WEST UNIVERSITY OF TIMIŞOARA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
MASTER STUDY PROGRAM: ARTIFICIAL INTELLIGENCE AND DISTRIBUTED COMPUTING

IMPLEMENTATION OF CLOUD ARCHITECTURED APPLICATIONS USING NOSQL DATABASES

Intermediate study (report)

Coordinator:
Lect. Dr. Pop Daniel

Graduate:
Sorin Carcalicea

TIMIŞOARA
2022

Contents

List of Figures

Chapter 1

Abstract

Sorin Carcalicea, Software Developer, West University of Timisoara
Abstract of Master's Thesis, Submitted 24 January 2022
Implementation of Cloud Architecture Applications Using NoSQL Databases

Both academics and industry are constantly innovating in the field of database technology. The needs of the successful pioneers of both web-scale apps and infrastructure for search and advertising drove the development of NoSQL. Because SQL technology did not fulfil the expectations of these applications, each of the early companies created unique databases to meet their requirements. The majority of these were created in-house and then released as open source. Some chose to keep their information private.

The purpose of this paper is that we need a clear picture of when to use an SQL database and when to use a NoSQL database, also we need to know which is more efficient in terms of performance by implementing CRUD operations in a web cloud application. For this we are going to analyze different cloud providers for software development and NoSQL database support.

The goal of our study is to implement a web cloud application in the field of public transportation which falls under the field of smart cities using NoSQL databases and SQL databases and determine which is the best for this study case.

1.1 Introduction

In this paper we are going to implement a web application which will be deployed in cloud and the main purpose of the application will be to show the public transportation vehicles in real time on an interactive map. We noticed that there are a lot of public transportation systems but almost none of them are showing to the user an interactive map with data that updates in real time.

The application will be implemented in such a way that it will be able to run using an NoSQL database as well as an SQL database and performance measurements will be done in order to determine which database is more suitable for this kind of application. An example of such an application could be Uber, which displays in real time where the carrier is and how fast it will arrive.

The scope of this project is to advance is the field of SQL vs NoSQL, we

believe that every developer must know when to use a SQL or a NoSQL database, based on the volume of the data that it will be processed by the application. We also think that we can bring benefit in the world of public transportation and smart cities, where everyone can see, in real time, where a public transportation vehicle is and most importantly if it is stuck in traffic. [4]

As you can see later in this paper, there were some researches done in the field of performance evaluation of SQL vs NoSQL databases and basically no researches were done regarding the real time representation of public transportation so our aim is to cover both fields in one paper.

1.2 State of The Art

The state-of-the-art performance analysis and smart cities public transportation are as follows:

The authors of paper [1] recommend that remote patient monitoring and rehabilitation activities be carried out in satellite medical centres or directly in residents' homes. This paper defines the phrase Tele-Rehabilitation as a Service (TRaaS), which falls under the umbrella of smart cities. This type of service creates healthcare big data from patients' remote rehabilitation equipment, which must be processed in the hospital Cloud. The performance of four NoSQL databases was evaluated, and the document approach was found to be the best fit for the case study.

In paper [3], the authors propose utilising MongoDB as a NoSQL database and MySQL as a SQL database to compare the performance of NoSQL and relational databases. This paper focuses on the advantages of NoSQL databases over relational databases in the context of big data analysis by comparing the performance of various queries and commands in both systems using two separate data sets of varying sizes.

In paper [8], the authors advocate highlighting the municipality's impact in ensuring the city's transportation needs in accordance with the smart city idea. The authors sought to test if choosing the correct carrier to accommodate other smart city aspects may make public transit more appealing. Reducing the use of individual transportation will improve the city's air quality and the population's overall quality of life.

In paper [4], in this reference there are some details about smart cities provided by the European Commission. It is described what a smart city is and what it can do. There is also a section regarding transportation which is poor and reflects where we are in terms of public transportation from the point of view of a smart city.

1.3 Motivation

The motivation of this paper is that we need to clearly know which database type, SQL or NoSQL, to use based on the volume of the information that is processed by the application. Also the field of IoT and smart cities is evolving and it is in trend right now.

Because if am a software developer and passionate about smart cities or anything that is smart thank to the aid of the technology that it is using, I decided that this could be an area of interest to many people and we could all benefit from such a system, where we could clearly see where a public transportation vehicle is located in real time.

The automotive industry is rapidly changing and basically in the future, more and more electric vehicles will be present on the roads. For such a vehicle it will be extremely easy to add a GPS tracker, or maybe future cars will all be build with a GPS tracker incorporated. For existing cars a Raspberry Pi and a GPS module could be added as well or a mobile phone could be used.

Studying what is available on the market today, we can see that there are few to none applications which display exactly where the carrier from the public transport vendor is. Most of the software applications that the public transportation companies have today are very limited in terms of real time data. Most of them are displaying a relative time of arrival but none of them display an interactive map with rel time location of each vehicle.

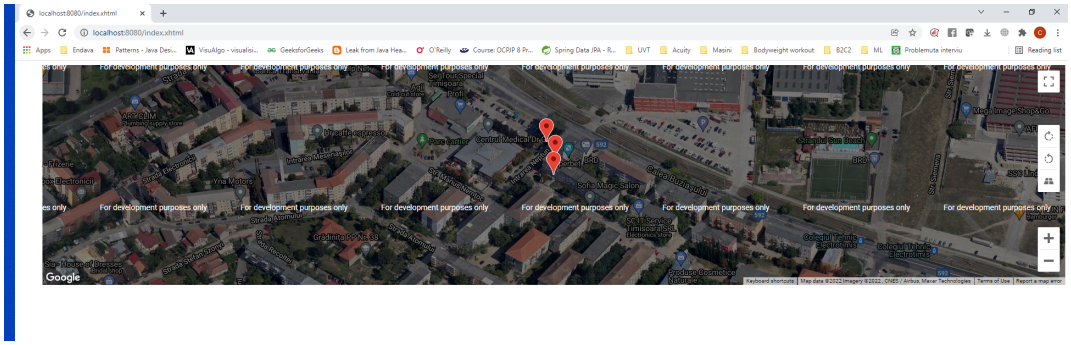
Chapter 2

Implementation of the application

As an application we have chosen to implement a web application which can switch the DB at run time. The application will be focusing on the public transport sector, basically it will be able to display where every car is located on a map. In order to implement this we will use Java for the implementation of the back end, Spring Data which is part of Spring Framework for the implementation of the persistence layer and for the graphical user interface we will use Primefaces, which is a Java library meant to implement the graphical user interface and it is based on JSF technology.

The data base layer is implemented using Sping Data because of the flexibility, we will need to switch the database from SQL to NoSQL based on a setting or something. Spring Framework is composed out of multiple modules and one module of interest is Spring Data JPA which handles the persistence layer for us, we just need to plug in some configurations, such as host and port of data base and credentials for the data base connection such as user name and port. From the module Spring Data JPA we are going to use two or more 'sub modules', one for the SQL persistence and another one for the NoSQL persistence. The first sub module is called 'spring-boot-starter-data-jpa' and it will be used as a dependency in our project for the SQL persistence and the other sub module is called spring-boot-starter-data-redis' and it will be used as a dependency as well in our project for the NoSQL persistence. For the data base switch we will implement a strategy pattern which will load the data base that we want for the current run based on a property that will be defined in a configuration file.

The application will receive as an input a set of data from GPS sensors which then will be stored into a DB and then the data will be retrieved by the user in order to see where the public transport car is. In the screenshot below you can see that we managed to simulate some GPS coordinates and then generated a map with pins, representing positions on the map.



2.1 Designing an application which can be able to switch database at runtime

Designing an application which can be able to switch database at runtime ...

2.2 Performance comparison between SQL and NoSQL

Details about Performance comparison between SQL and NoSQL ...

2.3 What and why?

Details about What and why? ...

Chapter 3

Architecture of the application

3.1 Spring Framework

Spring Framework [6] is a Java framework that supports the development of Java applications by providing complete infrastructure support. Spring takes care of the infrastructure, allowing you to concentrate on your application.

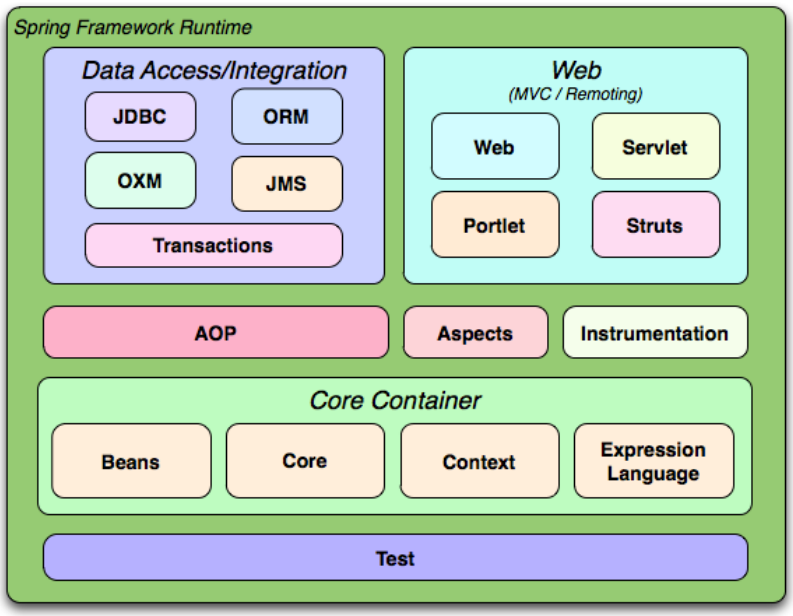
Spring allows you to create apps out of "plain old Java objects" (POJOs) and non-invasively deploy corporate services to POJOs. This feature applies to both complete and partial Java EE as well as the Java SE programming model.

As an application framework, here are some examples of how you can take advantage of the Spring platform:

- Without having to deal with transaction APIs, you may make a Java method execute in a database transaction.
- Without having to deal with remote APIs, you can turn a local Java method into a remote procedure.
- Without having to interact with JMX APIs, you can turn a local Java method into a management activity.
- Without having to deal with JMS APIs, you can turn a local Java method into a message handler.

Modules

The Spring Framework is made up of around 20 modules, each with its own set of functionality. As indicated in the figure, these modules are divided into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, and Test.



Overview of the Spring Framework

3.2 Database scheme

Details about Database scheme ...

3.3 Architecture of the application

Details about Architecture of the application ...
Maybe use micro-services ???

Chapter 4

Technologies used for the development of the application

4.1 Java Stack Technologies

Details about Java Stack Technologies ...

4.2 Nosql Services

Details about Nosql Services ...

4.3 SQL Services

Details about SQL Services ...

Chapter 5

Deployment of the application(CI/CD)

Overview

CI/CD [9] (continuous integration/continuous delivery) is a way for delivering apps to clients on a regular basis by incorporating automation into the app development process. Continuous integration, continuous delivery, and continuous deployment are the three key principles associated with CI/CD. CI/CD is a solution to the challenges that development and operations teams face while integrating new code (AKA "integration hell").

CI/CD, in particular, adds continuous automation and monitoring across the app lifecycle, from integration and testing through delivery and deployment. These approaches are referred to as a "CI/CD pipeline" when they are combined, and they are supported by development and operations teams working together in an agile manner using either a DevOps or a site reliability engineering (SRE) strategy.

What is the distinction between CI and CD (and other CDs)?

There are a number possible meanings for the abbreviation CI/CD. The "CI" in CI/CD stands for continuous integration, which is a developer automation method. New code changes to an app are built, tested, and merged to a common repository on a regular basis with successful CI. It's a solution to the problem of having too many app branches in development at the same time that could conflict.

Continuous delivery and/or continuous deployment are related ideas that are sometimes used interchangeably in the CI/CD acronym. Both are about automating pipeline stages farther down the line, although they're sometimes used independently to show how much automation is taking place.

Continuous delivery often means that a developer's modifications to an application are immediately bug checked and published to a repository (such as GitHub or a container registry), from which the operations team may deploy them to a live production environment. It's a solution to the issue of lack of visibility and communication between development and business teams. To that end, the goal of continuous delivery is to make it as easy as possible to deploy new code.

Continuous deployment (another alternative "CD") refers to a developer's

changes being automatically released from the repository to production, where they can be used by customers. It tackles the issue of operations staff being overburdened with manual processes that hinder app delivery. It extends the benefits of continuous delivery by automating the pipeline's next stage.

CI/CD can refer to merely the related practises of continuous integration and continuous delivery, or it can refer to all three connected practises of continuous integration, delivery, and deployment. To make matters more confusing, the term "continuous delivery" is sometimes used to refer to both continuous delivery and continuous deployment methods.

In the end, it's probably not worth your time to get mired down in these semantics—just remember that CI/CD is a process that involves adding a high degree of ongoing automation and continuous monitoring to app development, and it's commonly depicted as a pipeline.

What the phrases refer to varies depending on how much automation has been incorporated into the CI/CD pipeline on a case-by-case basis. Many businesses begin by using CI and then progress to automated delivery and deployment in the future, such as as part of cloud-native apps.

Our expertise can assist your company in establishing the techniques, tools, and culture required to more effectively modernise existing applications and develop new ones.

Continuous integration

The goal of modern application development is to have numerous developers working on different aspects of the same programme at the same time. However, if a company is set up to combine all branching source code on a single day (known as "merge day"), the work that results can be tedious, manual, and time-consuming. That's because when a developer working alone makes a change to an application, there's a potential it'll clash with other modifications being made at the same time by other developers. If each developer has created their own local integrated development environment (IDE), rather than the team agreeing on a single cloud-based IDE, the problem can be exacerbated.

Continuous integration (CI) enables developers to merge changes to a shared branch, or "trunk," more frequently—even daily. Once a developer's modifications to an application have been merged, the changes are validated by generating the app automatically and running various levels of automated testing, often unit and integration tests, to confirm the changes haven't broken the programme. This entails testing everything from classes and functions to the various modules that make up the programme as a whole. If automated testing uncovers a conflict between new and existing code, continuous integration (CI) makes it easier to fix errors fast and frequently.

Continuous delivery

Continuous delivery automates the deployment of validated code to a repository after the builds and unit and integration testing are automated in CI. As a result, it's critical that CI be already built into your development pipeline if you want to have a successful continuous delivery process. The purpose of continuous delivery is to have a codebase that is always ready to go into production.

Every level of continuous delivery incorporates test automation and code release automation, from merging code changes through delivering production-ready builds. At the end of the procedure, the operations team is able to swiftly and efficiently deploy an app to production.

Continuous deployment

Continuous deployment is the final stage of a mature CI/CD workflow. Continuous deployment is an extension of continuous delivery, which automates the release of a production-ready build to a code repository. It also automates the release of an app to production. Continuous deployment relies significantly on well-designed test automation because there is no manual gate at the point of the pipeline before production.

Continuous deployment, in practise, means that a developer's change to a cloud application could go live minutes after being written (assuming it passes automated testing). This makes it much easy to acquire and implement user feedback on a regular basis. All of these connected CI/CD methods, when combined, make application deployment less dangerous, as it's easier to deliver changes to apps in little chunks rather than all at once. However, because automated tests would need to be built to handle a range of testing and release phases in the CI/CD pipeline, there will be a significant upfront investment.

What are some common CI/CD tools?

5.1 Maven

Maven [2], a Yiddish phrase that means "collector of knowledge," originated as an attempt to streamline the Jakarta Turbine project's construction processes. There were multiple projects, each with its own Ant build files, that were slightly different from one another. CVS was used to check in JARs. We needed a common way to build projects, a clear definition of what a project was, a simple way to publish project information, and a means to exchange JARs across several projects.

As a result, you now have a tool that can be used to create and manage any Java-based project. We hope that we have built something that will aid in the understanding of any Java-based project and make the day-to-day job of Java developers easier.

Maven's Objectives

Maven's main purpose is to help developers understand the entire status of a development project in the quickest amount of time possible. Maven tackles a

number of issues in order to achieve this goal:

- Streamlining the delivery process
- Providing a standardised build system
- Delivering high-quality project data
- Better development approaches are encouraged.

Making the build process simple

While Maven does not remove the need to understand the underlying mechanics, it does shelter developers from many of the complexities.

Providing a standardized build system

Maven uses its project object model (POM) and a set of plugins to create a project. Once you've learned how to build one Maven project, you'll be able to build any Maven project. When navigating many projects, this saves time.

Providing high-quality project data

Maven provides useful project information that is derived in part from your POM and in part from the sources of your project. Maven, for example, can provide:

- Directly from source control, a change log is generated.
- Sources with cross-references.
- The project's mailing lists are administered by the project.
- Dependencies utilised by the project.
- Unit test reports including coverage.

Third-party code analysis packages also provide Maven plugins that supplement Maven's basic information with their own findings.

Providing guidelines for best practices development

Maven's goal is to compile current best practises development concepts and make it simple to steer a project in that direction.

The formulation, execution, and reporting of unit tests, for example, are all part of the standard Maven build cycle. As a guide, the following current unit testing best practises were used:

- Maintaining a distinct yet parallel source tree for test code.
- Locating and running tests using test case naming conventions.

- Instead of configuring the build for test preparation, test cases should set up their environment.
- Maven can also help with project process issues like release and issue management.

Maven also provides some instructions for laying up the directory structure of your project. Once you've mastered the layout, you'll be able to traverse other Maven-based projects with ease.

While it takes an opinionated approach to project layout, historical considerations may prevent some projects from fitting into this framework. While Maven is designed to be adaptable to the needs of various projects, it cannot accommodate every circumstance without jeopardizing its goals.

If your project's build structure is unique and cannot be restructured, you may have to forego some functionality or abandon Maven entirely.

What is Maven Not?

Some of the following things about Maven might be familiar for you:

- Maven is a site-building and documentation platform.
- Maven is an Ant extension that allows you to download dependencies.
- Maven is a collection of reusable Ant scriptlets.

While Maven does these things, as you can see in the "What is Maven?" section above, these aren't the only characteristics it has, and its goals are rather different.

5.2 Jenkins

Jenkins® [5] is an open source automation server. By automating the software development process, enterprises can save time and money. Jenkins is a tool that manages and controls software delivery processes across the whole lifecycle, including build, document, test, package, stage, deployment, static code analysis, and more.

Jenkins may be configured to monitor any code changes in GitHub, Bitbucket, or GitLab and do an automatic build using Maven and Gradle. You can use container technology like Docker and Kubernetes to run tests and then take actions in production like rolling back or forward.

Jenkins History

While working for Sun Microsystems, Kohsuke Kawaguchi founded the Jenkins project (formerly called Hudson) in 2004. Kohsuke was a Sun engineer who was fed up with his team's fury every time his code broke the build. Jenkins was designed as a technique to perform continuous integration – that is, to test his code before committing it to the repository to ensure everything was working properly.

Jenkins was quickly adopted by his colleagues as they witnessed what he was accomplishing. Kohsuke made it open source, resulting in the Jenkins project, which quickly grew in popularity around the world.

Jenkins Today

Jenkins, which was originally created by Kohsuke for continuous integration (CI), now orchestrates the complete software delivery process, which is referred to as continuous delivery. For certain businesses, automation goes even farther, allowing for continuous deployment. Software delivery is considerably accelerated when continuous delivery (CD) is combined with a DevOps culture.

Jenkins is the most extensively used continuous delivery solution, owing to its versatility and active community. Jenkins can be integrated with nearly any tool, including all of the best-of-breed solutions used throughout the continuous delivery process, thanks to the Jenkins community's more than 1,700 plugins. Jenkins continues to gain traction as the leading solution for software process automation, continuous integration, and continuous delivery, with more than 165,000 active installations and 1.65 million users worldwide as of February 2018.

CloudBees and the Jenkins Community

CloudBees is an active member of the Jenkins community and contributes significantly to the project's success. CloudBees employs a number of important contributors to the Jenkins project. CloudBees is the lead sponsor of DevOps World — Jenkins World, the annual user conference for the Jenkins community, in support of the community. Jenkins World brings people together, provides a sense of belonging, helps users to learn from one another, and promotes community growth. CloudBees has also aided the growth of Jenkins Area Meetups (JAMs) around the world, donating the JAMS to the CDF so that they can continue to thrive.

CloudBees support and product engineers contribute code to the Jenkins project on a regular basis, participate in Jenkins chats and project meetings, and contribute to the Jenkins project email lists. Jenkins X, Jenkins Pipeline, and Jenkins 2 are just a few of the significant projects CloudBees has worked on. Furthermore, all updates made by CloudBees in the open source code are submitted back to the project, allowing us all to benefit from an ever-better Jenkins experience.

Continuous Information is a monthly newsletter published by CloudBees for the Jenkins community. It's chock-full of community tips tricks, articles, presentations, global events, and a slew of other resources. Subscribe to receive it in your inbox once a month.

Jenkins and the Continuous Delivery Foundation

The Linux Foundation established the Continuous Delivery Foundation (CDF) in 2019. In partnership with the Jenkins and Jenkins X communities, Google, the Linux Foundation, and others, CloudBees led the launch initiative.

The CDF is dedicated to fostering, growing, and supporting open source projects, best practices, and industry standards relating to continuous delivery.

Jenkins, Jenkins X, Spinnaker, and Tekton are just a few of the open source projects hosted by the CDF. More projects are likely to join, with the goal of forming a continuous delivery (CD) ecosystem to develop specifications and initiatives around portability and interoperability. CloudBees is a founding member of the initiative, and it continues to contribute technology and resources to it.

5.3 Git

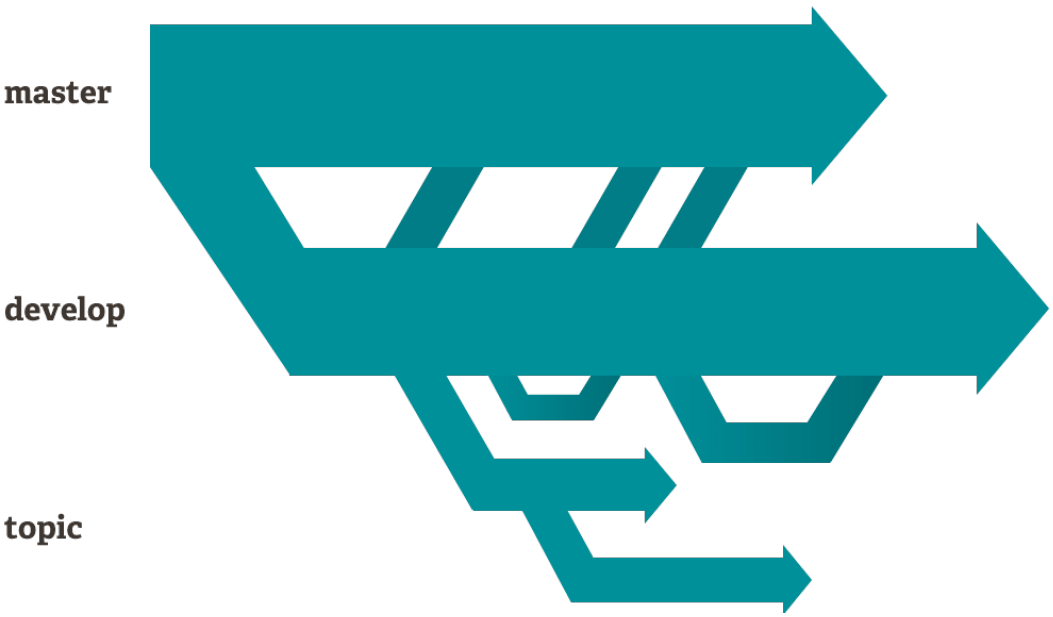
Branching and Merging

The branching model of Git [7] is what really sets it different from practically every other SCM out there.

Git supports and encourages the creation of several local branches that are completely independent of one another. It takes seconds to create, merge, and delete those development lines.

As a result, you can do things like:

- Context Switching with No Friction. To test an idea, create a branch, commit a few times, switch back to where you branched from, apply a patch, switch back to where you're exploring, and merge it in.
- Codelines based on roles. Create a branch that only contains what goes to production, a branch that you merge work into for testing, and a few smaller branches for day-to-day work.
- Workflow that is based on features. Create different branches for each new feature you're working on so you can transition between them effortlessly, then delete each branch after the feature is integrated into your main line.
- Experimentation with Disposables. Create a branch to experiment in, discover it's not going to work, and simply remove it, abandoning the task (even if you've pushed other branches in the meantime).



You don't have to push all of your branches to a remote repository, for example. You have the option of sharing just one, a couple, or all of your branches. This allows people to experiment with new ideas without having to worry about how and when they will integrate them or share them with others.

Some of this can be accomplished using other methods, but the job is far more complicated and error-prone. Git makes this process incredibly simple, and once learned, it transforms the way most engineers operate.

Chapter 6

Conclusion

To add conclusions here

Bibliography

[1] Celesti Antonio, Lay-Ekuakille Aimé, Wan Jiafu, Fazio Maria, Celesti Fabrizio, Romano Agata, Bramanti Placido, and Villariae Massimo. Information management in iot cloud-based tele-rehabilitation as a service for smart cities: Comparison of nosql approaches. 2019.

[2] Varanasi Balaji. Introducing maven. 2019.

[3] Josea Benymol and Abraham Sajimon. Performance analysis of nosql and relational databases with mongodb and mysql. 2020.

[4] European Commission. Smart cities. 2021.

[5] Walls Craig. Spring in action, sixth edition. 2017.

[6] Walls Craig. Spring in action, sixth edition. 2020.

[7] Git. About git. 2021.

[8] Bubelíny Oliver and Kubina Milan. Impact of the concept smart city on public transport. 2021.

[9] RedHat. Redhat, what is ci/cd? 2021.