UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES

Comparison of Back-end Frameworks for the Development of Scalable and Efficient Web Applications

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniero en Tecnologías de la Información

Author:

Steeven Geovanny Sanchez Pozo

Tutor:

Francisco Javier Hidrobo Torres Ph.D.

Urcuquí - Diciembre de 2024

Autoría

Yo, **Steeven Geovanny Sanchez Pozo**, con cédula de identidad 0603955824, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor/a del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Diciembre de 2024.

Firmado electrónicamente por:
STEEVEN GEOVANNY
SANCHEZ POZO

Steeven geovanny Sanchez pozo

CI: 0603955824

Autorización de publicación

Dedicación

Dedico este trabajo a mis padres Edith y Giovanni,
quienes siempre confiaron en mi y me guiaron con sus enseñansas y buena educación.

A mis hermanos, Alexis and Genesis,
por el amor y constante apoyo. Siempre estaré agradecido.

Y a mi familia por su incondicional amor.
Thanks for always believing in me.

Acknowledgment

Gracias a la Universidad Experimental Yachay por ser una universidad de excelencia. El estudio y la dedicación que tienen con los estudiantes es muy satisfactorio ypuedo decir que la experiencia vivida en esta universidad es única y no la voy aolvidar nunca.

Agradecimiento especial a mi tutor Francisco Hidrobo, por estar pendentiente y ayudarme en todo lo que necesite a lo largo de esta etapa, dandome motivación e inspiración para culminar mi proceso de titulación.

Gracias a mis amigos Amanda, Oswald, Karen y Michelle quienes estubieron junto a mi en las primeras etapas de esta vida estudiantil. A Henry, Pedro, Jhonatan, Andres, Marcos, Mauro y Karen, quienes fueron mis compañeros de fiesta y de estudio en carrera. A mis amigos varios como los Friendtasmas, Laila, Juli, Klever, Harvey y Eliana que me dieron emociones musicales y espaciales. A mis amigos de juego que hicieron mi vida Universitaría mucho más amena y especial Jaime, Micro, Monni, Jacko, Tevi. y finalmente a María que me ayudo en momento dificiles, me soportó muchas veces y me apoyó en esta parte importate de mi vida universitaria.

Finalmente Gracias a mis padres y mis hermanos los cuales amo y respeto mucho. Espero poder contar con ustedes toda mi vida y que todos los sueños que tenemos se nos cumplan.

Resumen

Los frameworks de trabajo para el back-end son herramientas esenciales para el desarrollo de aplicaciones web modernas y eficientes. Sin embargo, existe una amplia variedad de frameworks disponibles, lo que puede dificultar la elección del más adecuado para un proyecto específico. En este trabajo, proponemos un marco metodológico que permite comparar herramientas de back-end (frameworks para back-end). Esta metodología se utiliza para realizar una comparación exhaustiva de varios Frameworks de back-end. Básicamente, este enfoque estructurado utiliza dimensiones y métricas definidas, destacando atributos cruciales como facilidad de desarrollo, seguridad, rendimiento, escalabilidad, mantenibilidad, costo, ecosistema y compatibilidad, portabilidad y popularidad, entre otros, para desarrollar aplicaciones web escalables y eficientes. A través de un ejemplo práctico de esta metodología, la tesis tiene como objetivo simplificar y perfeccionar el proceso de selección de frameworks de back-end. Al hacerlo, ofrece a los desarrolladores una guía esencial al enfrentar decisiones críticas y desafíos inherentes al adoptar nuevos frameworks de back-end.

Palabras Clave:

Back-end frameworks, Marco metodológico, Dimensión, Métricas, Guía, Aplicaciones web, Selección de Framework.

Abstract

Back-end frameworks are essential tools for the development of modern and efficient web applications. However, there is a wide variety of frameworks available, which can make it difficult to choose the most suitable one for a specific project. In this work, we propose a methodological framework that allows to compare back-end tools (back-end framework). This methodology is used to comprehensive comparison of several back-end frameworks,at its core, this structured approach harnesses defined dimensions and metrics, underscoring pivotal attributes like easy of development, security, performance, scalability, maintainability, cost, ecosystem and compatibility, portability, and popularity amongst others to develop scalable and efficient web applications. Through a practical example of this methodology, the thesis aims to simplify and refine the process of selecting back-end frameworks. In doing so, it provides developers with an essential guide when facing critical decisions and challenges inherent in adopting new back-end frameworks.

Keywords:

Back-end frameworks, Methodology Framework, Dimension, Metrics, Guide, Web applications, Framework selection.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Software development has experienced significant growth in recent decades, going through various stages of evolution. In its early stages, the primary focus was on desktop application development, which ran locally on individual computers. These applications were often monolithic and relied on client-server architectures [1].

With the rise of the Internet and the World Wide Web, web applications began to gain popularity in the 1990s. Unlike desktop applications, web applications run in a browser and communicate with a remote server, allowing users to access them from any device with an Internet connection [1]. The transition to web applications brought with it a number of advantages, such as ease of updating and maintenance, scalability, and the ability to support multiple platforms [2].

As web applications continued to evolve, new technologies and architectural patterns emerged to address the growing complexity and demands of modern software development. One such pattern is the microservices architecture, which involves breaking down an application into small, independent services that communicate with each other through APIs [3]. This approach offers benefits such as improved scalability, fault tolerance, and easier maintenance compared to monolithic architectures [4].

The growing importance of web applications led to the development of back-end frameworks, which are an integral part of modern application development. back-end frameworks provide a structure and a set of tools that facilitate the creation and maintenance of web and mobile applications [5]. These frameworks help developers handle crucial aspects of web application development, such as database management, user authentication, and API implementation [6].

Over the years, numerous back-end frameworks have been developed to address different needs and preferences in software development. Some of the most popular back-end frameworks include Django (Python), Ruby on Rails (Ruby), Express (Node.js), Laravel (PHP), and ASP.NET (C#) [6]. Each of these frameworks has its own advantages and disadvantages in terms of performance, scalability, ease of use, and other factors that may influence a developer's choice.

The selection of an appropriate back-end framework is crucial for a project's success, as it can affect the productivity of the development team, code quality, and the ability to scale and maintain the application over time [5]. However, due to the large number of available frameworks and differences in their features and capabilities, it can be challenging for developers and organizations to make informed decisions about which framework to use.

Several studies have investigated different approaches to comparing and evaluating back-end frameworks. For example, Kaur and Rani (2016) [5] proposed a multi-criteria evaluation framework based on the Analytic Hierarchy Process (AHP) method for comparing web frameworks. Chandra and Borah (2017) [7] conducted a qualitative comparative analysis of five popular back-end frameworks based on criteria such as ease of use, scalability, and performance. Despite these efforts, there is still a need for comprehensive and objective methods for comparing back-end frameworks to support developers and organizations in their decision-making process.

Furthermore, the rise of cloud computing has introduced new considerations for selecting back-end frameworks. Cloud platforms such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) offer various services and tools that can be leveraged by back-end frameworks to enhance application performance, security, and scalability [8]. Therefore, it is necessary to have an idea of which is the best back-end framework to use for the development, and what are the specifications required by the user to know if it fits the corresponding needs, including deployment requirements (possibly in the cloud).

In response to this need, in this thesis a methodology framework is proposed that allows developers and organizations to effectively evaluate and compare back-end frameworks. By providing a systematic and objective approach, this methodology will help decision makers identify the most suitable back-end framework for their specific project requirements, ultimately leading to more successful and efficient software development processes.

1.2 Problem Statement

The development of web and mobile applications is a complex process that involves selecting appropriate technologies and tools to meet the specific needs and requirements of the project [1]. Among the many decisions developers must make is the choice of the back-end framework, which is crucial to the success of the project [5]. The incorrect selection of a framework can lead to performance, scalability, maintenance, and cloud service compatibility issues, which in turn can increase costs and development time [6].

There are numerous back-end frameworks available in the market, each with its own advantages and disadvantages in terms of performance, scalability, ease of use, and other factors that may influence the developer's choice [5]. Additionally, the growing importance of cloud computing and the adoption of microservices architectures have added additional layers of complexity to the selection of the appropriate back-end framework [3].

Despite the abundance of information and comparisons available online, developers often face difficulties in making an informed decision due to the lack of a systematic and standardized comparison methodology that allows evaluating and comparing different back-end frameworks objectively [2].

In this way, the primary challenge tackled in this study is the absence of a methodology that empowers developers to assess and differentiate various back-end frameworks effectively. This includes considering pivotal factors such as performance, scalability, user-friendliness, compatibility , and the integration of micro-services architectures [3]. Addressing this issue would equip developers with a tool to make data-driven choices and pick the best-suited back-end framework for their endeavors, subsequently enhancing software quality while diminishing expenses and development duration [1].

1.3 Objectives

In this thesis, we seek to address the challenges faced by developers when selecting a suitable back-end framework for their web and mobile application projects. The overall objective is to design a methodology framework that allows for the a quantitative measurement of various dimensions and metrics relevant to evaluating the accessibility and suitability of a back-end framework in comparison to others. By establishing both a general objective and specific objectives, we aim to provide developers with a valuable tool for making informed decisions and selecting the most suitable back-end framework for their projects.

1.3.1 General Objective

The general objective of this thesis is to design a comprehensive and well-founded methodology framework that allows developers to quantitatively measure various dimensions and metrics relevant to evaluating the accessibility and suitability of a back-end framework in comparison to others.

This methodology should be applicable to any back-end framework and provide a systematic and standardized approach to assessing critical aspects, such as performance, scalability, ease of development, security, popularity, portability, maintainability, cost, ecosystem and compatibility. Furthermore, the proposed methodology aims to be flexible enough to adapt to the changing needs and trends in web and mobile application development. It's important to mention that it should be feasible to develop a web-based tool that simplifies the implementation of this methodology, thereby offering developers a valuable tool for making informed decisions and selecting the most suitable back-end framework for their specific projects.

1.3.2 Specific Objectives

To achieve the general objective, the following specific objectives are proposed:

1. Define a set of evaluation criteria to objectively compare the selected frameworks.

2. Identify the relevant dimensions and metrics for evaluating the accessibility of back-end frameworks, considering aspects such as performance, scalability, ease of use,

security, popularity, portability, maintainability, cost, ecosystem, and compatibility.

3. Design a methodology based on the identified dimensions and metrics that allows developers to effectively and objectively evaluate and compare different back-end frameworks.

4. Identify the most popular and widely used back-end frameworks in the web and mobile application development industry.

5. Apply the proposed methodology to the back-end frameworks identified as the most popular, to demonstrate its applicability and usefulness in selecting the most suitable framework for a specific project.

By achieving these objectives, we expect to develop a robust and effective methodology framework that allows developers to compare and evaluate back-end frameworks objectively and based on relevant criteria. In the process of addressing the challenge of selecting the best back-end frameworks, the application of this methodology is envisioned to improve software quality, reduce development costs and time, and facilitate the adoption of suitable technologies and architectures for each project.

1.4 Phases of Problem Solving

In this section, we will discuss the different phases of problem-solving that will be followed throughout this thesis to address the challenges associated with selecting a suitable back-end framework for web and mobile application projects.

- The first phase involves identifying the problem, which in this case is the difficulty developers face when selecting a suitable back-end framework for their projects. This phase includes understanding the various factors that influence the choice of a back-end framework, such as performance, scalability, ease of use, among others.

- In the second phase, we will collect and analyze data related to popular back-end frameworks and their features. This may include conducting surveys, interviews, or gathering information from online sources. The collected data will help us identify the most popular and widely used back-end frameworks in the web and mobile application development industry.

- Based on the collected data and identified dimensions and metrics, we will design a methodology framework that allows developers to effectively and objectively evaluate and compare different back-end frameworks. This methodology will be designed to be applicable to any back-end framework and provide a systematic and standardized approach to assessing critical aspects.
- Once the methodology framework is developed, we will apply it to the most popular back-end frameworks identified in the data collection and analysis phase. This will demonstrate the applicability and usefulness of the methodology in selecting the most suitable framework for a specific project.

1.5 Scope and Limitations

In this section, we will discuss the scope of this thesis and the limitations associated with the proposed methodology framework for selecting a suitable back-end framework for web and mobile application projects.

Scope

The scope of this thesis includes the following aspects:

- Identifying relevant dimensions and metrics for evaluating back-end frameworks, focusing on performance, scalability, ease of use, cloud service compatibility, and adoption of microservices architectures.
- Designing a comprehensive methodology framework that allows developers to effectively and objectively evaluate and compare different back-end frameworks based on the identified dimensions and metrics.
- Applying the proposed methodology framework to the most popular and widely used back-end frameworks in the web and mobile application development industry.
- Proposing recommendations and best practices for selecting back-end frameworks based on the results obtained by applying the developed methodology framework.

Limitations

Despite the thorough approach taken in this thesis, there are some limitations to consider:

1. Methodoly framework may not cover all possible dimensions and metrics relevant to evaluating back-end frameworks. Some factors, such as specific project requirements or developer preferences, may not be fully addressed.

2. The identification of the most popular back-end frameworks is based on data collected at a specific point in time, which may change as new frameworks emerge or existing ones evolve.

3. The proposed recommendations and best practices may not be applicable to all projects or situations, as they are based on the results obtained from applying the methodology to a specific set of popular back-end frameworks.

This thesis focuses on the development of a methodology based on a quantitative model to aid in decision making related to the selection of back-end frameworks in web and mobile application projects.

Chapter 2

Theoretical Framework

The rapidly evolving world of software development is underpinned by a myriad of theories, principles, and established best practices. To understand the process of selecting a back-end framework and the myriad factors that influence this decision, it is imperative to ground our inquiry in a robust theoretical framework. This section delves into the foundational theories and concepts that inform our understanding of software development, back-end technologies, and the considerations that drive the adoption of one framework over another. By anchoring our methodology in these theories, we seek to ensure that our approach is both informed by past knowledge and tailored to address contemporary challenges. The following discussion will explore key theoretical constructs, their relevance to our study, and how they collectively shape the lens through which we approach the primary objective of this thesis.

2.1 Fundamentals of Web Technology

Web technology has undergone a revolutionary evolution since its inception, enabling effective and efficient access and distribution of information on the network. This section explores the fundamentals of web technology, focusing on key elements that have driven its development.

2.1.1 Historic Development

In its early days, the World Wide Web (WWW) was proposed by Tim Berners-Lee in 1989, becoming a hypertext system that facilitated the interconnection of documents through hyperlinks. The web is based on the client-server model, where web browsers act as clients and request information from web servers that store and distribute content using protocols like HTTP (Hypertext Transfer Protocol) [9].

The foundation of the web is built upon markup languages, with HTML (HyperText Markup Language) being the primary language used to structure content and present it visually in web browsers. CSS (Cascading Style Sheets) is responsible for defining presentation and layout, while JavaScript enables the addition of interactivity and dynamism to the content [10].

Over time, the web evolved to support real-time interaction through technologies like AJAX (Asynchronous JavaScript and XML), significantly enhancing the user experience [11]. Additionally, the adoption of open standards and the establishment of the World Wide Web Consortium (W3C) ensured interoperability and accessibility for all users, regardless of their device or platform [10].

The emergence of server-side technologies such as PHP, Ruby on Rails, and Node.js allowed the creation of more complex and scalable web applications. These technologies provided the ability to process data on the server before sending it to the browser, leading to the development of interactive and real-time web applications [12].

The web also underwent a significant transformation with the introduction of Application Programming Interfaces (APIs), enabling communication and data exchange between different web applications .This led to the development of Service-Oriented Architecture (SOA) and microservices architectures, improving application flexibility and modularity [10].

The exponential growth in the use of mobile devices drove the development of responsive and adaptive web design, allowing websites to adjust automatically to different screen sizes and devices [13]. Furthermore, the introduction of technologies like Progressive Web Apps (PWA) facilitated the creation of web applications that behave like native mobile apps, providing a more app-like user experience [12].

If you're embarking on the journey of creating a modern app or website, it's essential to grasp the two fundamental components that make it all work seamlessly. The first of these crucial elements is referred to as the ”front-end.” This pivotal component takes charge of everything related to the user experience, shaping what the user perceives, determining when to retrieve data from the server, and facilitating various interactions and modifications. In essence, it's the bridge between the user and the application's functionalities.[14]

The second indispensable piece of the puzzle is known as the "back-end." This encompasses the servers and data that the front-end relies on, manipulates, and ultimately showcases to the user. In other words, it serves as the engine powering the entire system, handling the logic and processing that goes on behind the scenes.[14]

2.1.2 Basic Operation

Web technology operates as an interconnected system where multiple components interact to deliver a seamless user experience. To comprehensively understand its basic operation, one can envision a typical flow initiated when a user accesses a web application through a browser.

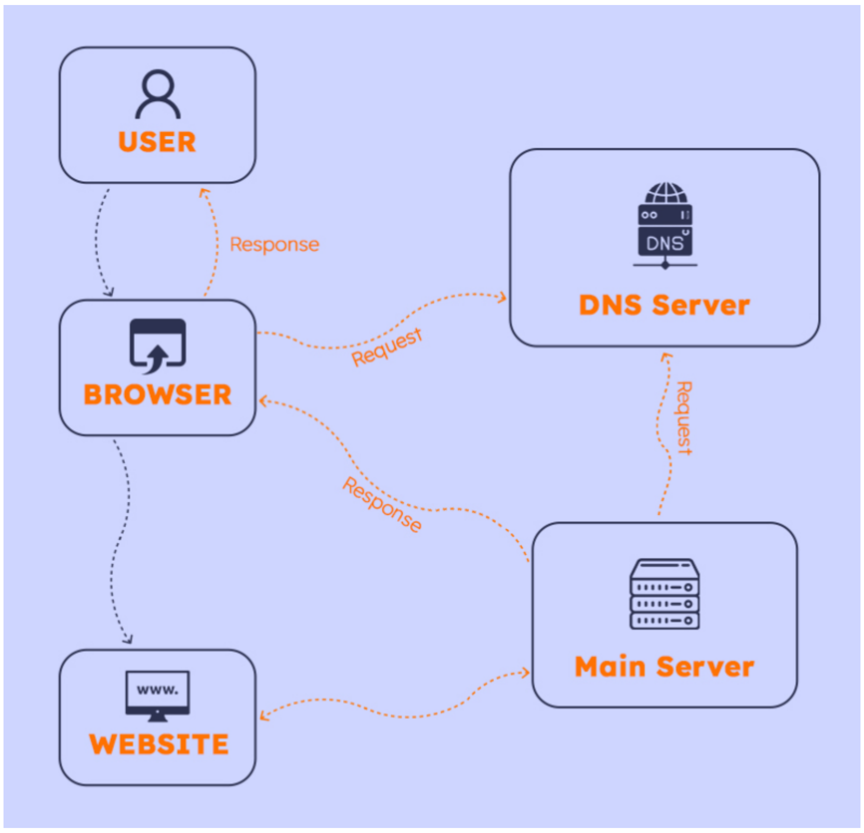

Figure 2.1: Basic Operation of Web Application

User: An individual who interacts with computer systems, applications, or websites. In the context of web applications, a user is someone who navigates a website using a browser.

Browser: Software application used to access, retrieve, and present content from the web. Examples include Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. The browser interprets web content, primarily HTML, CSS, and JavaScript, to display web pages to users.

DNS Server: Stands for ”Domain Name System Server”. It is a server that translates human-friendly domain names (like ”www.example.com”) into IP addresses, which are used for routing data across the internet.

Website: A collection of related web pages, including multimedia content, that are identified with a common domain name. A website is accessible via a public internet protocol (IP) network, such as the internet, or a private local area network (LAN).

Main Server: Often referred to as a ”web server”. It's a server that hosts websites and provides content to users via browsers. The main server processes incoming requests from users and sends back the appropriate response, typically in the form of web pages, images, or other media.

Request: In web communication, a request is made by a user's browser to a server to retrieve or send information. This can be a request to view a web page, download a file, or submit data to be processed by the server.

Response: The data sent by the server to the user's browser following a request. This can be in the form of a web page, an image, a video, or other types of data. The response tells the browser what content to display or how to process the results of the user's request.

The interplay of these components ensures the delivery of content, data retrieval and manipulation, and real-time interaction, forming the backbone of the modern web experience [15].

2.1.3 Front-end and Back-end

Front-end: The front-end, often referred to as the “client-side”, is the part of a web application that users interact with directly. It’s everything that the user experiences while navigating through a website – from graphics and design to how data is presented. Front-end development involves using languages such as HTML, CSS, and JavaScript [16].
Back-end: The back-end, sometimes known as the ”server-side”, is the part of a web application that operates behind the scenes. It handles the database operations, application logic, server configuration, authentication, and more. Back-end development typically involves server-side languages such as PHP, Ruby, Python, Java, and .NET [16].

Differences

In software development, understanding the key differences between the front-end and back-end is essential, as these components form the core of any application or website. Below is a comparative table that highlights the distinctive features and responsibilities of each, illustrating how these two areas interact and complement each other to create a cohesive and functional user experience.

Front-end	**Back-end**
Direct interaction with the user	Operates behind the scenes
Uses languages like HTML, CSS, and JavaScript	Uses server-side languages like PHP, Ruby, Python, Java
Focused on user interface and experience	Focused on server, application, and database operations
Manages how information is presented to the user	Manages how data is stored and ensures data integrity
Examples: Web design, animations, buttons	Examples: User authentication, server configuration, database interactions

Table 2.1: Comparison between Front-end and Back-end

2.2 State of Art

To ensure a robust and reliable back-end for develop an application, making the right choice when it comes to the back-end web framework is of paramount importance. Selecting the best back-end web framework available to developers can significantly impact the performance, scalability, and overall success of your project, making it a decision that should be taken seriously and thoughtfully. By carefully evaluating the available options and aligning them with your project's specific requirements, you pave the way for a powerful and impressive digital experience that users will love and appreciate. So, don't underestimate the significance of this decision, as it lays the foundation for your application's success in the competitive and dynamic world of modern technology.

In the literature, there are several studies that compare back-end frameworks for web application development. For example, study conducted by Savvycom Team in 2021 [17] compared the most famous Node.js frameworks: Koa, Express, and Hapi. It presents a detailed comparison, evaluating the performance of each framework in terms of speed, scalability, and ease of use. To do this, the article analyzes the code of each framework in its basic configuration, middleware, and installation. Finally, a simple but effective code is used to measure its speed, which is a common example of a "Hello World" program. Additionally, the article highlights the importance of not only considering the performance of a framework but also its ease of use and documentation.

Another study conducted in 2021 by ValueCoders [18] compares the CodeIgniter vs CakePHP vs Yii vs Laravel frameworks based on different parameters such as popularity, where they analyze the websites created and used by these frameworks. They compare the architecture of each one, in this case, using the Model-View-Controller (MVC) pattern, database integration, and how many databases each framework supports such as MySQL or PostgreSQL, ease of use, and development cost. Additionally, they compare features such as the database model, programming language, license, among others.

The following article written by John Wheeler [19]discusses a detailed comparison between three Python web frameworks: Django, Flask, and Pyramid. It describes the main features of each framework, their strengths and weaknesses, and offers advice on when to use one or the other depending on the project's needs. The article also provides code examples and explains how each framework handles the database, authentication, security, scalability, and other important aspects. Additionally, it offers an overview of how these frameworks work and performs small demos where we can observe the performance of each, providing a comprehensive and useful insight for anyone looking to choose the right Python framework for their web project.

Another significant contribution was the article written by Marin Kaluža, Marijana Kalanj, and Bernard Vukelić [20] explains that the use of frameworks can enhance the quality and standardization of web applications, leading to improved efficiency and reduced costs. The back-end frameworks they focus on are Laravel, Rails, Django, and Spring. They evaluate factors such as ease of use, scalability, security, performance, community and support, documentation, flexibility and customization, and integration with other technologies. In order to assess these factors, they utilize criteria such as code generator, project health, developer's perception, developer's availability, business trends, templates, I18n and l10n, testing, validation, mobile phone/iPhone support, among others.

In 2023, PrimerPy conducted an analysis of four popular Python web frameworks, including Django, Flask, FastAPI, and Tornado [21]. The article delved into the unique attributes of each framework, outlining their strengths and weaknesses. It also examined factors such as performance, ease of use, and scalability. The insights provided by this article can guide developers in choosing the most suitable framework based on their project requirements and objectives, emphasizing the need to evaluate development speed, community support, and compatibility with emerging technologies.

A comprehensive comparison of various web development frameworks was published by Monocubed in 2023 [22]. This comparison covered Django, Flask, Ruby on Rails, Express, and Laravel. The article provided an in-depth analysis of each framework's key features, advantages, and drawbacks. Additionally, it explored aspects such as performance, user-friendliness, and scalability. This comparison underscored the significance of selecting an appropriate framework tailored to the specific needs of a project and the skill set of the development team.

2.3 Overview of Back-end Frameworks

In the following section, we will delve into some of the most renowned back-end frameworks in contemporary web development. Among them are Laravel, Django, Node.js, and Spring. These frameworks have not only risen in popularity due to their robust features and capabilities but also because of the vast communities that have contributed to their evolution. As we trace their historical progression, we will highlight the pivotal moments and innovations that have positioned them as cornerstones in the realm of back-end development.

2.3.1 Laravel

Laravel is a popular PHP framework known for its elegant and robust architecture. It follows the Model-View-Controller (MVC) pattern, providing a structured and organized approach to web application development. It was developed in 2011 by Taylor Otwell as a spin-off of Symfony [23]. Laravel incorporates features like routing, ORM, caching, and authentication, enhancing productivity and maintainability. It also offers a modular architecture, allowing developers to leverage reusable components and create scalable applications. Laravel's architecture is influenced by various design patterns such as Dependency Injection and Facades, ensuring flexibility and testability [24]. Its extensive documentation and active community make it a preferred choice for developers.

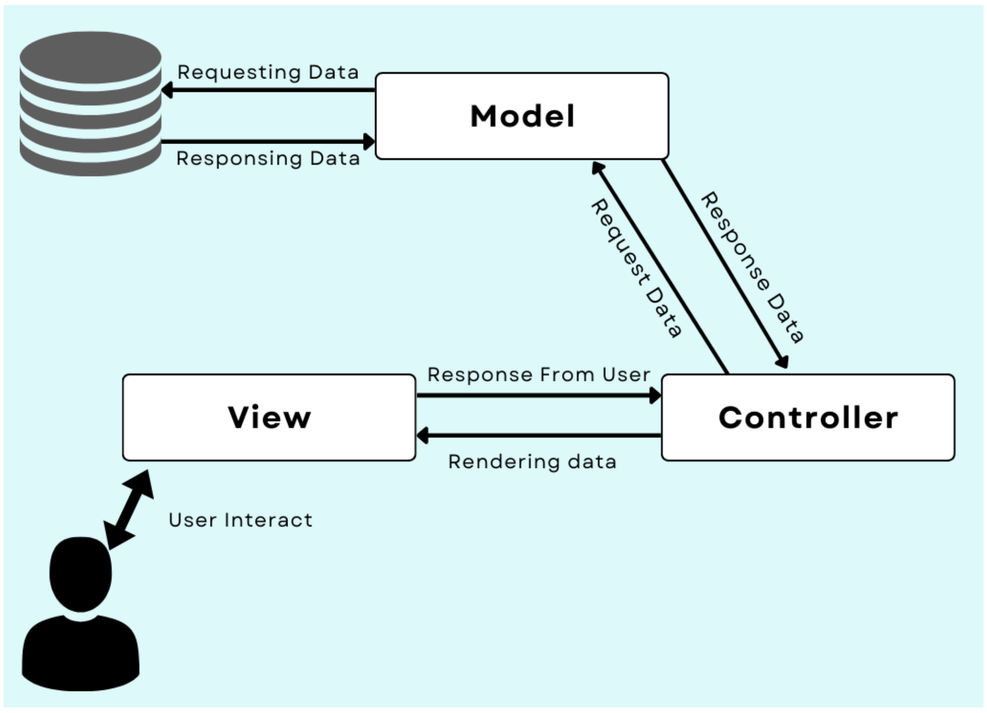

Figure 2.2: Model-View-Controller (MVC) architecture in Laravel

Laravel has gone through several versions, each introducing new features and improvements [25]:

- Laravel 1 was released in June 2011, followed by Laravel 2 in November 2012.
- Laravel 3 was released in February 2013 and introduced a modular architecture and better support for third-party libraries.
- Laravel 4 was released in May 2013 and included improvements in performance, modularity, and scalability.
- Laravel 5 was released in February 2015 and introduced significant changes, including a new directory structure, middleware, and the Blade templating engine.
- Laravel 6 was released in September 2019, followed by Laravel 7 in March 2020 and version 8 in September 2020. These releases brought various new features, such as Laravel Vapor, Laravel Sanctum, and Laravel Jetstream, to name a few.
- Laravel 9 was released in September 2022 and brought significant changes to the framework's internal architecture, including improvements to routing, middleware, and the service container.
- Laravel 10 is expected to be released in February or March 2023 .

Each version of Laravel has contributed to its evolution, addressing developer needs and enhancing the framework's capabilities. With each release, Laravel continues to solidify its position as one of the leading PHP frameworks in the industry [26].

Laravel is a PHP web development framework that has experienced significant growth in popularity since its release in 2011. Over the years, it has gained a solid reputation and become one of the most popular choices for web application development in PHP.

One of the main reasons behind Laravel's popularity is its focus on code elegance, simplicity, and readability. With concise and expressive syntax, Laravel allows developers to write clean and maintainable code. This has attracted a wide community of developers who appreciate the efficiency and clarity that the framework provides [25].

Another key factor that has contributed to Laravel's popularity is its extensive set of features and built-in tools. Laravel offers a variety of functionalities such as routing, session handling, authentication, caching, and more, enabling developers to efficiently build robust and feature-rich web applications. Additionally, Laravel provides a unified and consistent API that makes it easy to interact with external services and integrate third-party libraries [27]. The Laravel community is active, supportive, and always willing to help newcomers. There are abundant resources available, including official documentation, online tutorials, blogs, and educational videos. Furthermore, numerous Laravel-dedicated events and conferences take place worldwide, where developers can share knowledge, learn from experts, and network.

The popularity of Laravel is also reflected in the numbers. The official Laravel repository on GitHub has a significant number of 73.400 stars , indicating widespread adoption and community support. Additionally, downloads of Laravel-related packages in the PHP package repository, Packagist, show a high level of interest and active usage of the framework [28].

In the job market, the demand for developers with Laravel experience has significantly increased. Job postings and freelance projects requiring Laravel skills are plentiful, further reinforcing its status as a popular and relevant choice in the web development field. For instance, in the second week of May 2023, there were around 900 job applications on LinkedIn in Latin America. Considering the continuous growth in job offers, learning Laravel can be highly advantageous for job seekers.

In addition, Laravel provides a built-in code generator called ”Artisan” [29] that allows developers to generate code for various tasks, such as creating controllers, models, migrations, views, and more. This code generator saves developers time and effort by automating the creation of repetitive code components, allowing them to focus on implementing the application's specific business logic. Laravel also provides a powerful templating engine called Blade [30], which enables developers to write clean and expressive templates with features like template inheritance, conditionals, loops, and more .

Laravel is well-known for its emphasis on code maintainability and the structure of web applications. The framework promotes good development practices such as clear separation of responsibilities, the use of design patterns, and a modular architecture. These aspects contribute to improving code maintainability in the long term.

Laravel is a renowned web development framework known for its emphasis on security. It incorporates various features and tools to assist developers in safeguarding their applications against common vulnerabilities. Here are some of the security measures employed by Laravel [25]:

1. Cross-Site Request Forgery (CSRF) Protection: Laravel includes built-in protection against CSRF attacks. It generates a unique token for each session and verifies it for every POST, PUT, DELETE, and PATCH request, thereby preventing CSRF attacks .

2. SQL Injection Protection: Laravel employs prepared statements and parameter binding techniques to defend against SQL injection attacks. By doing so, it ensures that malicious attempts to manipulate database queries are thwarted.

3. Cross-Site Scripting (XSS) Protection: Laravel automatically escapes output values within views to mitigate the risk of XSS attacks. Additionally, it provides methods for input data cleansing and filtering.

4. Data Validation: Laravel incorporates a built-in validation system that simplifies the validation of input data. Developers can define validation rules for input fields, and Laravel automatically validates the data, displaying error messages upon validation failure.

5. Password Hashing: Laravel employs secure hashing algorithms to store passwords securely. Passwords are never stored in plain text but are instead stored as hashes in the database.

6. Route Protection and Authorization: Laravel offers an intuitive routing system that enables the protection of specific routes or groups of routes behind authentication. Furthermore, Laravel provides functions for role-based and permission-based authorization.

7. Error Handling and Logging: Laravel incorporates robust error handling and logging mechanisms. Errors are automatically logged and can be configured to trigger email notifications in the event of application errors in production environments.

2.3.2 Django

Django is a high-level Python web framework that allows for rapid development of secure and maintainable websites. It follows the Model-View-Controller (MVC) Fig 2.2 architectural pattern and includes a powerful Object-Relational Mapping (ORM) system for interacting with databases. The MVC pattern separates the application into three interconnected components: Model, View, and Controller. The Model is responsible for managing the application's data and business logic, the View is responsible for rendering the user interface, and the Controller is responsible for handling user requests and updating the Model and View accordingly. [31]

Django also includes several other components that contribute to its overall architecture, including:

- URL routing: Django uses a URL routing system that maps incoming requests to the appropriate Controller or View.
- Template engine: Django's template engine provides a way to separate the application logic from the presentation logic.
- Object-relational mapping (ORM): Django's built-in ORM provides a convenient way to interact with a database, allowing developers to work with objects rather than raw SQL.
- Middleware: Django's middleware provides a way to add functionality to the request/response processing pipeline.

Django provides many built-in features for common web development tasks, such as user authentication, form handling, and URL routing, as well as support for handling static and media files. It also has a thriving ecosystem of third-party packages for extending its functionality. [32]

Throughout its development, each version has introduced new features and improvements, addressing the growing demands of developers and technological advancements. The framework's consistent updates have enhanced development efficiency, streamlined processes, and expanded its functionality [32]:

- Django 1.0: This was the first stable release of Django, and it introduced many of the features that have made Django popular, such as the Django ORM, the admin interface, and the URL routing system.
- Django 1.2: This release added support for multiple databases, making it easier to write applications that use more than one database back-end.
- Django 1.3: This release added support for class-based views, which provide a more flexible way to define views than the traditional function-based views.
- Django 1.4: This release introduced the new timezone-aware datetime module, which makes it easier to work with datetime objects across different timezones.
- Django 1.5: This release added support for custom user models, making it easier to create applications that use alternative user models beyond the built-in User model.
- Django 1.6: This release added support for template context processors, which allow you to add variables to the template context automatically.
- Django 1.7: This release introduced migrations, a new system for managing database schema changes that replaced the previous south library.
- Django 1.8: This release added support for complex query expressions, making it easier to write complex database queries using the Django ORM.
- Django 1.9: This release introduced support for password validation, making it easier to enforce password policies in your applications.
- Django 2.0: This release dropped support for Python 2, making it a Python 3-only release. It also introduced several new features, including support for window functions in the ORM.
- Django 3.0: This release introduced experimental support for asynchronous views, and it added support for MariaDB as a database back-end.
- Django 3.1: This release added support for JSONField, a new database field type that allows you to store JSON data in the database.

- Django 3.2: This release introduced support for signed cookies, which can be used as an alternative to traditional session-based authentication.
- Django 4.0: Expected to be released in late 2022 or early 2023, this release is expected to introduce several new features, including improved support for model field validation and improved support for database schema changes.

Django, the Python web framework, has experienced substantial popularity and widespread adoption, supported by various numbers and studies, making it one of the leading choices for web development. According to the 2021 Python Developers Survey conducted by the Python Software Foundation, Django ranked as the most popular web framework among Python developers, with over 40% of respondents indicating its usage [33]. The Stack Overflow Developer Survey 2021 [34] also highlighted Django's popularity, listing it as one of the most loved frameworks. Furthermore, GitHub showcases Django's widespread adoption, with its repository amassing over 70,600 stars and thousands of forks, indicating active community engagement and contributions.

Beyond surveys and statistics, Django's popularity can be attributed to its numerous strengths. Its emphasis on clean and maintainable code, along with its comprehensive documentation, contributes to its appeal among developers. Django's robust feature set, including built-in components like authentication, form handling, and database modeling, enhances productivity and accelerates development. The framework's adherence to best practices, such as the Model-View-Controller (MVC) architectural pattern, promotes scalability and code organization. [31]

Django's extensive community support has also played a vital role in its popularity. The Django community is highly active, with numerous resources available for developers. The official Django documentation provides comprehensive guides and references, while community-driven platforms like Django Packages offer a vast collection of reusable apps and tools. Furthermore, the Django community organizes regular events, such as DjangoCon, where developers gather to share knowledge, discuss best practices, and showcase their projects . The official Laravel reposi- tory on GitHub has a significant number of 73.400 stars.[35]

Django's popularity is further evident in its widespread industry adoption. It powers numerous high-traffic websites and applications, including popular platforms like Instagram, Pinterest, and Mozilla. Django's reliability, security, and scalability make it an attractive choice for organizations of all sizes. Its versatility allows it to cater to diverse domains, such as e-commerce, social media, and content management systems.

In terms of job market demand, Django developers continue to be in high demand. Websites like Indeed and LinkedIn feature numerous job postings requiring Django expertise 1500 in the second week of May 2023. Freelancing platforms, such as Upwork, also showcase a significant number of Django-related projects, indicating the continuous need for skilled professionals.

One of the notable features of Django is its built-in code generator called "django-admin." This command-line tool allows developers to generate code for various tasks, such as creating models, views, templates, forms, and more. The code generator automates the creation of repetitive code components, saving developers time and effort during the development process [32].

Django includes a powerful templating engine that goes by the name of "Django Templates." It allows developers to create clean and expressive templates for rendering HTML dynamically. The templating engine supports features like template inheritance, conditional statements, loops, filters, and more. This enables developers to build flexible and reusable templates for their web application's user interface.[36]

Similar to Laravel, Django also emphasizes code maintainability and follows best practices in software development. It encourages the separation of concerns and follows the Don't Repeat Yourself (DRY) principle. Django promotes modular and reusable code through its application structure, which allows developers to divide their project into smaller, self-contained components.

Django is renowned for its emphasis on security and provides several built-in features and tools to help developers protect their web applications against common vulnerabilities. Here are some of the security measures incorporated in Django [32]:

1. Cross-Site Scripting (XSS) Protection: Django employs XSS protection by default, automatically escaping output values in views. This helps prevent XSS attacks by ensuring that any malicious code is not executed in end-users' browsers.

2. Cross-Site Request Forgery (CSRF) Protection: Django includes built-in CSRF protection to prevent CSRF attacks. It generates unique CSRF tokens for each form and verifies that the tokens match in POST, PUT, DELETE, and PATCH requests, ensuring that requests are only sent from application-generated forms and not from malicious third parties.

3. SQL Injection Protection: Django provides a secure Object-Relational Mapping (ORM) that prevents SQL injection by safely executing database queries. Django's ORM uses query parameters and prepared statements to ensure that user-supplied data is properly escaped and treated.

4. Form Validation: Django includes a built-in form validation system that helps ensure user-entered data adheres to required rules and constraints. This helps prevent injection attacks and ensures the integrity and validity of the entered data.

5. Secure Password Handling: Django provides secure functions and methods for password storage and handling. It utilizes strong hashing algorithms and salting techniques to securely store passwords in the database.

6. Access Control and Authorization: Django offers a flexible access control and authorization system, allowing developers to define permissions and roles to restrict access to certain parts of the application. This ensures that only authorized users can access specific functionalities and sensitive data.

7. Error Handling and Event Logging: Django provides robust error handling and event logging mechanisms. Errors are logged by default in the application logs, facilitating issue identification and resolution. Additionally, email notifications can be configured to receive alerts for unexpected errors in the production environment.

2.3.3 Nodejs

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside of a web browser. It was first released in 2009 by Ryan Dahl, and has since become a popular platform for building server-side applications and network applications [37]. Node.js uses an event-driven, non-blocking I/O model that makes it efficient and lightweight, and it has a large and active community of developers contributing to its growth and development.[38]

Node.js follows a modular and event-driven architecture, which makes it well-suited for building scalable and high-performance applications. At its core, Node.js is built around a single-threaded event loop, which handles all I/O operations asynchronously. This allows Node.js to handle a large number of requests without blocking, making it an ideal choice for building real-time applications such as chat applications or online gaming platforms.

Node.js also includes a built-in module system that allows developers to load and use external libraries and packages, making it easy to extend and customize Node.js applications. The module system is based on CommonJS, a module specification that defines a standard format for organizing and distributing JavaScript modules.

Node.js also includes a number of built-in modules that provide functionality for tasks such as file I/O, networking, and cryptography. In addition, Node.js supports the use of external modules through package managers such as npm.[38]

Overall, Node.js's architecture is focused on providing a lightweight, efficient platform for building server-side applications that can handle a high volume of I/O operations. Its modular design and support for external libraries and packages make it easy to extend and customize to meet the needs of a wide range of applications.[39]

This are the nNodejs versions that was releases.[40]

- Node.js 0.1.0 (May 2009): Initial release of Node.js.
- Node.js 0.4.0 (March 2011): Introduced the Node Package Manager (npm).
- Node.js 0.8.0 (June 2012): Significant performance improvements, including the addition of the libuv library for handling I/O.
- Node.js 0.10.0 (March 2013): Added support for streams2, which improved handling of large streams of data.
- Node.js 0.12.0 (February 2015): Upgraded V8 JavaScript engine to version 3.28, which brought significant performance improvements.

- Node.js 4.0.0 (September 2015): Merged Node.js and io.js codebases, bringing together the two separate projects.
- Node.js 6.0.0 (April 2016): Introduced support for the ECMAScript 2015 (ES6) specification, including new language features such as classes and arrow functions.
- Node.js 8.0.0 (May 2017): Introduced support for async/await, a new language feature that simplifies asynchronous programming.
- Node.js 10.0.0 (April 2018): Upgraded V8 engine to version 6.6, which brought performance improvements and new language features such as rest/spread properties and asynchronous iteration.
- Node.js 12.0.0 (April 2019): Upgraded V8 engine to version 7.4, which brought improved performance and new language features such as optional chaining and nullish coalescing.
- Node.js 14.0.0 (April 2020): Upgraded V8 engine to version 8.1, which brought performance improvements and new language features such as top-level await and import().
- Node.js 16.0.0 (April 2021): Upgraded V8 engine to version 9.0, which brought performance improvements and new language features such as the WeakRefs API.

The security in the Node.js framework is of utmost importance to protect web applications and ensure data integrity. Here are some common security measures that can be implemented while developing applications with Node.js:

1. Input Validation and Filtering: It is crucial to validate and filter user input to prevent malicious code injection attacks such as SQL injections or cross-site scripting (XSS) attacks. Node.js provides libraries and modules like Express-validator and Joi for input validation [41].
2. Proper Dependency Management: Verifying the authenticity and reliability of third-party packages and keeping dependencies up to date are essential security practices. Using security tools like vulnerability scanning can help identify and address potential security issues in the used packages [42].

3. Protection against Denial-of-Service (DoS) Attacks: Implementing measures to protect against DoS attacks is crucial for ensuring the availability and stability of Node.js applications. Techniques such as rate limiting can be applied to limit the number of requests from a single client, and utilizing DoS protection services at the infrastructure level can help mitigate such attacks [42].

4. Authentication and Authorization: Implementing a secure authentication and authorization system is vital for protecting sensitive resources and data. Node.js provides libraries like Passport.js, which offers flexible authentication strategies. Properly implementing authorization mechanisms ensures that only authorized users can access specific functionalities [43].

5. Encryption and Secure Password Storage: Utilizing strong encryption algorithms and techniques for storing passwords is essential to safeguard user credentials. Node.js offers libraries like bcrypt and crypto for secure password hashing and encryption [42].

6. Session Management: Secure session management is crucial for preventing session-related vulnerabilities. Implementing secure session handling techniques, such as using secure session identifiers, proper session storage, and expiration policies, can mitigate session-related security risks [42].

7. Logging and Auditing: Implementing logging and auditing mechanisms helps track and monitor application activities, aiding in identifying and investigating security incidents. Libraries like Winston and Bunyan provide logging capabilities for Node.js applications [44].

8. Regular Updates and Patching: Keeping Node.js and its dependencies up to date is essential to benefit from the latest security patches and bug fixes. Staying informed about security updates and promptly applying them helps mitigate potential vulnerabilities [42].

2.3.4 Spring

Spring is an open-source application framework for Java used to build enterprise-level applications. It provides a comprehensive programming and configuration model for modern Java-based enterprise applications. Spring's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform [45]. The framework's primary goal is to simplify the development of Java applications and promote good programming practices by enabling developers to focus on the "business logic" of their application, rather than on infrastructure and integration tasks. It was first released in 2002 and has since become one of the most popular Java frameworks in use today. [46]

Spring architecture is based on the Model-View-Controller (MVC) design pattern, which separates the application into three components:

- Model: This represents the data and business logic of the application. It is responsible for managing data persistence, validation, and processing.
- View: This is responsible for rendering the user interface. It receives data from the model and presents it to the user.
- Controller: This handles user input and manages the flow of data between the model and the view. It is responsible for mapping incoming requests to the appropriate handlers and returning responses to the client.

In addition to the MVC pattern, Spring also includes several other key components:

- Dependency Injection (DI): This is a design pattern that allows components to be loosely coupled, making them easier to test and maintain. Spring's DI container manages the creation and configuration of objects and their dependencies.
- Aspect-Oriented Programming (AOP): This is a programming paradigm that enables developers to modularize cross-cutting concerns, such as logging and security, into reusable components.
- Spring Security: This is a powerful security framework that provides authentication, authorization, and other security features to Spring-based applications.

- Spring 1.0 (2004): The initial release of Spring, designed to be a lightweight alternative to existing J2EE frameworks. Key features included Inversion of Control (IoC) and Aspect-Oriented Programming (AOP).
- Spring 2.0 (2006): Introduced the concept of ”JavaConfig”, which allowed developers to configure Spring using pure Java code instead of XML files. Also added support for scripting languages and improved AOP capabilities.
- Spring 3.0 (2009): Introduced the Spring Expression Language (SpEL) for more powerful configuration options, added support for RESTful web services, and improved the Spring MVC framework with support for RESTful controllers.
- Spring 4.0 (2014): Dropped support for Java 5 and introduced support for Java 8 features such as lambda expressions. Also introduced support for reactive programming with the Spring WebFlux framework.
- Spring 5.0 (2017): Dropped support for Java 6 and introduced support for Java 9 features such as the Java Platform Module System (JPMS). Also introduced support for reactive streams and the Spring WebClient for non-blocking HTTP communication.
- Spring 6.0 (in development): Will drop support for Java 8 and introduce support for Java 17 features. Also expected to introduce improvements to the Spring WebFlux framework and simplify the Spring programming model. [46]

The Spring framework provides several security measures to protect applications from common security threats. These measures can be implemented to ensure the confidentiality, integrity, and availability of the application’s resources. Here are some key security measures for the Spring framework:

1. Authentication and Authorization: Spring Security offers robust authentication and authorization mechanisms. It provides various authentication options such as form-based authentication, token-based authentication using technologies like JSON Web Tokens (JWT), and integration with external providers such as OAuth 2.0. Additionally, Spring Security enables fine-grained access control using roles and permissions [47].

2. Protection against Common Security Attacks: Spring Security includes features to mitigate common security vulnerabilities such as Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), SQL Injection, and File Inclusion attacks. These protections are implemented through request filtering, input validation, and secure coding practices [47].

3. Session Management: Spring Security provides session management capabilities to handle user sessions securely. It allows configuring session policies, such as session timeout and invalidation, and ensures proper association of sessions with authenticated users [47].

4. Password Encryption: Spring Security offers utilities for secure password storage. It supports secure hashing algorithms and salting techniques to protect user passwords from unauthorized access. Storing passwords securely reduces the risk of compromising user accounts in the event of a security breach [47].

5. Auditing and Logging: Spring provides logging and auditing features that enable tracking and monitoring of application activities. This helps in identifying security incidents, investigating issues, and complying with regulatory requirements. Logging frameworks like Logback and Log4j are commonly used with Spring applications [47].

6. Brute Force Attack Prevention: Spring Security provides mechanisms to prevent brute force attacks, such as limiting the number of failed login attempts and temporarily blocking user accounts after multiple failed attempts. These measures protect user accounts from unauthorized access [47].

Chapter 3

Methodology Framework

This chapter aims to establish a systematic approach, based on quantitative model, for evaluating and selecting the most appropriate back-end framework for development projects. With an ever-increasing range of options, making an informed choice becomes crucial yet complicated. This chapter provides a structured approach that relies on carefully selected dimensions and metrics, offering an adaptable and weighted evaluation. By following this methodology, stakeholders can objectively compare different frameworks, taking into account both technical and organizational factors critical to the project's success.

3.1 General Objective

The main goal of this methodology is to offer a standardized and measurable procedure that helps to choose the most fitting back-end framework, serving developers and decision makers. It allows for aligning the framework's functionalities meticulously with the project's requirements, minimizing inaccuracies, and optimizing efficacy. This approach facilitates informed decisions based on predefined criteria and ensures the selected framework aligns seamlessly with the project's goals and needs, fostering enhanced outcomes and efficiency.

3.2 Relevance

The choice of a back-end framework is often a long-term commitment and can significantly affect both the development speed and the quality of the final product. A wrong choice can lead to scalability issues, increased costs, and delays. Therefore, it is crucial to have a clear and systematic process for assessing different frameworks based on a range of factors such as performance, security, compatibility, and community support.

3.3 Procedure Specification

This section will describe the detailed procedure that will be followed to specify, evaluate and select back-end frameworks according to a set of predetermined criteria. Key terminologies, such as dimension, metric, weight, and rating, will be defined and explained and used to categorize and quantify various aspects of the frameworks. The process for identifying relevant dimensions and metrics and for collecting relevant data from each framework under consideration will be detailed. Subsequently, we will discuss how quantitative scores will be assigned to each metric and how dimension scores will be calculated, in addition to assigning weights and calculating the final score. Finally, the method of comparison and selection of the most appropriate framework will be described based on the final weighted scores. In summary, this section will provide a comprehensive overview of the methodology to be employed for the systematic evaluation of back-end frameworks.

3.3.1 Basic Definitions

This section defines basic concepts that are crucial to understanding the criteria and processes that will be described later, thus establishing a common basis of understanding for the objective evaluation and comparison of the different back-end development frameworks that will be analyzed.

- **Dimension:** Refers to a broad and comprehensive evaluation category, such as security, performance, or scalability, in which similar features or attributes of a back-end framework are grouped together for evaluation and comparison. It is a general criterion that makes it easier to organize specific and detailed aspects of the framework that are being examined.
- **Metric:** Within each dimension, metrics are specific attributes or characteristics that can be evaluated to determine the effectiveness or quality of the framework on a

particular dimension. For example, in the security dimension, relevant metrics could include authentication, validation, and implemented testing methods, thus offering a more granular and detailed approach in evaluating each dimension.

- **Weight:** This term represents the degree of importance or relevance assigned to each dimension during the evaluation process. The weights directly influence the final rating of the framework, allowing the results to be adjusted according to the specific priorities and needs of the project or user.
- **Rating:** It is a quantitative value that summarizes the adequacy or suitability of a back-end framework based on the evaluation of metrics of its various dimensions. The score is calculated using a formula that integrates both the individual values of each metric and dimension and the weights assigned to them, providing a numerical summary of the overall performance of the framework in question.

3.3.2 Identification of Dimensions and Metrics

Before beginning the evaluation of back-end frameworks, it is essential to carefully identify the relevant dimensions and metrics that will serve as evaluation criteria. In this step, a list of relevant dimensions such as performance, scalability, ease of use, security, popularity, portability, maintainability, cost, ecosystem and compatibility, which are vital for the success of the project, is made. Within each dimension identified, specific metrics are determined, such as authentication for security or loading times for performance, which will allow for a more detailed and focused evaluation. This identification process is crucial to ensure that all important aspects of the framework are comprehensively and fairly evaluated, and to provide meaningful insights that will guide the final selection of the back-end framework.

3.3.3 Data Gathering

Once the relevant dimensions and metrics are identified, the next step in the evaluation process is the collection of data for each back-end framework under consideration. Data is obtained through methods such as empirical testing, documentation review, and consultation with subject matter experts. This step is essential since the accuracy and relevance of the data collected will directly influence the validity of the final results. By gathering accurate and relevant data, objective and informed evaluations can be made that accurately reflect the capabilities and limitations of each framework, ensuring a well-informed final selection tailored to the needs of the project.

3.3.4 Quantitative Scoring

In this phase of the procedure, each identified metric is evaluated and assigned a score on a scale from 0 to 1, allowing intermediate values such as 0.25, 0.5 or 0.75 for a more refined evaluation. This quantitative approach provides a means to objectively compare the different frameworks, offering clarity and precision in the results, and facilitating the identification of specific strengths and weaknesses in each framework evaluated.

3.3.5 Dimension Rating

In this process, the average score of each metric within a dimension is calculated to obtain the score for that dimension. Each metric M_{ij} within the dimension i receives a score, and then all these scores are added together and divided by the total number of metrics n in that dimension, as shown in the formula:

$$D_i = \frac{\sum_{j=1}^{n} M_{ij}}{n} \tag{3.1}$$

Where:

- D_i is the rating of dimension i.
- M_{ij} is the rating of metric j in dimension i.
- n is the total number of metrics in dimension i.

This scoring method provides a consolidated numerical summary that reflects the overall performance of the framework on each dimension evaluated, allowing detailed and direct comparisons between different frameworks and facilitating a detailed analysis of how each framework performs in specific areas. Therefore, an understandable and fair evaluation is ensured, essential to inform the final selection of the most appropriate framework for the project.

3.3.6 Weight Assignment

In this crucial step, weights are assigned to each metric, reflecting its relative importance in the specific use case or project at hand. This mapping is essential to ensure that the most critical aspects of the framework have a proportional impact on the final grade. The weights are intended to refine and personalize the evaluation process, allowing users to prioritize based on their particular needs and preferences, thus providing a final result that is truly representative and relevant to the project context.

3.3.7 Final Rating Calculation

Using the next formula, the final rating for each framework is calculated:

$$F = \sum_{i=1}^{m} D_i \times P_i \tag{3.2}$$

Where:

- F is the framework's final rating.
- D_i is the rating of dimension i.
- P_i is the weight assigned to dimension i.
- m is the total number of dimensions.

This integrative calculation takes into account both the score obtained in each dimension and the relative importance assigned to each of them, thus providing a comprehensive and balanced reflection of the overall performance of the framework in relation to the specific needs of the project. This final step is crucial to derive a single numerical value that allows you to objectively compare different frameworks and select the fittest one based on clear and fair criteria.

3.3.8 Comparison and Selection

Once the final scores are calculated, they are compared to each other to determine which of the evaluated frameworks is the most suitable. The framework with the highest weighted score is recommended for use, as this indicates that it best meets the project requirements and preferences. This phase is essential for informed decision making, as it allows for a direct and objective evaluation of the frameworks based on their performance on different dimensions and metrics, thus ensuring that the final choice is aligned with the objectives and needs of the project.

3.3.9 Evaluation Process

The evaluation process begins by assigning scores to each metric associated with a dimension, using a scale of 0, 0.5, and 1. Intermediate values, such as 0.25 and 0.75, can also be utilized if a more nuanced measurement is necessary or if the user desires a more detailed analysis based on the fulfillment of each metric.

- **Metrics Scoring:** Every individual metric related to a dimension is scored based on its compliance and efficacy within the evaluated framework.
- **Dimension Rating:** Subsequently, the average of the metrics scores is calculated to acquire the dimension's rating using the formula, as stated in equation 3.1:

$$D_i = \frac{\sum_{j=1}^{n} M_{ij}}{n}$$

Once ratings for all the dimensions are obtained, weights are assigned according to user-defined priorities. This phase aims to depict the relative importance of each dimension within the specific context of the project.

- **Final Rating Calculation:** The formula to derive the final rating of the framework is denoted in equation 3.2:

$$F = \sum_{i=1}^{m} D_i \times P_i$$

This comprehensive process ensures that every aspect of the framework is accurately evaluated and weighted, facilitating a fair and precise comparison among different frameworks and aiding in the selection of the framework best aligned with the project's goals and needs.

3.4 Generic Example

Before delving into the complexities of our analysis, it is imperative to establish an understanding of our approach through a generic example, illustrating the applied methodology in a simplified manner. In this section, a hypothetical scenario is presented to showcase the comparison of two frameworks, designated as Framework A and Framework B, based on three predefined dimensions. For each dimension, two metrics are considered to evaluate and contrast the frameworks systematically. The intention is to offer a clear and concise depiction of how the methodology functions in practice, enabling an assessment and final rating of each framework. It should be noted that the comprehensive explanations of each dimension and metric will be furnished in Chapter 4 (See Chapter 4 for more details).

Suppose we are evaluating two frameworks and considering 3 dimensions: Dimension 1, Dimension 2, and Dimension 3. Each metric has 2 metrics.

Framework A:

- Dimension 1: Metric 1 (0.75), Metric 2 (0.5)
- Dimension 2: Metric 1 (0.25), Metric 2 (1)
- Dimension 3: Metric 1 (0.5), Metric 2 (0.75)

Framework B:

- Dimension 1: Metric 1 (0.5), Metric 2 (0.5)
- Dimension 2: Metric 1 (0.75), Metric 2 (0.75)
- Dimension 3: Metric 1 (0.25), Metric 2 (0.5)

If we assign a weight of 0.5 to Dimension 1, 0.3 to Dimension 2, and 0.2 to Dimension 3, we can calculate each framework's final rating using the previously mentioned formulas.

$$F_A = (0.625 \times 0.5) + (0.625 \times 0.3) + (0.625 \times 0.2) = 0.625$$

$$F_B = (0.5 \times 0.5) + (0.75 \times 0.3) + (0.375 \times 0.2) = 0.5625$$

Based on the calculations, Framework A has a higher final rating (0.625) compared to Framework B (0.5625).

Now, let's create the comparison table:

Comparison Table

FW	Met 1.1	Met 1.2	Met 2.1	Met 2.2	Met 3.1	Met 3.2
A	0.75	0.5	0.25	1	0.5	0.75
B	0.5	0.5	0.75	0.75	0.25	0.5

Table 3.1: Metric ratings for Frameworks A and B

FW	Dim 1 (Weight: 0.5)	Dim 2 (Weight: 0.3)	Dim 3 (Weight: 0.2)
A	0.625	0.625	0.625
B	0.5	0.75	0.375

Table 3.2: Dimension ratings and weights for Frameworks A and B

Final ratings:

$$F_A = 0.625$$

$$F_B = 0.5625$$

Based on the table and calculations, the **Framework A** is recommended over the **Framework B** for this particular scenario due to its higher final rating.

With these calculations and the table, one can clearly and concisely discern which framework is more suitable based on the defined dimensions and weights in this example. Of course, in real-world applications, multiple additional factors may need to be considered, and a more detailed evaluation may be required.

Choosing a back-end framework is a critical decision for the success of any development project. Multiple criteria need to be considered, from technical aspects to organizational factors. With so many options available, it's essential to have a systematic and objective methodology to guide this choice. The importance of this methodology lies in its ability to adapt to the specific needs of a user or project, assigning weights to the different dimensions based on their priorities.

Chapter 4

Dimensions and Metrics

Choosing a back-end framework is a very important decision in the software development process, often shaping the project's architecture, scalability, and long-term maintainability. With a multitude of options available, it's essential to evaluate frameworks based on a series of well-defined dimensions, ensuring alignment with project requirements and team expertise. This chapter elucidates key dimensions, each with associated quantitative metrics. These dimensions provide a comprehensive guideline for developers and decision-makers to systematically assess, compare, and ultimately select the most suitable back-end framework for their specific needs.

4.1 Ease of Development

Ease of Development pertains to the simplicity and expedience with which developers can understand, learn, and use the framework to create and maintain applications. This dimension is vital as it can significantly impact the development time and the quality of the final product.

- **Metric: Programming Language**
  - *Meaning:* Indicates whether the framework's programming language is well-known by the team.
  - *Where to Find:* Discuss with the team and consult skill sets.
  - *Measurement:*

    * Very familiar: 1
    * Moderately familiar: 0.5
    * Not familiar: 0

- **Metric: Maturity**
  - *Meaning:* Indicates whether the technology is well-established or something new that is still being tested.
  - *Where to Find:* Review the history of the framework and the community backing it.
  - *Measurement:*
    * Well-established: 1
    * In development: 0.5
    * New or under testing: 0
- **Metric: Code Generators and Scaffolding**
  - *Meaning:* Tools that allow developers to rapidly generate code for common components such as models, views, and controllers.
  - *Where to Find:* Consult the framework's documentation and available tools.
  - *Measurement:*
    * Extensive support and efficiency: 1
    * Moderate support: 0.5
    * No support or inefficient: 0
- **Metric: Documentation Quality**
  - *Meaning:* Availability and completeness of official documentation.
  - *Where to Find:* Official website or GitHub repository.
  - *Measurement:*
    * full measure: 1 (Comprehensive, well-organized documentation)
    * Partially measure: 0.5 (Some useful documentation, but lacking in areas)

    * Not measure: 0 (Poor or non-existent documentation)

- **Metric: Community Support**
  - *Meaning:* Availability of community resources, tutorials, and forums.
  - *Where to Find:* Online communities, Stack Overflow, GitHub.
  - *Measurement:*
    * full measure: 1 (Strong community support and abundant resources)
    * Partially measure: 0.5 (Moderate community support and some resources)
    * Not measure: 0 (Limited or no community support)
- **Metric: Learning Curve**
  - *Meaning:* Time and effort required to become proficient.
  - *Where to Find:* Personal experience, online tutorials, and courses.
  - *Measurement:*
    * full measure: 1 (Easy to learn and become proficient)
    * Partially measure: 0.5 (Moderate learning curve)
    * Not measure: 0 (Steep learning curve)

4.2 Security

Security is a paramount dimension that assesses the framework's ability to safeguard against unauthorized access and other potential vulnerabilities, ensuring data integrity and confidentiality. This dimension is pivotal as it directly impacts the trust users place in the application and its adherence to data protection regulations.

- **Metric: Authentication**
  - *Meaning:* Support for standard authentication protocols like OAuth, JWT.
  - *Where to Find:* Check the framework documentation or built-in features.
  - *Measurement:*

    * Full measure: 1 (Supports multiple standard authentication protocols)
    * Partially measure: 0.5 (Supports one standard authentication protocol)
    * Not measure: 0 (No standard authentication protocol support)

- **Metric: Authorization**
  - *Meaning:* Ease of implementing access controls at a resource level.
  - *Where to Find:* Framework documentation and user reviews.
  - *Measurement:*
    * Full measure: 1 (Rich, flexible authorization features)
    * Partially measure: 0.5 (Basic authorization features)
    * Not measure: 0 (Poor or no authorization features)
- **Metric: Attack Mitigation**
  - *Meaning:* Resistance to common attacks like SQL Injection, XSS, CSRF.
  - *Where to Find:* Security audit reports and framework documentation.
  - *Measurement:*
    * Full measure: 1 (Resistant to multiple types of common attacks)
    * Partially measure: 0.5 (Resistant to at least one type of common attack)
    * Not measure: 0 (No resistance to common attacks)
- **Metric: Encryption**
  - *Meaning:* Support for encrypting data in transit and at rest.
  - *Where to Find:* Framework documentation.
  - *Measurement:*
    * Full measure: 1 (Supports both data-in-transit and data-at-rest encryption)
    * Partially measure: 0.5 (Supports either data-in-transit or data-at-rest encryption)
    * Not measure: 0 (No support for encryption)

- **Metric: Auditing and Logging**
  - *Meaning:* Ability to perform detailed tracking of security-related activities.
  - *Where to Find:* Framework features or third-party integrations.
  - *Measurement:*
    - Full measure: 1 (Detailed, configurable security logging and auditing)
    - Partially measure: 0.5 (Basic logging capabilities)
    - Not measure: 0 (No security logging and auditing)
- **Metric: Reported security issues**
  - *Meaning:* Indicates the framework's vulnerability and security robustness. Number of reported security issues in the past year.
  - *Where to Find:* Security advisory databases, framework's official channels.
  - *Measurement:*
    - Few or no reported issues (0-3): 1
    - Moderate number of issues (4-7): 0.5
    - Many issues (>7): 0

4.3 Scalability

Scalability refers to the ability of a framework to expand and manage increased load gracefully, ensuring that the performance of the application remains optimal as the demand grows. It's an essential property to consider, especially for applications anticipated to experience varied or growing user traffic over time.

- **Metric: Horizontal Scalability**
  - *Meaning:* The ability of the framework to handle more requests by adding more machines in the pool.
  - *Where to Find:* Through stress tests and framework documentation.
  - *Measurement:*

    * Full measure: 1 (Easily scales horizontally with little to no configuration)
    * Partially measure: 0.5 (Supports horizontal scaling but requires manual configuration)
    * Not measure: 0 (Does not support horizontal scaling)

- **Metric: Vertical Scalability**
  - *Meaning:* The ability to handle more requests by adding more power (CPU, RAM) to an existing machine.
  - *Where to Find:* Through performance tests and documentation.
  - *Measurement:*
    * Full measure: 1 (Easily scales vertically without application changes)
    * Partially measure: 0.5 (Supports vertical scaling but may require application changes)
    * Not measure: 0 (Does not support vertical scaling)
- **Metric: Load Balancing**
  - *Meaning:* The ability to distribute incoming network traffic across multiple servers.
  - *Where to Find:* Framework documentation and feature set.
  - *Measurement:*
    * Full measure: 1 (Built-in load balancing features or easy integration with external load balancers)
    * Partially measure: 0.5 (Can be manually configured for load balancing)
    * Not measure: 0 (No load balancing support)
- **Metric: Microservices Architecture Support**
  - *Meaning:* Capability to break up the application into loosely coupled, independently deployable components.
  - *Where to Find:* Framework features, documentation, and community support.

  - *Measurement:*
    - Full measure: 1 (Full support for microservices architecture)
    - Partially measure: 0.5 (Some support, but may require third-party tools)
    - Not measure: 0 (No support for microservices)

- **Metric: State Management**
  - *Meaning:* Ability to manage user sessions and data across multiple servers.
  - *Where to Find:* Framework documentation or third-party libraries.
  - *Measurement:*
    - Full measure: 1 (Built-in robust state management features)
    - Partially measure: 0.5 (Can handle state but may require third-party libraries)
    - Not measure: 0 (Poor or no state management features)

4.4 Performance

Performance of a framework is a pivotal consideration as it directly impacts user experience and resource utilization. Efficient frameworks ensure swift responses, even under substantial load, guaranteeing user satisfaction and optimal resource allocation.

- **Metric: Latency**
  - *Meaning:* The time it takes for the framework to respond to a single request.
  - *Where to Find:* This can be found using monitoring tools, load testing, benchmarks, official documentation.
  - *Measurement:*
    - Full measure: 1 (Less than 100 ms)
    - Partially measure: 0.5 (Between 100 ms and 300 ms)
    - Not measure: 0 (More than 300 ms)
- **Metric: Throughput**

  - *Meaning:* The number of requests that the system can handle per second.
  - *Where to Find:* Measured through load testing and performance monitoring, benchmarks, official documentation.
  - *Measurement:*
    * Full measure: 1 (More than 1000 requests per second)
    * Partially measure: 0.5 (Between 500 and 1000 requests per second)
    * Not measure: 0 (Less than 500 requests per second)

- **Metric: CPU Usage**
  - *Meaning:* Percentage of CPU time required for operations.
  - *Where to Find:* Using system resource monitoring tools, benchmarks.
  - *Measurement:*
    * Full measure: 1 (CPU usage less than 30%)
    * Partially measure: 0.5 (CPU usage between 30% and 60%)
    * Not measure: 0 (CPU usage above 60%)

- **Metric: Memory Usage**
  - *Meaning:* Amount of RAM used during operations.
  - *Where to Find:* Through system resource monitoring or profiling tools, benchmarks.
  - *Measurement:*
    * Full measure: 1 (Memory usage less than 70%)
    * Partially measure: 0.5 (Memory usage between 70% and 85%)
    * Not measure: 0 (Memory usage above 85%)

4.5 Popularity

Popularity can serve as an indirect indicator of a framework's reliability, user-friendliness, and community support. A popular framework is likely to have a large, active community that can offer support, plugins, and contribute to the framework's improvement and stability. Popularity is often associated with the quality and quantity of documentation and learning resources available, which can significantly impact the ease of adoption and use.

- **Metric: Number of stars/forks on platforms like GitHub.**
  - *Meaning:* Indicates community trust and engagement.
  - *Where to Find:* Framework's GitHub or similar repositories.
  - *Measurement:*
    - Many stars/forks (>20k): 1
    - Moderate stars/forks (10k-20k): 0.5
    - Few stars/forks (<10k): 0
- **Metric: Community activity and responsiveness.**
  - *Meaning:* Gauge of community support.
  - *Where to Find:* Forums, Stack Overflow, and GitHub issues.
  - *Measurement:*
    - Very active and responsive: 1
    - Moderately active: 0.5
    - Barely active or unresponsive: 0

4.6 Portability

Portability is a pivotal dimension that deals with the ability of the software framework to be transferred and adapted to different environments easily. This adaptability allows for enhanced flexibility and broadens the scope and reach of the application, facilitating deployment in varied ecosystems and ensuring optimal functionality across different platforms and environments.

- **Metric: Cross-Platform Support**

  - *Meaning:* Ability to run the application on multiple platforms without major code modifications.
  - *Where to Find:* Official documentation and developer community forums.
  - *Measurement:*
    * Full measure: 1 (Runs on multiple platforms without code modifications)
    * Partially measure: 0.5 (Runs on multiple platforms but requires some code modifications)
    * Not measure: 0 (Not designed for cross-platform support)

- **Metric: Containerization Support**
  - *Meaning:* Ease of packaging the application and its dependencies into a container.
  - *Where to Find:* Framework documentation and community discussions.
  - *Measurement:*
    * Full measure: 1 (Offers native or easy support for containerization)
    * Partially measure: 0.5 (Supports containerization but with additional configuration)
    * Not measure: 0 (Does not support containerization)

- **Metric: Cloud Compatibility**
  - *Meaning:* How well the framework integrates with cloud services.
  - *Where to Find:* Framework documentation, cloud service documentation, and developer testimonials.
  - *Measurement:*
    * Full measure: 1 (Excellent compatibility with major cloud providers)
    * Partially measure: 0.5 (Some compatibility issues with major cloud providers)
    * Not measure: 0 (Poor or no cloud compatibility)

- **Metric: Database Agnosticism**

  - *Meaning:* Flexibility to work with different types of databases.
  - *Where to Find:* Framework documentation and feature set.
  - *Measurement:*
    - Full measure: 1 (Supports multiple types of databases out of the box)
    - Partially measure: 0.5 (Limited to certain types of databases but extensible)
    - Not measure: 0 (Tied to a specific database with no flexibility)

4.7 Maintainability

Maintainability is a critical dimension, reflecting the ease with which a framework can accommodate modifications, enhancements, and debugging. High maintainability ensures that developers can efficiently comprehend, update, and augment the codebase to adapt to evolving requirements or address issues.

- **Metric: Code Complexity**
  - *Meaning:* The ease with which the codebase can be understood, modified, and extended.
  - *Where to Find:* Code analysis tools, code reviews.
  - *Measurement:*
    - Full measure: 1 (Low complexity, easy to understand code)
    - Partially measure: 0.5 (Moderate complexity, some areas are hard to understand)
    - Not measure: 0 (High complexity, code is difficult to understand)
- **Metric: Modularity**
  - *Meaning:* The extent to which the application is divided into independent, interchangeable modules.
  - *Where to Find:* Codebase structure, framework documentation.
  - *Measurement:*

        - full measure: 1 (Highly modular design)
        - Partially measure: 0.5 (Some modularity, but also monolithic components)
        - Not measure: 0 (Monolithic design, low modularity)

- **Metric: Testability**
    - *Meaning:* How conducive the framework is to unit testing, integration testing, and end-to-end testing.
    - *Where to Find:* Framework documentation, community forums, codebase.
    - *Measurement:*
        - Full measure: 1 (Framework provides or integrates well with testing tools)
        - Partially measure: 0.5 (Testing possible but may require significant effort or third-party tools)
        - Not measure: 0 (Poorly suited for testing)
- **Metric: Long-Term Support**
    - *Meaning:* Availability of long-term support for the framework, including security updates and bug fixes.
    - *Where to Find:* Official announcements, framework website, community forums.
    - *Measurement:*
        - Full measure: 1 (Guaranteed long-term support)
        - Partially measure: 0.5 (Some level of support, but not guaranteed long-term)
        - Not measure: 0 (No long-term support)

4.8 Cost

Cost is a pivotal dimension as it assesses the economic feasibility and impact of adopting a specific framework. It is crucial for organizations to evaluate the cost dimension meticulously to ensure sustainable and efficient deployment of resources.

- **Metric: Licensing**
  - *Meaning:* Cost associated with the framework's license, if applicable.
  - *Where to Find:* Official framework website, documentation, or pricing page.
  - *Measurement:*
    - ∗ Full measure: 1 (No licensing cost)
    - ∗ Partially measure: 0.5 (Moderate licensing cost)
    - ∗ Not measure: 0 (High licensing cost)
- **Metric: Server Resources**
  - *Meaning:* Requirements and costs associated with CPU, memory, and storage.
  - *Where to Find:* Framework documentation, case studies, real-world usage.
  - *Measurement:*
    - ∗ Full measure: 1 (Minimal requirements and low cost)
    - ∗ Partially measure: 0.5 (Moderate requirements and costs)
    - ∗ Not measure: 0 (High requirements and elevated costs)
- **Metric: Migration Cost**
  - *Meaning:* Time and resources required to migrate from another system.
  - *Where to Find:* Developer testimonials, documentation, community forums.
  - *Measurement:*
    - ∗ Full measure: 1 (Easy and low-cost migration)
    - ∗ Partially measure: 0.5 (Moderate in terms of time and resources for migration)
    - ∗ Not measure: 0 (Difficult and costly to migrate from another system)
- **Metric: Learning Cost**
  - *Meaning:* Time required for the team to get up to speed with the framework.
  - *Where to Find:* Community forums, developer feedback, personal experience.

- *Measurement:*
  - Full measure: 1 (Quick learning curve)
  - Partially measure: 0.5 (Moderate time to learn)
  - Not measure: 0 (Steep learning curve)

4.9 Ecosystem and Compatibility

The Ecosystem and Compatibility dimension evaluates the adaptability and integration capability of the framework within the existing technology ecosystem. The evaluation of this dimension is crucial to ensuring that the framework can effortlessly integrate with other technologies and tools, facilitating a cohesive and seamless development environment.

- **Metric: Libraries and Tools**
  - *Meaning:* Ease of integrating external libraries and third-party tools.
  - *Where to Find:* Framework documentation, developer communities, and tutorials.
  - *Measurement:*
    - Full measure: 1 (Very easy to integrate)
    - Partially measure: 0.5 (Moderately easy to integrate)
    - Not measure: 0 (Difficult to integrate)
- **Metric: Front-end Compatibility**
  - *Meaning:* Ease of integration with front-end frameworks and libraries.
  - *Where to Find:* Framework documentation, community forums, and example projects.
  - *Measurement:*
    - Full measure: 1 (Very easy to integrate)
    - Partially measure: 0.5 (Moderately easy to integrate)
    - Not measure: 0 (Difficult to integrate)

- **Metric: APIs and Connectivity**
  - *Meaning:* Ease of connecting to external services via APIs.
  - *Where to Find:* Framework documentation, API documentation of external services, developer testimonials.
  - *Measurement:*
    - ∗ Full measure: 1 (Very easy to connect)
    - ∗ Partially measure: 0.5 (Moderately easy to connect)
    - ∗ Not measure: 0 (Difficult to connect)
- **Metric: Updates and Maintenance**
  - *Meaning:* Frequency and quality of updates to the framework.
  - *Where to Find:* Framework release notes, community forums, and feedback.
  - *Measurement:*
    - ∗ Full measure: 1 (Frequent and high-quality updates)
    - ∗ Partially measure: 0.5 (Infrequent or medium-quality updates)
    - ∗ Not measure: 0 (Rare and poor-quality updates)

4.10 Metrics Resume

In the following table, we present a systematic and detailed summary of the key metrics used to evaluate back-end frameworks. These metrics span across various critical dimensions such as Ease of Development, Security, Scalability, Performance, Popularity, Portability, Maintainability, Cost, and Ecosystem and Compatibility. Each metric is assessed on a quantifiable scale, enabling a comprehensive and objective comparison of different frameworks. This tabular compilation serves as a valuable tool for developers, architects, and decision-makers in selecting the most appropriate back-end framework for their specific project needs, taking into consideration the multifaceted aspects of software development.

Dimension	**Metric**	**Measurement Scale**
Ease of Development	Programming Language	1, 0.5, 0
	Maturity	1, 0.5, 0
	Code Generators and Scaffolding	1, 0.5, 0
	Documentation Quality	1, 0.5, 0
	Community Support	1, 0.5, 0
	Learning Curve	1, 0.5, 0
Security	Authentication	1, 0.5, 0
	Authorization	1, 0.5, 0
	Attack Mitigation	1, 0.5, 0
	Encryption	1, 0.5, 0
	Auditing and Logging	1, 0.5, 0
	Reported Security Issues	1, 0.5, 0
Scalability	Horizontal Scalability	1, 0.5, 0
	Vertical Scalability	1, 0.5, 0
	Load Balancing	1, 0.5, 0
	Microservices Architecture Support	1, 0.5, 0
	State Management	1, 0.5, 0
Performance	Latency	1, 0.5, 0
	Throughput	1, 0.5, 0
	CPU Usage	1, 0.5, 0
	Memory Usage	1, 0.5, 0
Popularity	Number of Stars/Forks on GitHub	1, 0.5, 0
	Community Activity and Responsiveness	1, 0.5, 0
Portability	Cross-Platform Support	1, 0.5, 0
	Containerization Support	1, 0.5, 0
	Cloud Compatibility	1, 0.5, 0
	Database Agnosticism	1, 0.5, 0
Maintainability	Code Complexity	1, 0.5, 0
	Modularity	1, 0.5, 0
	Testability	1, 0.5, 0
	Long-Term Support	1, 0.5, 0
Cost	Licensing	1, 0.5, 0
	Server Resources	1, 0.5, 0
	Migration Cost	1, 0.5, 0
	Learning Cost	1, 0.5, 0
Ecosystem and Compatibility	Libraries and Tools	1, 0.5, 0
	Frontend Compatibility	1, 0.5, 0
	APIs and Connectivity	1, 0.5, 0
	Updates and Maintenance	1, 0.5, 0

Table 4.1: Summary of Back-End Framework Evaluation Metrics

Chapter 5

Application of Methodological Framework: A Practical Example

This chapter seeks to articulate in a cohesive and practical way the foundations established in the previous chapters, integrating the methodology detailed in Chapter 3 with the dimensions and metrics outlined in Chapter 4. The intention is to develop a structured and comprehensive evaluative process that allows an evaluation balanced and multifaceted of different frameworks, contemplating aspects such as functionality, performance, maintainability, cost and compatibility in the technological ecosystem. This comprehensive approach aims to provide a robust and adaptable tool to guide the informed selection of back-end frameworks, effectively aligning their capabilities and characteristics with the specific requirements and objectives of software development projects. Through the careful implementation of this methodology, we aim to maximize the value and minimize the risks associated with the adoption of new frameworks, thus contributing to the success and sustainability of technological projects.

5.1 Description of Project

This section seeks to exemplify the application of the developed methodology for the selection of a suitable framework for a project to make an application web to support an "Inventory Management System". The project pertains to a company specializing in inventory management, handling products sourced globally. The system necessitates a NoSQL database and is estimated to require good controller of back-end calls to ensure optimal functionality.

5.1.1 Project Requirements

Given the project's mandate to efficiently manage vast inventory data from diverse global sources, the system demands high scalability, reliability, and peak performance. In addition, due to the significant volume of daily back-end calls, it's imperative for the system to be robust and responsive, ensuring the uninterrupted operation of the inventory system. The choice of a NoSQL database emphasizes the importance of selecting a framework that not only integrates seamlessly with such databases but also offers robust support.

The project team consists of two members. One member predominantly possesses expertise in JavaScript, while the other is proficient in PHP and Python. Java is not within their comfort zone. Taking into consideration the team's size, the system should prioritize ease of development and maintainability. Moderate cost, compatibility with various APIs, and a structured approach to foster an optimal development experience are also critical factors. Therefore, the team is in pursuit of a framework that aligns with both their technical requirements and their current skill set, thereby streamlining the development and deployment of their initiative.

5.1.2 Framework Selection

For the practical implementation and assessment of the proposed methodology, we have chosen four prominent and widely used frameworks in contemporary software development: Laravel, Django, Spring, and Nodejs [2.3]. Each of these frameworks represents different paradigms, programming languages, and ecosystems, thus providing a diverse and representative field for the application of our evaluative methodology.

- **Laravel:** This PHP framework is recognized for its elegant syntax and robustness, being a preferred choice for many developers aiming to build modern and scalable web applications.
- **Django:** Representing the Python language, Django is notable for its speed and its "batteries-included" philosophy, offering a wide array of built-in features.

- **Spring:** This robust Java framework is known for its flexibility and its capability to develop large-scale enterprise applications, providing a vast ecosystem of tools and extensions.
- **Nodejs:** Node.js, utilizing the Nest framework, allows for the development of efficient and scalable applications with JavaScript (or TypeScript), benefiting from Node.js's event-driven programming paradigm.

5.1.3 Evaluation Results

The evaluation focused on dimensions such as Scalability, Performance, Maintainability, Cost, Ecosystem and Compatibility and Easy of development, with each dimension being meticulously measured against the predefined metrics in chapter 4.

Ease of Development Analysis

Ease of Development evaluates the simplicity and expedience with which developers can employ the framework to create and maintain applications.

1. **Laravel**: is known for its ease of development, aiding developers to produce quality applications efficiently.

    - **Programming Language (PHP):**
        - **Score:** 0.5
        - **Analysis:** Although PHP is not the primary language for the team, its widespread use and simplicity may mitigate initial unfamiliarity. PHP's design facilitates quick development cycles.
    - **Maturity:**
        - **Score:** 1
        - **Analysis:** Laravel's established presence and active community reflect its stability and reliability as a web development framework.
        - **References:** [25].
    - **Code Generators and Scaffolding (Artisan):**

  - **Score:** 1
  - **Analysis:** The Artisan command-line tool enhances developer productivity through efficient code generation and scaffolding.
  - **References:** [29].

- **Documentation Quality:**
  - **Score:** 1
  - **Analysis:** Laravel's documentation is comprehensive and organized, ensuring accessible information for effective framework usage.
  - **References:** [48].

- **Community Support:**
  - **Score:** 1
  - **Analysis:** The Laravel community provides extensive support through various platforms, fostering a collaborative problem-solving environment.
  - **References:** [49], [50].

- **Learning Curve:**
  - **Score:** 1
  - **Analysis:** Laravel's design and abundant resources lower the learning curve, making it accessible for developers to quickly become proficient.
  - **References:** [49].

2. **Django** Django is acclaimed for its developer-friendly environment and adherence to the ”Don't Repeat Yourself” principle, streamlining the development process.

   - **Programming Language (Python):**
     - **Score:** 0.5
     - **Analysis:** The team's familiarity with Python, although not specifically for backend development, is beneficial due to Python's readability and simplicity.

   - **Maturity:**

  - **Score:** 1
  - **Analysis:** Django's well-established status and extensive community support highlight its reliability and robustness for web development.
  - **Reference:** [32]

- **Code Generators and Scaffolding:**
  - **Score:** 1
  - **Analysis:** Django offers efficient command-line tools for code generation, enhancing development speed and efficiency.
  - **Reference:** [32]

- **Documentation Quality:**
  - **Score:** 1
  - **Analysis:** The framework's documentation is noted for its excellence, comprehensiveness, and organization, making it an invaluable resource.
  - **Reference:** [32]

- **Community Support:**
  - **Score:** 1
  - **Analysis:** Django's large and active community provides an extensive support network and abundant resources for developers.
  - **Reference:** [32]

- **Learning Curve:**
  - **Score:** 0.5
  - **Analysis:** The comprehensive feature set of Django results in a moderate learning curve, but the abundance of learning resources available mitigates this challenge.
  - **Reference:** [32]

3. **Spring** Spring framework is recognized for its extensive suite of features and tools that aim to facilitate the development process, albeit with certain considerations regarding its learning curve and documentation.

- **Programming Language (Java):**
  - **Score:** 0
  - **Analysis:** Java's moderate familiarity within the team and its not being the language of choice could pose challenges in leveraging Spring to its full potential.
- **Maturity:**
  - **Score:** 1
  - **Analysis:** Spring's high maturity and well-established presence in the enterprise development landscape are indicative of its reliability and comprehensive community support.
  - **Reference:** [51]
- **Code Generators and Scaffolding:**
  - **Score:** 0.5
  - **Analysis:** While Spring offers tools for code generation and scaffolding, they are not as extensive or intuitive as those provided by some other frameworks.
  - **Reference:** [51]
- **Documentation Quality:**
  - **Score:** 0.5
  - **Analysis:** The documentation, though comprehensive, is often considered overwhelming for newcomers, impacting the accessibility of the framework.
  - **Reference:** [51]
- **Community Support:**
  - **Score:** 1
  - **Analysis:** Spring benefits from very strong community support, offering a wealth of resources, forums, and discussion groups to aid developers.
  - **Reference:** [51]
- **Learning Curve:**

  - **Score:** 0.5
  - **Analysis:** The framework's broad array of features and configurations introduces a steep learning curve, although the abundance of educational resources available may help mitigate this challenge.
  - **Reference:** [51]

4. **Node.js** Node.js stands out for its versatility and developer-friendly environment, leveraging the widely known and favored JavaScript.

   - **Programming Language (JavaScript):**
     - **Score:** 1
     - **Analysis:** JavaScript's familiarity and popularity among the team members significantly contribute to Node.js's accessibility and ease of adoption.
   - **Maturity:**
     - **Score:** 0.5
     - **Analysis:** Node.js exhibits a mature yet rapidly evolving ecosystem, providing a balance between stability and innovation.
     - **Reference:** [38]
   - **Code Generators and Scaffolding:**
     - **Score:** 1
     - **Analysis:** The availability of efficient tools for code generation and scaffolding enhances developer productivity and facilitates rapid application development.
     - **Reference:** [52]
   - **Documentation Quality:**
     - **Score:** 1
     - **Analysis:** Node.js's documentation is noted for its clarity, conciseness, and organization, making it an invaluable resource for developers.
     - **Reference:** [38]

- **Community Support:**
    - **Score:** 1
    - **Analysis:** The Node.js community is known for its increasing support, offering a plethora of resources, forums, and discussion platforms.
    - **Reference:** [38]
- **Learning Curve:**
    - **Score:** 1
    - **Analysis:** Node.js provides an intuitive and straightforward learning path, significantly lowering the barrier to entry for new developers.
    - **Reference:** [38]

Scalability Analysis

Scalability is a crucial property to evaluate, especially for applications anticipated to experience varied or growing user traffic over time. The frameworks chosen for analysis are Laravel, Django, Spring, and Node.js with Nest.

1. **Laravel** exhibits a blend of features that cater to both horizontal and vertical scalability, with additional strengths in load balancing and microservices architecture support.

    - **Horizontal Scalability:**
        - **Score:** 0.5
        - **Analysis:** While Laravel supports horizontal scalability, achieving it may necessitate manual configuration efforts to optimize performance across multiple instances.
        - **Reference:** [48]
    - **Vertical Scalability:**
        - **Score:** 0.5
        - **Analysis:** Laravel facilitates vertical scaling, though this might require adjustments in the application to fully leverage increased server resources.

    - **Reference:** [48]

- **Load Balancing:**
    - **Score:** 1
    - **Analysis:** The framework integrates seamlessly with external load balancers, enhancing the distribution of incoming network traffic across multiple servers.
    - **Reference:** [53]
- **Microservices Architecture Support:**
    - **Score:** 1
    - **Analysis:** Laravel offers comprehensive support for developing applications following the microservices architecture, facilitating modular and scalable application development.
    - **Reference:** [48]
- **State Management:**
    - **Score:** 0.5
    - **Analysis:** State management in Laravel is feasible, yet it may rely on the integration of third-party libraries for optimal handling.
    - **Reference:** [48]

2. **Django** is celebrated for its comprehensive scalability features, underpinned by its ”batteries-included” philosophy.

    - **Horizontal Scalability:**
        - **Score:** 1
        - **Analysis:** Django facilitates effortless horizontal scaling, requiring minimal configuration to adapt to increased traffic by distributing the load across multiple servers.
        - **Reference:** [54]
    - **Vertical Scalability:**

  - **Score:** 1
  - **Analysis:** It supports seamless vertical scalability, enabling efficient use of additional server resources without the need for application modifications.
  - **Reference:** [54]

- **Load Balancing:**
  - **Score:** 1
  - **Analysis:** Django includes built-in load balancing features, optimizing the distribution of requests to ensure high availability and reliability.
  - **Reference:** [54]

- **Microservices Architecture Support:**
  - **Score:** 1
  - **Analysis:** The framework provides robust support for the microservices architecture, facilitating the development of scalable and modular applications.
  - **Reference:** [54]

- **State Management:**
  - **Score:** 1
  - **Analysis:** Django comes with powerful, built-in state management capabilities, ensuring consistent application states across distributed environments.
  - **Reference:** [54]

3. **Spring** is distinguished by its convention-over-configuration philosophy and exemplary support for microservices, making it a powerful choice for scalable application development.

   - **Horizontal Scalability:**
     - **Score:** 0.5
     - **Analysis:** Spring supports horizontal scalability, facilitating the distribution of load across multiple servers, although achieving optimal configuration may necessitate specific adjustments.

    - **Reference:** [55]

- **Vertical Scalability:**
    - **Score:** 1
    - **Analysis:** The framework enables applications to scale vertically with ease, leveraging additional resources without necessitating changes to the application itself.
    - **Reference:** [46]
- **Load Balancing:**
    - **Score:** 1
    - **Analysis:** Spring includes built-in load balancing features, optimizing resource utilization and enhancing application responsiveness.
    - **Reference:** [56]
- **Microservices Architecture Support:**
    - **Score:** 1
    - **Analysis:** With Spring Cloud, Spring offers robust support for microservices architecture, allowing for the development of highly scalable and flexible applications.
    - **Reference:** [57]
- **State Management:**
    - **Score:** 1
    - **Analysis:** Spring provides multiple effective strategies for application state management, ensuring data consistency across distributed components.
    - **Reference:** [46]

4. **Node.js** excels in providing a flexible architecture conducive to creating scalable, testable, and maintainable applications.

    - **Horizontal Scalability:**
        - **Score:** 1

  - **Analysis:** Node.js enables effortless horizontal scaling, allowing applications to handle increased loads by adding more instances with minimal or no configuration required.
  - **Reference:** [43]

- **Vertical Scalability:**
  - **Score:** 1
  - **Analysis:** It efficiently leverages additional server resources to scale vertically, enhancing performance without necessitating changes to the application code.
  - **Reference:** [43]

- **Load Balancing:**
  - **Score:** 0.5
  - **Analysis:** While Node.js applications can integrate with external load balancers, this setup may require additional configuration for optimal distribution of traffic.
  - **Reference:** [43]

- **Microservices Architecture Support:**
  - **Score:** 1
  - **Analysis:** Node.js is inherently suited for microservices architecture, supporting the development of scalable, loosely coupled services.
  - **Reference:** [43]

- **State Management:**
  - **Score:** 0.5
  - **Analysis:** Effective state management in Node.js often relies on integrating third-party solutions to maintain application state across distributed systems.
  - **Reference:** [43]

Performance Analysis

Performance of a framework is pivotal as it directly impacts user experience and resource utilization. Efficient frameworks ensure swift responses, even under substantial load, guaranteeing user satisfaction and optimal resource allocation.

1. **Laravel** is celebrated for its ability to deliver efficient and scalable solutions, marked by its performance metrics.

    - **Latency:**
        - **Score:** 0.5
        - **Analysis:** Laravel generally exhibits response times between 100 ms and 300 ms, balancing speed and resource efficiency.
        - **Reference:** [58]

    c

    - **Throughput:**
        - **Score:** 1
        - **Analysis:** It is capable of handling more than 1000 requests per second, demonstrating high throughput under load.
        - **Reference:** [59]
    - **CPU Usage:**
        - **Score:** 1
        - **Analysis:** Laravel's optimized performance results in CPU usage typically less than 30
        - **Reference:** [60]
    - **Memory Usage:**
        - **Score:** 0.5
        - **Analysis:** Memory utilization often ranges between 70% and 85%, indicating a moderate footprint.
        - **Reference:** [60]

2. **Django** stands out for its performance optimizations and capabilities in supporting rapid application development.

   - **Latency:**
     - **Score:** 1
     - **Analysis:** Django is highly efficient, boasting response times of less than 100 ms, which significantly enhances user experience by providing swift feedback.
     - **Reference:** [61]
   - **Throughput:**
     - **Score:** 0.5
     - **Analysis:** It processes between 500 and 1000 requests per second, demonstrating a balanced throughput capacity suitable for moderate to high traffic applications.
     - **Reference:** [61]
   - **CPU Usage:**
     - **Score:** 0.5
     - **Analysis:** Django's CPU usage lies between 35% and 40%, indicating a moderate level of resource consumption that allows for additional processes to run efficiently.
     - **Reference:** [62]
   - **Memory Usage:**
     - **Score:** 1
     - **Analysis:** Memory efficiency is a key strength, with typical usage under 70% of available resources, ensuring that applications remain lightweight and performant.
     - **Reference:** [63]

3. **Spring** is renowned for its capability to develop stand-alone, production-grade applications, showcasing exceptional performance metrics as follows.

- **Latency:**
  - **Score:** 1
  - **Analysis:** Spring applications exhibit rapid response times, often less than 100 ms, ensuring quick interactions for end-users.
  - **Reference:** [64]
- **Throughput:**
  - **Score:** 1
  - **Analysis:** Demonstrates the ability to efficiently manage more than 1000 requests per second, highlighting its high throughput capacity.
  - **Reference:** [64]
- **CPU Usage:**
  - **Score:** 0.5
  - **Analysis:** Spring's CPU utilization ranges between 40% and 60%, indicating a moderate level of efficiency.
  - **Reference:** [51]
- **Memory Usage:**
  - **Score:** 1
  - **Analysis:** It excels in memory management, with utilization typically remaining below 70%, contributing to its performance efficiency.
  - **Reference:** [51]

4. **Node.js** is efficient and versatile, designed for building scalable, maintainable, and testable applications.

   - **Latency:**
     - **Score:** 0.5
     - **Analysis:** Node.js typically exhibits response times ranging between 100 ms and 300 ms, reflecting a balance between speed and processing efficiency.
     - **Reference:** [65]

- **Throughput:**
  - **Score:** 1
  - **Analysis:** Demonstrates a high capacity for managing workload, efficiently handling more than 1000 requests per second.
  - **Reference:** [65]
- **CPU Usage:**
  - **Score:** 1
  - **Analysis:** Node.js is optimized for CPU resource management, maintaining usage below 30%, which contributes to its overall performance efficiency.
  - **Reference:** [66]
- **Memory Usage:**
  - **Score:** 0.5
  - **Analysis:** Memory consumption typically falls between 70% and 85%, indicating moderate efficiency in memory utilization.
  - **Reference:** [66]

Maintainability Analysis

Maintainability is a critical dimension, reflecting the ease with which a framework can accommodate modifications, enhancements, and debugging. High maintainability ensures that developers can efficiently comprehend, update, and augment the codebase to adapt to evolving requirements or address issues.

1. **Laravel** excels in providing clean, readable, and highly maintainable code solutions.

- **Code Complexity:**
  - **Score:** 1
  - **Analysis:** Laravel boasts a low-complexity, well-structured codebase, facilitating ease of understanding and modification.
  - **Reference:** [25]
- **Modularity:**

  - **Score:** 1
  - **Analysis:** Its highly modular design promotes efficient and independent enhancements, allowing for the seamless integration of new features or services.
  - **Reference:** [48]

- **Testability:**
  - **Score:** 1
  - **Analysis:** Laravel provides extensive support for a variety of testing tools and methodologies, ensuring robustness and reliability through comprehensive test coverage.
  - **Reference:** [48]

- **Long-Term Support:**
  - **Score:** 1
  - **Analysis:** The framework guarantees ongoing support, regular updates, and security patches, contributing to its stability and reliability over time.
  - **Reference:** [25]

2. **Django** with its ”batteries-included” philosophy, provides highly maintainable and coherent code structures.

   - **Code Complexity:**
     - **Score:** 0.5
     - **Analysis:** While Django presents moderate complexity in certain aspects, it compensates with comprehensive documentation that aids in navigating and understanding its intricacies.
     - **Reference:** [54]

   - **Modularity:**
     - **Score:** 1

  - **Analysis:** The framework is inherently designed for high modularity, encouraging the use of reusable components and fostering a scalable application structure.
  - **Reference:** [54]
- **Testability:**
  - **Score:** 1
  - **Analysis:** Django provides a robust testing framework that facilitates various levels of testing, from unit to integration tests, ensuring the development of reliable applications.
  - **Reference:** [54]
- **Long-Term Support:**
  - **Score:** 1
  - **Analysis:** With its commitment to long-term support, Django regularly releases updates and maintains extensive documentation, which underpins a stable and evolving ecosystem.
  - **Reference:** [54]

3. **Spring** provides a comprehensive and well-structured solution for building maintainable enterprise applications.

- **Code Complexity:**
  - **Score:** 1
  - **Analysis:** Spring is designed to minimize code complexity, enhancing readability and simplifying maintenance efforts.
  - **Reference:** [46]
- **Modularity:**
  - **Score:** 1
  - **Analysis:** It champions modular development, allowing for the creation of independent, interchangeable modules that facilitate scalable and flexible application architectures.

- **Reference:** [46]

- **Testability:**
  - **Score:** 0.5
  - **Analysis:** While Spring supports a wide range of testing methodologies, integrating with certain tools and configurations may necessitate additional setup.
  - **Reference:** [46]
- **Long-Term Support:**
  - **Score:** 1
  - **Analysis:** Spring provides robust long-term support, ensuring regular updates and comprehensive documentation to support its community.
  - **Reference:** [46]

4. **Node.js** focuses on modular, scalable development, providing maintainable code structures and extensive testing capabilities.

   - **Code Complexity:**
     - **Score:** 0.5
     - **Analysis:** Node.js features moderate complexity in certain areas, necessitating a deeper understanding, yet it is balanced by extensive documentation and community resources.
     - **Reference:** [38]
   - **Modularity:**
     - **Score:** 1
     - **Analysis:** It adopts a highly modular design philosophy, promoting scalable and flexible development practices through reusable and independently manageable components.
     - **Reference:** [38]
   - **Testability:**

- **Score:** 1
- **Analysis:** Node.js integrates seamlessly with a variety of robust testing tools, accommodating diverse testing methodologies to ensure comprehensive validation of applications.
- **Reference:** [38]

- **Long-Term Support:**
  - **Score:** 1
  - **Analysis:** Demonstrates a strong commitment to long-term support, evidenced by consistent updates and the provision of security patches, ensuring the framework remains current and secure.
  - **Reference:** [38]

Cost Analysis

Cost is a pivotal dimension as it assesses the economic feasibility and impact of adopting a specific framework. It is crucial for organizations to evaluate the cost dimension meticulously to ensure sustainable and efficient deployment of resources.

1. **Laravel** being open-source and providing extensive resources, ensures cost-effectiveness throughout the development lifecycle.

   - **Licensing:**
     - **Score:** 1
     - **Analysis:** Laravel is an open-source framework with no licensing cost, making it highly cost-effective for development projects.
     - **Reference:** [25]
   - **Server Resources:**
     - **Score:** 1
     - **Analysis:** Laravel has minimal server resource requirements and offers optimal performance, ensuring low operational costs.
     - **Reference:** [48]

- **Migration Cost:**
  - **Score:** 0.5
  - **Analysis:** Migrating to Laravel may require moderate effort, depending on the complexity and scale of the existing system, leading to potential migration costs.
  - **Reference:** [48]
- **Learning Cost:**
  - **Score:** 1
  - **Analysis:** Laravel has a quick learning curve due to extensive documentation and strong community support, reducing the learning cost for development teams.
  - **Reference:** [48]

2. **Django** offers a comprehensive framework with a focus on minimizing the cost associated with the development and maintenance of web applications.

- **Licensing:**
  - **Score:** 1
  - **Analysis:** Django is available under an open-source license, eliminating licensing costs for development projects.
  - **Reference:** [54]
- **Server Resources:**
  - **Score:** 0.5
  - **Analysis:** Django has moderate server resource requirements, and associated costs depend on the scale and complexity of the application.
  - **Reference:** [32]
- **Migration Cost:**
  - **Score:** 1
  - **Analysis:** Django provides tools and documentation for smooth migration, reducing potential migration costs.

  - **Reference:** [32]

- **Learning Cost:**
  - **Score:** 1
  - **Analysis:** Django offers a gentle learning curve with extensive resources and tutorials, minimizing the learning cost for development teams.
  - **Reference:** [32]

3. **Spring** , with its comprehensive ecosystem, may involve varying costs based on the chosen modules and deployment scale.

- **Licensing:**
  - **Score:** 1
  - **Analysis:** Spring is open-source with no additional licensing cost, contributing to cost-effectiveness.
  - **Reference:** [51]
- **Server Resources:**
  - **Score:** 0
  - **Analysis:** Spring typically requires higher server resources and costs due to its extensive feature set and capabilities.
  - **Reference:** [46]
- **Migration Cost:**
  - **Score:** 0.5
  - **Analysis:** Migration costs for Spring can vary depending on the existing system and the complexity of the migration process.
  - **Reference:** [46]
- **Learning Cost:**
  - **Score:** 0.5
  - **Analysis:** Spring demands a moderate amount of time to master due to its extensive capabilities and diverse modules.

    - **Reference:** [46]

4. **Node.js**, optimizes performance and resources, ensuring cost-effective development.

   - **Licensing:**
     - **Score:** 1
     - **Analysis:** Node.js is open-source and free to use, contributing to cost-effectiveness.
     - **Reference:** [38]
   - **Server Resources:**
     - **Score:** 1
     - **Analysis:** Node.js is optimized for performance with minimal resource requirements, reducing operational costs.
     - **Reference:** [43]
   - **Migration Cost:**
     - **Score:** 1
     - **Analysis:** Node.js offers smooth migration paths with extensive support and documentation, minimizing migration costs.
     - **Reference:** [43]
   - **Learning Cost:**
     - **Score:** 0.5
     - **Analysis:** Node.js requires a moderate learning curve, especially for developers new to TypeScript, which may slightly impact initial development costs.
     - **Reference:** [43]

Ecosystem and Compatibility Analysis

The Ecosystem and Compatibility dimension evaluates the adaptability and integration capability of the framework within the existing technology ecosystem.

1. **Laravel** boasts a rich ecosystem and ensures high compatibility with various technologies and tools.
    - **Libraries and Tools:**
        - **Score:** 1
        - **Analysis:** Laravel offers an extensive array of libraries and tools that are easily integrable, enhancing development capabilities.
        - **Reference:** [48]
    - **Front-end Compatibility:**
        - **Score:** 1
        - **Analysis:** Laravel provides smooth integration with popular front-end frameworks and libraries, facilitating comprehensive web application development.
        - **Reference:** [48]
    - **APIs and Connectivity:**
        - **Score:** 1
        - **Analysis:** Laravel has excellent support for connecting with external services via APIs, enhancing application functionality and versatility.
        - **Reference:** [48]
    - **Updates and Maintenance:**
        - **Score:** 1
        - **Analysis:** Laravel receives regular and high-quality updates, ensuring ongoing maintenance support and compatibility with evolving technologies.
        - **Reference:** [48]
2. **Django** excels in providing a diverse ecosystem and compatibility with various technologies.
    - **Libraries and Tools:**
        - **Score:** 1

  - **Analysis:** Django offers a robust set of libraries and tools available for integration, enhancing development capabilities.
  - **Reference:** [32]

- **Front-end Compatibility:**
  - **Score:** 0.5
  - **Analysis:** Django has moderate ease in integrating with some front-end frameworks, which may require additional configuration in some cases.
  - **Reference:** [32]

- **APIs and Connectivity:**
  - **Score:** 1
  - **Analysis:** Django provides efficient connectivity to external services through well-documented APIs, enhancing application versatility.
  - **Reference:** [32]

- **Updates and Maintenance:**
  - **Score:** 1
  - **Analysis:** Django ensures consistent delivery of quality updates and effective maintenance, ensuring long-term stability.
  - **Reference:** [32]

3. **Spring** provides a comprehensive ecosystem and high-level compatibility, with a focus on enterprise-level services.

- **Libraries and Tools:**
  - **Score:** 0.5
  - **Analysis:** Spring offers a good variety of libraries and tools, although some libraries may be complex to integrate, impacting ease of development.
  - **Reference:** [51]

- **Front-end Compatibility:**
  - **Score:** 0.5

  - **Analysis:** Spring is compatible with various front-end technologies but may require additional configurations, which can add complexity to the development process.
  - **Reference:** [51]
- **APIs and Connectivity:**
  - **Score:** 1
  - **Analysis:** Spring offers easy connectivity with a range of external services via APIs, enhancing application integration capabilities.
  - **Reference:** [51]
- **Updates and Maintenance:**
  - **Score:** 0.5
  - **Analysis:** Spring provides regular updates, but the quality and maintenance can sometimes be inconsistent, leading to potential challenges in long-term support.
  - **Reference:** [51]

4. **Node.js**, leveraging Node.js, offers a flexible ecosystem and a high degree of compatibility.

- **Libraries and Tools:**
  - **Score:** 1
  - **Analysis:** Node.js provides a wide range of easily integrable libraries and tools, enhancing development capabilities.
  - **Reference:** [38]
- **Front-end Compatibility:**
  - **Score:** 1
  - **Analysis:** Node.js offers effortless integration with numerous front-end frameworks and libraries, ensuring seamless development experiences.
  - **Reference:** [38]

- **APIs and Connectivity:**
  - **Score:** 1
  - **Analysis:** Node.js provides superior support for API integrations with external services, facilitating robust connectivity solutions.
  - **Reference:** [38]
- **Updates and Maintenance:**
  - **Score:** 1
  - **Analysis:** Node.js receives frequent, high-quality updates and excellent maintenance support, ensuring reliability and security.
  - **Reference:** [38]

5.1.4 Select Framework

Based on the results obtained from documentation, benchmarks, blogs and websites, we will apply formula **3.1** to obtain the calculation of each dimension, to the final rating we apply formula **3.2**. Since we have six dimensions and considering ease of development to be the most crucial aspect of this project, we have assigned a weight of 0.25 to this dimension. For the other five dimensions, we have selected a weight of 0.15 each. This gives us a cumulative total of 1 in the sum of the final weights.

1. **Laravel**

   - **Scalability:** $D_{1.1} = \frac{0.5+0.5+1+1+0.5}{5} = 0.7$
   - **Perfomance:** $D_{1.2} = \frac{0.5+1+1+0.5}{4} = 0.75$
   - **Maintanability:** $D_{1.3} = \frac{1+1+1+1}{4} = 1$
   - **Cost:** $D_{1.4} = \frac{1+1+0.5+1}{4} = 0.875$
   - **Ecosystem and Compatibility:** $D_{1.5} = \frac{1+1+1+1}{4} = 1$
   - **Ease of Development:** $D_{1.6} = \frac{0.5+1+1+1+1+1}{6} = 0.9166$

   Final Rating: $F_1 = \frac{0.7\times0.15+0.75\times0.15+1\times0.15+0.875\times0.15+1\times0.15+0.9166\times0.25}{6} = 0.8779$

2. **Django**

- **Scalability:** $D_{2.1} = \frac{1+1+1+1+1}{5} = 1$
- **Perfomance:** $D_{2.2} = \frac{1+0.5+0.5+1}{4} = 0.75$
- **Maintanability:** $D_{2.3} = \frac{0.5+1+1+1}{4} = 0.875$
- **Cost:** $D_{2.4} = \frac{1+0.5+1+1}{4} = 0.875$
- **Ecosystem and Compatibility:** $D_{2.5} = \frac{1+0.5+1+1}{4} = 0.875$
- **Ease of Development:** $D_{2.6} = \frac{0.5+1+1+1+1+0.5}{6} = 0.833$

Final Rating: $F_2 = \frac{1\times0.15+0.75\times0.15+0.875\times0.15+0.875\times0.15+0.875\times0.15+0.833\times0.25}{6} = 0.8645$

3. **Spring**

- **Scalability:** $D_{3.1} = \frac{0.5+1+1+1+1}{5} = 0.9$
- **Perfomance:** $D_{3.2} = \frac{1+1+1+0.5}{4} = 0.875$
- **Maintanability:** $D_{3.3} = \frac{1+1+0.5+1}{4} = 0.875$
- **Cost:** $D_{3.4} = \frac{1+0+0.5+0.5}{4} = 0.5$
- **Ecosystem and Compatibility:** $D_{3.5} = \frac{0.5+0.5+1+0.5}{4} = 0.625$
- **Ease of Development:** $D_{3.6} = \frac{0+1+0.5+0.5+1+0.5}{6} = 0.5833$

Final Rating: $F_3 = \frac{0.9\times0.15+0.875\times0.15+0.875\times0.15+0.5\times0.15+0.625\times0.15+0.5833\times0.25}{6} = 0.712075$

4. **Nodejs**

- **Scalability:** $D_{4.1} = \frac{1+1+0.5+1+0.5}{5} = 0.8$
- **Perfomance:** $D_{4.2} = \frac{0.5+1+1+0.5}{4} = 0.75$
- **Maintanability:** $D_{4.3} = \frac{0.5+1+1+1}{4} = 0.875$
- **Cost:** $D_{4.4} = \frac{1+1+1+0.5}{4} = 0.875$
- **Ecosystem and Compatibility:** $D_{4.5} = \frac{1+1+1+1}{4} = 1$
- **Ease of Development:** $D_{4.6} = \frac{1+0.5+1+1+1+1}{6} = 0.9166$

Final Rating: $F_1 = \frac{0.8\times0.15+0.75\times0.15+0.875\times0.15+0.875\times0.15+1\times0.15+0.9166\times0.25}{6} = 0.8179$

Comparison Table

Table 5.1 show a comparative illustrating the results gleaned from the analysis of four back-end frameworks. This table delineates the conclusive rankings of each framework, assisting in the selection of a final candidate. The insights derived from this comparative analysis are pivotal in making an informed and optimal choice for our project needs.

Dimensions	**Laravel**	**Django**	**Spring**	**Nodejs**	**Weight**
Scalability	0.7	1	0.9	0.8	0.15
Performance	0.75	0.75	0.875	0.75	0.15
Maintainability	1	0.875	0.875	0.875	0.15
Cost	0.875	0.875	0.875	0.875	0.15
Ecosystem and Compatibility	1	0.875	0.675	1	0.15
Ease of Development	0.92	0.83	0.58	0.92	0.25
Final Raiting	**0.88**	**0.86**	**0.71**	**0.82**	

Table 5.1: Rating of Dimensions for each back-end seleted framework

Finally, after evaluating the obtained results among Laravel, Spring, Node, and Django, specifically for this project, we can discern that Laravel emerges as the most suitable choice, with Django not far behind. It is essential to remember that these results serve as a mere guideline for making decisions when initiating a new project. However, any among Django, Node, and Laravel can be considered a commendable option to commence development, as they all hold significant merit in the realm of back-end frameworks. Keep in mind that the ultimate choice should align with the specific requirements and goals of your project.

Chapter 6

Conclusions

This thesis embarked on the ambitious endeavor to address the prevalent challenges encountered by developers in selecting an apt back-end framework for web and mobile application projects. The overarching objective was to craft a meticulous and multifaceted methodology framework tailored to enable the quantitative measurement of assorted dimensions and metrics pivotal for appraising the accessibility and suitability of a back-end framework relative to its counterparts.

This methodology framework has been curated with the intention of providing a systematic and standardized approach, allowing for the nuanced evaluation of critical components such as Ease of Development, Security, Scalability, Performance, Popularity, Portability, Maintainability, Cost, Ecosystem and Compatibility. It is intended to be universally applicable to any back-end framework, ensuring it caters to evolving trends and needs in application development.

6.1 Contributions and Implications

The realizations of the specific objectives have led to the establishment of a framework that offers not just theoretical insights but practical implications in choosing back-end frameworks, ensuring objective and criteria-based evaluations and comparisons are made. It is crucial to note that the application of this methodology in the selection of back-end frameworks is meticulously designed with an aim to elevate software quality, decrease development costs and time, and foster the adoption of suitable technologies and architectural frameworks, tailored specifically for each project. This advancement is poised to bring about a transformative change in the field, enhancing overall development efficacy and output quality.

Based on the results procured from the application of this developed methodology framework, nuanced recommendations will be provided, serving as a robust aid in the decision-making processes for individual developers and enterprises alike. These recommendations are destined to be a rich reservoir of insights, assisting in the discerning selection of the most suitable back-end framework tailored to the unique requisites of specific projects. The practical implications of these recommendations are profound, offering invaluable guidance and contributing to informed and optimal decision-making in real-world developmental scenarios.

6.2 Future Adaptations and Refinement

In light of the advanced methodological framework developed in this research, significant future initiatives are envisioned to further expand and optimize the efficacy and applicability of the proposed method. A key project on the horizon is the conception and realization of a comprehensive web application that serves as a portal for accessing detailed information about various back-end frameworks.

This application aspires to offer an intuitive and highly structured visual interface, allowing users not only to access crucial data but also to experience and test the frameworks in an effective and simplified manner. Such a platform will democratize access to vital information, fostering a deeper understanding and more accurate evaluation of the various framework options available, thus facilitating more informed and tailored decisions for each project's specific needs.

The implementation of a web application will be a revolutionary step, providing an invaluable tool for developers and enterprises, and solidifying the proposed methodology as a standard in back-end framework selection. This initiative not only responds to the current needs of the industry but also anticipates and adapts to future trends and evolutions in the field of web and mobile application development, maintaining its relevance and contribution in a constantly changing technological environment.

Finally, the future application will be designed with a focus on user experience, ensuring that users can navigate and utilize the platform intuitively and effectively, thereby maximizing its benefit and impact on the development community. Subsequent developments and enhancements of this platform will be guided by emerging needs and user feedback, ensuring its coherent evolution and its continued resonance with the demands and expectations of the industry.

6.3 Final Reflections

This thesis stands as a navigational compass for developers in the complex journey of back-end framework selection, offering a methodology that is structured, analytical, resilient, and adaptable to the shifting sands of technological innovations. It illuminates pathways to optimal developmental approaches and improved outcomes, with its applicative and evolutionary nature promising to reshape developmental methodologies and uplift the standards of web and mobile application development in the times to come.

In closing, it is pivotal to emphasize that the insights and recommendations rendered by this methodology are to be perceived as guiding lights, not as rigid prescriptions, allowing for a spectrum of flexibility and discernment in alignment with individual project needs and developer inclinations. The continual enhancement and expansion of this methodology are anticipated to embrace and address the intricate and diverse demands of the dynamic realm of application development, ensuring its sustained relevance and utility.

Bibliography

[1] N. Balani, “Web application development – evolution and future,” *International Journal of Computer Science and Information Technologies*, vol. 6, no. 3, pp. 2951–2955, 2015.

[2] A. Paliwal and A. Sharma, “Comparison of web development technologies: Adobe flex and microsoft silverlight,” *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 4, pp. 872–875, 2013.

[3] S. Newman, *Building Microservices: Designing Fine-Grained Systems.* O’Reilly Media, Inc., 2015.

[4] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, “Microservices: yesterday, today, and tomorrow,” *Present and Ulterior Software Engineering*, vol. 32, pp. 95–118, 2017.

[5] N. Kaur and R. Rani, “Comparative analysis of web application development frameworks: Ruby on rails and django,” *International Journal of Advanced Research in Computer Science*, vol. 7, no. 3, pp. 25–28, 2016.

[6] S. Shah and V. Patel, “A comparative study of web application development frameworks: Laravel and django,” *International Journal of Engineering and Techniques*, vol. 4, no. 1, pp. 1–5, 2018.

[7] A. Chandra and P. Borah, “Comparative analysis of popular backend web development frameworks,” in *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET).* IEEE, 2017, pp. 1–6.

[8] S. K. Garg and R. Buyya, “Framework for the assessment of the environmental impact of cloud computing services,” *Information Systems Frontiers*, vol. 13, no. 4, pp. 525–540, 2011.

[9] T. Berners-Lee, “Information management: A proposal,” https://www.w3.org/History/1989/proposal.html, CERN, 1989.

[10] World Wide Web Consortium, “Making the web work,” 2021. [Online]. Available: https://www.w3.org/

[11] G. J. James, “Ajax: A new approach to web applications,” 2005. [Online]. Available: https://www.semanticscholar.org/paper/Ajax%3A-A-New-Approach-to-Web-Applications-Garrett/c440ae765ff19ddd3deda24a92ac39cef9570f1e

[12] E. Etemad and M. Zamanian, “Progressive web apps: A new level of web experience,” in *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*, 2017.

[13] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, “A survey of mobile malware in the wild,” https://doi.org/10.1145/2381934.2381936, 2012.

[14] A. Lekh, “Best backend frameworks review,” Jan 2021. [Online]. Available: https://impressit.io/blog/best-backend-frameworks

[15] M. Gurevich, D. Nosov, and N. Krichfalushiy, “Web application architecture: Types amp; components,” Nov 2021. [Online]. Available: https://www.azoft.com/blog/web-application-architecture/

[16] D. Comunicación, “¿qué es backend y frontend?” Sep 2019. [Online]. Available: https://descubrecomunicacion.com/que-es-backend-y-frontend/

[17] S. Team, “The best node.js framework: Koa vs express vs hapi - detailed comparison,” *Savvycom Software*, 2021. [Online]. Available: https://medium.com/savvycom/the-best-node-js-framework-koa-vs-express-vs-hapi-detailed-comparison-46cc85207d65

[18] ValueCoders. (2021) Codeigniter vs cakephp vs yii vs laravel: Which php framework is right for you? [Online]. Available: https://www.valuecoders.com/blog/technology-and-apps/codeigniter-vs-cakephp-vs-yii-vs-laravel/

[19] J. Wheeler, "Django vs. flask vs. pyramid: Choosing a python web framework," 2016, [Online; accessed 4-April-2023]. [Online]. Available: https://medium.com/featurepreneur/flask-django-or-pyramid-choose-the-right-python-framework-for-your-project-3a9dd23ee51c

[20] M. Kaluža, M. Kalanj, and B. Vukelić, "Comparison of back-end frameworks for web application development," *Zbornik Veleučilišta u Rijeci*, vol. 7, no. 1, pp. 317–332, 2019.

[21] PrimerPy, "Which python web framework to choose for 2023: Django, flask, fastapi, or tornado?" 2023. [Online]. Available: https://primerpy.medium.com/which-python-web-framework-to-choose-for-2023-django-flask-fastapi-or-tornado-9d05860adfe3

[22] Monocubed, "Web development framework comparison: Django, flask, ruby on rails, express, and laravel," 2023. [Online]. Available: https://www.monocubed.com/blog/web-development-framework-comparison/

[23] Statistics and Data, "Most popular backend frameworks 2012-2023," 2023, accessed on March 5, 2023. [Online]. Available: https://statisticsanddata.org/data/most-popular-backend-frameworks-2012-2023/

[24] J. Aman, "Laravel design pattern: Simplifying web development," 2023. [Online]. Available: https://amanj0314.medium.com/laravel-design-pattern-simplifying-web-development-d84115bba19d

[25] Laravel, "Laravel - the php framework for web artisans," Laravel Sofware Foundation, 2011, accessed on March 5, 2023. [Online]. Available: https://laravel.com/

[26] DreamzTech, "Why laravel is the most popular php framework in 2023," *DreamzTech Blog*, 2023. [Online]. Available: https://blog.dreamztech.com/why-laravel-is-the-most-popular-php-framework-in-2023/

[27] Laravel, “Laravel,” https://packagist.org/packages/laravel/, Acceso en línea, consultado el 15 de mayo de 2023.

[28] Packagist, “Laravel packages,” 2023, accessed: November 2023. [Online]. Available: https://repo.packagist.org/packages/laravel/

[29] Laravel. (Accessed: 2023) Laravel documentation - artisan console. [Online]. Available: https://laravel.com/docs/artisan

[30] L. Blade. (Accessed: 2023) Laravel documentation - blade templates. [Online]. Available: https://laravel.com/docs/blade

[31] InterviewBit. (2021) Django architecture explained. [Online]. Available: https://www.interviewbit.com/blog/django-architecture/

[32] D. S. Foundation, *Django Documentation*, Django Software Foundation, 2023. [Online]. Available: https://docs.djangoproject.com/

[33] JetBrains, “Python developers survey 2021 results,” 2021, accessed: November 14, 2023. [Online]. Available: https://lp.jetbrains.com/python-developers-survey-2021/

[34] JetBrains, “JetBrains Python Developers Survey 2021,” https://lp.jetbrains.com/python-developers-survey-2021/, Acceso en línea, consultado el 15 de mayo de 2023.

[35] Django, “Django GitHub Repository,” https://github.com/django/django, Acceso en línea, consultado el 15 de mayo de 2023.

[36] D. S. Foundation, *Django Documentation*, Django Software Foundation, 2023. [Online]. Available: https://www.djangoproject.com/community/

[37] R. Dahl, “A history of node.js,” *Built In Node*, May 2019. [Online]. Available: https://builtinnode.com/2019/05/04/a-history-of-node-js/

[38] N. S. Foundation, *Node*, Node Sofware Foundation, 2009, accessed: March 6, 2023. [Online]. Available: https://nodejs.org/

[39] D. Haynes, "Node.js architecture and 12 best practices for node.js development," *Scout APM*, 2021. [Online]. Available: https://scoutapm.com/blog/nodejs-architecture-and-12-best-practices-for-nodejs-development

[40] Node.js, "Releases," https://nodejs.org/en/about/releases/, 2022, accessed: March 6, 2023.

[41] Node.js security. Node.js Foundation. [Online]. Available: https://nodejs.org/en/docs/guides/security-best-practices/

[42] RisingStack. Node.js security best practices. [Online]. Available: https://blog.risingstack.com/node-js-security-best-practices/

[43] N. S. Foundation, *Node Documentation*, Node Sofware Foundation, 2009, accessed: March 6, 2023. [Online]. Available: https://nodejs.org/docs/latest/api/

[44] Twilio. Logging in node.js with winston and bunyan. [Online]. Available: https://www.twilio.com/blog/guide-node-js-logging

[45] Spring Boot. Spring boot documentation. Spring Sofware Foundation. [Online]. Available: https://docs.spring.io/spring-boot/index.html

[46] S. Team, *Spring Framework Documentation*, Pivotal Software, Inc., 2002–present. [Online]. Available: https://docs.spring.io/spring-framework/docs/current/reference/html/

[47] Spring Security. Spring security. [Online]. Available: https://docs.spring.io/spring-security/site/docs/5.4.2/reference/html5/

[48] L. Documentation, "Laravel documentation," Laravel Sofware Foundation, 2011, accessed on March 5, 2023. [Online]. Available: https://laravel.com/docs/

[49] "Laracasts - the laravel learning platform," https://laracasts.com.

[50] L. News. (Accessed: 2023) Getting started with laravel artisan. [Online]. Available: https://laravel-news.com/category/tutorials

[51] Spring, "Spring project official website." [Online]. Available: https://spring.io/

[52] "Npm and node.js tools," https://www.npmjs.com/, accessed: 2024-01-05.

[53] L. Framework, "Integrating laravel with load balancers," https://laravel-news.com/.

[54] D. Official, "Django project official website." [Online]. Available: https://www.djangoproject.com/

[55] "Spring framework horizontal scalability," https://spring.io/projects/spring-cloud, accessed: 2024-02-05.

[56] "Spring cloud load balancing," https://spring.io/guides/gs/spring-cloud-loadbalancer/, accessed: 2024-02-05.

[57] "Microservices with spring cloud," https://spring.io/microservices, accessed: 2024-02-05.

[58] L. Community, "Laravel speed," 2023. [Online]. Available: https://laracasts.com/discuss/channels/testing/laravel-speed

[59] "Scaling laravel applications for high throughput," https://laravel-news.com/category/tutorials.

[60] "Optimizing laravel cpu usage," https://laracasts.com/, accessed: 2024-02-05.

[61] "Optimizing django response times," https://docs.djangoproject.com/en/stable/topics/performance/, accessed: 2024-02-05.

[62] "Django (moderately) high cpu usage at idle," https://code.djangoproject.com/ticket/30372, accessed: 2024-02-06.

[63] "Reducing memory usage in django," https://www.guguweb.com/2020/03/27/optimize-django-memory-usage/, accessed: 2024-02-06.

[64] I. Anghel, "Latency vs throughput in spring boot applications," accessed: 2024-02-07. [Online]. Available: https://softwareg.com.au/blogs/computer-hardware/spring-boot-high-cpu-usages

[65] E. Genedy, “Latency and throughput in node.js,” accessed: 2024-03-06. [Online]. Available: https://egenedy.hashnode.dev/latency-and-throughput-in-nodejs

[66] Node.js. Node.js official documentation. Accessed: 2024-03-22. [Online]. Available: https://nodejs.org/en/docs/