployment.
ReIM	Invoice Matching.
RESA	Sales Audit.
RIB	Retail Integration Bus.
RMS	Retail Merchandising System.

RPM Retail Price Management.

Chapter 1

Introduction

The growing complexity of the digital world, including the increasing amount of data, the proliferation of connected devices, and the emergence of new technologies, accelerated the development of more complex software applications. This has led development and use of batch processing, which is a method of processing large sets of data in a specific order, typically in non-interactive mode. This allows for the efficient and orderly processing of large amounts of data, and it is particularly useful for tasks such as data analysis, data warehousing, and business intelligence.

The use of batch processing has become an essential tool in the modern data-driven world, allowing businesses and organizations to extract insights and knowledge from large and complex data sets.

Increasing data volumes and the growing complexity of data analyses, create new demands for batch processing on distributed systems. Effective operation of these systems is challenging when facing uncertainties about the performance of jobs and tasks under varying resource configurations (Witt et al. 2019).

1.1 Context

In the digital age, businesses increasingly rely on data-driven insights. This reliance necessitates the development of systems capable of processing and analyzing large volumes of data efficiently. These systems, responsible for executing numerous jobs, must have robust active monitoring, effective dependency, and criticality management, adherence to strict time constraints, and the ability to autonomously recover from a significant proportion of adverse technological incidents. Furthermore, these systems need to be agile, capable of swiftly adapting to new conditions and promptly responding to changes in the business environment (Ammon et al. 2010).

Batch processing, the execution of a group of batch programs or jobs, has become crucial due to the need to process large volumes of data, interface with external systems, and perform internal maintenance. Batch programs can swiftly process large quantities of data with minimal impact on system performance, often scheduled to run during idle or low-activity periods (Young 2021).

Scheduling software allows jobs to be executed/configured in a specific order, respecting inter-process restrictions. If a job fails, an administrator must identify and rectify the error, then manually execute the failed program (Young 2021).

The specific challenge in the context of Retail Consult, as discussed in the problem section, managing these complex batch processing tasks within a distributed environment poses unique challenges, due to their diverse client base and specific technological requirements.

1.2 Problem

At Retail Consult, a multitude of retail tool packages are deployed using Oracle Retail technology, which necessitates daily and nightly batch processing. These processes, indispensable to Retail Consult's clients, must conclude before the commencement of each business day.

The batch processing in Oracle Retail is composed of a set of jobs with diverse functionalities. These batch programs can be categorized into several types:

- Upload programs that import data from external systems into the database;
- Download programs that extract and format data from the Retail Management System (RMS) for use by other programs;
- System maintenance programs that perform functions such as updating the system date;
- Functional maintenance programs that process data specific to a functional area (Young 2021);

Furthermore, the operation of these batches adheres to a strict schedule, called the batch window, during periods when online systems are least active. The order of task execution is critical in this process (Young 2021).

Each product is executed in phases by agents on specific machines, respecting job dependencies to maintain data integrity. Jobs are uniquely configured for each client, aligning with their business objectives (Young 2021).

Each batch can comprise approximately 300-900 programs belonging to different applications of the Oracle Retail suite, namely:

- Retail Allocation (ALLOC);
- Invoice Matching (ReIM);
- Retail Merchandising System (RMS);
- Retail Price Management (RPM);
- Sales Audit (RESA);
- Retail Integration Bus (RIB);
- Business Intelligence Publisher (BIP);
- Oracle Data Integrator (ODI);

The following diagram gives a high-level overview of the processing phases of the RMS batch cycle:

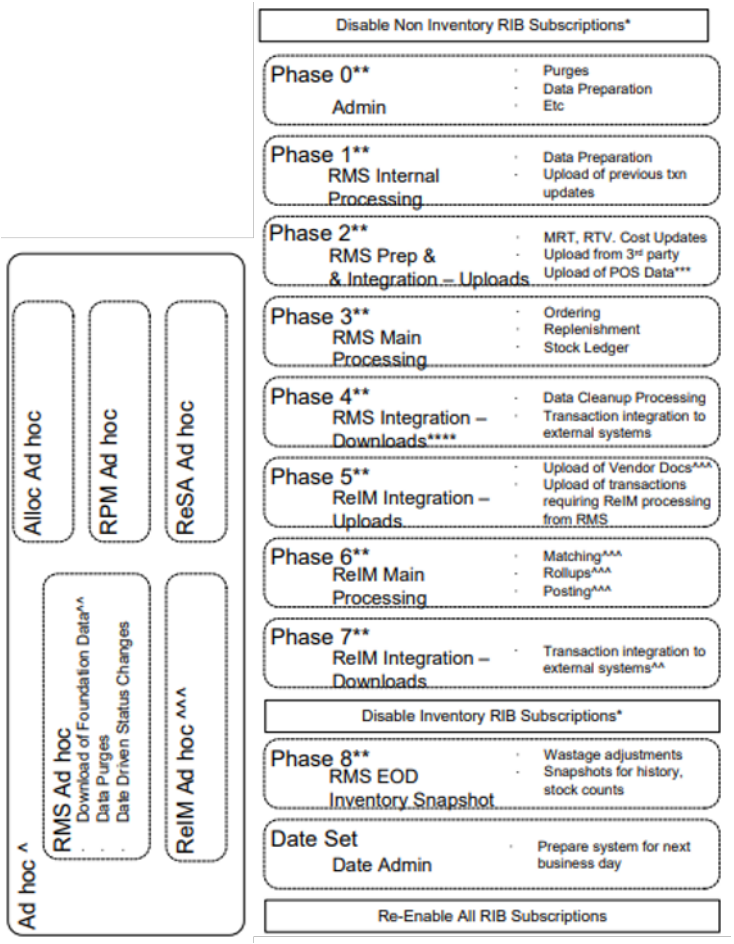

Figure 1.1: Integrated Merch Batch Cycle Phase Overview (Oracle 2018)

However, the problem escalates due to the following complexities:

- The jobs are written in varying technologies (Java, KornShell, and C), making them challenging to manage, schedule, monitor, and execute.
- Some jobs are compiled code, producing executable files, while others aren't, causing inconsistencies in their operation and outputs. These inconsistencies make it tough to monitor job status and performance effectively and complicate troubleshooting.
- These jobs might also need integration with other applications, adding to the intricacy of the batch processing orchestration.

Addressing these complexities requires a sophisticated software solution. Specifically, a prototype equipped with:

Job Scheduler Intelligent Process Engine (IPE): To manage the order and execution of varying jobs across diverse technologies.

User Interface: To seamlessly integrate Artificial Intelligent (AI) modules with the Job Scheduler, enhancing adaptability and efficiency.

The envisioned AI modules are:

- **Configurations**: Enables editing and validation of active batch configurations and the generation of batch scheduling solutions.
- **Scheduling**: Allows for the selection and analysis of batch scheduling solution characteristics. It should also permit solution submission for the next batch's execution.
- **Forecasting**: Offers a view of batch job execution history and forecasts, incorporating dynamic customer context variables used in training forecasting models.
- **Management/Disruptions**: Facilitates monitoring and management of batch execution, including error detection, lock detection, and unexpected job behavior, and suggests potential corrections.
- **Monitoring**: Components provided by Grafana to observe and track the batch processing metrics.

1.3 Objectives

The primary objective of this dissertation is to develop and validate a software solution that addresses the challenges faced by Retail Consult in managing, scheduling, monitoring, and troubleshooting diverse batch processing jobs. This solution aims to not only simplify these processes but also to enhance them through the integration of AI modules.

The specific objectives to achieve this are:

1. Design and develop a prototype that includes:
    - A user interface for seamless monitoring of batch processing.
    - Integration capabilities with a job scheduler to manage job order and execution.
2. Incorporate existing AI modules—Optimization, Disruptions, and Forecasting—into the prototype, ensuring their synergy with the Intelligent Batch Processing System (IBPS).
3. Validate the prototype in the Oracle Retail environment, ensuring it offers dynamic scaling and effective monitoring.

1.4 Expected Results

This project aims to develop a prototype that not only addresses the complexities of batch processing but also enhances the user experience and system performance. The success criteria for the prototype are as follows:

- **User Experience:** The interface should be intuitive and user-friendly, allowing administrators or users to seamlessly monitor and interact with the system. Moreover, it should be accessible from various devices and support diverse user interaction modes.
- **Performance:** The prototype should optimize system resources, ensuring efficient batch processing. The Optimization and Forecasting modules should collaboratively enhance performance and resource allocation.
- **Reliability:** The system should be dependable, with the Disruptions module identifying and mitigating potential issues to ensure smooth operations.

1.5 Document Structure

This document is organized into the following chapters:

- **State of the art-** presents the review of the current state of the literature regarding other job schedulers that already exist and are mentioned;
- **Value Analysis -** describes and evaluates the opportunities and value of the solution proposed with this dissertation;
- **Problem Analysis and Solution Design -** presents the project's functional and non-functional requirements, as well as the architectural design;
- **Implementation -** details the implementation process, mentioning the system architecture, development environment and technologies used, and the implementation details;
- **Testing -** demonstrate the tests designed to verify the functioning of the prototype by running a batch;
- **Conclusions -** presents this project's achieved objectives and what are its limitations. It also mentions some aspects that can be addressed in the future to improve the work that was done;

Chapter 2

State of the art

Job scheduling solutions are critical software tools used in batch processing to manage the execution of jobs in a computing environment. There are several job scheduling solutions available in the market, each with its unique set of features and capabilities. In this section, we will discuss the state of the art of job scheduling solutions by comparing software tools that are frequently mentioned in a NASA study, users' highest-rated software using the G2 methodology, and software widely used in High-Performance Computing.

2.1 Job scheduling solutions

Batch processing scheduling solutions are software tools used to automate repetitive batch jobs, typically performed by organizations for large data processing, database updates, report generation, and other time-consuming tasks.

The aim is to improve efficiency, and accuracy, and minimize manual intervention, ensuring batch jobs run smoothly and on time. These solutions typically include job scheduling, resource allocation, load balancing, error handling, and reporting.

The NASA Ames Research Center's Numerical Aerodynamic Simulation (NAS) supercomputer facility has been searching for a robust Job Management System (JMS) to support parallel jobs. NAS produced the NAS Requirements Checklist for Job Queuing/Scheduling Software, which provides a baseline set of requirements for JMSs, based on input from various organizations and agencies. This report evaluates leading JMS systems based on the checklist (Jones and Brickell 1997).

Recent studies highlight the importance of adaptability, scalability, and resilience in job scheduling solutions. With the increasing complexity of IT infrastructures and the diverse nature of computational tasks, it's imperative for job scheduling solutions to be both robust and flexible. For instance, the integration of artificial intelligence and machine learning techniques in job schedulers can optimize resource allocation dynamically based on real-time system states and predictive analytics (Smith and Doe 2021).

Several popular software are available to assist businesses and organizations in managing growth and expansion. The best software for a particular organization depends on factors such as its infrastructure size, the applications it runs, and its budget.

In section 2.5 provides a detailed comparison of the selected tools, highlighting their key features.

2.2 Mentioned in NAS

Historically, schedulers have been present for decades, some of them mentioned in the NAS study, which continues to be used today despite constant changes over time.

According to NAS the job scheduling solutions in the following Table 2.1 were evaluated in the report.

Most mentioned tools
Computing in Distributed Networked Environments (CODINE)
Distributed Queueing System (DQS)
LoadLeveler (LL)
Load Sharing Facility (LSF)
Network Queueing Env (NQE)
Portable Batch System (PBS)

Table 2.1: The most mentioned system in NAS

Figure 2.1 is an adaptation from the (_UEUEING 1994), providing a graphical depiction of the relationships between the different systems. The Network Queueing System (NQS) is considered the progenitor of all the systems. The system is still in production and development if an arrow continues outward. If a system feeds linearly into another, it indicates a name change. If a system branches from another's line, the new system has evolved from the former one.

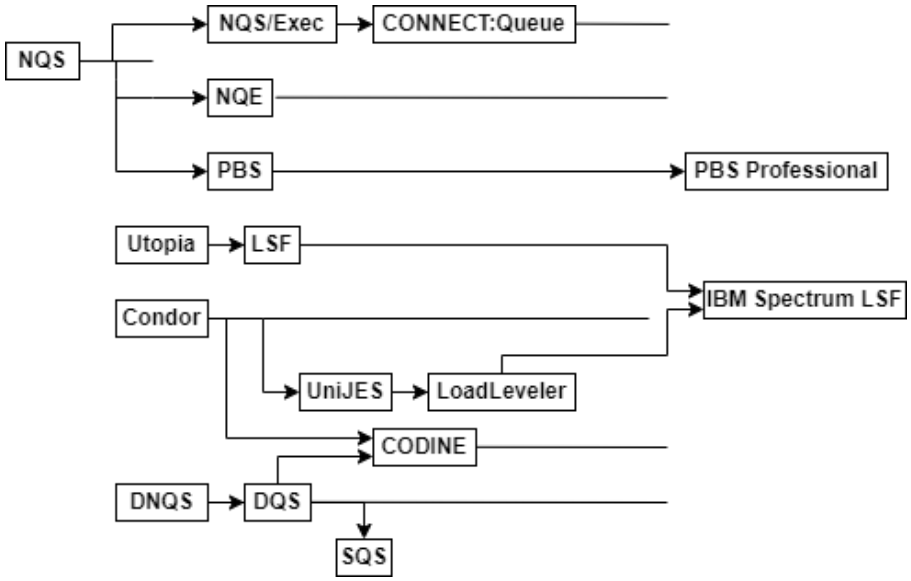

Figure 2.1: Evolution

Of these systems, LL and LSF are both still available and marketed by IBM under the IBM Spectrum LSF product line. The Portable Batch System is also still available and is now known as PBS Professional.

CODINE, DQS, and NQE are also mentioned in the report, but it is unclear if they are still available on the market.

2.3 Highest Rated

The methodology used by G2 (G2 2023) for evaluating software is called the G2 Research Scoring Methodology. It is a comprehensive and objective approach that takes into account multiple factors to determine the overall score of a software product. This methodology is designed to be fair, comprehensive, and objective, taking into account a variety of factors to provide an overall score for each product.

The evaluation process begins with the collection of user reviews and ratings, which are used to generate an overall score for each product. The scores are based on a combination of factors, including ease of use, features and functionality, customer support, and the product's overall value.

In addition to user reviews, G2 also considers the market presence and the quality of the product, which includes factors such as the vendor's size and stability, as well as the level of innovation and product development.

Exclusion criteria are not explicitly stated by G2. However, G2 may exclude certain products from consideration if they do not meet their criteria for inclusion, which include having a minimum number of user reviews and meeting certain standards for user engagement, and reviewing quality and the product must be commercially available and actively marketed or have insufficient information available to evaluate.

Overall, the G2 Research Scoring Methodology is designed to provide a fair and objective evaluation of software products, based on a combination of user feedback, market presence, and product quality. The methodology is regularly updated to reflect changes in the market and to ensure that G2's evaluations remain relevant and accurate.

Based on the G2 Research Scoring methodology, which takes into account factors such as user satisfaction, market presence, and product quality, these three solutions presented in the following Table 2.2 were selected.

Highest Rated tools	**G2 Satisfaction Score**
Redwood Software	93%
JAMS Enterprise Job Scheduler	90%
ActiveBatch Workload Automation	88%

Table 2.2: Highest Rated software in G2

2.4 High-Performance Computing

High-Performance Computing (HPC) refers to the use of supercomputers and other high-end computer systems to solve complex computational problems that would be impractical to solve with traditional computing methods. HPC systems are designed to handle large amounts of data and perform calculations at extremely high speeds, making them ideal for a wide range of scientific, engineering, and business applications. HPC can generate insights that would not be possible through other means, and simulations can replace or augment

hazardous or expensive experiments. It also has the potential to suggest new experiments that would not have been possible to conduct otherwise (Vetter 2017).

The focus on HPC often centers around its largest architectures and benchmark results like TOP500, but the support from the international scientific and engineering community is much broader. This community has spent decades developing scientific simulation methods and software, which now serve as the foundation for HPC (Vetter 2017).

The capacity to process large volumes of data is a key distinguishing factor for job scheduling software. In NAS, the evaluated schedulers were used in HPC environments, which is a critical factor to consider.

The graphical representation of the selected software options that utilize HPC can be seen in Figure 2.2, which includes the software mentioned in 2.2 and 2.3. It's important to note that Redwood Software was not selected for comparison, as it does not fulfill the criteria for inclusion based on HPC usage.

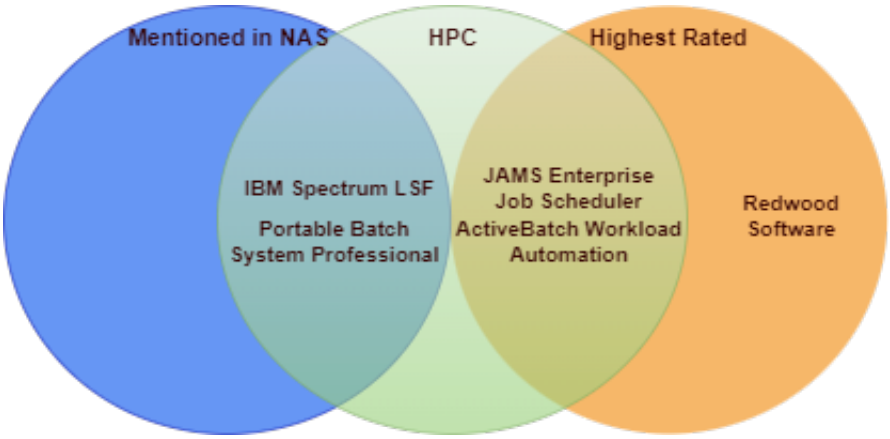

Figure 2.2: Selection of software used in HPC

2.5 Software Comparison

Job scheduling software has become a vital tool for many organizations to manage and automate their business processes. With an increasing number of options available, choosing the right job scheduling software can be a daunting task for organizations. This subsection of the thesis aims to provide a comparative analysis of several job scheduling software options currently available on the market. This analysis takes into account the selection process that was previously conducted to narrow down the options.

According to "A Taxonomy of Job Scheduling on Distributed Computing Systems" (Lopes and Menascé 2016), the components of a scheduling problem in distributed computing systems include workload, resources, and scheduling requirements, and these were utilized after selecting the software for comparison.

Workload: This component defines the consumers of the resources and is composed of jobs, which are collections of computational tasks. Each job has a certain number of tasks that need to be executed.

Resources: The resources component consists of a set of distributed nodes or computers that are required to execute the workload. These resources are connected by a high-speed

network and may be organized in computing clusters or data centers. Nodes can only communicate through message exchange, and each node has one or more processing cores, main memory, storage devices, and network access.

Scheduling requirements: This component determines the scheduling goal and other requirements that must be met by the solution. The scheduling goal is usually to optimize one or more performance metrics affected by scheduling decisions. Other important scheduling requirements include the scheduling level, which determines the granularity or level of detail considered when making a scheduling decision. There are two levels of scheduling decisions: job and task.

The proposed taxonomy characterizes a scheduling problem (2.3) consisting of 17 static features that fall into three groups: workload, resources, and scheduling requirements.

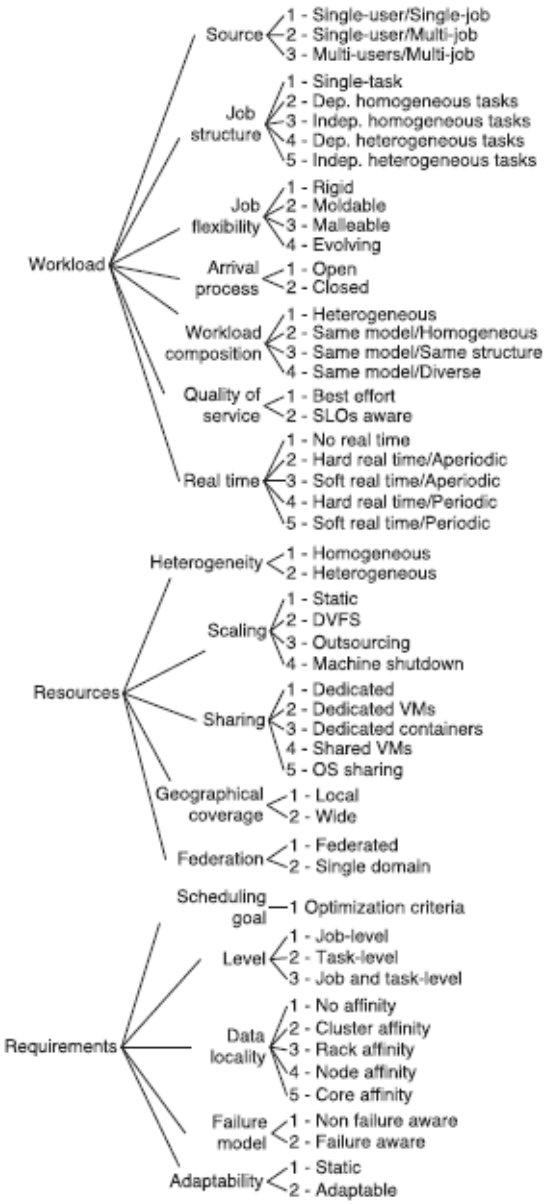

Figure 2.3: Summary of static features related to a scheduling problem (Lopes and Menascé 2016)

Workload Description:

- Job source: Determines if the workload comes from a single user or multiple users and if it consists of single or multiple jobs.
- Job structure: Defines the number of tasks per job, the dependency relations among tasks, and communication needs. This feature distinguishes between single-task, independent homogeneous multi-task, independent heterogeneous multi-task, dependent homogeneous multi-task, and dependent heterogeneous multi-task jobs.
- Job flexibility: Determines whether jobs have fixed, moldable, malleable, or evolving computing requirements.
- Arrival process: Specifies whether the workload is open or closed.
- Workload composition: Determines the programming model and the types of relationships among tasks. Workload composition may be homogeneous or diverse, depending on the similarity of the jobs in terms of their structure, number of tasks, and resource demands.
- Quality of service: Specifies whether jobs are associated with service level agreements (SLAs) and require service level objectives (SLOs) to be met.
- Real time: Specifies whether the workload consists of real-time or non-real-time jobs, and whether the tasks are periodic or aperiodic

Resource Description:

- Resource heterogeneity: Resources may be homogeneous or heterogeneous in terms of processing power, storage, and networking capabilities.
- Resource scaling: Resources may have fixed or dynamic infrastructure in terms of processing capacity, and may allow rapid capacity changes in response to variations in the workload.
- Sharing: Resources may be shared among different jobs, either through dedicated nodes, virtual machines, containers, or operating system sharing.
- Geographical coverage: Resources may be local or wide depending on the geographically coverage of its nodes.
- Federation: Resources may be federated, shared among different administrative domains in a coordinated fashion, which brings new challenges to scheduling in terms of security, geographical issues, and the opportunistic usage of idle resources.

Scheduling Requirements:

- Scheduling goal: The objective of the scheduling solution is to maximize or minimize a particular metric or multiple metrics of interest, such as makespan, resource utilization, throughput, deadlines, or energy costs.
- Scheduling level: The level at which scheduling is done, which can be at the job level, task level, or both.
- Data locality: Whether the scheduling policies take into account the location of data to optimize the scheduling decisions. This can include cluster affinity, rack affinity, node affinity, and core affinity.

- Failure model: Whether the scheduling solution is designed to be failure-aware or non failure-aware, which can impact the need for task reassignments in case of node failures.
- Adaptability: Whether the scheduling policies can adapt to changes in workload or resources, allowing the solution to adjust to changing conditions.

Table 2.3 provides a comprehensive comparison of four prominent job scheduling software options: JAMS Enterprise Job Scheduler, ActiveBatch Workload Automation, IBM Spectrum LSF, and PBS Professional. Based on the features highlighted:

- All four software options support both single and multiple users.
- Each software is equipped to handle various job structures, demonstrating flexibility in job scheduling.
- While all the software options offer moldability in terms of job flexibility, it's worth noting that only IBM Spectrum LSF doesn't list fixed flexibility, and PBS Professional is the only one that doesn't support moldability beyond malleability.
- All the tools have consistent support for both open and closed arrival processes and can handle both homogeneous and diverse workload compositions.
- Quality of service with SLAs and SLOs is a common feature, but IBM Spectrum LSF only mentions SLAs.
- All software options cater to both real-time and non-real-time job scheduling requirements.
- In terms of resource descriptions, they all exhibit similar capabilities in handling resource heterogeneity, scaling, sharing, geographical coverage, and federation.
- When considering scheduling requirements, all tools are poised to manage multiple metrics of interest, can operate on both job-level and task-level, and are failure-aware. However, only IBM Spectrum LSF does not mention task-level scheduling.
- In terms of data locality, Cluster affinity and rack affinity are supported across most tools, with IBM Spectrum LSF only mentioning Cluster affinity.

In conclusion, while all four software options provide robust job scheduling capabilities, subtle differences in their features might make one more suitable than the others depending on specific organizational requirements. It's essential for organizations to evaluate these nuances when selecting a job scheduling software.

Table 2.3: Comparison between selected software

Feature	Software			
	JAMS Enterprise	**ActiveBatch**	**IBM Spectrum**	**PBS Professional**
Source	Single and multiple users	Single and multiple users	Single and multiple users	Single and multiple users
Job structure	All types	All types	All types	All types
Job flexibility	Fixed, moldable, malleable	Fixed, moldable, malleable	Moldable, malleable	Malleable
Arrival process	Open or closed	Open or closed	Open or closed	Open or closed
Workload composition	Homogeneous and diverse	Homogeneous and diverse	Homogeneous and diverse	Homogeneous and diverse
Quality of service	Yes, with SLAs and SLOs	Yes, with SLAs and SLOs	Yes, with SLAs	Yes, with SLAs and SLOs
Real time	Both real-time and non-real-time	Both real-time and non-real-time	Both real-time and non-real-time	Both real-time and non-real-time
Resource heterogeneity	Homogeneous, heterogeneous	Homogeneous, heterogeneous	Homogeneous, heterogeneous	Homogeneous, heterogeneous
Resource scaling	Fixed and dynamic infrastructure	Fixed and dynamic infrastructure	Dynamic infrastructure	Dynamic infrastructure
Sharing	Dedicated nodes, VMs, containers	Dedicated nodes, VMs, containers	Dedicated nodes, VMs, containers	Dedicated nodes, VMs, containers
Geographical coverage	Local and wide	Local and wide	Local and wide	Local and wide
Federation	Federated resources	Federated resources	Federated resources	Federated resources
Scheduling goal	Multiple metrics of interest	Multiple metrics of interest	Multiple metrics of interest	Multiple metrics of interest
Scheduling level	Job-level, task-level, or both	Job-level, task-level, or both	Job-level and task-level	Job-level and task-level
Data locality	Cluster affinity, rack affinity	Cluster affinity, rack affinity	Cluster affinity	Cluster affinity
Failure model	Failure-aware	Failure-aware	Failure-aware	Failure-aware
Adaptability	Can adapt to changes	Can adapt to changes	Can adapt to changes	Can adapt to changes

Chapter 3

Value Analysis

Assessing the success of a business often involves evaluating its ability to create value and innovate. For many companies, making decisions about investing in new products can be challenging. To aid in this process, value analysis can be employed.

3.1 Value for the Customer

Customer value is a combination of the benefits a customer receives from a product or service and the cost they pay to acquire it. These benefits can include both functional and emotional aspects such as convenience, reliability, aesthetics, customer service, and social status. The cost can include both the monetary cost as well as any effort or time expended to acquire and use the product or service (Oh 1999).

Therefore, customer value is not only about the price of a product or service but also about how well it meets the needs and expectations of the customer in relation to the costs they incur. A product or service that provides high customer value is perceived as offering greater benefits than the cost required to acquire it, leading to greater customer satisfaction and loyalty (Oh 1999).

Perceived value is the customer's assessment of the overall benefits they receive from a product or service in relation to the cost or sacrifice required to obtain it. It is a subjective and multidimensional concept that includes not only the product's functional attributes but also the emotional, social, and symbolic benefits that the customer derives from it. Perceived value is influenced by various factors, such as the customer's needs and preferences, the product's quality and price, the competition, and the marketing messages and communication strategies used to promote the product. Perceived value is a critical factor in determining customer satisfaction, loyalty, and repurchase intentions, and it plays a vital role in the success of a product or service in the marketplace (Sánchez-Fernández and Iniesta-Bonillo 2007).

The value of this solution is a robust job scheduling system that helps the customer in the batch executions, reducing costs and risks associated

To evaluate the value for the customer was created in Table 3.1 with the benefits and sacrifices associated with this project.

Benefits	**Sacrifices**
Robust job scheduler system	System's cost

Table 3.1: Benefits and Sacrifices to the customer

3.2 Value Proposition

A value proposition is a statement that describes the benefits a company's products or services offer to its customers. It is the way a company sets itself apart from its competitors and provides the reason why customers choose to buy from that particular company. The value proposition is specific to a particular customer segment and represents an overall view of a company's bundle of products and services. It highlights how the company packages and offers items of value, such as products, services, and complementary value-added services, to fulfill the specific needs of that customer segment (Osterwalder and Pigneur 2003).

The value proposition can be deconstructed using a tool that includes a value map to illustrate the features of the offering, including products and services, pain relievers, and gain creators. Additionally, a customer profile is included which outlines a specific customer segment, including their jobs, pains, and gains (Osterwalder, Pigneur, et al. 2015):

- **Customer Jobs:** Refers to the tasks, problems, or needs that customers are trying to address or fulfill in their work or personal lives. These jobs can be functional, social, or emotional in nature, and understanding them is crucial for creating products and services that effectively meet customer needs (Osterwalder, Pigneur, et al. 2015).
- **Customer Pains:** Refers to the negative emotions, undesired costs, and situations that customers experience or fear. These can be both functional and emotional, such as frustrations, annoyances, fears, or risks associated with trying to accomplish their jobs or achieve their desired gains (Osterwalder, Pigneur, et al. 2015).
- **Customer Gains:** Refers to the outcomes and benefits that customers expect or desire from using a product or service. These gains can be functional, emotional, or social (Osterwalder, Pigneur, et al. 2015).
- **Products and Services:** Refers to the tangible or intangible items that a company offers to meet the needs of a customer segment. They can be physical products, digital products, services, software, or any combination of these. They are the building blocks of a company's value proposition and are designed to provide specific benefits to customers (Osterwalder, Pigneur, et al. 2015).
- **Gain Creators:** Refers to the elements of a product or service that generate benefits for customers beyond simply relieving their pains. They are features that make the product or service more valuable to the customer, such as additional functionalities, improved performance, enhanced experience, and better design. (Osterwalder, Pigneur, et al. 2015).
- **Pain Creators:** Refers to the products or services that help customers alleviate their pains or challenges. (Osterwalder, Pigneur, et al. 2015).

In Figure 3.1, the value proposition canvas illustrates the value map component, which pertains to the proposed job scheduler system in this dissertation. Meanwhile, the customer segment pertains to customers of Retail Consult.

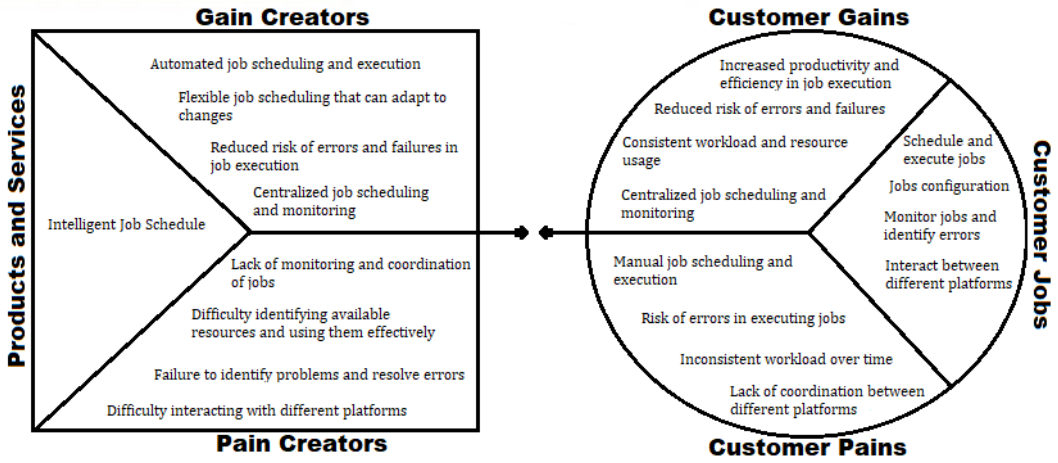

Figure 3.1: Value Proposition Canvas (Osterwalder, Pigneur, et al. 2015).

3.3 Quality Function Deployment

According to (Chan and Wu 2002) Quality Function Deployment (QFD) as a comprehensive approach that helps translate customer requirements into appropriate technical specifications. The authors then provide a historical perspective on QFD, highlighting its origins in Japan in the late 1960s and early 1970s and its eventual popularity in the United States during the 1980s.

Based on this project's objectives, described in the first chapter, the customer needs are as follows:

- Distributed Computation
- Batch Configuration
- Monitoring
- Visualization
- Integration

To fulfill the customer's needs, the following technical requirements were defined:

- User Interface
- Machine Learning Integration
- Parallel Processing
- Real-time Monotoring
- Email notification
- Job Configuration

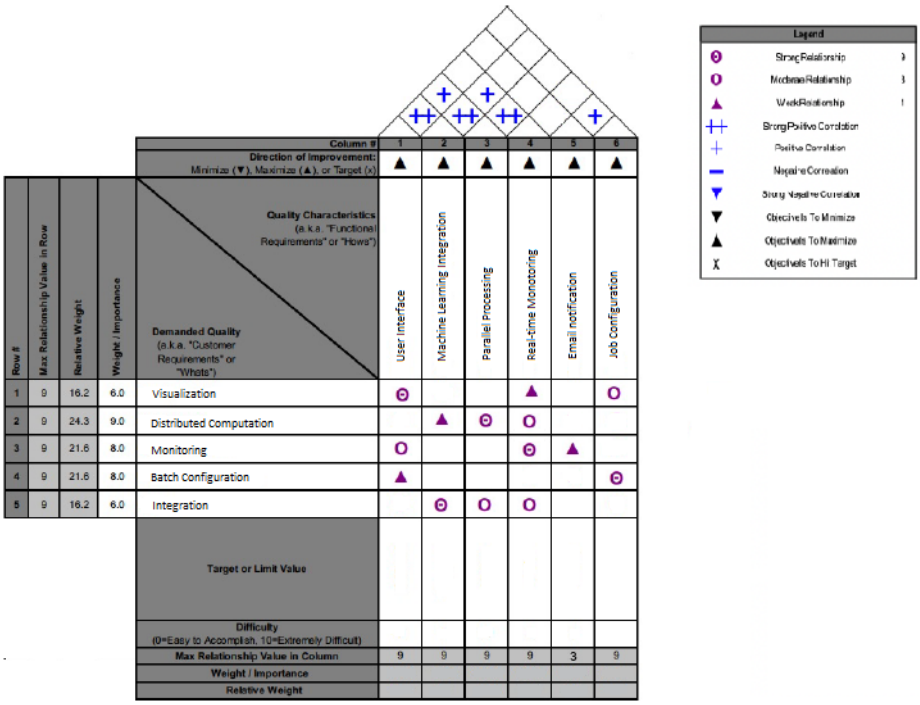

Figure 3.2: Quality Function Deployment

Chapter 4

Problem Analysis and Solution Design

The purpose of this chapter is to delve deep into the analytical and architectural aspects of the proposed system. It seeks to establish a firm foundation by understanding the essential requirements and by laying out the groundwork for the technological blueprint. As you navigate through this chapter, you'll be introduced to the methods employed to elicit requirements, the various actors interacting with the system, and a comprehensive list of functional and non-functional requirements. Moreover, we'll also explore the architectural considerations that ensure the system's robustness and adaptability. By the end of this chapter, the reader should have a clear understanding of what the system aims to achieve, how it intends to function, and the technological backbone that supports its operations.

4.1 Requirements Engineering

This section defines and documents the requirements for the system to develop. During the requirements elicitation process, various techniques were applied, namely:

- **Interviews:** used to understand what each user needs;
- **Workshops:** used to define requirements with multiple stakeholders when their vision;
- **Observations:** used to understand the entire process of batch processing and how it can be improved as well as corrected;
- **Document analysis:** required when possible to uncover new information;
- **System interface analysis:** used to understand how the batch processing works when a job schedule executing in an Oracle Retail environment;

4.1.1 Actors of the system

An actor is a role played by an external user or system that interacts with the system, usually through a use case. Table 4.1 specifies and describes the actors of the current prototype.

Actor	Description
Client	Access to visualization and authorization of automatic process of the systems responsible for AI algorithms.
Functional Team	Access to visualization, configurations, and optimization of the batch.
Infra Team and CMS Team	Access to visualization, disruptions, and monitoring.
IPE (Job Scheduler)	Responsible for providing, and receiving information as well as executing the batch scheduled by IBPS.

Table 4.1: Actors

4.1.2 Functional Requirements

Functional requirements specify what a system is supposed to do and are used to gauge system functionality in response to external stimuli Anandakumar, Arulmurugan, and Onn 2019. These requirements are integral to system development and offer stakeholders a clear insight into the expected system behavior. In this section, we outline the functional requirements for different components of the proposed system.

Job Scheduling (Optimization module) ensures that plans are executed in the most efficient manner, respecting the various constraints and maximizing resource utilization. The functional requirements associated with optimization are outlined in Table 4.2.

Identifier	Description	Actors
JS-REQ-01	Selection and Analysis of Escalation Solutions	Client, Infra Team, CMS Team
JS-REQ-02	Implement Escalation Solution	Client, Infra Team, CMS Team

Table 4.2: Job Scheduling Functional Requirements

With these functional requirements:

- The system must allow analyzing the characteristics and gains of a Scheduling solution - whether in batch duration or in the optimal use of resources (agents).
- The system must allow the submission of one of the Escalation solutions (chosen by the user) so that it can be implemented in the next batch execution.

Configurations are pivotal in ensuring the precision and efficiency of batches and optimizations. They provide a structured approach for defining batch parameters, validating adjustments, and formulating optimal scheduling resolutions. These configurations are not only crucial for the system's core functionality but also offer adaptability and resource optimization. The detailed functional requirements pertaining to configurations are tabulated in Table 4.3.

Identifier	Description	Actors
CONFIG-REQ-01	Configure and Modify Client's Batch	Client, Functional Team
CONFIG-REQ-02	Validate and Save Batch Configuration Modifications	Client, Functional Team
CONFIG-REQ-03	Generate Optimal Batch Scheduling Solutions	Client, Functional Team

Table 4.3: Configurations Functional Requirements

With these functional requirements:

- The system must allow adding new configurations, or replacing configurations of a Customer phase, based on the configuration model (or example) of other customers.
- The system should allow checking whether there are inconsistencies in the Batch's dependencies and only allow saving new changes if there are no errors caused by them.
- The system must allow generating scheduling solutions for batch jobs, respecting the client's active configurations, and making effective use of Agent resources.

Monitoring is a linchpin in maintaining system health and ensuring optimum performance, especially in Oracle environments. Personalized metrics, aptly tailored to the nuances of Oracle systems, are indispensable for preemptive system management. The detailed functional requirements associated with monitoring are depicted in Table 4.4.

Identifier	Description	Actors
MTR-REQ-01	Monitor environment resources	Infra Team, CMS Team
MTR-REQ-02	View collected data	Infra Team, CMS Team
MTR-REQ-03	Configuration of alerts	Infra Team, CMS Team
MTR-REQ-04	Configure custom queries and filters	Infra Team, CMS Team

Table 4.4: Monitoring Functional Requirements

In line with these functional requirements:

- The system must proactively monitor essential resources in real-time, including metrics like CPU, memory, and storage.
- The system should facilitate viewing the aggregated data through intuitive panels and graphical representations, utilizing platforms like Grafana.
- The system must empower users with the capability to configure personalized alerts, rooted in specific metrics and events.
- For the sake of insightful data analysis, the system should offer the flexibility to devise custom queries and filters, enabling the extraction of pertinent information from logs.

Forecasting plays a crucial role in determining future activities and understanding the potential trends in batches and jobs. Accurate forecasting allows better resource allocation, improved planning, and optimized job scheduling, resulting in enhanced system efficiency. This section lays out the detailed functional requirements related to forecasting, as presented in Table 4.5.

Identifier	Description	Actors
FCST-REQ-01	View Execution of Batch and Jobs	Infra Team, CMS Team
FCST-REQ-02	View Client's Business Context	Infra Team, CMS Team
FCST-REQ-03	View Estimated Durations of Batch and Jobs for Upcoming Days	Infra Team, CMS Team
FCST-REQ-04	Check Jobs with the Longest Estimated Duration for the Next Batch	Infra Team, CMS Team
FCST-REQ-05	View Jobs from the Critical Path of the Batch	Infra Team, CMS Team
FCST-REQ-06	Alert on Potential Anomalies in Batch/Job Duration	Infra Team, CMS Team
FCST-REQ-07	View Estimated Transactions/Price Changes for the Upcoming Days	Infra Team, CMS Team

Table 4.5: Forecasting Functional Requirements

With these functional requirements:

- The system should facilitate the visualization of the execution duration of a job over various batch days, filtering by phase and job name.
- The system must enable the viewing of the evolution of dynamic business variables, such as the number of transactions and the client's price changes.
- The system must provide insights into the expected duration of batches and jobs for the upcoming days, helping teams plan and allocate resources accordingly.
- The system should allow users to pinpoint the jobs with the longest projected duration in the next batch, ensuring preemptive measures can be taken if needed.
- This functionality is designed for understanding the critical path of the batch, helping teams prioritize and manage jobs effectively.
- The system should have the capability to alert users about potential anomalies or irregularities in the duration of batches or individual jobs, safeguarding against unexpected disruptions.
- By viewing estimated transactions and anticipated price changes, teams can strategy and prepare for the demands of the upcoming days.

Disruption functional requirements are instrumental in diagnosing, recovering from, and preventing errors and disruptions. These requirements outline how the system should handle errors and discrepancies, especially in the context of Oracle DB, ORM, and associated software error codes. The associated requirements are articulated in Table 4.6.

Identifier	Description	Actors
DRPT-REQ-01	Identify errors	Infra Team, CMS Team
DRPT-REQ-02	Detect anomalies and errors based on past patterns	Infra Team, CMS Team
DRPT-REQ-03	Notify of errors	Infra Team, CMS Team
DRPT-REQ-04	Map error recovery actions	Infra Team, CMS Team
DRPT-REQ-05	Run automatic recovery	Infra Team, CMS Team
DRPT-REQ-06	Log incidents and auto-healing action history	Infra Team, CMS Team
DRPT-REQ-07	Monitor in real-time	Infra Team, CMS Team
DRPT-REQ-08	Detect locks by job name	Infra Team, CMS Team
DRPT-REQ-09	Immediately notify locks	Infra Team, CMS Team

Table 4.6: Disruption Functional Requirements

With these functional requirements:

- The system must identify errors based on log analysis and error coding patterns, specifically from Oracle DB, ORM, and other associated software error catalogs.
- The system must detect and classify jobs as errors based on forecasting model patterns, comparing execution time, exceptions, and anomalies to notify stakeholders.
- The system must send immediate notifications to stakeholders responsible for the detected errors so they can address them appropriately.
- The system must allow manual and automatic error recovery logging.
- The system must attempt to automatically recover failed jobs or forward them for alternative processing.
- The system must record incidents that occurred during auto-healing and maintain a history of the actions taken for subsequent analysis.
- The system must provide real-time monitoring of resource status and locks to detect potential blockages.
- The system must detect database locks and associate the job name through log analysis and blockage information to notify stakeholders.
- The system must send immediate notifications to teams responsible for identified blockages so they can address them appropriately.

4.1.3 Non-functional Requirements

While functional requirements define what a system is supposed to do, Non-Functional Requirements (NFR) detail how the system achieves this. NFR, sometimes termed "quality attributes", govern the operation of systems and outline constraints, performance goals, and other intrinsic system behaviors Chung et al. 2009. These requirements not only address the system's functionality but also its overarching qualities, ensuring its usability, reliability, and performance Glinz 2007.

The subsequent sections elucidate the non-functional requirements essential for the proposed system's optimal performance:

- **Performance:** The system should respond to user requests within two seconds under normal operating conditions and should support concurrent access by multiple users.
- **Reliability:** The system should have an uptime of more than 90% and should be capable of recovering from failures without data loss.
- **Usability:** The user interface should be intuitive, requiring no more than one hour of training for new users.
- **Testability:** The system should be designed for easy testing, both at the component and the overall system level.
- **Flexibility:** The system should be designed to accommodate future modifications, such as the addition of new modules or integration with additional systems.
- **Scalability:** As the demand for the system increases (either in terms of data volume or user count), it should handle the growing load without noticeable performance degradation.

4.2 Architectural Design

The architectural design is fundamental for understanding the system's structure, its components, and their interactions. This section will provide insights into the layout and deployment of our system components as identified in the functional requirements. With the integration of the IBPS and the Oracle Retail component as central pillars, we aim to offer a cohesive, efficient, and scalable system architecture.

4.2.1 Logical View

The Logical View addresses the system's high-level structure and its main components. Here, we will provide a graphical representation that shows the interconnections between the IBPS components identified in the functional requirements and how they interface with the Oracle Retail component. This view assists in comprehending the modularity and the responsibilities of each component.

Refer to Figure 4.1 for the visual representation of the logical view. In general, the Job Scheduler provides job configurations, execution histories as well as errors and system states. Configurations are essential for the optimization component and execution histories for forecasting. The forecast component provides values that are then used in the optimization and disruption components. The disruption component will provide the jobs that still need to be executed to the optimization component in order to determine a more effective schedule and save time in the rest of the batch. This disruption module will provide automatic corrections to the scheduler, which will provide a list of errors and different states of the system. The monitoring component provides environment resources information and alerts of the system.

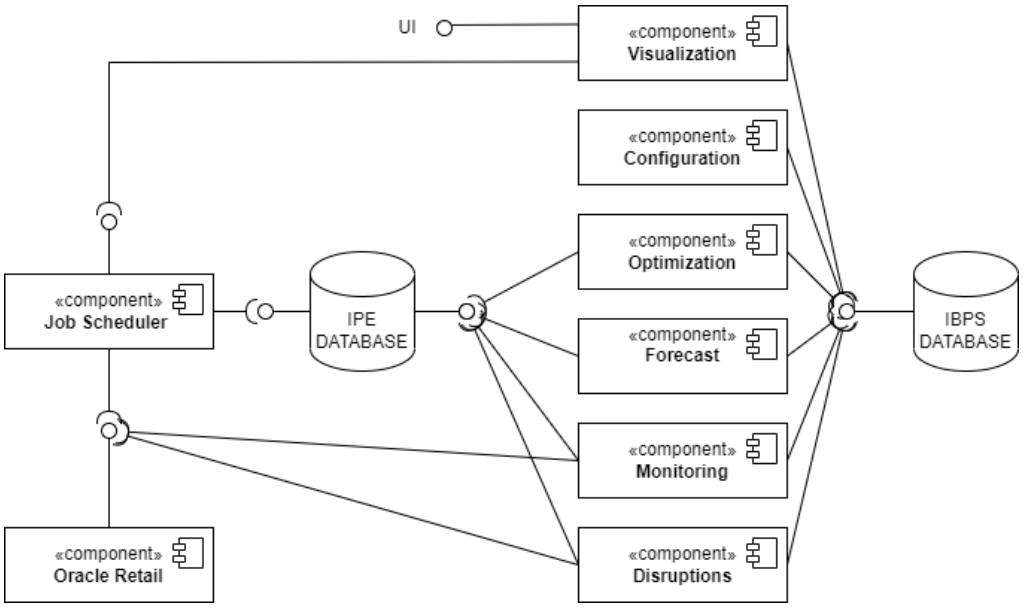

Figure 4.1: Logical view of the system components.

4.2.2 Deployment View

The Deployment View showcases the system's runtime configuration in terms of hardware and software. This view is crucial for understanding how the system components, including IBPS and Oracle Retail, will be distributed across servers, networks, and other infrastructure elements. The Deployment View ensures that the system remains robust, performant, and easily maintainable.

Figure 4.2 illustrates the system's deployment setup. The system will be deployed in the Oracle Retail environment for the execution of batch processing.

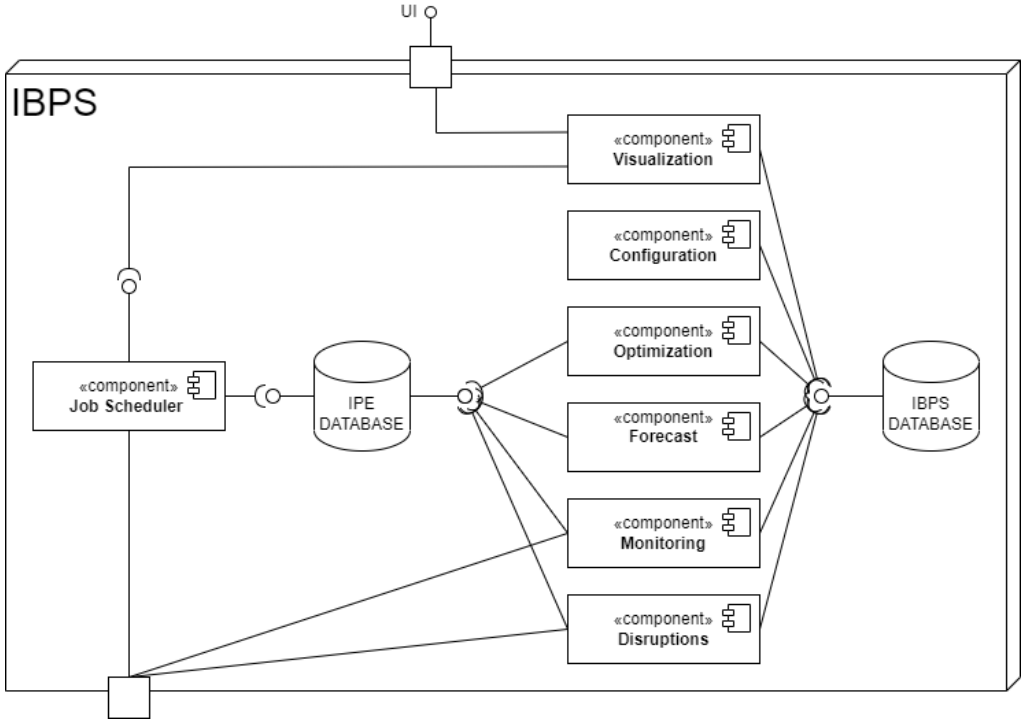

Figure 4.2: Deployment view of the system components.

Chapter 5

Implementation

Following the delineation of requirements and architectural design in previous chapters, this chapter delves into the nuances of the actual implementation. This exploration provides insights into the tools, methodologies, and pivotal decisions that shaped the system's realization.

5.1 System Architecture

The IBPS system was architected following the principles of a **microservices architecture**. This architectural style divides the system into small, independent services that communicate over well-defined APIs (Newman 2015).

- **Scalability:** Individual microservices can be scaled based on their workload, leading to more efficient resource use(Richardson 2018).
- **Flexibility:** The decoupled nature of microservices allows for independent development and deployment cycles, accelerating feature releases (Newman 2015).
- **Resilience:** Isolating functionalities into separate services can improve fault isolation. A failure in one service can be handled without affecting the entire system (Richardson 2018).
- **Technology Diversity:** Microservices allow for using the right tool for the job, meaning services can be implemented using different programming languages or databases (Newman 2015).

5.2 Development Environment and Technologies

This section elaborates on the specific technologies and tools employed during the development phase, offering the rationale behind each choice.

5.2.1 Programming Languages

- **Java:** Java was the primary language used to develop the scheduler. Known for its portability, scalability, and a wide ecosystem of libraries, Java remains a leading choice for developing efficient and scalable back end services Ekman and Jensen 2018).

- **Python:** Python was selected for its simplicity, readability, and dynamic capabilities. Python's extensive libraries and frameworks, particularly in data science and web development, have made it a popular language for a wide range of applications (Van Rossum and Drake 2018).

5.2.2 Frameworks and Libraries

- **Dash and Plotly:** The front end was powered by Dash by Plotly. Dash enables the creation of interactive web applications using pure Python, while Plotly provides versatile data visualization tools. Combined, they ensure a responsive and visually appealing user interface (Shetty and Wright 2018).
- **Grafana:** Grafana was employed for the monitoring module due to its prominence in the field of real-time data monitoring and analytics. Its extensibility, combined with its rich visualization capabilities, makes it a top choice for creating comprehensive and customizable monitoring dashboards (Labs 2023).

5.2.3 Database System

The Oracle database, an integral component of the Oracle Retail environment, was employed. Oracle's robustness, scalability, and suite of advanced features make it suitable for handling intricate and large-scale retail operations (Mohan and Oberoi 2019).

5.3 Implementation Details

This segment dissects the specifics of the system components and their interactions.

5.3.1 IBPS Component

Once deployed, the IBPS component integrates effortlessly with the Oracle Retail environment. Such a setup champions efficient job scheduling, with the master scheduler dispatching jobs to various agents. These agents subsequently execute the tasks on dedicated machines, including RMS, SIM, RESA, among others.

5.3.2 Job Scheduler-Frontend communication

Endpoints form the bridge connecting the scheduler to the frontend. Leveraging these endpoints, the front end can instigate AI module tasks, enhancing system versatility.

5.3.3 Integration with Oracle Retail

The scheduler interfaces with the Oracle Retail component, harnessing its robust database system. Tight integration between the scheduler and the Oracle Retail component is maintained. Such synergy ensures fluid data exchanges, facilitating tasks ranging from job execution to results retrieval.

5.4 Conclusion

This chapter delves implementation of the IBPS system. Starting with the architecture, emphasis is placed on the adoption of a microservices architecture, which offers significant

advantages in terms of scalability, flexibility, resilience, and technological diversity. These principles guide the system's modularity and efficiency.

The development environment and the technologies used are then outlined. The choice of Java and Python as programming languages reflects the pursuit of efficiency and versatility. Additionally, the use of frameworks and libraries such as Dash, Plotly, and Grafana exemplifies the emphasis on interactive visualization and real-time monitoring. The decision to employ the Oracle database system aligns with the need for robustness and scalability, especially in the context of the Oracle Retail environment.

Specific aspects of the implementation are detailed, from the integration of the IBPS component with the Oracle Retail environment to the communication between the job scheduler and the user interface. The chapter underscores the synergy between the various components and the importance of fluid integration for the system's success.

Chapter 6

Testing

Testing plays a pivotal role in assessing the robustness of any system. In the case of the IBPS, the consequences of deploying an untested or poorly-tested system can be catastrophic. The primary focus here is to ensure seamless communication between the User Interface and the Job Scheduler (IPE) and to validate the flawless integration and functionality of the AI modules. This chapter delves into the meticulous process followed to achieve these objectives.

After this introduction, each section provides a brief description and then presents specific test scenarios, the challenges faced, and the results obtained. Each test case has been designed to provide a comprehensive evaluation, covering all facets of the prototype.

6.1 Job Scheduling

Ensuring the functionality and performance of the Job Scheduler is vital for the system's successful operation. A rigorous testing process has been designed to validate the requirements specified for this component. By utilizing a set of test cases (TCs), we aim to comprehensively assess each requirement and ensure the system's behavior aligns with stakeholders' expectations.

The Job Scheduler, as delineated in the functional requirements, plays a pivotal role in efficiently managing and scheduling tasks across different technologies. It is paramount to assess the system's capability to not only manage its routine tasks but also to handle unexpected challenges. To this end, the following test cases have been designed:

FR Identifier	TC Identifier	Test Case - Client Profile
JS-REQ-01	TC 01	View the graph of active batch configurations
	TC 02	View execution sequence and parallelism of scheduling solutions
	TC 03	View agent occupancy of scheduling solutions
	TC 04	View gains in batch duration with optimization of scheduling solutions
	TC 05	View content of files of scheduling solutions
JS-REQ-02	TC 06	Submit a scheduling solution in the IPE

Table 6.1: Test cases for Job Scheduler Functional Requirements

6.1.1 Test Case 01: The graph of active batch configurations Visualization

The Figure 6.1 represents the TC01

Description:
The user should be able to view a graph that represents the active batch configurations, generated in the Configurations tab. The blue nodes represent the jobs, the red nodes represent the phases, and the connections signify the dependencies between jobs and phases.

Pre-Condition:

1. `T_BATCH_CONFIG_ACTIVE` must have configurations for the dashboard client (in our case, a client with ID 4).
2. The graph must have been generated in the Configurations tab when scheduling solutions were created by clicking the "Scheduling" button.

Procedure:

1. Click on the 'Configs Graph' button at the end of the tab.
2. Check the graph, observing some jobs (blue nodes), phases (red nodes), and dependencies (connections).

Expected Result:
Upon clicking the button, a popup should appear with the Graph of the active batch configurations - blue nodes, red nodes, and connections between nodes.

Result:
Passed

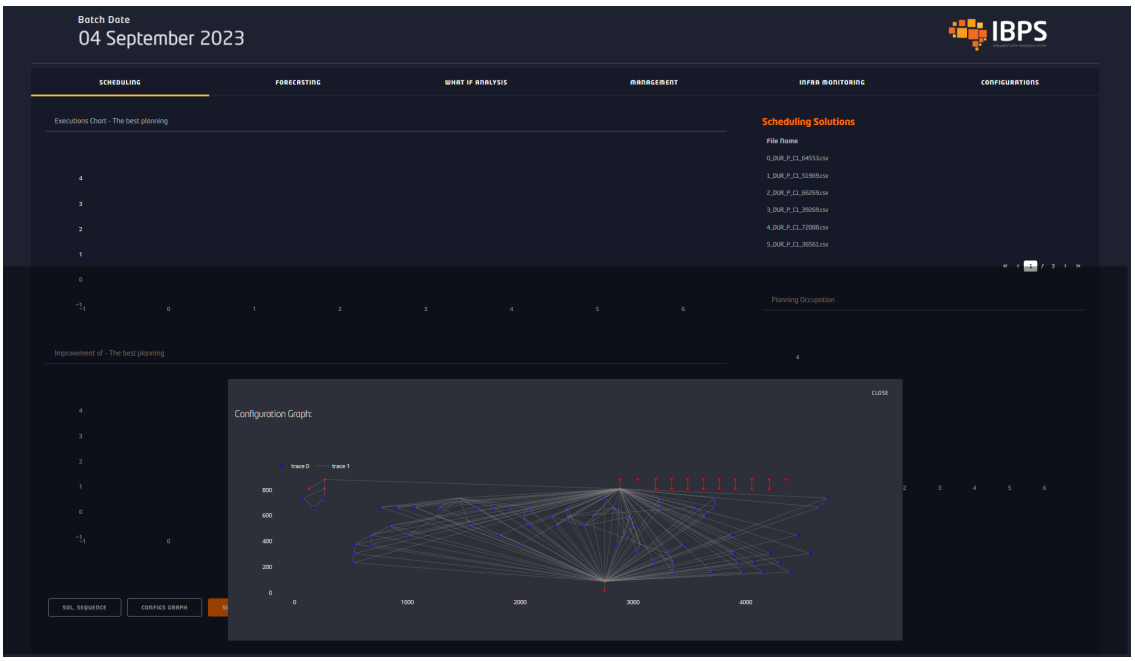

Figure 6.1: Show Active Configurations Graph

6.1.2 Test Case 02: Execution Sequence and Parallelisms Visualization

The Figure 6.2 represents the TC02

Description:
The user should be able to select a Scheduling solution and observe the execution graph and the respective parallelisms performed by it in order to optimize the Batch duration.

Pre-Condition:

1. The 'Scheduling Solutions' list must have solutions previously generated in the Configurations tab.

Procedure:

1. Select one of the solutions from the 'Scheduling Solutions' list.
2. Observe the graph in the 'Executions Chart - The best planning' section.

Expected Result:
Upon selecting one of the Scheduling solutions, the graph in the 'Executions Chart - The best planning' section should update with the information from the chosen solution.

Result:
Passed

6.1.3 Test Case 03: Planning Occupation Visualization

The Figure 6.2 represents the TC03

Description:
The user should be able to select a Scheduling solution and observe the agent occupancy with the jobs to be executed, taking advantage of the resources in an optimized way.

Pre-Condition:

1. The 'Scheduling Solutions' list must have solutions previously generated in the Configurations tab.

Procedure:

1. Select one of the solutions from the 'Scheduling Solutions' list.
2. Observe the graph in the 'Planning Occupation' section.

Expected Result:
Upon selecting one of the Scheduling solutions, the graph in the 'Planning Occupation' section should update with the information from the chosen solution.

Result:
Passed

6.1.4 Test Case 04: Batch Duration Improvement Visualization

The Figure 6.2 represents the TC04

Description:
The user should be able to select a Scheduling solution and observe the reduction in the total batch duration in the event that this is applied.

Pre-Condition:

1. The 'Scheduling Solutions' list must have solutions previously generated in the Configurations tab.

Procedure:

1. Select one of the solutions from the 'Scheduling Solutions' list.
2. Observe the graph in the 'Improvement of - The best planning' section.

Expected Result:
Upon selecting one of the Scheduling solutions, the graph in the 'Improvement of - The best planning' section should update with the information from the chosen solution.

Result:
Passed

Figure 6.2: Show Scheduling Solutions Executions, Parallelisms, Occupation of Agents

6.1.5 Test Case 05: Solution Sequence File Visualization

The Figure 6.10 represents the TC05

Description:
The user should be able to view the tabular data stored in the file for each solution, which indicates the order of job execution and the agents to which these jobs were allocated.

Pre-Condition:

1. The 'Scheduling Solutions' list must have solutions previously generated in the Configurations tab.

Procedure:

1. Select one of the solutions from the 'Scheduling Solutions' list.
2. Click on the 'Sol. Sequence' button.
3. Observe the table information displayed in the popup.

Expected Result:
The popup should update and display a table with the data stored in the file of the selected Scheduling solution.

Result:
Passed

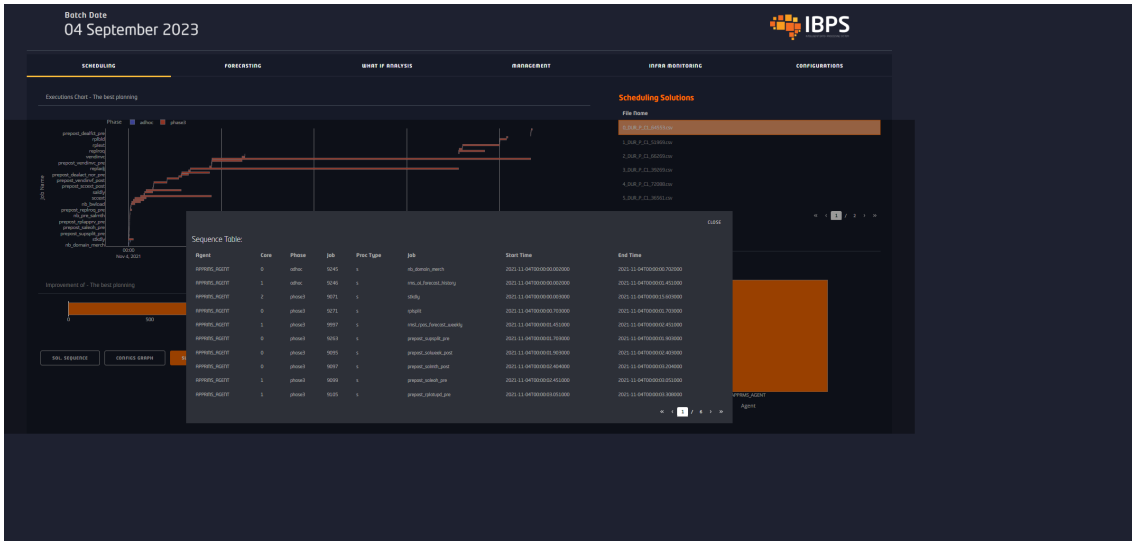

Figure 6.3: Show Scheduling Solutions File Content

6.1.6 Test Case 06: Solution Submission in IPE

The Figure 6.11 represents the TC06

Description:
After analyzing the characteristics of each Scheduling solution using the described charts, the user should be able to select their preferred solution and submit it to the IPE to be implemented in the next batch run.

Pre-Condition:

1. The 'Scheduling Solutions' list must have solutions previously generated in the Configurations tab.
2. Tables T_PROCESS_INTERMEDIATE, T_PROCESS_DEPS_INTERMEDIATE, T_SCHEDULES_INTERMEDIATE must be created in the IBPS database.
3. Tables PE_PROCESSES, PE_PROCESS_DEPS, and PE_SCHEDULES must be created in the IPE database and used by the IPE tool.

Procedure:

1. Select the solution you want to implement.
2. Click on the 'Submit' button.
3. Observe the message with table updates.

Expected Result:
After 3 minutes, the batch should start being executed by the IPE and the status of its jobs should start updating in the 'Management' tab.

Result:
Passed

Figure 6.4: Submit Escalation Solution to IPE

6.2 Configurations

Ensuring the functionality and reliability of the Configuration component is crucial for the optimal performance of the system. Rigorous testing has been set in motion to verify the requirements specifically outlined for this integral element. Through a diverse range of test cases (TCs), our objective is to holistically gauge each stipulated requirement and guarantee that the system's responses resonate with stakeholder specifications.

As underlined in the functional requirements, the Configurations component is indispensable in the fine-tuning and adjustment of client-specific parameters. It's vital to ascertain the system's proficiency in regular operations and its adaptability in face of unforeseen challenges. In light of this, the subsequent test cases have been conceptualized:

RF Identifier	**TC Identifier**	**Test Case - Client Profile**
CONFIG-REQ-01	TC 01	Select Configuration Template of a Phase for a Client
	TC 02	Add a Phase's Configuration to Active Configurations
	TC 03	Replace Active Configurations of a Phase
CONFIG-REQ-02	TC 04	Validate Active Configurations
	TC 05	Detect Configuration Inconsistencies - Nonexistent or Inactive Dependencies
	TC 06	Detect Configuration Inconsistencies - Loops Caused by Dependencies
	TC 07	Save Changes in Active Configurations
CONFIG-REQ-03	TC 08	Generate Scheduling Solutions
	TC 09	Ensure Coherence of Scheduling Solutions with the Saved Active Configurations

Table 6.2: Test cases for Configurations Functional Requirements

6.2.1 Test Case 01: Select Configuration Template of a Phase for a Client

The Figures 6.5 represents the TC01.

Description:
The system should allow users to view templates (examples) of phase configurations from different clients.

Pre-Condition:

1. `T_BATCH_CONFIG_ALL` must be populated with template configuration data of different phase settings from various clients.

Procedure:

1. Select `RMS` from the Products dropdown.
2. Select `RMS_PHASE_3` from the Phases dropdown.
3. Select `RPL` from the Sub-Phases dropdown.
4. Choose a client from the Clients dropdown - 4, 5, 98, or 99 - and check the template configuration table.
5. Select another client, different from the previous one, from the Clients dropdown and check the template configuration table.

Expected Result:
The table with the selected template configurations should update with the settings of the selected client for the chosen phase.

Result:
Passed

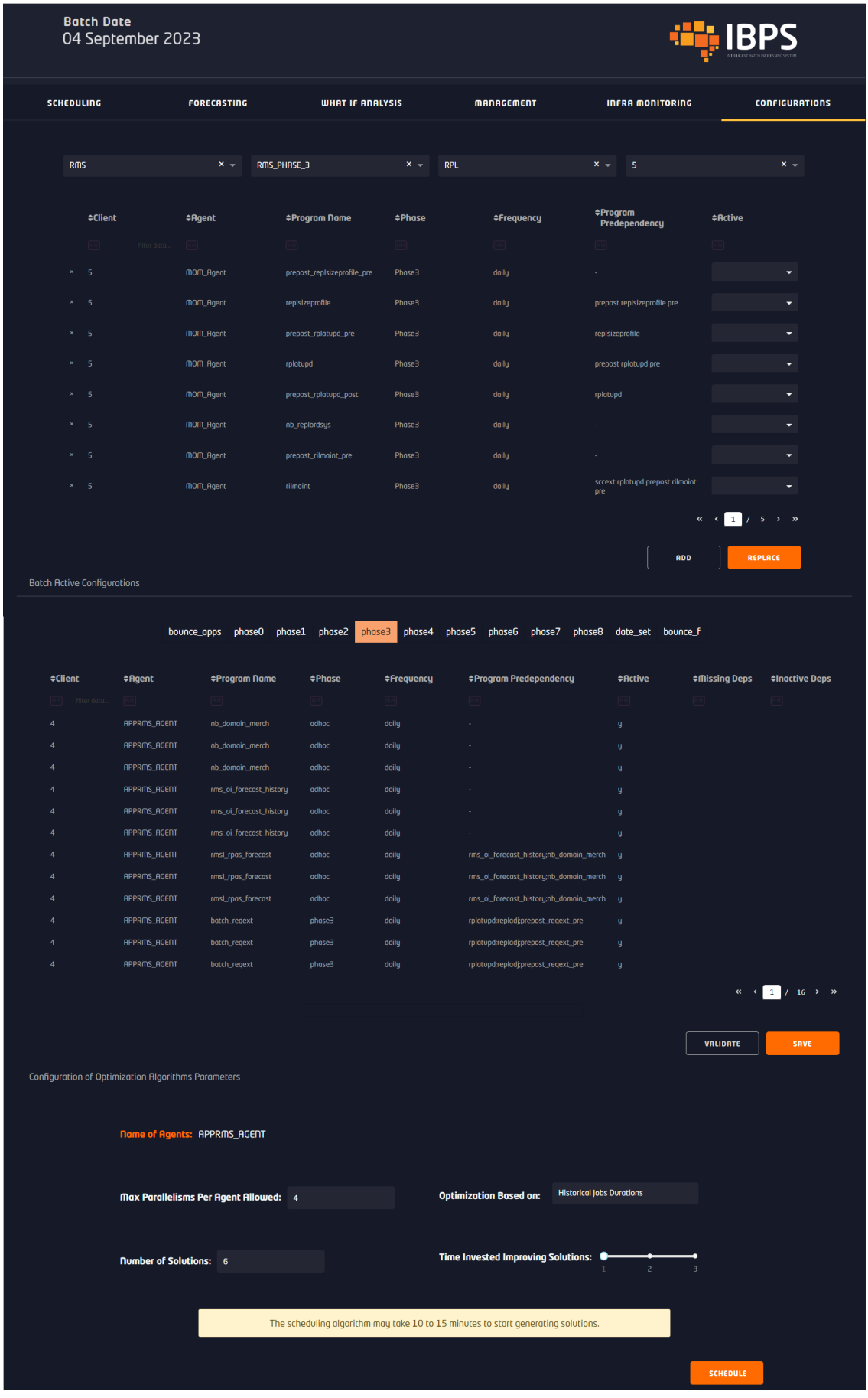

Figure 6.5: Select Template Configurations for RMS_PHASE_3 (Client 5)

6.2.2 Test Case 02: Add a Phase's Configuration to Active Configurations

The Figure 6.6 and Figure 6.7 represents the TC02.

Description:
The system should allow adding a pre-configured phase to the client's active batch configurations displayed on the dashboard. Additionally, it should let the user determine which jobs from that phase should remain active.

Pre-Condition:

1. `T_BATCH_CONFIG_ALL` must be populated with template configuration data of different phase settings from various clients.
2. The table `T_BATCH_CONFIG_ACTIVE` must be created in the IBPS database and should have a structure similar to the `T_BATCH_CONFIG_ALL`.

Procedure:

1. Filter the table in the active configurations section by the 'Phase' column with the string 'phase0'.
2. Select 'RMS' from the Products dropdown.
3. Select 'RMS_PHASE_0' from the Phases dropdown and '-' from the Sub-Phases dropdown.
4. Choose client 4 from the Clients dropdown.
5. Click on the 'Add' button.
6. Check the active configurations table.

Expected Result:
Initially, when filtering by 'phase0', the active configurations table is empty. At the end of the test, after clicking the 'Add' button, this table should contain the 'phase0' configurations selected from the template configurations.

Result:
Passed

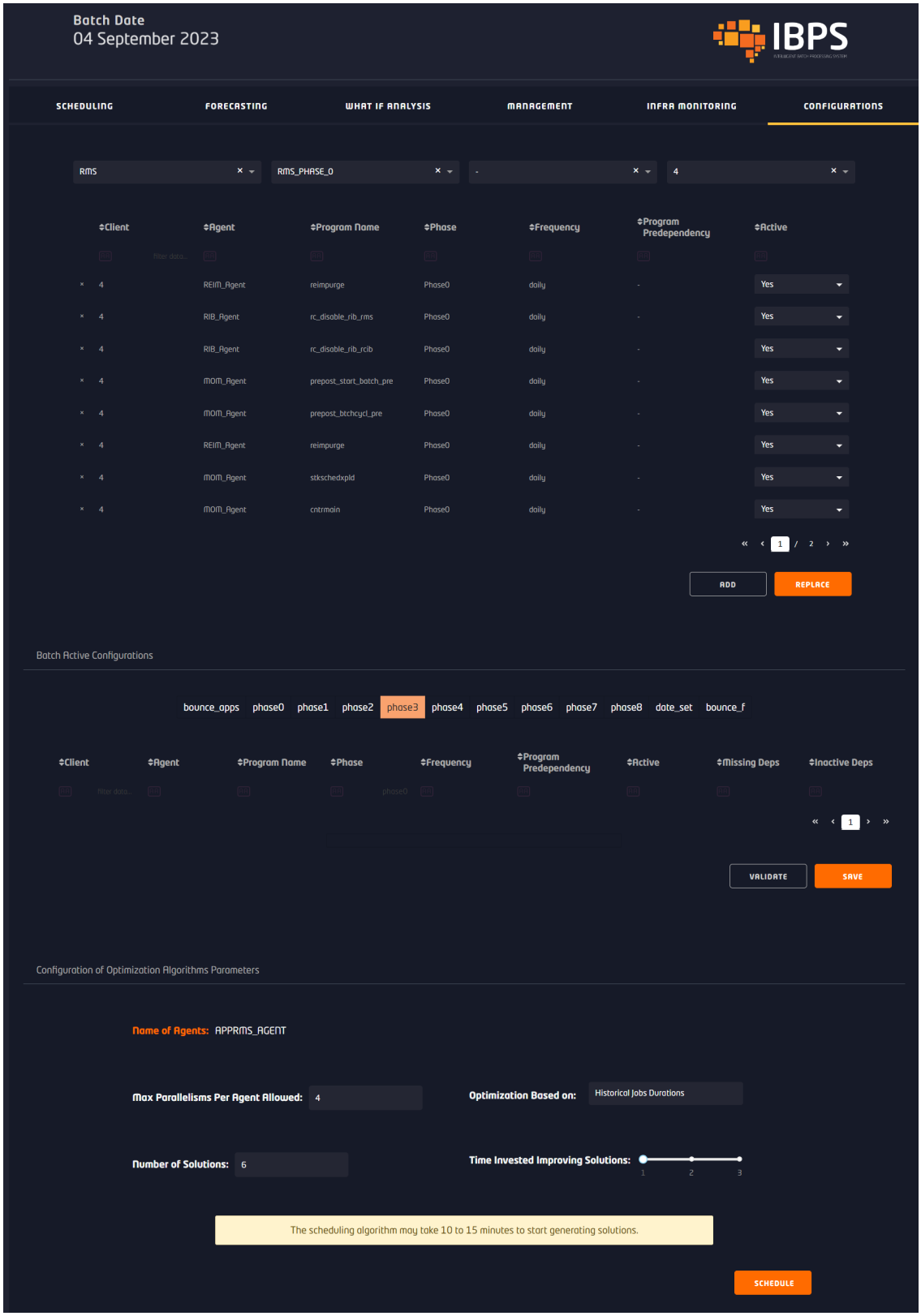

Figure 6.6: Select Template Configurations for RMS_PHASE_0

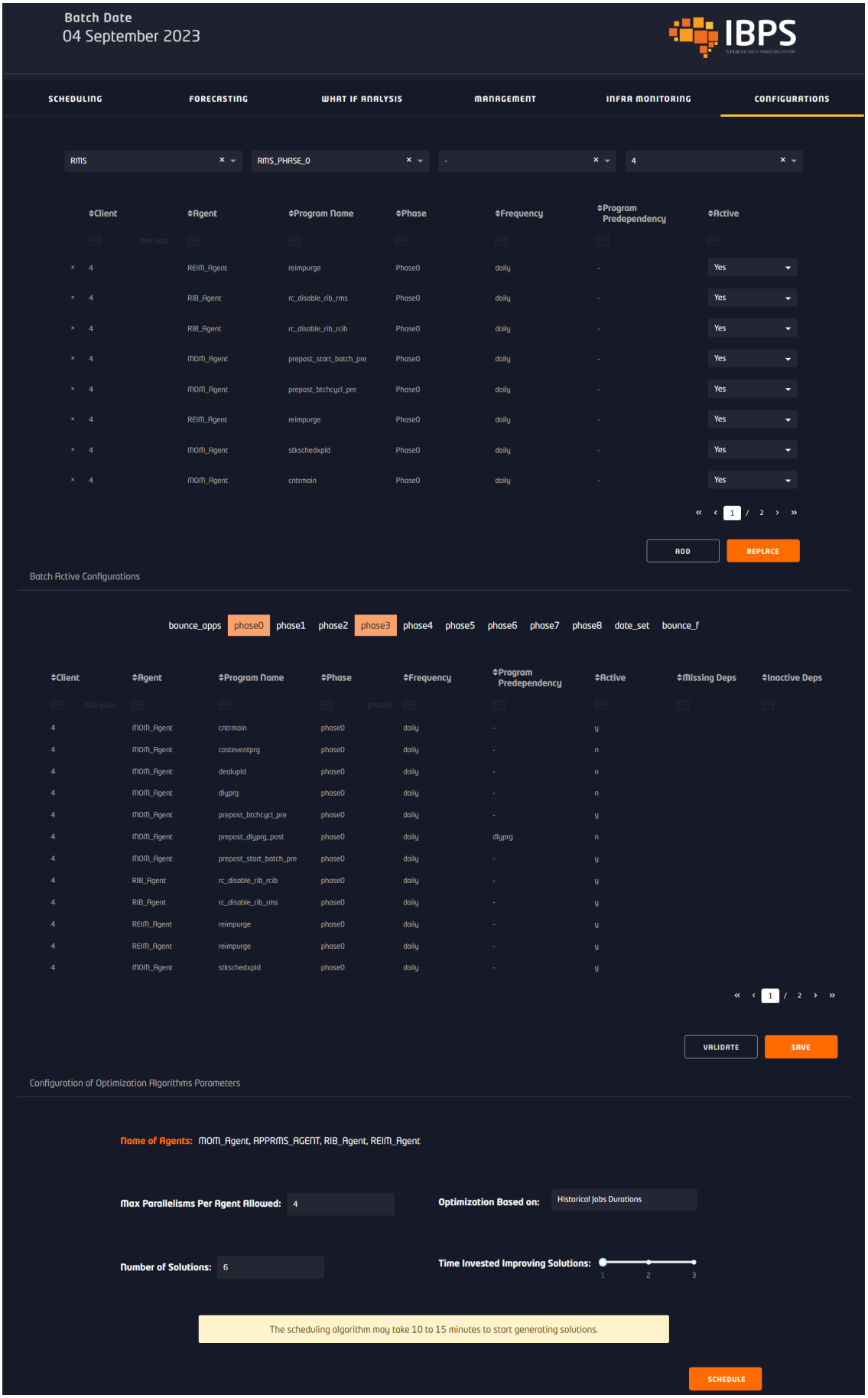

Figure 6.7: Added Selected Phase to Active Configurations

6.2.3 Test Case 03: Replace Active Configurations of a Phase

The Figure 6.8 represents the TC03.

Description:
When a phase is already set in the active batch, the system should allow its replacement with the configurations of that same phase from another template (possibly from a different client).

Pre-Condition:

1. `T_BATCH_CONFIG_ALL` must be populated with template configuration data from different phases of different clients.
2. `T_BATCH_CONFIG_ACTIVE` must be populated with configuration data for the phase that we want to replace for client 4.

Procedure:

1. Perform test TC 02 to add 'phase0' to active configurations.
2. Filter 'phase0' in active configurations and verify them.
3. Select 'RMS' from the Products dropdown.
4. Select 'RMS_PHASE_0' from the Phases dropdown and '-' from the Sub-Phases dropdown.
5. Choose client 5 from the Clients dropdown.
6. Activate some jobs from the template configurations in the 'Active' column.
7. Click the 'Replace' button and check in the active configurations section if these were updated in 'phase0'.

Expected Result:
When filtering active configurations for 'phase0' in the Phase column, they should contain the configurations selected from client 5 for that phase, with the active jobs selected in step 6.

Result:
Partially passed - did not allow job configuration activations.

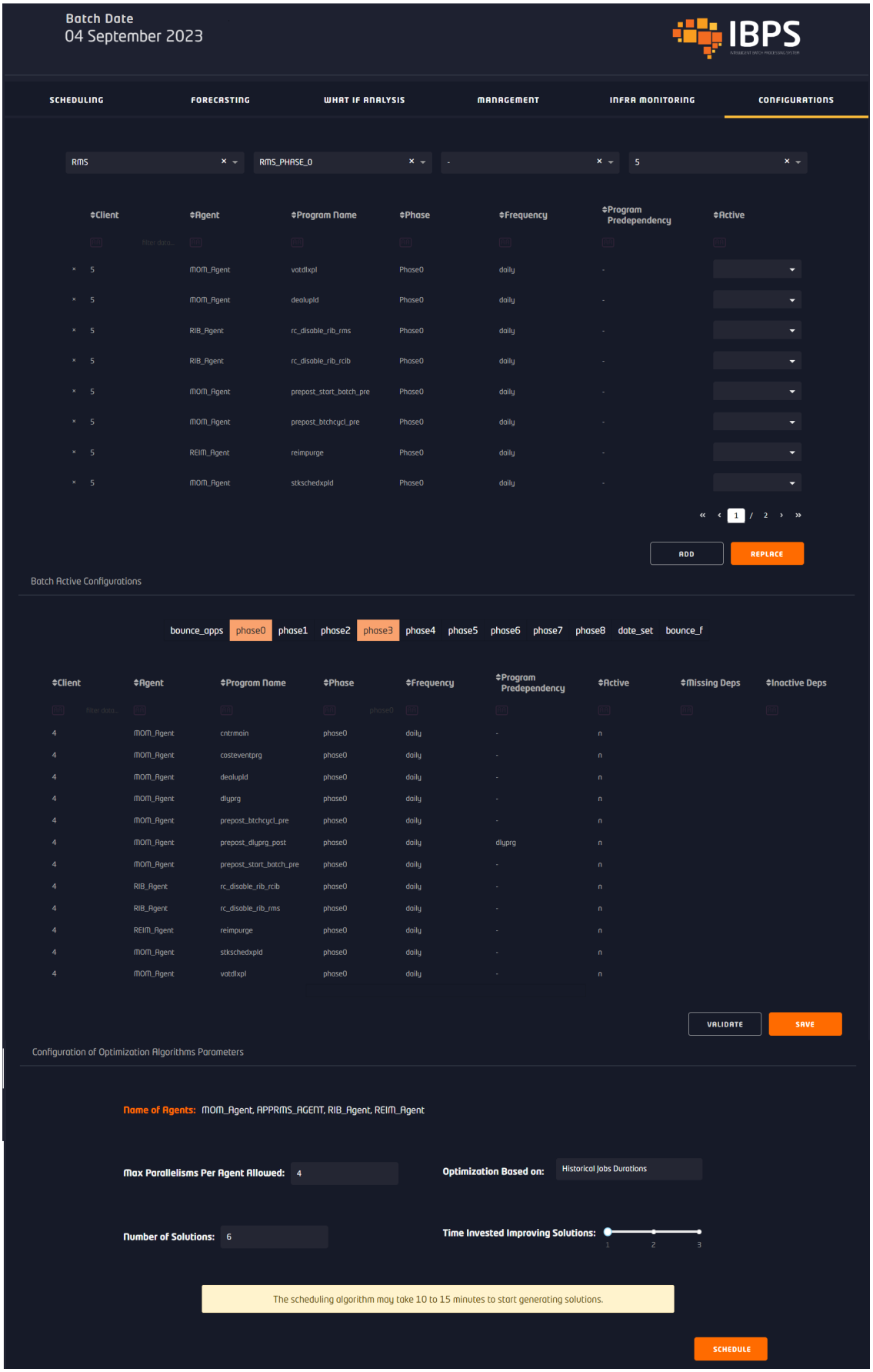

Figure 6.8: Replace Active Configurations for RMS_PHASE_0

6.2.4 Test Case 04: Validate Active Configurations

The Figure 6.9 represents the TC04.

Description:
After making changes to the active batch configurations, the system should validate them, ensuring there are no inconsistencies in job dependencies so that they can be applied in batch optimization and execution.

Pre-Condition:

1. `T_BATCH_CONFIG_ACTIVE` must be populated with error-free active configurations for client 4.

Procedure:

1. Refresh the Configurations tab (in case changes have been made to the active configurations that have not been saved).
2. Click on 'Validate' in the section of active batch configurations.
3. Check for a success message.

Expected Result:
Success message indicating configurations are valid.

Result:
Passed

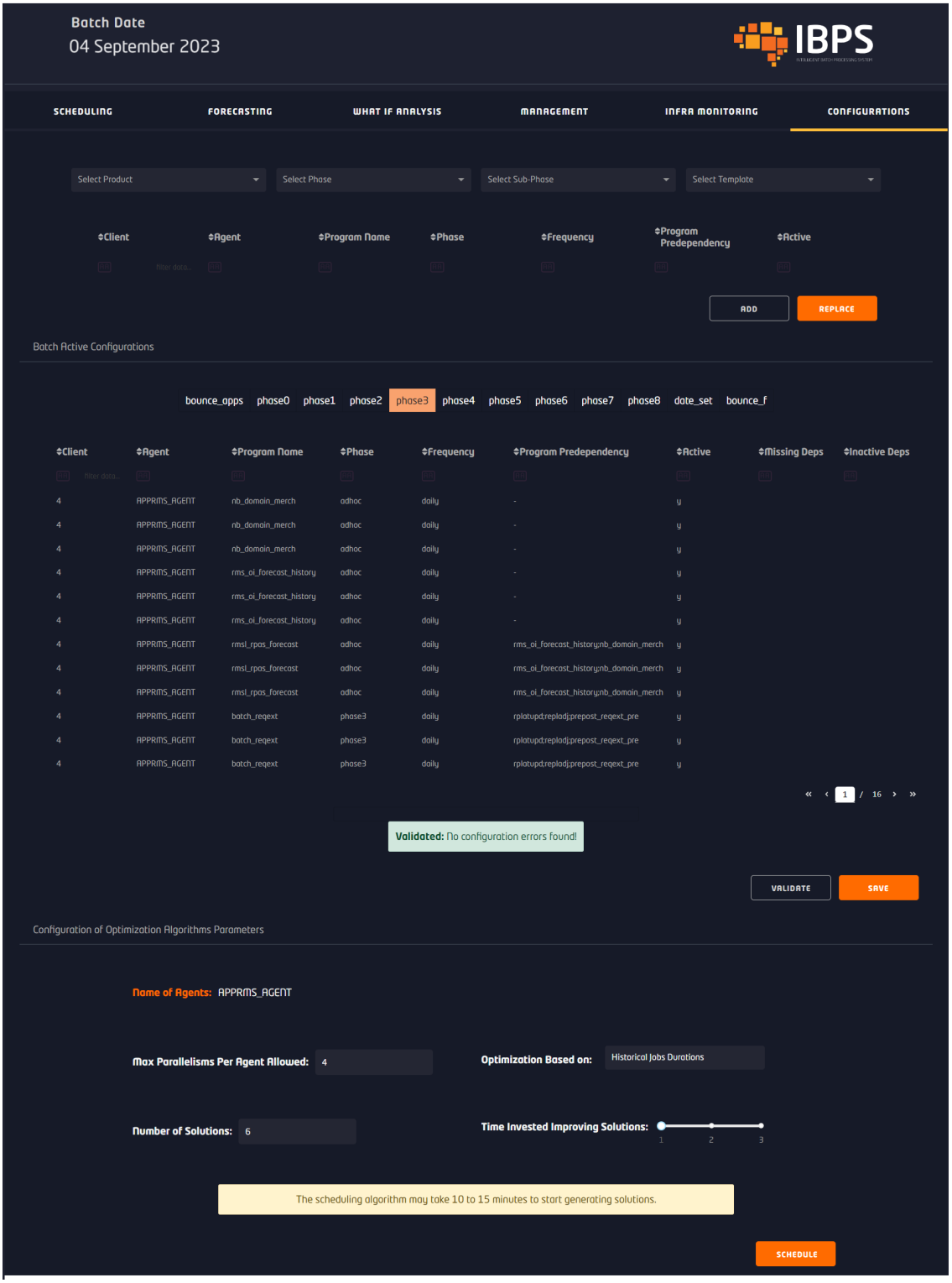

Figure 6.9: Validation of Active Configurations

6.2.5 Test Case 05: Detect Configuration Inconsistencies - Nonexistent or Inactive Dependencies

The Figure 6.10 represents the TC05.

Description:
The system should notify the user of jobs that depend on jobs that either don't exist or are inactive in the batch configurations.

Pre-Condition: None

Procedure:

1. Refresh the Configurations tab (in case changes were made to the active configurations that have not been saved).
2. At the beginning of the tab, select 'RMS', 'RMS_PHASE_3', and 'RPL' in the Product, Phase, and Sub-Phase dropdowns respectively.
3. Select client 99.
4. Click the 'Replace' button.
5. Click 'Validate' in the section of active batch configurations.

Expected Result:
An error message indicating the number of non-existent and/or inactive dependencies. Additionally, rows of jobs with dependency errors in the active configurations table should be highlighted differently.

Result:
Passed

Figure 6.10: Finding Errors in Dependencies

6.2.6 Test Case 06: Detect Configuration Inconsistencies - Loops Caused by Dependencies

The Figure 6.11 represents the TC06.

Description:
The system should notify the user of the existence of loops caused by dependencies between jobs.

Pre-Condition: None

Procedure:

1. Refresh the Configurations tab (in case changes were made to the active configurations that have not been saved).
2. At the beginning of the tab, select 'RMS', 'RMS_PHASE_3', and 'RPL' in the Product, Phase, and Sub-Phase dropdowns respectively.
3. Select client 98.
4. Click the 'Replace' button.
5. Click 'Validate' in the section of active batch configurations.

Expected Result:
An error message indicating the number of existing loops.

Result:
Passed

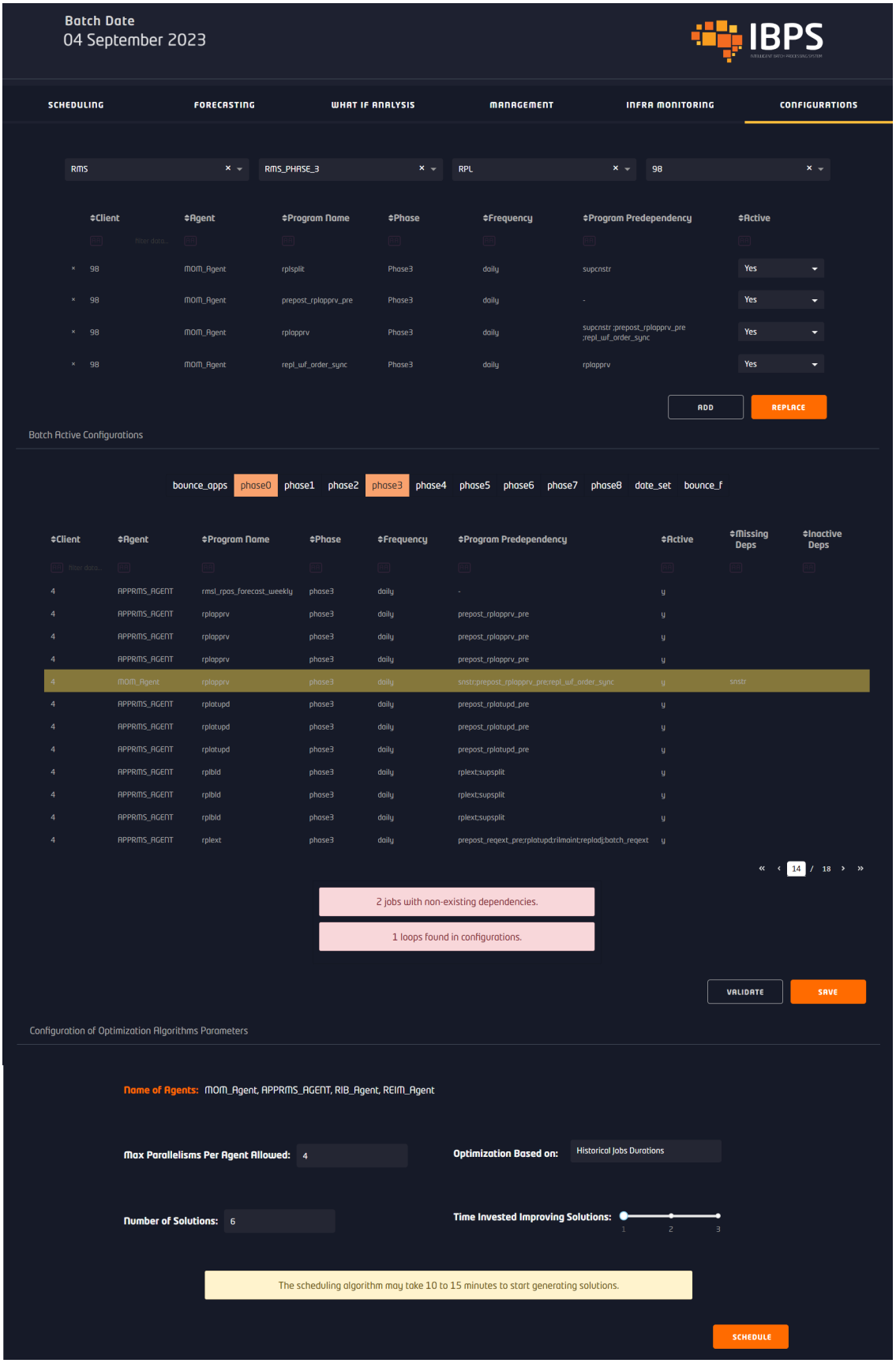

Figure 6.11: Detect Configuration Inconsistencies - Loops Caused by Dependencies

6.2.7 Test Case 07: Save Changes in Active Configurations

The Figure 6.12 and Figure 6.13 represents the TC07.

Description:
After changes have been made to the active batch configurations, the system should allow their validation, checking for inconsistencies in the dependencies between jobs, so that they can be applied in the optimization and execution of the batch.

Pre-Condition:

1. `T_BATCH_CONFIG_ALL` must be populated with template configuration data from different phases of different clients.

Procedure:

1. Select `RMS` from the Products dropdown.
2. Select `RMS_PHASE_3` from the Phases dropdown.
3. Select `RPL` from the Sub-Phases dropdown.
4. Choose a client from the Clients dropdown - 4, 5, 98, or 99 - and check the template configuration table.
5. Select another client, different from the previous one, from the Clients dropdown and check the template configuration table.

Expected Result:
The table with the selected template configurations should be updated with the settings of the selected client for the chosen phase.

Result:
Passed

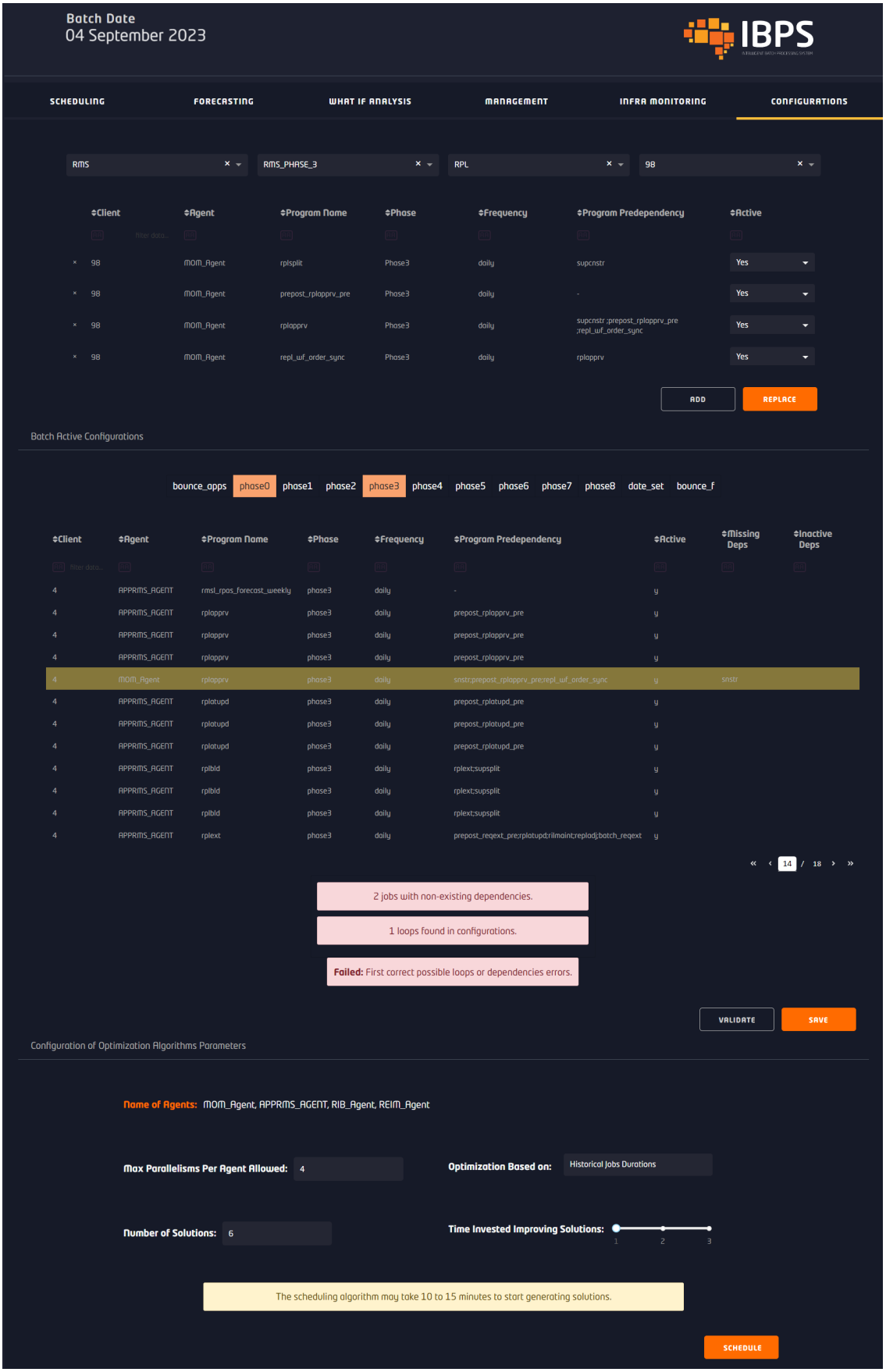

Figure 6.12: Preventing the User to Save Configurations with Errors

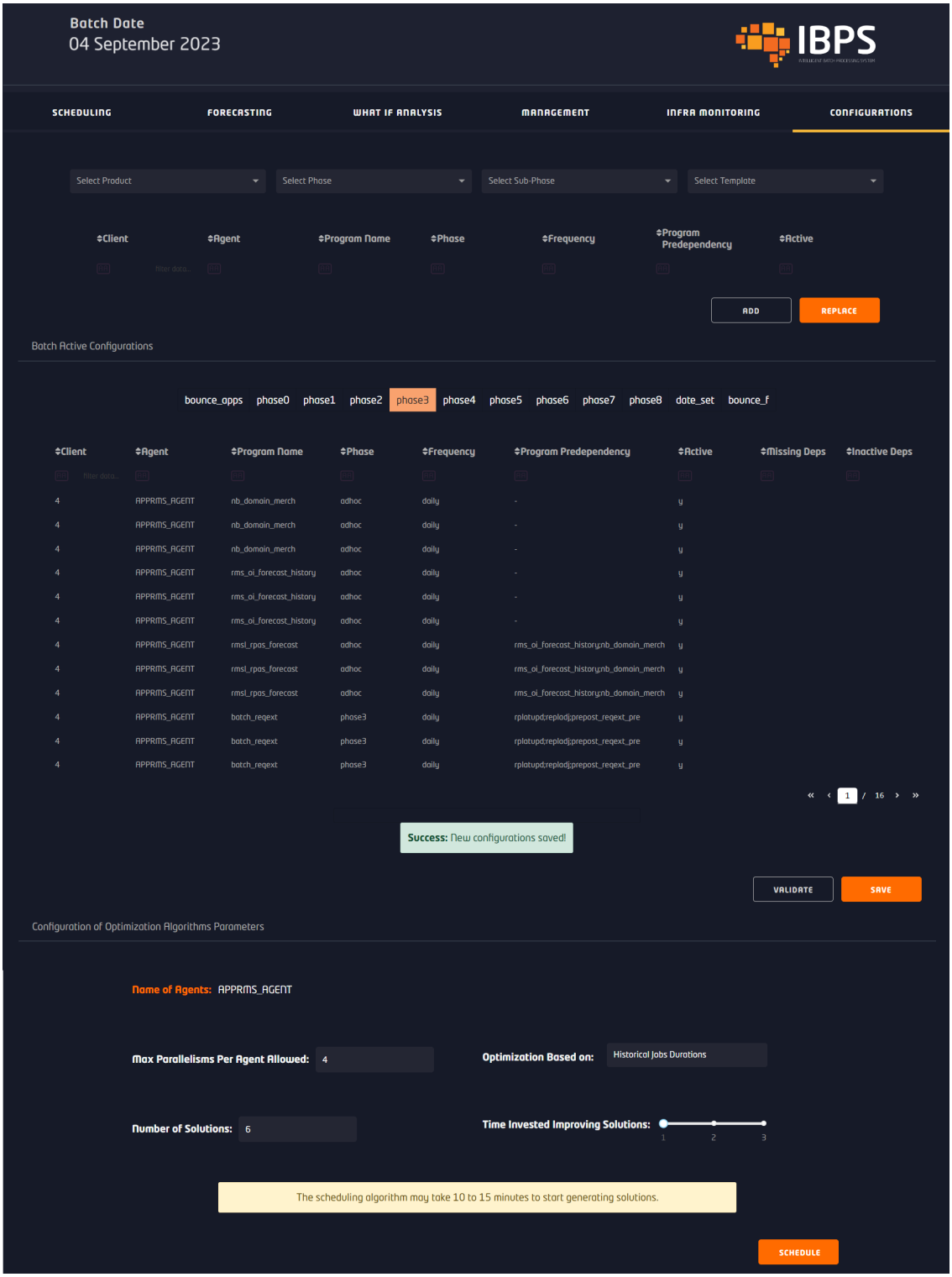

Figure 6.13: Saving Successfully Validated Configurations

6.2.8 Test Case 08: Generate Scheduling Solutions Dependencies

The Figure 6.19 represents the TC08.

Description:
The system should allow the user to generate the desired number of solutions, based on the active client configurations managed in the dashboard. These solutions can only be generated when any valid modification made to the client's active configurations (with the 'Add' or 'Replace' buttons) has been updated in the database using the 'Save' button.

Pre-Condition:

1. `T_BATCH_CONFIG_ACTIVE` must be populated with valid configurations for client 4.
2. The configurations in the section of active configurations must be updated in the `T_BATCH_CONFIG_ACTIVE` table, meaning there should be no unsaved changes in the database.

Procedure:

1. Refresh the Configurations tab (in case changes were made to the active configurations that have not been saved).
2. Select the number of solutions to be generated in 'Number of Solutions'.
3. Choose the 'AI Predicted Jobs Durations' option under 'Optimization Based On', so that the Scheduling algorithms are based on the times predicted by the Forecasting module.
4. Click the 'Schedule' button and wait for 5 to 10 minutes.
5. Switch to the 'Scheduling' tab and check the list of solutions ('Scheduling Solutions').

Expected Result:
In the 'Scheduling' tab, the 'Scheduling Solutions' list should contain a number of csv files equal to the number of solutions generated in the 'Configurations' tab. These files should be saved in the following folder: /oracle/TEST/Forecast/data/scheduling-solutions/

In the /oracle/TEST/Forecast/data/ folder, a 'configs_graph.obj' file should be generated, containing the graph of the active batch configurations.

Result:
Passed

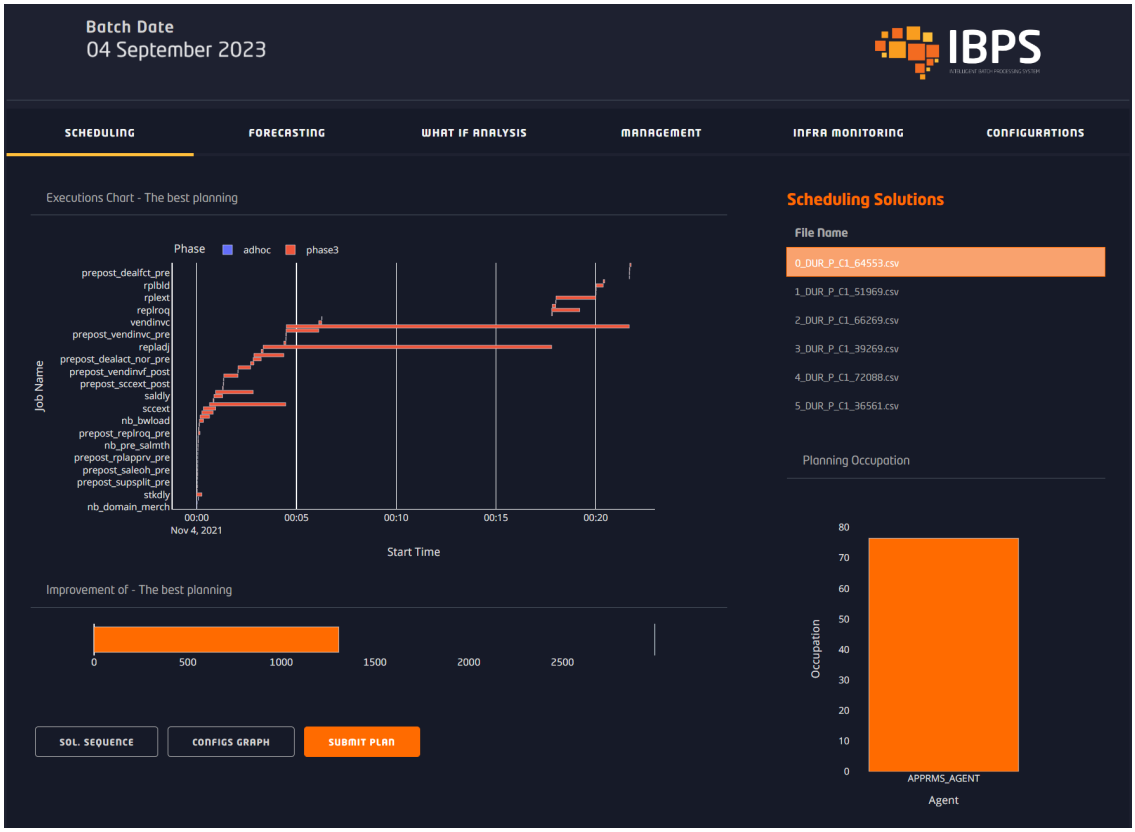

Figure 6.14: Generated Solutions in the Configuration Tab

6.2.9 Test Case 09: Ensure Coherence of Scheduling Solutions with the Saved Active Configurations

Figure 6.15 represents the first part of TC09, while Figure 6.16 represents the second part.

Description:
The system should only generate solutions when all modifications made to the client's active configurations are saved in the database (using the 'Save' button).

Pre-Condition:

None

Procedure:

1. Select 'RMS', 'RMS_PHASE_0', and '-' in the 'Product', 'Phase', and 'Sub-Phase' dropdowns respectively.
2. Select client 4 in the 'Client' dropdown.
3. Click the 'Add' button.
4. Try to generate Scheduling solutions by clicking the 'Scheduling' button.

Expected Result:
Error Message - The user must update the modifications, correct errors if they exist, and save them in the database with the 'Save' button before generating Scheduling solutions.

Result:
Passed

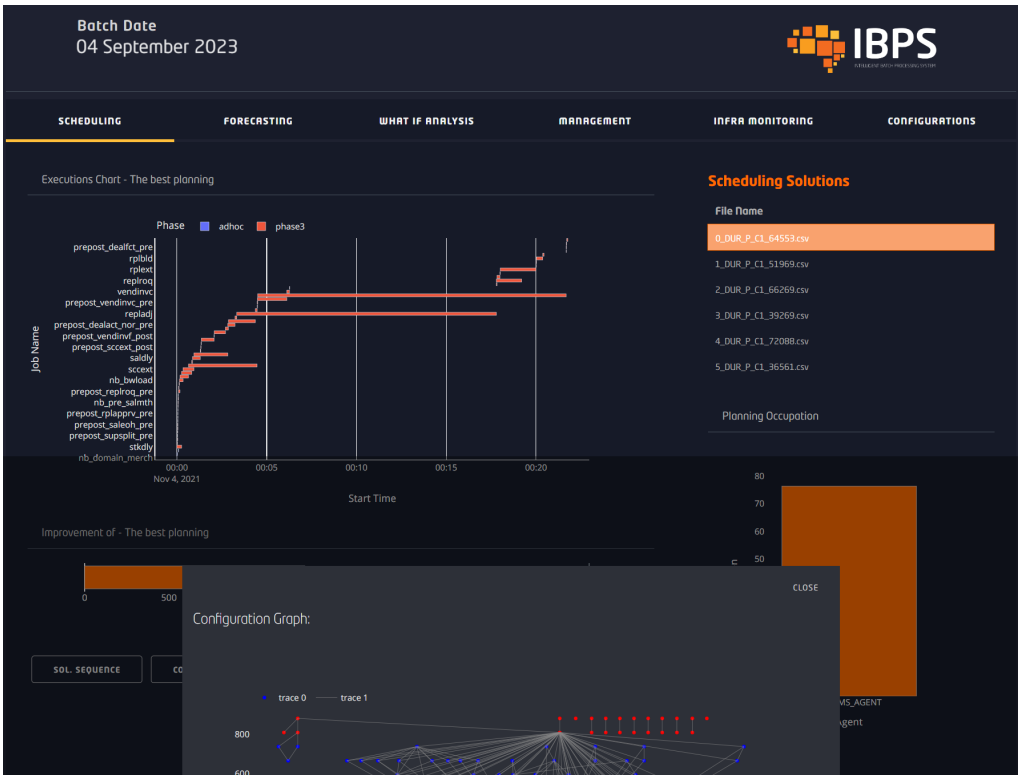

Figure 6.15: TC09 - Selection of Active Configurations in the Configurations Tab

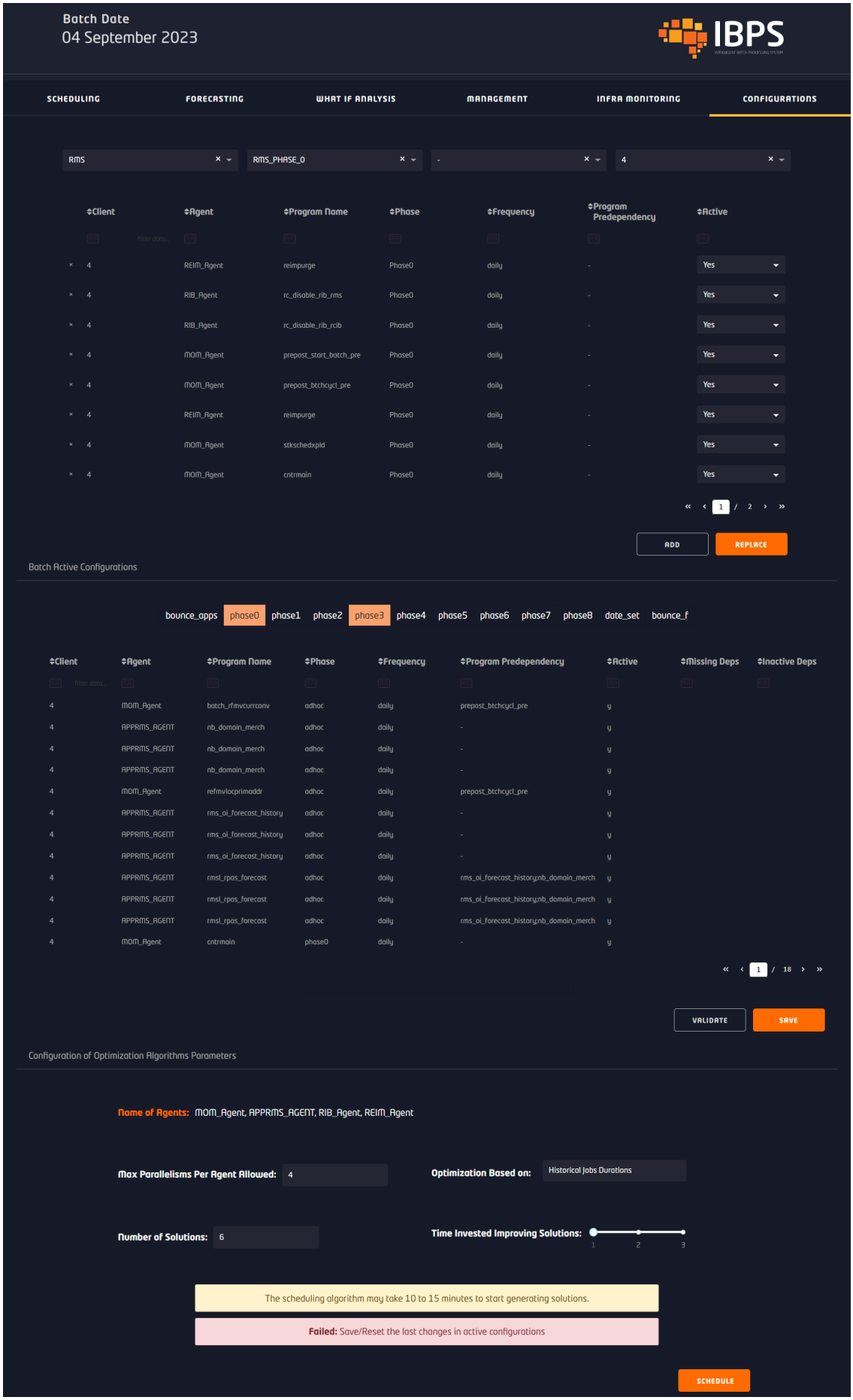

Figure 6.16: TC09 - Error Message Prompting to Save Changes Before Scheduling

6.3 Monitoring

Ensuring real-time tracking and assessment of the Monitoring component is essential for the system's overall resilience and efficiency. Comprehensive testing procedures have been put in place to confirm that the system adheres to the clearly defined requirements for this critical module. Through a series of test cases (TCs), our goal is to systematically evaluate each specified requirement and ensure that the system's behavior matches stakeholder expectations.

As detailed in the functional requirements, the Monitoring component is central for real-time surveillance and management of system performance metrics. It's of utmost importance to assess the system's ability in regular operations and its flexibility in responding to unexpected challenges. In view of this, the following test cases have been crafted:

RF Identifier	TC Identifier	Test Case - Client Profile
MTR-REQ-01	TC 01	Monitor environment resources
MTR-REQ-02	TC 02	View collected data
MTR-REQ-03	TC 03	Configure custom queries and filters
MTR-REQ-04	TC 04	View dynamic graphs
MTR-REQ-05	TC 05	Configure alerts

Table 6.3: Test cases for Monitoring Functional Requirements

6.3.1 Execution and Results

A total of 5 functionalities were tested from a system test scenario, considering the test case design defined for the Monitoring modules (see Table 6.3). Any discrepancies or anomalies were identified and rectified during the execution of each test scenario. Tests for individual cases were performed iteratively until an "approved" result was achieved.

Table 6.4 presents the integrated test scenario of the Monitoring module. For clarity, Table 6.4 furnishes the details of the execution, procedures, and expected outcomes of the tests.

6.3.2 Execution and Results

A total of 5 functionalities were tested from a system test scenario, considering the test case design defined for the Monitoring modules (see Table 4.4). Any discrepancies or anomalies were identified and rectified during the execution of each test scenario.

Table 6.4 presents the integrated test scenario of the Monitoring module. For clarity, Table 6.4 furnishes the details of the execution, procedures, and expected outcomes of the tests.

Code	TC 01 to TC 05
Test Scenario	Integrated test of the Monitoring module.
Description	This use case scenario has been constructed to test the Monitoring functionalities within the scheduling system, replicating a typical Monitoring use case scenario. The scenario entails real-time monitoring of system metrics including CPU, memory, and storage.
Pre-Conditions	The system must be active and operational, ensuring the Monitoring module can access system metrics and that the user can define and activate alerts.
Procedure	1 - Activate the Monitoring module within the scheduling system. 2 - Initiate real-time monitoring of specified metrics. 3 - Configure custom queries, filters, and alerts as per requirements.
Expected Results	Real-time monitoring metrics are accessible, and visualizations display current system health. Alerts are triggered based on defined conditions, and the user receives notifications of these alerts promptly.
Result	Approved

Table 6.4: Monitoring Module Test Scenario (TC 01 to TC 05)

6.4 Forecasting

Ensuring the functionality and reliability of the Forecasting component is paramount for the accurate prediction and analysis of future events in the system. Comprehensive testing has been initiated to validate the requirements particularly designed for this key component. Through a spectrum of test cases (TCs), we aim to exhaustively evaluate each defined requirement and confirm that the system's behaviors align with the expectations of stakeholders.

As highlighted in the functional requirements, the Forecasting component plays a pivotal role in predicting and assessing future events based on historical data. Determining the system's accuracy in such predictions and its resilience in evolving scenarios is crucial. Hence, the following test cases have been devised:

6.4.1 Test Case 01: List Customer Phases

The Figure 6.17 represents the TC01.

Description:
The dropdown for phases should list, as options for the user to select, the different phases of the client's nighttime batch.

Pre-Condition:

- T_PROCESS_EXECUTIONS_FILTERED contains processed, filtered, and updated data from the executions of the client's nighttime batch jobs.

Procedure:

RF Identifier	TC Identifier	Test Case - Client Profile
FCST-REQ-01	TC 01	List Client Phases
	TC 02	List Jobs from a Client's Phase
	TC 03	View Job Execution History
	TC 04	View Total Batch Execution History
FCST-REQ-03	TC 05	Generate Job Execution Predictions
	TC 06	Inform Confidence Level of Specific Job Predictions
	TC 07	Generate Total Batch Execution Predictions
FCST-REQ-04, FCST-REQ-05	TC 08	List Jobs with Highest Predicted Duration
FCST-REQ-02	TC 09	List Client Transaction Types
	TC 10	List Client Price Change Types
	TC 11	View Client Transaction History
	TC 13	View Client Price Change History
	TC 15	View Job Workload History
RF07	TC 12	View Client Transaction Predictions
	TC 14	View Client Price Change Predictions

Table 6.5: Test cases for Forecasting Functional Requirements

1. Click on the phases dropdown and check the list.

Expected Result:
The dropdown displays a list with phases from the client's nighttime batch.

Result:
Passed

6.4.2 Test Case 02: List Jobs from a Customer Phase

The Figure 6.17 represents the TC02.

Description:
The dropdown for jobs should list, as options for the user to select, the different jobs that run in one of the phases selected by the client in TC 01.

Pre-Condition:

- T_PROCESS_EXECUTIONS_FILTERED loaded with processed, filtered, and updated data from the executions of the client's nighttime batch jobs.

Procedure:

1. Select a phase of the client.
2. Click on the jobs dropdown and check the list.

Expected Result:
The dropdown displays a list with the names of the client's jobs belonging to the selected phase.

Result:
Passed

6.4.3 Test Case 03: Visualize Job Execution History

The Figure 6.17 represents the TC03.

Description:
Update the execution chart with historical information on the durations of the selected job.

Pre-Condition:

- T_PROCESS_EXECUTIONS_FILTERED loaded with processed, filtered, and updated data from the executions of the client's nighttime batch jobs.

Procedure:

1. Select a phase of the client.
2. Select one of the jobs listed from the phase.
3. Check the forecast chart on the left side.

Expected Result:
The forecast chart should draw a line with historical values of the duration of the job selected by the user up to the date of the last batch execution.

Result:
Passed

6.4.4 Test Case 04: Visualize Batch Total Execution History

The Figure 6.18 represents the TC04.

Description:
Update the execution graph with historical information on the client's total Batch durations.

Pre-Condition:

1. T_PROCESS_EXECUTIONS_FILTERED filled with executions of the client's nighttime batch jobs up to the current date.
2. José's service to receive job durations on a given Batch day and with these calculate the critical path and the total Batch duration on that day.

Procedure:

1. Leave the phase and/or job dropdown empty.
2. Check the forecast chart on the left side.

Expected Result:
The forecast chart should draw a line with historical values of the client's total Batch duration up to the date of its last execution.

Result:
Passed

6.4.5 Test Case 05: Generate Job Execution Forecasts

The Figure 6.17 represents the TC05.

Description:
Select the best model and generate forecasts for the selected job.

Pre-Condition:

1. T_VALIDATION_FORECASTS updated with validation results - MAE and MASE metrics - to select the model to be used.
2. Table with trained models to load and use in the client's job forecasts.
3. T_DURATION_MODEL loaded with models trained to predict the different jobs of the client's Batch.
4. T_PROCESS_EXECUTIONS_FILTERED, T_JOB_AGENT, T_HOST_MAPPING, T_HOST_PROPERTIES, and T_PARALLELISMS, updated with data to create features used as input in the models.
5. T_DYN_BUSINESS_MODEL loaded with SARIMA models trained to predict future client transactions and price changes, to be used as input in the job duration forecast models.

Procedure:

1. Select a phase of the client.
2. Select one of the jobs listed from the phase.
3. Check the forecast chart on the left side.
4. Check the summary panel on the right side.

Expected Result:
The forecast chart should trace predictions and confidence intervals for the duration of the selected job in the client's next Batch days, generated by the most suitable model to predict it.

Result:
Passed

6.4.6 Test Case 06: Inform Confidence Level of Job Forecasts

The Figure 6.17 represents the TC06.

Description:
Display the degree of certainty of job forecasts based on the validation result metrics.

Pre-Condition:

1. T_VALIDATION_FORECASTS updated with validation results - MAE and MASE metrics - of the selected job.

Procedure:

1. Observe the confidence intervals of the forecasts on the left forecast chart.

Expected Result:
The forecast chart should show shaded confidence intervals around the predictions of the duration of the selected job in the client's next batch days, generated by the most suitable model to predict it.

Result:
Passed

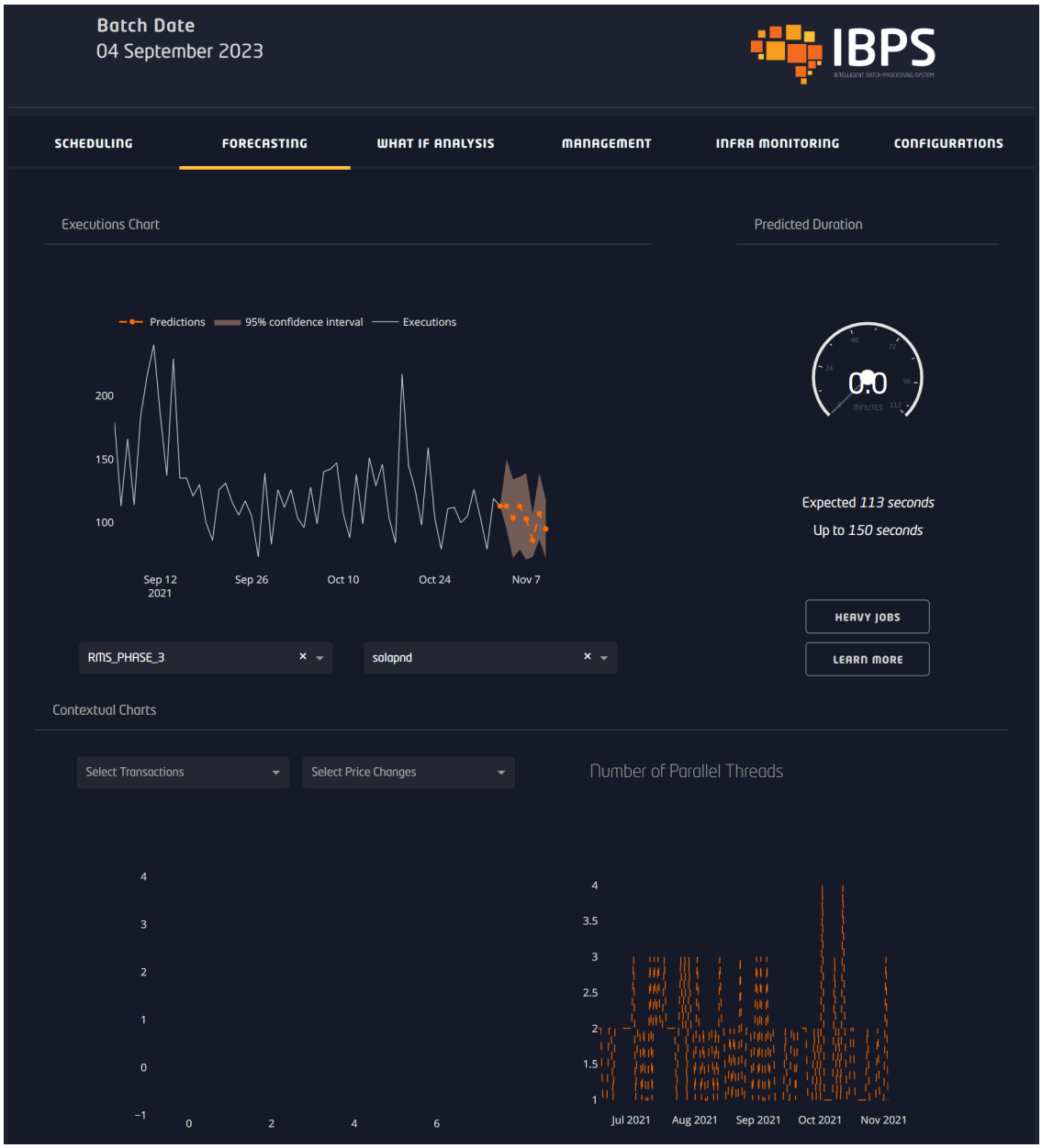

Figure 6.17: Historical and Predicted Durations for Batch Jobs

6.4.7 Test Case 07: Generate Forecasts for the Total Batch Executions

The Figure 6.18 represents the TC07.

Description:
Generate forecasts for the total Batch executions in the upcoming days, using the job forecasts in the upcoming batch days to calculate the critical path of the batch graph.

Pre-Condition:

1. All pre-conditions from TC 05 should be checked, in order to be able to generate forecasts for all the jobs in the client's night batch in the upcoming days.
2. José's service to receive predicted durations for all the client's jobs in the upcoming batch days, and use these timings in conjunction with the settings to generate the critical path and total batch time on these same dates.

Procedure:

1. Leave the phase and/or job dropdown empty.
2. Check the forecast chart on the left side.
3. Check the summary panel on the right side.

Expected Result:
The forecast chart should plot predictions for the total batch duration in the upcoming days for the client, combining the critical path generation service with the forecasts generated for the different batch jobs.

Result:
Passed

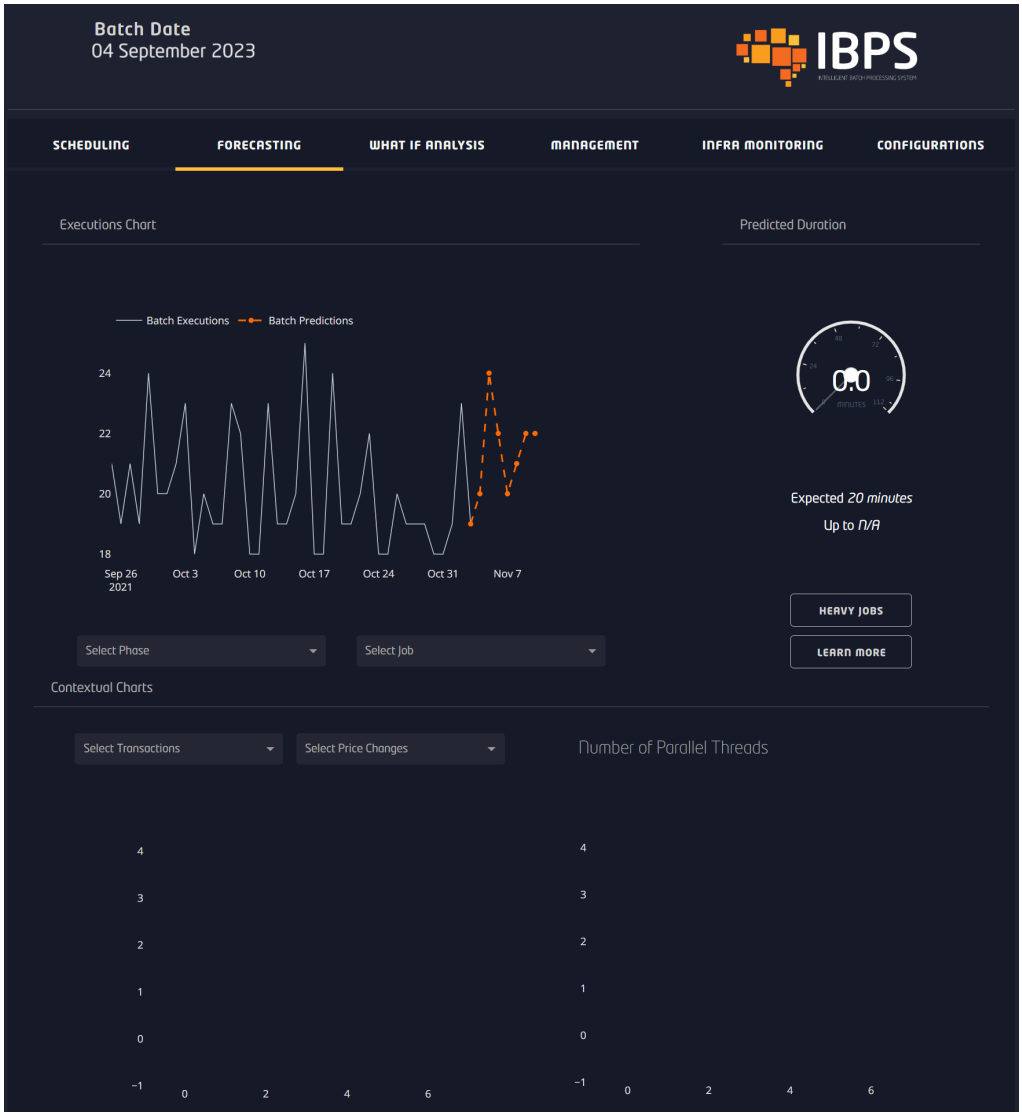

Figure 6.18: Historical and Predicted Durations for Batch Executions

6.4.8 Test Case 08: List Jobs with Longest Predicted Duration

The Figure 6.19 represents the TC08.

Description:
The "Heavy Jobs" button should list the jobs with the longest predicted duration for the client's next batch day.

Pre-Condition:

1. All pre-conditions from TC 05 should be checked, in order to be able to generate forecasts for all the jobs in the client's night batch in the upcoming days.
2. José's service to receive predicted durations for all the client's jobs in the upcoming batch days, and use these timings in conjunction with the settings to generate the critical path and total batch time on these same dates.

Procedure:

1. Click on the "Heavy Jobs" button and observe job lists in the "In General" and "In Critical Path" sections.

Expected Result:
Table with job names and their respective phase with the longest predicted durations for the next batch, sorted in descending order.

Result:
Passed

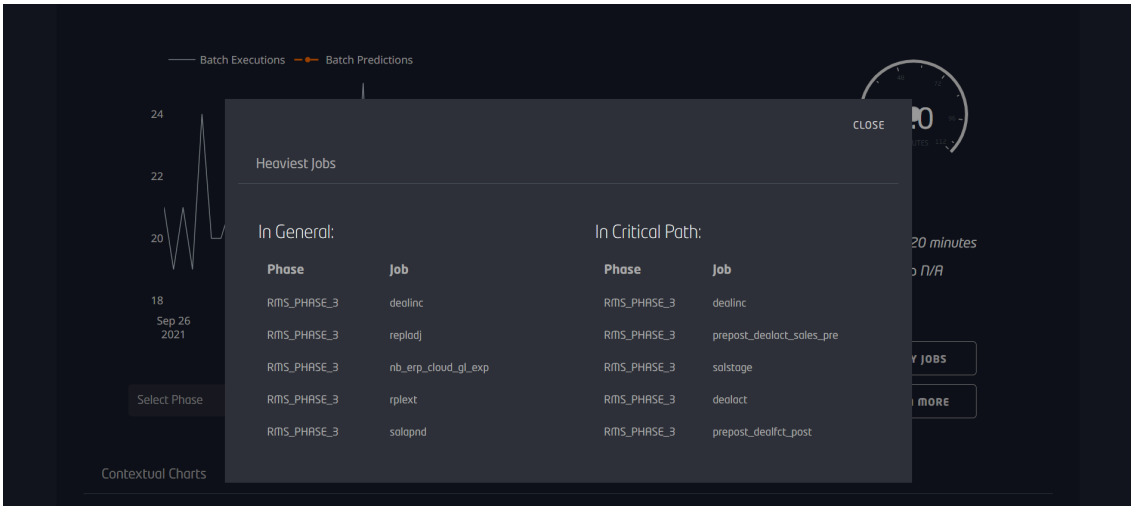

Figure 6.19: Predicted Heaviest Jobs

6.4.9 Test Case 09: List Client's Transaction Types

The Figure 6.20 represents the TC09.

Description:
The transaction dropdown should list, as selection options, the names of the different transaction types of the client.

Pre-Condition:

1. T_TRANSACTIONS_BY_DAY updated with the transaction history of each type, by date, up to the current date.

Procedure:

1. Click on the dropdown of the client's transaction types and check the list.

Expected Result:
The dropdown displays a list of different transaction types registered in the client.

Result:
Passed

6.4.10 Test Case 10: List Client's Price Change Types

The Figure 6.20 represents the TC10.

Description:
The price change dropdown should list, as selection options, the names of the different price change types of the client.

Pre-Condition:

1. T_PRICE_CHANGES updated with the price change history of all types, on different days of the client.

Procedure:

1. Click on the dropdown of the client's price change types and check the list.

Expected Result:
The dropdown displays a list with different types of price changes registered in the client.

Result:
Passed

6.4.11 Test Case 11: Visualize Client's Transaction History

The Figure 6.20 represents the TC11.

Description:
When selecting a type of transaction, the business dynamic variables graph should display a line with the historical data of the number of transactions of that type in the client.

Pre-Condition:

1. T_TRANSACTIONS_BY_DAY updated with the transaction history of different types up to the current date.

Procedure:

1. Select a transaction type.
2. Check the business dynamic variables graph.

Expected Result:
The business dynamic variables graph should display the count of the selected transaction type across different operating days of the client up to the current date.

Result:
Passed

6.4.12 Test Case 12: Visualize Client's Transaction Predictions

The Figure 6.20 represents the TC12.

Description:
When selecting a type of transaction, the business dynamic variables graph should display estimated values for that type of client transaction in the coming days.

Pre-Condition:

1. T_DYN_BUSINESS_MODELS loaded with SARIMA models trained to predict values for each type of client transaction in the coming days.

Procedure:

1. Select a transaction type.
2. Check the business dynamic variables graph.

Expected Result:
The business dynamic variables graph should display estimated values for the selected transaction type for the coming operating days of the client.

Result:
Failed

6.4.13 Test Case 13: Visualize Client's Price Change History

The Figure 6.20 represents the TC13.

Description:
When selecting a type of price change, the business dynamic variables graph should display a trace with historical information on the number of price changes of that type for the client.

Pre-Condition:

1. T_PRICE_CHANGES updated with the historical record of price changes of all types, on different days for the client.

Procedure:

1. Select a type of price change.
2. Check the business dynamic variables graph.

Expected Result:
The business dynamic variables graph should show the count of the selected price change type on different operating days for the client up to the current date.

Result:
Passed

6.4.14 Test Case 14: Visualize Client's Price Change Predictions

The Figure 6.20 represents the TC14.

Description:
When selecting a type of price change, the business dynamic variables graph should display estimated values for that type of price change for the client in the coming days.

Pre-Condition:

1. T_DYN_BUSINESS_MODEL loaded with SARIMA models trained to predict values of each type of price changes for the client in the coming days.

Procedure:

1. Select a type of price change.
2. Check the business dynamic variables graph.

Expected Result:
The business dynamic variables graph should show estimated values for the selected type of price change in the client's upcoming operational days.

Result:
Failed

6.4.15 Test Case 15: View Job Workload History

The Figure 6.20 represents the TC15.

Description:
When a client job is selected, the workload graph should display the history of parallelisms, represented by the Workload variable, up to the current day.

Pre-Condition:

1. T_PARALLELISMS loaded with historical parallelism of different jobs and the host.
2. T_JOB_AGENT, T_HOST_MAPPING, and T_HOST_PROPERTIES updated to associate the host of each job and thus the CPU, RAM, and parallelism in the different job executions.

Procedure:

1. Select a phase from the client.
2. Select one of the listed jobs of the phase.
3. Check the job workload graph.

Expected Result:
The Workloads graph should trace the historical evolution of the workload calculation of the selected job, which encompasses parallelisms, the number of threads, and the CPU, for the different Batch days of the client up to the current date.

Result:
Passed

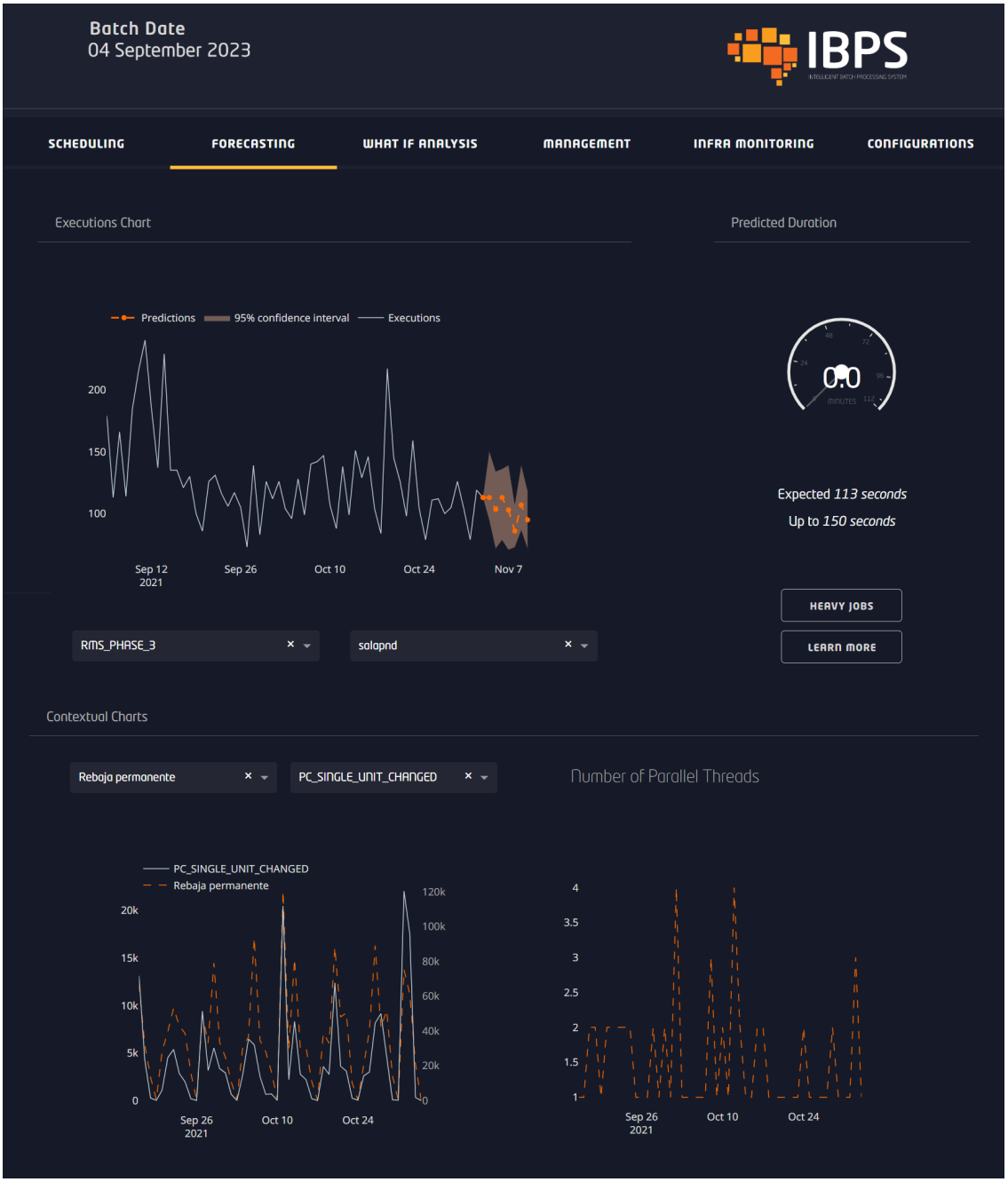

Figure 6.20: Contextual Variables

6.5 Disruption

Ensuring the functionality and reliability of the Disruption component is pivotal for effective error detection and management in the system. Rigorous testing has been initiated to validate the requirements specifically outlined for this essential module. Through a series of test cases (TCs), we aim to comprehensively evaluate each stated requirement and confirm that the system's behavior aligns with the desired outcomes.

As highlighted in the functional requirements, the Disruption component plays a central role in error detection, management, and recovery. It's paramount to assess the system's

competence in routine operations and its resilience in the face of unexpected disturbances. In light of this, the following test cases have been devised:

RF Identifier	TC Identifier	Test Case - Disruption Module
DRPT-REQ-01	TC 01	Identify error
DRPT-REQ-02	TC 02	Detect anomalies and errors based on previous patterns
DRPT-REQ-03	TC 03	Notify errors
DRPT-REQ-04	TC 04	Map error recovery actions
DRPT-REQ-05	TC 05	Execute automatic recovery
DRPT-REQ-06	TC 06	Log incidents and auto-healing action history
DRPT-REQ-07	TC 07	Monitor in real-time
DRPT-REQ-08	TC 08	Detect locks by job name
DRPT-REQ-09	TC 09	Immediately notify of locks

Table 6.6: Test cases for Disruption Functional Requirements

6.5.1 Execution and Results

A total of 9 functionalities were tested from a system test scenario, considering the test case design defined for the Disruption modules (see Table 4.6). Errors were identified and corrected during the execution of the test scenario by observing each test case individually, and their tests were exhaustively repeated until they achieved the "approved" result.

Table 6.7 presents the integrated test scenario of the IBPS Disruption module. As an example, Table 6.7 contains details of the execution, procedures, and expected test results.

Code	TC 01 to TC 09
Test Scenario	Integrated test of the Disruption module.
Description	This use case scenario is designed to test the functionalities of the IBPS scheduling system, simulating a real use case scenario of the Disruption module. The scenario includes the configuration of a batch composed of 10 jobs, each with specific error and auto-healing characteristics.
Pre-Conditions	(Provide the detailed description of the jobs, errors, and auto-healing as described above in the format)
Procedure	1 - Configure the batch with the 10 jobs described above. 2 - Start executing the batch in the IBPS scheduling system. 3 - Monitor the execution of the jobs.
Expected Results	(Provide the detailed expected results of the jobs as described above)
Result	Approved

Table 6.7: IBPS - Disruption Module Test Scenario (TC 01 to TC 09)

Table 6.8 lists the auto-healing execution configuration (Skip, Kill, or Start) by Job, depending on the planned error.

Id. Job	Type	Error Description	Action
Job 9	Script	System Error / Configuration - without logs	Skip – automatic simulation
Job 10	Script / Oracle Retail	Not expected duration – Overrun – Without error – 4min	Do nothing
Job 4	Script / Oracle Retail	Not expected duration – Overrun - Without error - 4 min	Kill
Job 1	Python	Normal / Lock – With manual intervention – Without error – after anomaly duration	None / Kill
Job 2	Python	Error simulation – SQL syntax – with LOG ORA	Skip
Job 3	Python	Error simulation – with invalid data – with LOG ORA	Skip

Table 6.8: Definition of auto-healing actions by Job and error

Figure 6.21 illustrates the interface of the IBPS system, specifically the Disruption module. We highlight below the functional requirements DRPT-REQ 01 to 09, previously listed in Table 4.6 of the test plan subsection 4.1.2. At the top of this image, information is summarized that allows real-time batch processing monitoring (DRPT-REQ 07). The summaries are consistent with the test scenario of the module described in Table 6, illustrating the total of 9 submitted jobs, of which 5 presented failures. In the subsequent panels, DRPT-REQ 01 to 06 and 08 and 09 can be observed, displaying the jobs that failed by error code, and the error execution history by correction action as previously defined in the test scenario.

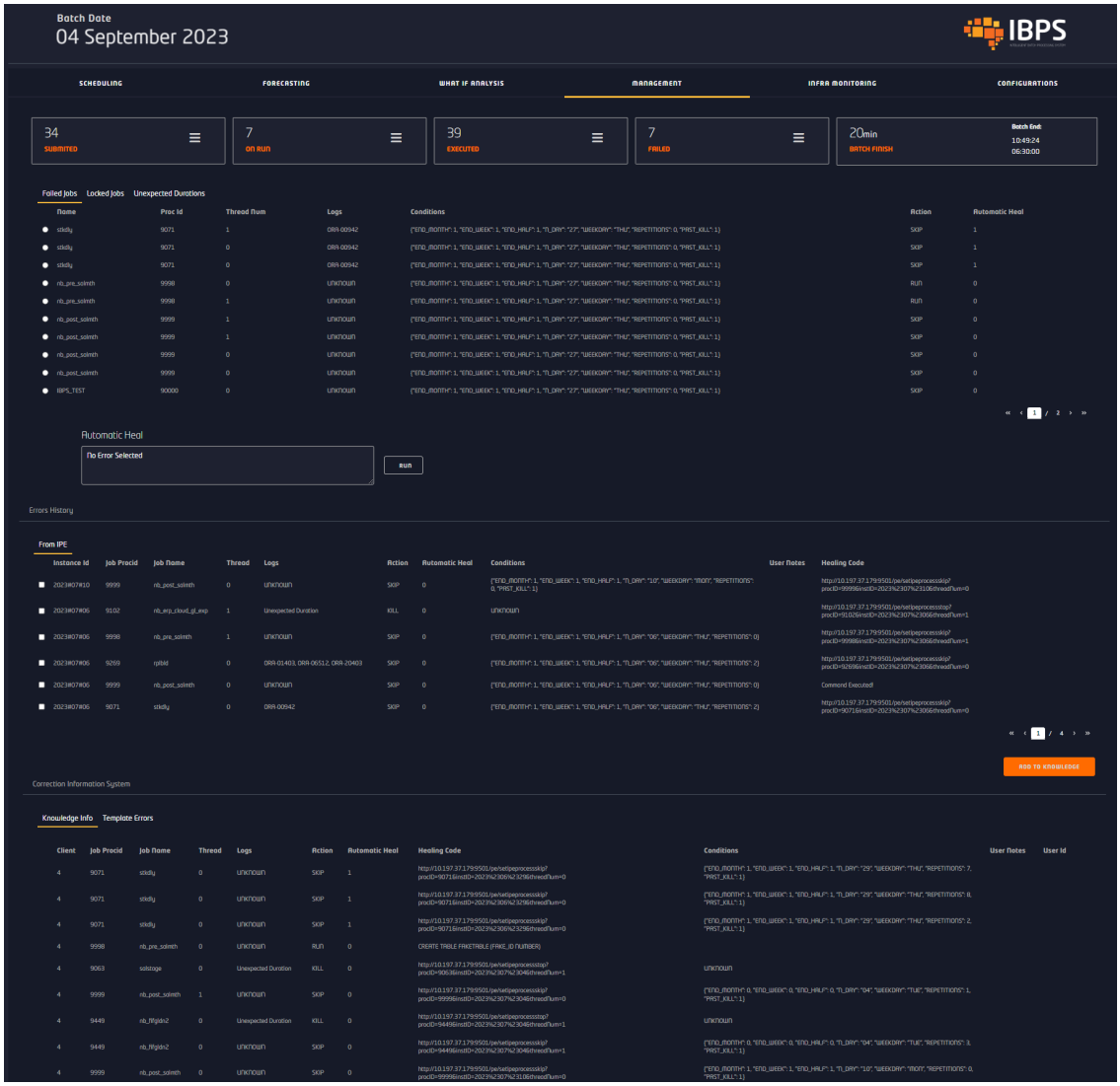

Figure 6.21: Module Disruption Management

6.6 Concluding Remarks

Throughout this chapter, a series of test cases have been presented that highlight the system's proficiency and robustness. They exemplify the seamless communication between the User Interface and the Job Scheduler (IPE) and the successful integration of AI modules. Overall, the results indicate that the system is ready for deployment. The importance of ensuring reliability in a system as complex as the IBPS is paramount.

The tests not only confirm the system's current efficacy but also lay the groundwork for future enhancements, ensuring that the system remains adaptable, resilient, and reliable. While most tests affirmed the system's capabilities, any test that did not pass successfully was given due attention. These setbacks, primarily resulting from minor issues, offered valuable insights. Each failure was treated as an opportunity to identify and rectify areas requiring improvement.

Chapter 7

Conclusion

This chapter has as its main objective to present the conclusions about this project, in particular, the achieved objectives, limitations and future work.

7.1 Achieved Objectives

Oracle Retail, with its intricate collection of modules, demands meticulous integration through batch processing. The implementation of a batch is not only time-consuming, often spanning several months, but is also critical to the operations of retailers. The stakes are high, as batches must conclude before business hours commence. This process often necessitates manual interventions by support teams to identify technological events, pinpoint their functional origins, and rectify any discrepancies. The necessity for close monitoring and swift interventions is paramount to ensure smooth operations and address issues without delay.

In light of these challenges, this project embarked on the development of the IBPS prototype. Traditional batch processing methods, although previously effective, now face limitations in the current digital landscape. The IBPS, or Intelligent Batch Processing System, stands as a testament to the power of integrating advanced Artificial Intelligence (AI) into batch processing. It offers a transformative approach to batch processing, enabling batches to be configured, optimized, scheduled, and monitored.

The objectives of this project were the design and implementation of the IBPS prototype, which incorporated a user interface tailored for seamless interaction with the job scheduler (IPE). Designed with the specific needs of Retail Consult and its clientele in mind, the prototype aimed to meet batch processing requirements effectively. A pivotal aspect was the integration of existing AI modules, enhancing the system's capabilities and facilitating management through the user interface. The system's performance and reliability were further affirmed through validation in a controlled Oracle Retail environment.

Reflecting on the principal aims outlined in section 1.4, the project's objectives have been notably achieved. The IBPS is equipped with a range of advanced features and signifies more than just an advanced batch processing tool; it represents a strategic shift in data management and processing. By integrating AI capabilities, it paves the way for innovative data analysis, automation, and process optimization.

The introduction of IBPS signifies more than just an advanced batch processing tool; it represents a strategic shift in data management and processing. By offering advanced functionalities and integrating AI capabilities, it paves the way for innovative data analysis, automation, and process optimization possibilities.

7.2 Limitations and Future Work

Although the project was successful, there are some aspects that could be addressed to improve or complement the work that was done.

One of the limitations at the moment is the correction of the tests failures found on tests cases.

The following enhancements and architectural patterns could be investigated further to bolster the robustness and efficiency of the system:

- **Optimized Communication:** Enhancing communication strategies can augment system responsiveness. Investigating the potential of an **API Gateway** might streamline and manage service requests, reducing the complexity of inter-service communication (Newman 2015).
- **Security Enhancements:** As the system grows, so do potential security vulnerabilities. A comprehensive security strategy encompassing authentication, authorization, and secure inter-service communication will fortify the system's defenses (Richardson 2018).
- **Advanced Design Patterns:** Patterns like **CQRS**, **Event Sourcing**, and **Domain Events** can be integrated to improve system flexibility and data management capabilities (Richardson 2018).
- **Database Management:** Implementing the **Database-per-Service** pattern can decouple services and their respective databases, ensuring data consistency and reducing potential data conflicts (Newman 2015).
- **Event-Driven Architecture:** Transitioning to an event-driven architecture using **Messaging** and **Domain Events** can make the system more reactive, allowing it to respond dynamically to changes and user interactions (Kleppmann 2017).

Refer to Figure 7.1 for the visual representation of the logical view following the approaches mentioned previously.

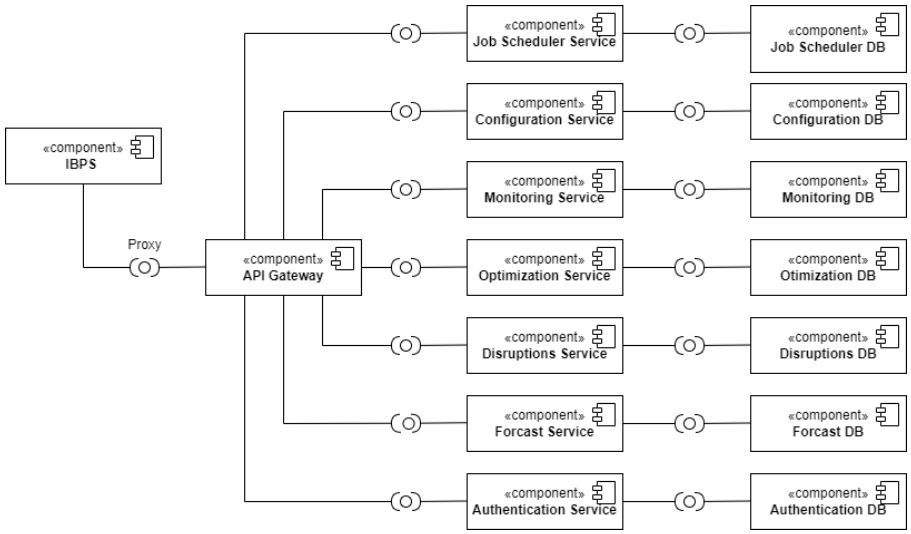

Figure 7.1: Logical view of the system components.

Figure 7.2 illustrates the system's deployment setup.

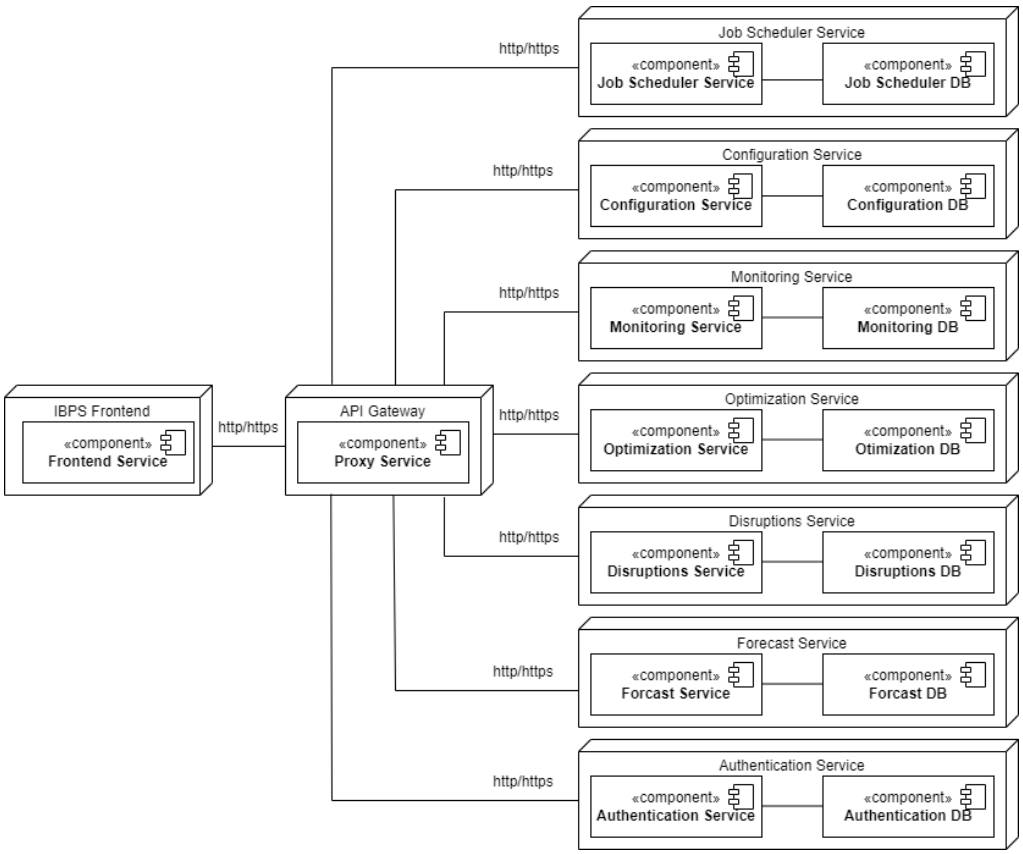

Figure 7.2: Deployment view of the system components.

Bibliography

_UEUEING, CLUSTER (1994). "A Comparison of Queueing, Cluster Distributed Computing Systems". In.

Ammon, Rainer von et al. (2010). "Integrating complex events for collaborating and dynamically changing business processes". In: *European Conference on a Service-Based Internet, International Conference on Service-Oriented Computing*. Springer, pp. 370–384.

Anandakumar, H, R Arulmurugan, and Chow Chee Onn (2019). "Computational intelligence and sustainable systems". In: *EAI/Springer Innovations in Communication and Computing*.

Chan, Lai-Kow and Ming-Lu Wu (2002). "Quality function deployment: A literature review". In: *European journal of operational research* 143.3, pp. 463–497.

Chung, Lawrence et al. (2009). *Non-functional requirements in software engineering*. Springer Science & Business Media.

Ekman, S. and K. Jensen (2018). "Java in modern computing: Applications, performance, and evolution". In: *Journal of Computer Science and Technology* 33.4, pp. 675–689.

G2 (2023). *G2 Research Scoring Methodologies*. url: `https://research.g2.com/methodology/scoring` (visited on 2023).

Glinz, Martin (2007). "On non-functional requirements". In: *Requirements Engineering* 15.1, pp. 55–66.

Jones, James Patton and Cristy Brickell (1997). *Second evaluation of job queuing/scheduling software: Phase 1 report*. Tech. rep. Citeseer.

Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media.

Labs, Grafana (2023). *Grafana Official Documentation*. Accessed: 2023. url: `https://grafana.com/docs/`.

Lopes, Raquel V and Daniel Menascé (2016). "A taxonomy of job scheduling on distributed computing systems". In: *IEEE Transactions on Parallel and Distributed Systems* 27.12, pp. 3412–3428.

Mohan, K. and S. Oberoi (2019). "Oracle database systems: A comprehensive study on performance, scalability, and robustness". In: *International Journal of Database Management Systems* 11.3, pp. 56–71.

Newman, S. (2015). *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media.

Oh, Haemoon (1999). "Service quality, customer satisfaction, and customer value: A holistic perspective". In: *International journal of hospitality management* 18.1, pp. 67–82.

Oracle (Nov. 2018). *Oracle® Retail Merchandising System Operations Guide Volume 1*. Operations Guide, Volume 1 - Batch Overviews and Designs. E87600-03.

Osterwalder, Alexander and Yves Pigneur (2003). "Modeling value propositions in e-Business". In: *Proceedings of the 5th international conference on Electronic commerce*, pp. 429–436.

Osterwalder, Alexander, Yves Pigneur, et al. (2015). *Value proposition design: How to create products and services customers want*. John Wiley & Sons.

Richardson, C. (2018). *Microservices Patterns: With examples in Java*. Manning Publications.

Sánchez-Fernández, Raquel and M Ángeles Iniesta-Bonillo (2007). "The concept of perceived value: a systematic review of the research". In: *Marketing theory* 7.4, pp. 427–451.

Shetty, R. and E. Wright (2018). "Dash and Plotly: Empowering data scientists in web application development". In: *Proceedings of the 10th International Conference on Data Visualization*, pp. 213–220.

Smith, John A. and Jane B. Doe (2021). "Dynamic Resource Allocation in Modern Job Scheduling Systems Using AI". In: *Journal of Advanced Computational Systems* 58.2, pp. 315–329.

Van Rossum, G. and F. L. Drake (2018). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.

Vetter, Jeffrey S (2017). "Contemporary high performance computing". In: *Contemporary High Performance Computing*. Chapman and Hall/CRC, pp. 3–11.

Witt, Carl et al. (2019). "Predictive performance modeling for distributed batch processing using black box monitoring and machine learning". In: *Information Systems* 82, pp. 33–52. issn: 0306-4379. doi: https://doi.org/10.1016/j.is.2019.01.006. url: https://www.sciencedirect.com/science/article/pii/S0306437918301765.

Young, Nathan (Aug. 2021). *Oracle® Retail Merchandising Batch Schedule*. Release 19.3.000. url: https://www.oracle.com/retail/merchandising/ (visited on 05/23/2023).