Building a Big Data Oriented Architecture for Enterprise Integration

Le Hoang Nam and Phan Duy Hung(✉)

FPT University, Hanoi, Vietnam
nam19mse13037@fsb.edu.vn, hungpd2@fe.edu.vn

Abstract. Digital transformation is happening across all industries and affecting all facets of our daily life. However, in many corporations, this important process is fragmented and is undertaken without a farsighted plan to take advantage of an invaluable resource: data. This can be due to a variety of reasons, for example, lack of funding, poor business vision, inappropriate consulting or deployment. Digital transformation is a considerable investment since it will determine the system's ability to grow and adapt to the company's changing requirements. To achieve that end, the architecture must be flexible both in development and deployment and must also be able to harness the ever-increasing data of the corporation. Among the widely used information system architectures being used in the world, Microservice is a standout with many advantages. The adaptation of this architecture to work with Big Data, as well as to tackle different aspects of a data system such as load-balancing, file handling and storage, etc. is a very practical area of research. This paper presents such an enterprise integration solution for a mega-corporation client in Vietnam, the An Pha Petrol Group Joint Stock Company, including the architecture and technologies used to build a comprehensive system that brings novel experiences to its 2,000 internal users. It consists of building the information infrastructure and system, super applications for both desktop and mobile devices to enhance the work performance and quality. The approaches and results of this paper are applicable to similar large enterprise solutions.

Keywords: Enterprise integration · Architecture · Big Data · Microservice

1 Introduction

Digital transformation is not just a trend among big corporations, but it is becoming a necessity for all companies in the age of Industry 4.0. Many are aware of the requirement but do not understand all the advantages or the know-hows. An information system requires 3 separate parts that must be tackled simultaneously: digital infrastructure, digitalized management system and digitalized production data. However, many companies are using fragmented and inefficient pieces of software to manage accounting, production, warehousing, etc., without the ultimate aim of controlling their biggest asset in this process: data. Meanwhile, they should have "started small" but always keep the "big picture" in mind. The small start may include things such as digitizing documents to

Y. Luo (Ed.): CDVE 2021, LNCS 12983, pp. 172–182, 2021.
https://doi.org/10.1007/978-3-030-88207-5_17

keep printing down, digitalize tasks assignment and progress tracking. These works will gradually make using software part of the company's culture. At the same time, using the data gathered with new technologies such as Big Data, Artificial Intelligence, Machine Learning, Blockchain, etc. to achieve the big picture of data-driven decision making in building a smart management system that can bring about better customer satisfaction, shorter time-to-market, increased efficiency and more.

Such a "big picture" information system, of course, requires a similarly grand vision from the directors of the company. And they too need an architect that can build an optimized and flexible system that can adapt to the growing demand of the company in both business and data.

Among the widely used information system architectures, Micro-service is one with many advantages. The microservice architecture is style that structures a system as a collection of services. It enables the delivery of large and complex systems by frequently adding loosely coupled, independent applications. The flexibility and high maintainability are the key features that make Micro-service extremely suitable for large systems.

Regarding building and deploying information system solutions for corporations, as well as optimizing solutions to the integration of different modules in a system, several related studies were found.

In chapter "Microservice Architecture" [1], Bob Familiar introduce an approach of designing microservices using Separation of Concern (SoC). SoC is a design principle for separating implementation into distinct layers corresponding to separate concern. Microservices architecture uses SoC to identify business concerns. Then the business & data layer of 3-Tier Architecture or Layered Architecture is vertically sliced into isolated & bounded context services, each with its own domain model and API.

Alexis Henry and Youssef Ridene in [2] state that the trade-off is essential in order to successfully migrate your business applications toward microservices. The work aims to drive readers through a journey by presenting a roadmap and methodology which has been used successfully in several projects. They guide readers through the typical microservice migration project by using migration patterns for managing service decomposition and data isolation and replication. Those patterns may be used iteratively in any order, therefore authors defined a reference architecture to sequence the building of your microservice architecture. Eventually they conclude with a use case from the real world.

Following previous work on the automated deployment of component-based applications, Mario Bravetti et al. present a formal model specifically tailored for reasoning on the deployment of microservice architectures [3]. The authors present a formal proof of decidability of the problem of synthesizing optimal deployment plans for microservice architectures, a problem which was proved to be undecidable for generic component-based applications. Then, given that such proof translates the deployment problem into a constraint satisfaction problem, they present the implementation of a tool that, by exploiting state-of-the-art constraint solvers, can be used to actually synthesize optimal deployment plans. The work evaluates the applicability of the tool on a realistic microservice architecture taken from the literature.

The above studies have shown the deployments of Micro-service for many different corporations. However, they have yet explored how to combine this architecture with a

Big Data processing business. They also have not gone into details about the technical requirements that are frequently needed for integration and information management solutions.

This paper gives a Micro-service architecture as implemented for large organizations with a Big-data oriented approach. In the different modules, the sub-modules that are frequently used and play an important role in the overall performance of the system are presented and analyzed in details. For example, the load-balancing, dockerization, file size optimization, specialized file handling, archiving, storage.

This paper presents the general architectures and technologies with a case study on a mega-corporation client of ours in Vietnam, the An Pha Petrol Group Joint Stock Company.

An Pha Petroleum Group JSC. [4] is a large corporation with many member companies. The expansion and merger that happened during the development process made it extremely difficult to manage internal information and an urgent concern. The whole corporation and its companies do not have a common platform to handle processes, procedures and paperwork. Internal tasks are either not digitized or are on different heterogeneous software platforms. That leads to difficulty in management and statistics for the company directors, as well as presents challenges in expanding the business in the future.

The paper describes the specific implementation applied to build a comprehensive system that brings new experiences to all of its 2,000 employees. It consists of building the information infrastructure and system, super applications for both desktop and mobile devices to enhance the work performance and quality. The approaches and results of this paper are applicable to similar large enterprise solutions.

With that aim in mind, this paper will introduce the Microservice architecture: the system design and how it is separated into small components. Add-on features that all businesses need are covered and how to integrate them with the system. Software continuous integration and deployment are introduced in the case of an on-premise deployment at An Pha. Database design is also considered here, as a cache layer ready for data mining & big data.

The remainder of the paper is organized as follows. Section 2 describes system requirements. The system design and implementation are presented in Sect. 3. Then, conclusions and perspectives are made in Sect. 4.

2 System Architecture and Requirements

2.1 Architecture Overview

The architecture of whole system is described in Fig. 1.

The whole system is divided into several subsystems, corresponding to a system on the head of group (Corporation Information System) and systems of the member companies (Member Company business system). This will distribute data and operations in each company subsystem located in different geographical locations. Thus, it reduces the latency of business operations on the software. These subsystems are all structurally similar, consists of main components listed as below:

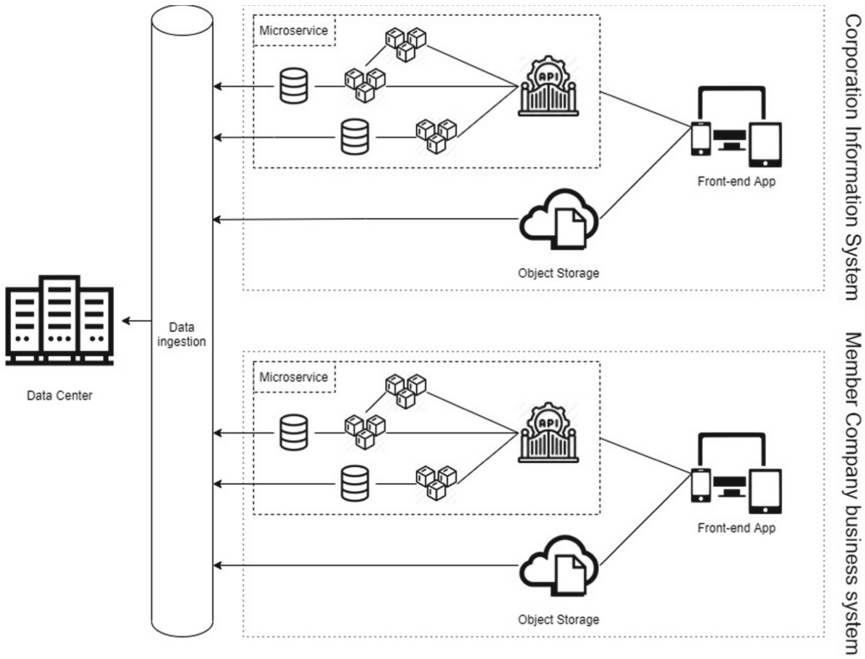

Fig. 1. Architecture overview.

- Front-end app: Applications on different platform (mobile & desktop) that interact directly with end-users. They are used as all-in-one applications with full provided features.
- Object storage: The space for storing all files of employees for paperwork, profiles, reports. The local storage server is used here instead of cloud storage because all files are confidential and need to be saved on premise.
- Microservice: Server side receive front-end request, process & save information in database. Here the microservice architecture is applied, with many services & 1 API Gateway.

The following components are used for mining, analysis data from subsystems:

- Data ingestion: A pipeline is responsible for moving data to the Data Center, where the data can be further enriched and analyzed.
- Data center: A cluster of many servers. Big data analysis and processing are performed here.

2.2 System Requirements

The system focuses on solving digital problems that businesses often encounter, with users being employees of the enterprise and system administrators (also belonging to the enterprise). The functional requirements are described in Table 1.

Table 1. Main categories of functional requirements

Type	Content
Internal communications	Manage internal news, events, activities, job openings
	Communication in groups
	Manage user profiles: roles, avatar, phone, email, etc.
Software modules for units/groups/departments have defined functions	Digitizing production materials: documents, templates, etc.
	Digitizing workflow, management system
	Ex: booking car, room; call center management; retail/wholesale management; etc.
Common management modules	Project management & task assignments
	Request processing
	Send and receive notifications according to the hierarchical model of corporation

3 System Design and Implementation

3.1 Servers

The system consists of many servers configured in Table 2.

Table 2. List of servers used in the system.

No	Configuration			Type	Number	Function	Environment
	CPU (Cores)	RAM (Gb)	HDD (TB)				
1	20	64	8	Physical	1	Database	PRODUCTION
2	20	64	8	Physical	1	Object storage	PRODUCTION
3	32	128	8	Physical	2	Microservice	PRODUCTION
4	20	128	8	Physical	5	Data center	PRODUCTION

3.2 Microservice Applications

The backend application consists of many services. The design, implementation and deployment of a service has the following characteristics:

- A service is an isolated component, performs features of a specific domain. The context, boundary of a domain is divided based on Domain-Driven Design (DDD) [5], closely follow the operations and activities of the divisions in the enterprise.
- The functions provided by a service are independent from that of others. That helps the software implementation to be able to execute in parallel. Thus, the digitization of each department's business in the enterprise can be flexible, depending on the priority of development orientation and their need.
- The implementation of a service is encapsulated as API, so the upgrade, maintenance can be done without affecting client-side.
- A service usually is a stateless application. That is, it doesn't cache any information, such as user's session. That helps the scaling of service into multiple instances, increases the availability of the system and remove the single point of failure.

Services are implemented in Spring boot [6] and are deployed as service instances via containerization [7]. Spring Boot's many purpose-built features make it easy to build and run your microservices in production at scale. And the microservice architecture is completed with the Spring Cloud – easing administration and boosting the fault-tolerance. The containerization technology used here is docker. Source code & dependencies of a service are packaged as a docker image and deployed as docker container.

The microservice architecture is described in Fig. 2:

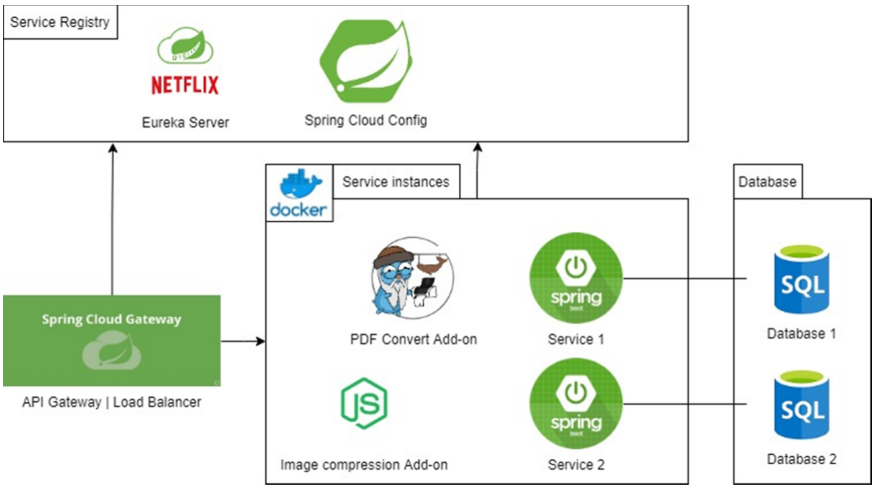

Fig. 2. Microservice architecture.

When creating applications, it's therefore worth optimizing Docker Images and Dockerfiles to help teams share smaller images, improve performance, and debug problems. A lot of verified images available on Docker Hub are already optimized, so it is always a good idea to use ready-made images wherever possible. With some services need to create an image of your own, several ways of optimization it for production is considered, for example: Base image with a smaller footprint, Cleanup commands, Static builds of libraries, Only necessary dependencies, No pip caching, Multi-stage builds, using ".dockerignore" files, dependencies caching. Such intelligent implementation of optimization strategies allowed us to reduce the Docker image size and increase in speed of image building and sharing.

Database architecture is database-per-service [8]. Changing one service's database does not impact any other services. Each service can choose the database type which is best suited for its need (SQL, NoSQL). With SQL database, we are using Microsoft SQL server. SQL table design don't have foreign key constraint, thus increasing flexibility by easily change schema design, data migration without affecting others.

Service registry is an essential component because all other services registered here when deploying. It connects all services & provide communication between them. Hence, API Gateway known the location of service instances on network for routing. API Gateway is a single-entry point of the server side, which receives all requests from client side and proxies/routes to the appropriate services. Thanks to API gateway, a service can be scaled into multiple instances to increase the availability, fault tolerance of system. Here, the technologies we used for Service Registry is Netflix Eureka [9] and API Gateway is Spring Cloud Gateway [10].

In addition to fully satisfying the business operations, other add-on features are also considered. These are normally featuring related to image processing and file processing. We designed each add-on feature as a service in the microservices, rather than SDK or dependencies in front-end apps. It has some advantages: reduction in cost and development time, reduction in the size of front-end applications, using the best libraries for extra requirements. There are various add-ons we have employed in this project for An Pha: Sharp [11] for image compression, Gotenberg [12] for PDF conversion, etc.

3.3 Object Storage

A very common necessity in software service is file storage and transfer. In order to be independent from any third party, to ensure the stability and support for API development and to keep cost down, this service has to be carefully selected.

In the project for An Pha, Minio [13] is used as object storage on system. Minio has the same working concepts as Amazon S3 [14]. Minio is used here instead of S3 because it is open source and can be easily installed on premise at the enterprise's server.

Minio provide an SDK for most commonly used programming languages. Because the files are private, the uploading & downloading is done via pre-signed URLs. A pre-signed URL is a URL that end-users to grant temporary access to a specific object. Using the URL, the user can read and write the object in a specific timeout. By using pre-signed URL, the download/upload files of front-end apps do not go through the backend but connects directly with the Minio, which reduces the load on the backend server, while still ensuring file security (Fig. 3).

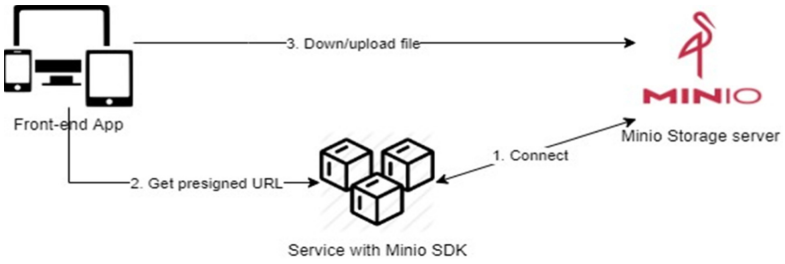

Fig. 3. Working with Minio by presigned URL.

3.4 Data Ingestion and Data Center

The Data Center & Data Ingestion is built as Hadoop Ecosystem. It consists of many elements to solve the big data problems, such as: HDFS, YARN, Spark, Pig, Hive, Kafka, etc. [15].

The data center built for An Pha includes 5 physical servers (with configurations shown in Table 1). All servers use RAID 0 for data storage in disk. In the HDFS concept, there are 1 Name node and 4 Data nodes. The HDFS replication factor here is 3. This data center can handle up to 10 terabytes of data. With a Server Cluster running on Hadoop, data center expansion to accommodate the corporation's needs is guaranteed.

Data ingestion is a process that collects data from various data sources, in an unstructured format and stores it somewhere to analyze that data. This data can be real-time or integrated in batches. Real-time data is ingested as soon it arrives, while the data in batches is ingested in some chunks at a periodical interval of time. To make this ingestion process work smoothly, we can use different tools at different layers which will help to build the data pipeline. Apache sqoop and Kafka are used for data ingestion.

Sqoop designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases of The Apache Software Foundation.

Apache Kafka have a core component as Kafka Connect API, introduced in version 0.9. It provides scalable and resilient integration between Kafka and other systems. The two options to consider are using the JDBC connector for Kafka Connect, or using a log-based Change Data Capture (CDC) tool which integrates with Kafka Connect. In this project, Sqoop and CDC tool integrated Kafka are used.

3.5 Front-End Applications

Taking up the most effort in company management solutions is usually the desktop application. It is written in JavaFX in order to optimize the human resources as the backend is developed in Java [16]. JavaFX is an open-source application platform for desktop application built on Java. The community, dependencies and libraries is excellent for all enterprise feature needs. JavaFX also support CSS to create a highly customized and desired user interface.

For the mobile application, React Native is selected because it is one of the most stable (release on 2015) and common framework [17]. It is an open-source framework for building cross-platform mobile apps using React, another framework created and

developed by Facebook. It uses JavaScript - a very popular programming language and provides a complete set of common view components and dependencies for enterprise app. The choice in React Native also helps in finding developers without maintaining two separate Android and iOS teams. A few screenshots of the desktop and mobile applications are shown below, in Figs. 4, 5 and 6.

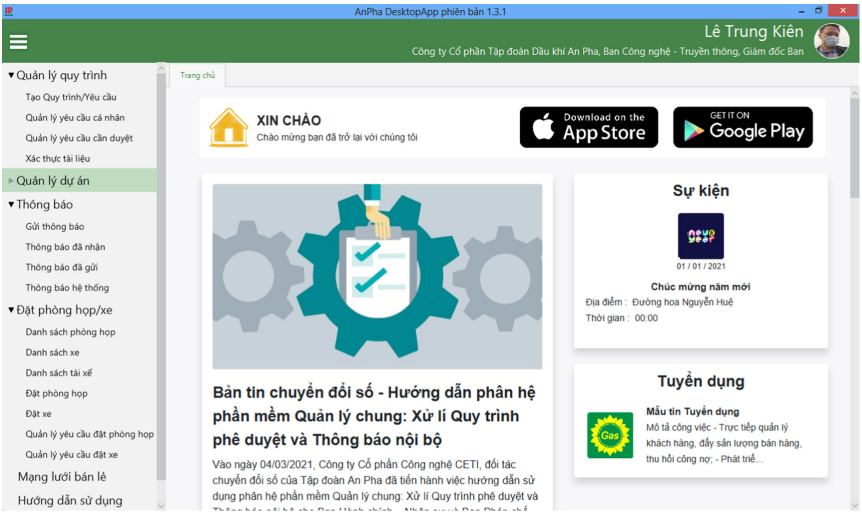

Fig. 4. The main screen of the application on the Desktop.

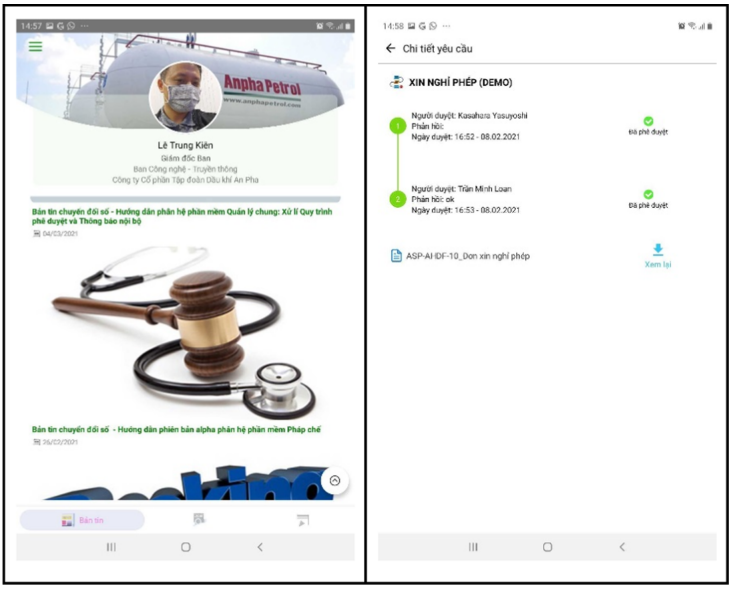

Fig. 5. The main screen of the application on the Mobile (left) and screen of software module "Request Processing" (right).

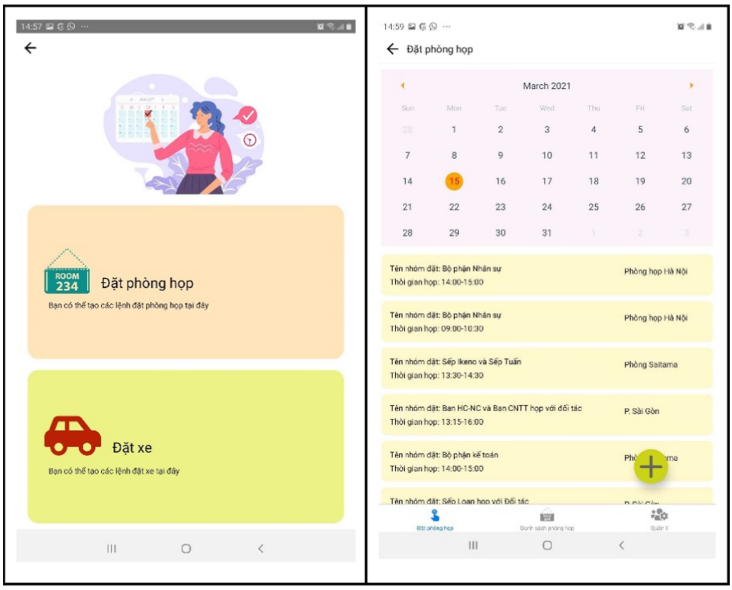

Fig. 6. Two screenshots of software module "Meeting room booking".

4 Conclusion and Perspectives

This research focuses on a practical problem of building a Micro-service architecture with a Big Data oriented approached. At the same time, this research also organizes the best practices to help with the optimization of enterprise integration such as load balancing, services for file transfer, storage, conversion, archiving and optimization of RAM for services.

This paper also presented a case study of a comprehensive digitalization solution and information system for An Pha Petroleum Group JSC, Vietnam.

The microservice architecture is introduced with common solutions as file storage, add-on features. Service design, implementation principles and deployment technologies are also described here. Above all, the database design oriented to the collection and processing of big data is considered.

This solution is also fully applicable for similar large enterprises and is also a good reference for research directions of Software engineering, Information System, etc. [18–20].

References

1. Familiar, B.: Microservice architecture. In: Microservices, IoT, and Azure, pp. 21–31 (2015). https://doi.org/10.1007/978-1-4842-1275-2_3
2. Henry, A., Ridene, Y.: Migrating to microservices. In: Bucchiarone, A., et al. (eds.) Microservices: Science and Engineering, pp. 45–72. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-31646-4_3

3. Bravetti, M., Giallorenzo, S., Mauro, J., Talevi, I., Zavattaro, G.: A formal approach to microservice architecture deployment. In: Bucchiarone, A., et al. (eds.) Microservices: Science and Engineering, pp. 183–208. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-31646-4_8
4. An Pha Petroleum Group JSC. https://anphapetrol.com/. Accessed 01 Mar 2021
5. Khemaja, M.: Domain driven design and provision of micro-services to build emerging learning systems. In: Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM 2016), pp. 1035–1042. Association for Computing Machinery, New York (2016). https://doi.org/10.1145/3012430.3012643
6. https://spring.io/projects/spring-boot. Accessed 01 Mar 2021
7. Yadav, A.K., Garg, M.L., Ritika: Docker containers versus virtual machine-based virtualization. In: Abraham, A., Dutta, P., Mandal, J.K., Bhattacharya, A., Dutta, S. (eds.) Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 3, pp. 141–150. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-1501-5_12
8. Henry, A., Ridene, Y.: Assessing your microservice migration. In: Bucchiarone, A., et al. (eds.) Microservices: Science and Engineering, pp. 73–107. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-31646-4_4
9. https://spring.io/projects/spring-cloud-netflix. Accessed 01 Mar 2021
10. https://spring.io/projects/spring-cloud-gateway. Accessed 01 Mar 2021
11. https://sharp.pixelplumbing.com/. Accessed 01 Mar 2021
12. https://thecodingmachine.github.io/gotenberg/. Accessed 01 Mar 2021
13. https://min.io/. Accessed 01 Mar 2021
14. https://aws.amazon.com/s3/. Accessed 01 Mar 2021
15. Mrozek, D.: Foundations of the Hadoop ecosystem. In: Mrozek, D. (ed.) Scalable Big Data Analytics for Protein Bioinformatics: Efficient Computational Solutions for Protein Structures, pp. 137–150. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98839-9_6
16. Chin, S., Johan, V., James, W.: The Definitive Guide to Modern Java Clients with JavaFX, Cross-Platform Mobile and Cloud Development (2019)
17. Akshat, P., Abhishek, N.: React Native for Mobile Development, Harness the Power of React Native to Create Stunning iOS and Android Applications (2019)
18. Hai, M.M., Hung, P.D.: Centralized access point for information system integration problems in large enterprises. In: Luo, Y. (ed.) CDVE 2020. LNCS, vol. 12341, pp. 239–248. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60816-3_27
19. Tae, C.M., Hung, P.D.: A collaborative web application based on incident management framework for financial system. In: Luo, Y. (ed.) CDVE 2020. LNCS, vol. 12341, pp. 289–301. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60816-3_32
20. Chung, N.N., Hung, P.D.: Logging and monitoring system for streaming data. In: Luo, Y. (ed.) CDVE 2020. LNCS, vol. 12341, pp. 184–191. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60816-3_21