LAB University of Applied Sciences

DEVELOPING BACK-END OF A WEB APPLICATION WITH NESTJS FRAMEWORK

Case: Integrify Oy’s student management system

LAB UNIVERSITY OF APPLIED SCIENCES LTD
Bachelor of Business Administration
Degree Programme in Business Information Technology
Autumn 2020
Anh Duc Pham

Abstract

Author(s) Pham, Anh Duc	Type of publication Bachelor's thesis	Published Autumn 2020
	Number of pages 46	

Title of publication

Developing back-end of a web application with NestJS framework
Case: Integrify Oy's student management system

Name of Degree

Bachelor of Business Administration

Abstract

Nowadays, it is essential for software development companies to find an appropriate backend framework for their projects since there are a wide variety of frameworks on the market right now.

The objective of this thesis is to determine the benefits and the drawbacks when applying NestJS - a Node.js backend framework to the case company's backend project.

The study's theoretical section introduced the background knowledge of related technical issues such as web application development, backend development, JavaScript, Node.js, NestJS.

Furthermore, this thesis presents a literature review section that reviewed data from other researchers or sources from books or online articles said about the pros and cons of NestJS. This data then acted as the interview structure guideline for the interview with the case company's experienced developers that worked with NestJS. The final result is the comparison and analysis of data between reported information from existing findings and the real-life data from the interviews with the developers.

The conclusion reveals that the advantages of NestJS are being an opinionated framework, being written in TypeScript, having solid Angular-like architecture, offering a useful command-line interface, high scalability, and well-designed automated testing. However, the framework has some limitations in its documentation, community support, and learning curve.

Keywords

NestJS, Node.js, JavaScript, web application development, backend development.

CONTENTS

1 INTRODUCTION

Since 1999 when the first concept of “web application” was introduced, the use of web applications has increased to a vast extent and many Internet users had used one before without even realizing it. Today, millions of people use applications via the web every day to carry out electronic banking tasks, file income taxes online, share pictures or videos on social media, communicate with others, etc. For example, Google Docs is a text management web application that stores data online and allows users to "download" the files onto the personal computer. (Nations 2020). Also, Netflix is the most popular streaming video platform in the world (FIPP 2019).

Regarding web application development, in this modern era, Node.js is a popular choice for building the server/backend of the web application. Node.js is a JavaScript runtime environment that shapes the basis of many popular application frameworks today. It is well-known for an event-driven, non-blocking model that allows for high-speed synchronized data-processing. (Jung 2019.) As a result, Node frameworks have become the go-to choice in many companies for building servers that require flexible, highly scalable data handling.

However, as displayed in the figure below, the NodeJS ecosystem has developed a lot, and there is a wide range of unique frameworks for the company to consider (Blondin 2019). Each framework offers different technical use cases for the applications and the action of choosing one of those needs to be carefully decided base on the project requirements and objectives.

Figure 1. The wide variety of Node.js frameworks (Arora 2020)

One of the rising Node.js frameworks is NestJS – a framework for developing scalable and effective server-side applications. NestJS is being used as a backend framework by many large enterprises in the world such as Adidas or Autodesk, etc. Nowadays, NestJS is the most popular Node.js framework on Github with more than 23000 Github stars. Star is a vital metric about the software project evolution on Github, developers often give stars to a project to show appreciation toward the creators (Borges & Valente 2018).

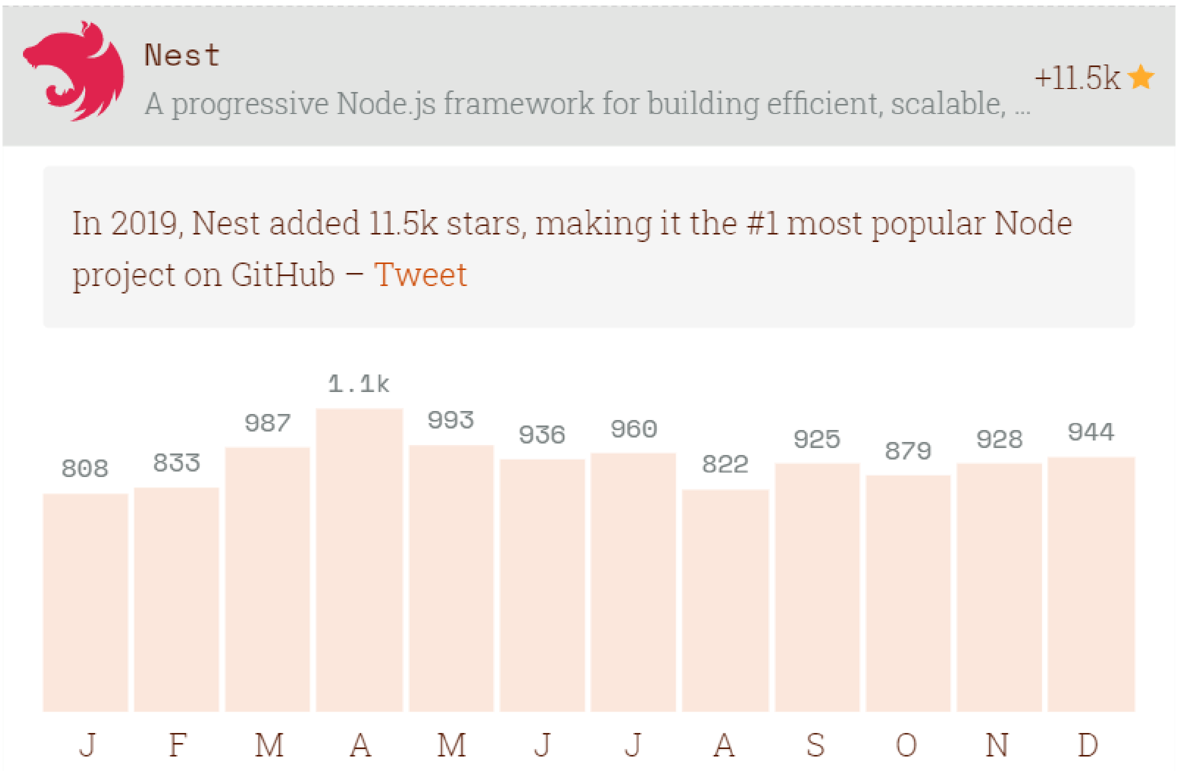


Figure 2. The rise of NestJS on Github in 2019 (RisingStarJS 2020)

However, because of the Node.js framework's diversity, before choosing a framework for a project, companies need to evaluate the adaptability of that framework in various aspects to make sure that it is the most appropriate on the market (Starling 2019). In terms of NestJS, there are not many specific academic articles about this new framework. Therefore, the goal of the thesis is to measure the pros and cons of Nest when utilizing the framework for a company’s backend application.

2 RESEARCH DESIGN

This chapter explains the selected research question, research type, research approach, research methods, and describes how data will be collected, processed, and analyzed. Finally, the chapter wraps up with the overall research framework.

2.1 Thesis objectives and research questions

The thesis provides a theoretical background that explains the concept of the web application, back-end development with JavaScript language, Node.js, NestJS, and technical development viewpoints. Moreover, the thesis presents research about the capability of NestJS (a popular Node.js framework) when applying it to a real-life backend project.

Consequently, the research question which the thesis aimed to answer is:

- What are the pros and cons of using the NestJS framework for a backend project?

There are many criteria to measure the pros and cons of a Node.js framework as new frameworks are continuously announced to the developer's community. One of the popular measures that define a successful development process is from Cleveroad – a software development company in America that has built a wide variety of web apps. According to Cleveroad (2017), the appropriate Node.js framework selection greatly relies on the overall architecture, documentation, performance, testing aspect. Besides that, the developers should also wisely analyze and understand the steepness of the learning curve and the support of the framework community. These are all the fundamental features of a Node.js framework that coders need to consider (Tachintha 2020).

As a result, to answer the research question, the thesis will adopt 6 measuring viewpoints from Cleveroad (2017) which is developed for all Node.js frameworks to measure the adaptability of NestJS. To be more precise, research of NestJS pros and cons when applying to the company's project will be conducted on 6 criteria:

- Architecture
- Testing
- Performance
- Documentation
- Community

- Learning curve

The more in-depth definition and impact of each viewpoint will be discussed in the Literature Review chapter of the research.

2.2 Research type

According to Creswell (2014, 3), there are two popular research types to answer this form of research question: Quantitative and Qualitative. Regarding quantitative research, the information from the quantitative study should be in the form of numbers, these numbers often come from surveys, questionnaires, observations checklists. In contrast, the qualitative research type reveals a dissimilar approach to methods of quantitative research. Qualitative type relies on text and image documents, has distinctive measures in data assessment, and develops on diverse designs even though the processes are related.

Based on the case study, an in-depth interview method is adopted to collect real-life data from experienced developers. Therefore, the thesis uses the qualitative research type, because the interview method cannot be assessed with numerical calculation.

2.3 Research approach

There is a wide variety of research approaches, but these three types are the most popular: deductive, inductive, and abductive**.** Firstly, deductive reasoning deals with certainty and contains analysis toward a convincing conclusion. Secondly, inductive reasoning comes with probability and requires reasoning toward results based on existing ideas. Thirdly, abductive reasoning goes with guesswork, includes hypothesis toward likely conclusions based on the finest presumption. In a nutshell, induction is examining the information from testing a hypothesis, the deduction would be applied when obtaining specific logical conclusions from the existing data, and abduction is generating a hypothesis. (DeMichele 2018.)

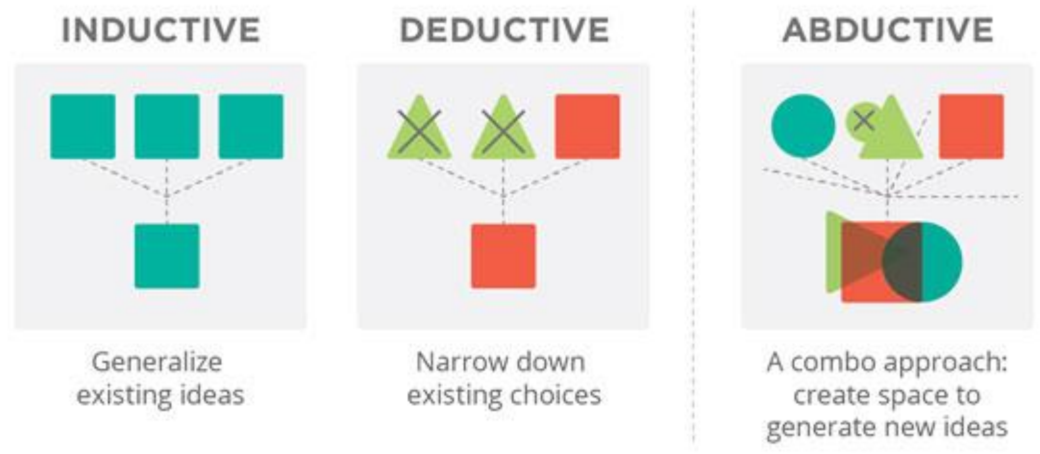

INDUCTIVE
DEDUCTIVE
ABDUCTIVE
Generalize existing ideas
Narrow down existing choices
A combo approach: create space to generate new ideas

Figure 3. Defining the inductive, deductive, and abductive methods of reasoning (Jokhio & Chalmers 2015)

In terms of the thesis, deductive is applied as the main research approach. This is because the thesis will assess the validity of the existing NestJS pros and cons theory. To be more specific, the NestJS benefits and drawbacks theory will be gathered in 6 mentioned aspects via existing academic books or online articles. Subsequently, this information will be tested with the confirmation or rejection of experienced developers via in-depth interviews to give out the finest conclusions/answers for the research question.

2.4 Data collection and data analysis

Intergrify Oy, a technology start-up in Helsinki, Finland is chosen for the thesis' case study. The author has been working with this company for the last few months and is in charge of developing a NestJS backend project here with his teammates.

2.4.1 Data collection

The literature review chapter of the thesis references information from existing books, online articles, journals about the pros and cons of NestJS. This knowledge base will be reviewed and listed based on the 6 mentioned measuring viewpoints. Furthermore, this pros and cons list will then act as the theory to test, measure, and be the guideline for structuring the developer's interview questions. All documents used in the literature review will be cited and stated thoroughly in the Reference section.

As a result, the research data will be collected from in-depth interviews with Integrify's developers who took part in building the NestJS backend project.

In-depth interviewing is one of the qualitative research techniques that include organizing personal interviews via a discovery-oriented method. This allows the interviewer to explore the respondent's opinions and perspectives on a specific topic. This method delves into respondents' behaviors, perceptions, and attitudes in terms of a situation or an issue. (Sbalchiero 2017.) The main benefit of in-depth interviews is that they provide much more thorough data than other data collection methods, such as surveys. Furthermore, the respondents often feel more comfortable having a discussion with the interviewers in a relaxed atmosphere that filling out a survey. (Boyce & Neale 2006.)

According to the Wallace Foundation, there is no standard number of interviewees in an in-depth interview, however, small numbers below 15 would be common. Therefore, the anticipated size for this in-depth interview process is 4 NestJS experienced developers. Because of the Covid-19 pandemic, the interviews will be performed via Internet application-based interview method. As a result, Zoom, Google Meet, and Slack are three chosen application to carry out these meetings. The author will schedule the meeting in advance for the coders who are inclined to join the process. Moreover, the identities of all developers will not be disclosed due to privacy purposes. The detailed plan of the interview questions will be shown in Chapter 5.

The author will videotape the interviews and record key interview information for the data analysis. Furthermore, the transcription of verbal data is not performed in real-time along with the interview but via filmed videos. In this case, the suitable transcription method is manual transcription. Manual transcription is the method by which human transcriptionists translate spoken words into the most readable and legible text type or vice versa. It is an appropriate option for precise, high-quality texts with a small number of speakers. (Worthy 2019.)

2.4.2 Data analysis

To answer the main research question, sentiment analysis will be performed to compare interview data with the report from the literature review and give out the finest result. Sentiment analysis deals with judgments, reactions, and emotions of given texts. This type of analysis is commonly used for the interview process to discover respondents' opinions, attitudes toward the questions. This is because sentiments are the most important characteristics of human behavior. Sentiments can be positive, negative, or neutral (Chakraborty, Bhattacharyya, Bag & Hassanien 2019, 127-147). Therefore, the standard measurement of sentiment analysis includes the calculation of the rate of how optimistic certain words are in each response (Thematic 2020).

Further details of the research analysis will be presented and demonstrated in Chapter 7.

2.5 Research framework

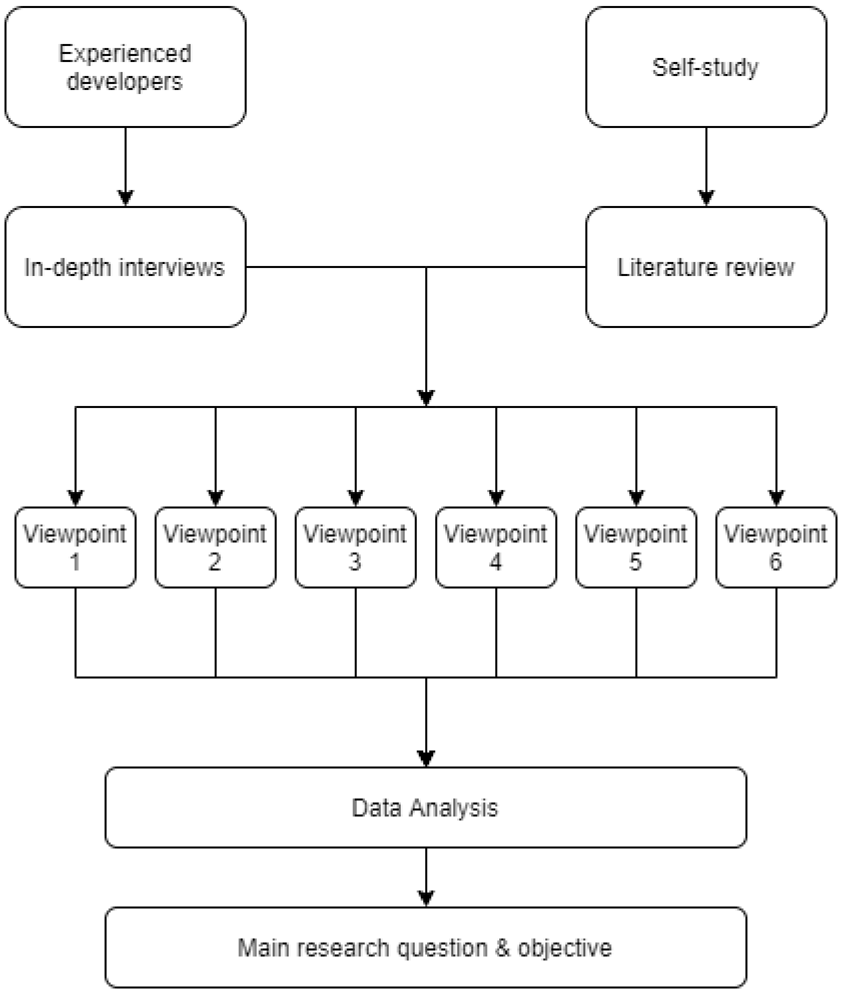


Figure 4. Research framework

Figure 4 describes the research framework of the thesis. To be more precise, a literature review about the pros and cons of the 6 mentioned aspects of NestJS from existing articles and books is presented. This knowledge base information then will be applied as guidance for the in-depth interview part. In-depth interviews are conducted to collect real-life opinions about the literature review's existing theory from 4 experienced developers. Finally, the interview data collected in 6 different viewpoints will be compared and analyzed with the theory to answer the main research question.

3 THEORETICAL BACKGROUND

3.1 Web application development

The term “web application” was first announced in 1999 as a new method to run application software on a web server. A web application is developed with web technologies that save and perform operations with data. Data operation actions take a crucial point in running a web application. These kinds of applications are accessed by web browsers and frequently have a strict authentication system. (Rouse 2019)

Regarding web application development, according to Johnston (2020), it is the procedure when creating a web application. The main phases of developing a web application are identifying the challenge, creating the answer, connecting with customers, applying a framework, developing, and assessing the app.

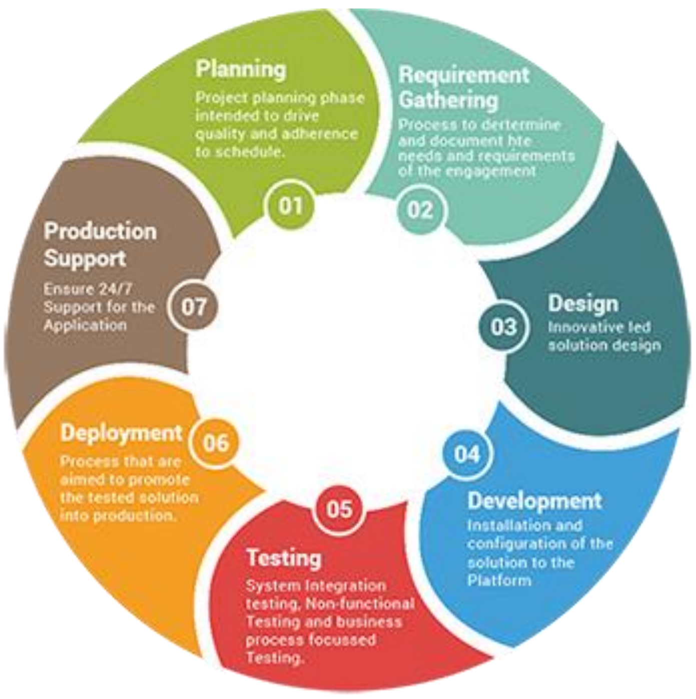

Planning
Project planning phase intended to drive quality and adherence to schedule.
01
Requirement Gathering
Process to dertermine and document hte needs and requirements of the engagement
02
Design
Innovative led solution design
03
Development
Installation and configuration of the solution to the Platform
04
Testing
System Integration testing, Non-functional Testing and business process focussed Testing.
05
Deployment
Process that are aimed to promote the tested solution into production.
06
Production Support
Ensure 24/7 Support for the Application
07

Figure 5. Web application development process (Vision Exault 2020)

3.2 Back-end development

When designing and developing a web application, the two most fundamental stages are frontend development and backend development. The frontend is largely focused on user-driven development such as the interface and the user experience. In contrast, back-end means server-side development with the primary focus on how the application performs. This development often involves a server, an application, and a database.

According to Ferguson (2020), the backend speaks with the frontend, delivering, and obtaining data to be presented on a web page. With any customer action on the web interface, the internet delivers a request to the server and receives some related data based on that request.

Consequently, back-end development often consists of these fundamental areas:

- Programming and scripting such as Python, PHP, Java, JavaScript, etc.
- Server architecture
- Database administration
- Scalability
- Security
- Testing
- Data transportation
- Backup

3.3 JavaScript language

3.3.1 History

JavaScript is a scripting or programming language that lets users execute complex features on web pages. It was first launched in 1995 when developers around the world were seeking a decent way to make web pages more dynamic. Since then, JavaScript along with HTML, CSS have become crucial parts of web application development to handle responsive, interactive elements and boosting the user experiences. (Aston 2015.)

From its rocky beginning, JavaScript has grown to be the most popular coding language on the planet. At present, roughly 94% of websites are developed with JavaScript. Stack Overflow – an online developer community surveyed 32000+ developers and established that JavaScript is the most common coding language in the recent 5 years. (Peters 2019.) Besides, according to GitHub’s Octoverse report (2019), there are more JavaScript code repositories than any other languages, and this number will be indeed on the rise in the future.

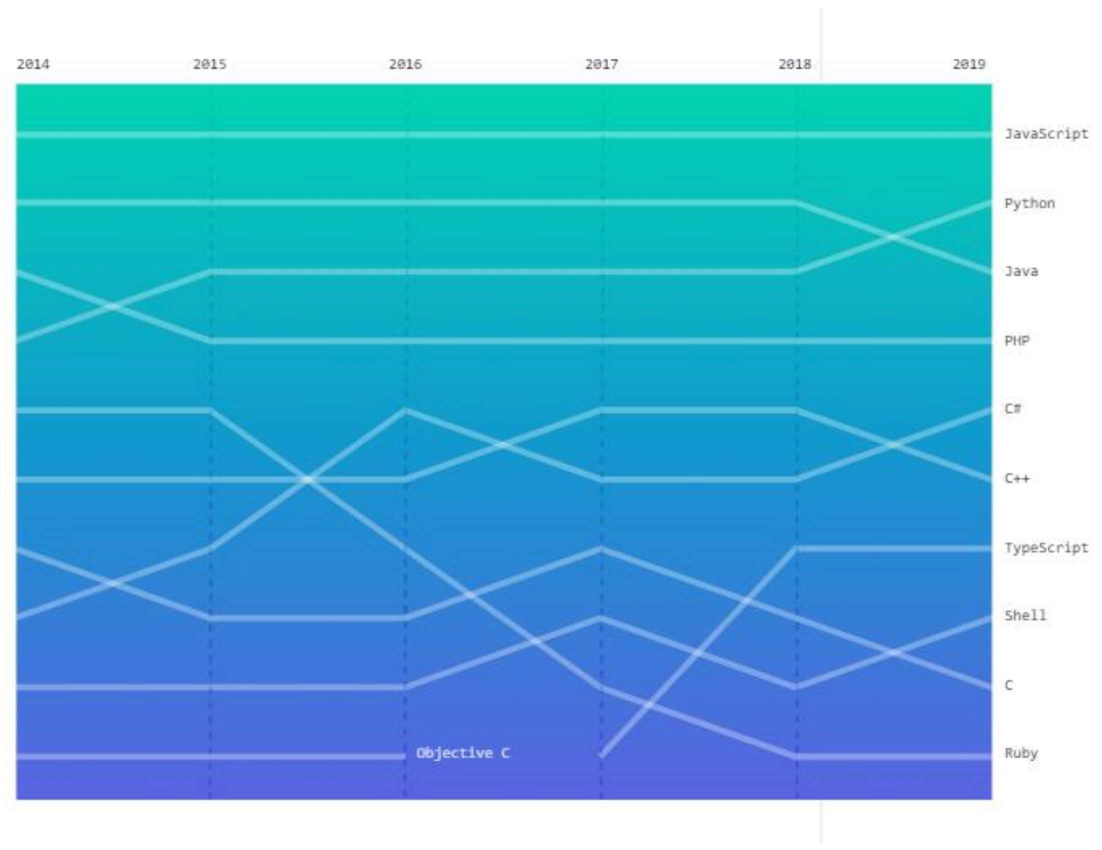


Figure 6. Most popular code repositories on GitHub (Octoverse 2019)

3.3.2 Development

Nowadays, JavaScript has become the most popular language in the world and developers began pushing the limits of what they could do with web applications. Eventually, there was the idea of using JavaScript as the server-side language since most of the coders are somewhat familiar with it and do not have to learn a new language, and that leads to the creation of Node.js. (Hayani 2018.)

3.4 Node.js

Node.js replaces the client-side rendering with server-side code implementation. To be more concise, Node.js gives websites the ability to run easier through all browsers because the JavaScript code is managed through the hosting server. (Adam 2017.)

3.4.1 History

In 2009, Ryan Dahl was obsessed with the theory of executing JavaScript outside the browser. After that, Ryan invented Node.js by using Chrome’s V8 engine - the sharpest JavaScript engine and embedding it with a C++ system. (Henderson 2019.) Chrome’s V8 successively improves Node's performance while removing problems with the middleman and offering compilation into native system code.

3.4.2 Development

NodeJS is an open-sourced environment with a great number of users all over the world that contribute to its repository or packages or modules. According to Stack Overflow (2020), Node.js is the most used framework by developers for building the frontend and backend of web applications.

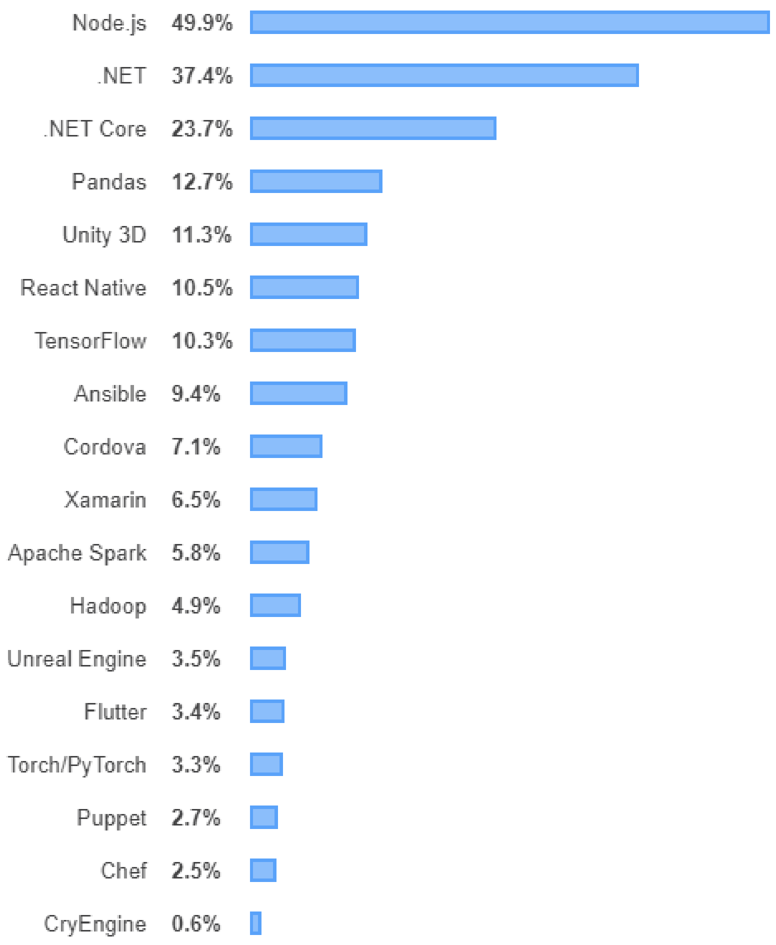


Figure 7. Most used frameworks by developers (Stack Overflow 2020)

Nowadays, Node.js is progressively becoming a go-to technology for numerous companies globally. As a result, more and more coders, especially web developers, have shifted to using Node.js to create more reliable applications. This has contributed to the evolution of the use of Node.js frameworks. Many professional JavaScript developers have developed a wide range of amazing frameworks over the years to quickly get started with Node.js when making web apps. (Oluyemi 2018.)

3.5 NestJS

3.5.1 History

NestJS (Nest) was developed by Kamil Myśliwiec with a mission to build effective, scalable Node.js backend applications. The framework supports JavaScript and TypeScript language. Moreover, it mixes components of FP (Functional Programming), OOP (Object Oriented Programming), and FRP (Functional Reactive Programming). (NestJS Documentation 2020)

3.5.2 Framework structure

The concepts below are the core fundamentals of NestJS to build a basic application:

- **NestCLI**: A command-line interface tool that not only speeds up the entire web app development but also automates the app initialization of the project. With only 2 commands in the figure below, the project directory will be created, node modules and a few other boilerplate files will be installed. (Bouchefra 2019.)
- **Controllers:** Manages requests and returns data to the client. A controller's mission is to get specific requests for the application. The routing mechanism determines which controller receives which requests. Occasionally, each controller has more than one route, and multiple routes may perform various acts. (NestJS Documentation 2020.)

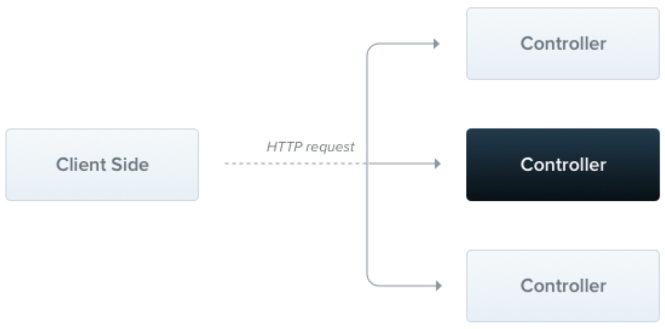


Figure 8. Example of how Nest's controllers work (NestJS Documentation 2020)

- To create a basic controller, Nest uses classes and decorators. **Decorators** associate classes with necessary metadata and allow Nest to build a routing map.

To define HTTP methods of given endpoint, there are: @Get(), @Post(), @Put(), @Delete(), etc. (NestJS Documentation 2020.)

- **Providers**: A fundamental concept in Nest. The main idea of a provider is that it can inject dependencies; this means objects can build various relationships with each other. In short, a provider is simply a class annotated with an @Injectable() decorator. (Samanta 2019.)
- **Services**: Responsible for operating the database, and is intended to be managed by the controller, so it could also be defined as a provider. Thus, NestJS decorate the class with @Injectable(). (Yadav 2020.)
- **Module**: A class with @Module() decorator. The module class is the place to connect Services, Controllers, etc. (Kiss 2020)

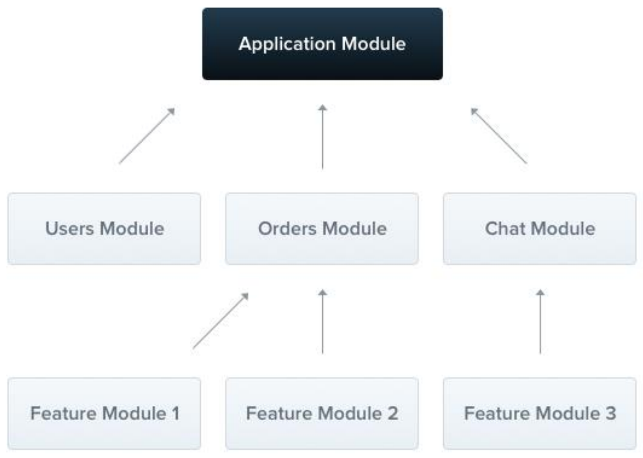


Figure 9. Example of how Nest's modules work (NestJS Documentation 2020)

- **Dependency injection**: A programming model where rather than creating dependencies, a class requires dependencies from outside sources. With this approach, the class will receive the needed dependency in the constructor. (Yeh 2020)

4 LITERATURE REVIEW

This chapter dives into the literature on the research topic: applying NestJS to the backend project. Research about the pros and cons of NestJS will be presented. The knowledge base information is reviewed via existing NestJS books and online articles on 6 aspects mentioned in chapter 2. These 6 viewpoints are architecture, performance, testing, documentation, community, and learning curve. As a result, each perspective will be studied and presented carefully in each sub-chapter. Furthermore, the concept of each viewpoint will be explained in general first and in NestJS's perspective after that. In the end, there will be a summary of the findings.

4.1 Architecture

An application architecture offers well-designed techniques and patterns to build an application. Furthermore, it provides coders a roadmap and best practices to follow and develop a well-structured app (Hat 2020). For example, folder structure and layer architecture are a few of the key factors that form the general application architecture.

As a result, in terms of NestJS architecture, there are 4 main advantages based on the existing articles and reports:

Firstly, NestJS is a highly opinionated framework**.** An opinionated framework is a framework where the architecture paths are already built, and the developers need to follow the provided structure guidelines without making any assumptions (Skowronski 2019). By defining an opinionated architecture that each coder must implement, it offers a collection of standards, resulting in an easily maintainable codebase (Yadav 2020). It already wraps the controller behind the scenes with a try-catch block, parses the request body, adds an error handler, middleware, logger, etc. Therefore, when scaffolding a new project, developers do not need to do these common things every single time. Furthermore, they must write the repositories, services, controllers in a guided way. (Rahman 2019.)

Secondly, NestJS applications are written in Typescript. With loosely typed languages like JavaScript or PHP, code can be written effortlessly by developers. Therefore, the risk of having bugs is very significant without strict typing. Nest, on the other hand, is probably the only well-structured system available that is completely written with Typescript. According to Kreuzer (2019), Typescript is a typed JavaScript version that supports developers during compilation time and runtime. It empowers developers with the JavaScript simplicity and the robustness of a typed language. For big applications, this robustness is particularly helpful. In short, Typescript is a scalable JavaScript. Typescript also offers users access to the new JavaScript features that have not yet been introduced. One of NestJS 's key goals is to build scalable applications, so it inherits the benefits of TypeScript.

Thirdly, NestJS has a solid Angular-like architecture**.** Most JavaScript backend frameworks do not cope with architecture. In Nest, the folder structure is based heavily on Angular. Consequently, this offers minimal downtime when creating a Nest service first. Each component gets its folder, modules, and main files residing in the root (plus some extra configuration files). This framework is as basic as it sounds and makes developers pay more attention to the design of endpoints and their customers instead of the application structure. (Earl 2019.)

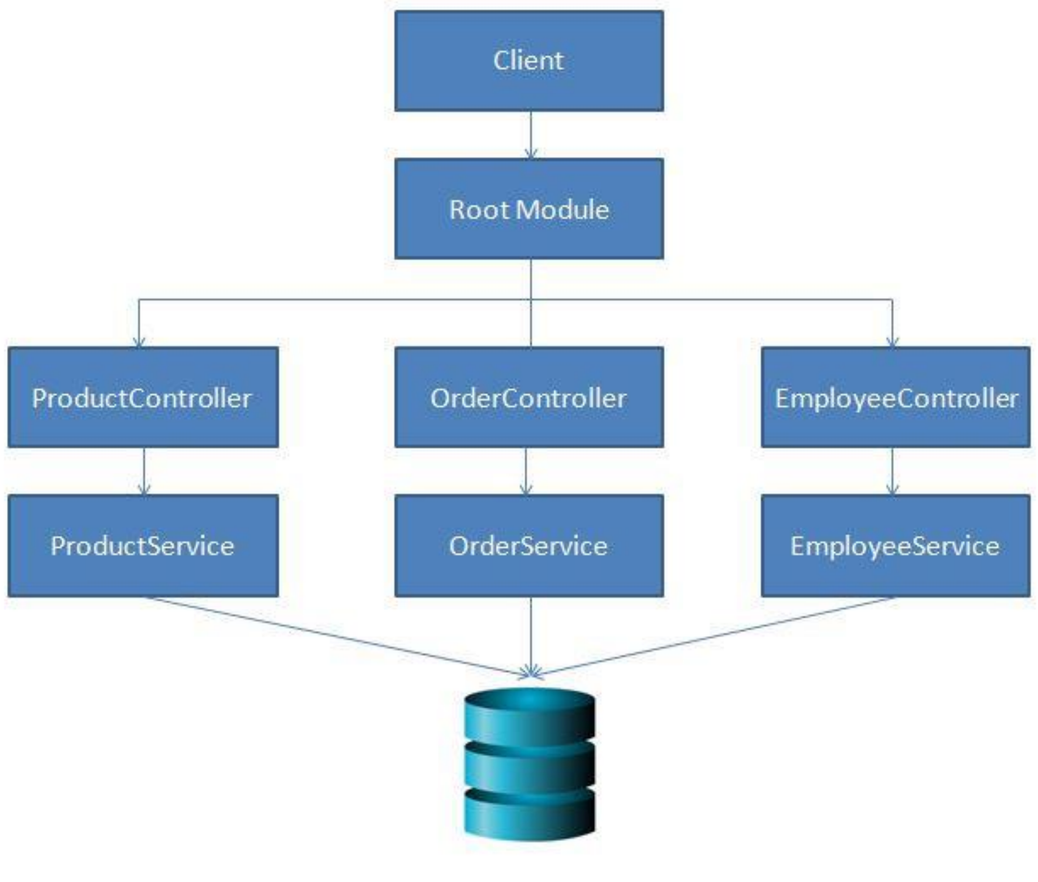


Figure 10. NestJS API architecture (Yadav 2020)

Finally, NestJS offers a command-line interface (CLI). To start and quickly support a project, Nest has its own CLI. Developers can quickly create modules, controllers, pipes, and middleware by using CLI without pain and therefore, boost productivity. (Yadav 2020.)

Table 1. Example of architecture modules that can be generated by NestCLI (Kreuzer 2019)

NestCLI command	Description
nest generate class	Generates a simple class and an according .spec file
nest generate controllers	Generates a class with the @Controller annotation. A controller can be used to create endpoints
nest generate filter	Generates an exception filter
nest generate guard	Generates a guard – this can be developed to decide if a request be processed or not
nest generate interceptor	Generates an interceptor to bind logic before/after the method performance
nest generate middleware	Generates a middleware – a function that runs before passing a request to its handler

In conclusion, according to the findings, there are four main advantages regarding the NestJS architecture: being a highly opinionated network, supporting TypeScript, having a solid Angular-like architecture, and offering a command-line interface.

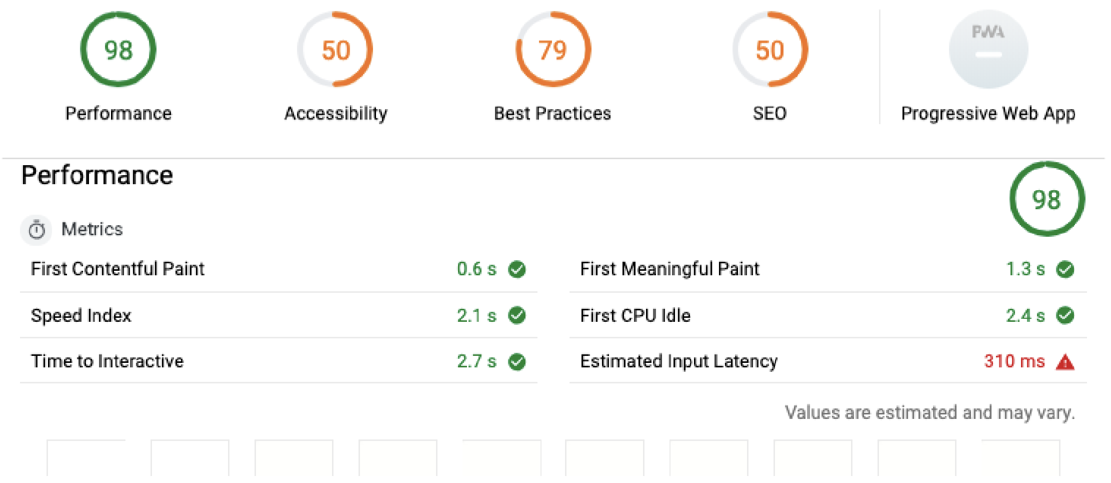
4.2 Performance

According to Smith and Williams (2001), there are two important aspects of software performance: responsiveness and scalability.

4.2.1 Responsiveness

Responsiveness is a system's ability to achieve its response time or throughput targets. In end-user systems, from a user perspective, responsiveness is frequently defined. Responsiveness, for instance, refers to the amount of time it takes for a user task to complete or the number of transactions that can be processed in a given amount of time. Responsiveness in real-time systems is a measure of how rapidly the system reacts to an event. (Smith & Williams 2001.)

Regarding NestJS, Nest also offers compatibility with other libraries, such as Fastify or Express.js, according to their official documentation. By implementing a system adapter whose primary purpose is to proxy middleware and handlers to suitable library-specific implementations, Nest achieves this system independence. Because Fastify and Express.js are two of the fastest Node.js frameworks, this results in boosting the overall responsiveness. (Rahman 2019.)



Performance

98

Metrics

First Contentful Paint	0.6 s ✓	First Meaningful Paint	1.3 s ✓
Speed Index	2.1 s ✓	First CPU Idle	2.4 s ✓
Time to Interactive	2.7 s ✓	Estimated Input Latency	310 ms ▲

Values are estimated and may vary.

Figure 11. Performance benchmark of NestJS from NodeSource (Parody 2019)

As can be seen from the figure above, according to Parody (2019), high responsiveness is an advantage of NestJS. When running a basic Hello World app, NestJS has very high responsiveness with a score of 98 in the report provided by Lighthouse – an automated tool for measuring web pages.

4.2.2 Scalability

Scalability is a system's ability to continue to achieve its reaction time or efficiency targets as the demand for software functions grows (Smith & Williams 2001).

NestJS is compatible with progressive JavaScript, developed with Typescript with and incorporates elements of these three concepts:

- OOP (Object Oriented Programming)
- FP (Functional Programming)
- FRP (Functional Reactive Programming).

This enables Nest.js as a highly scalable framework as Angular.js. Also, NestJS comes with its dependency injection container, like Angular. Moreover, NestJS makes the life of developers much simpler and happier when keeping the very testable server. Therefore, this framework is highly scalable (Oreofe 2019.)

4.3 Testing

Testing is a method of assessing a software application's performance to identify any software bugs. It tests whether the produced software meets the specified requirements and recognizes any flaws in the software for the best results. To find any holes, bugs, or incomplete specifications contrary to the actual requirements, developers often execute a testing system. (Sayantini 2020.)

According to McHenry (2020), the automated testing starter is a big advantage feature of NestJS since it increases the simplicity of testing in the application. NestJS creates an application starter automatically that comes with a default testing environment. Automated testing is believed to be an essential part of any serious effort to improve the software. Automation makes it possible during production to replicate individual tests or test suites quickly and easily. This helps ensure that every development fulfills the objectives of quality and efficiency. As a consequence, NestJS offers default testing tools to scaffold the default unit tests for components and e2e tests for applications automatically.

4.4 Documentation

Software engineering documentation is the umbrella term that covers all written documents and materials dealing with the growth and use of a software product. Documentation is required for all software development products, whether produced by a small team or a large company. Furthermore, the whole lifecycle of software development produces multiple types of documents. The purpose of documentation is to clarify product features, unify project-related data, and allow discussions of all major questions arising between developers and stakeholders. (AltexSoft 2018.)

In terms of NestJS, their official documentation has been revamped and is now fully written in Markdown syntax since 2019. Markdown syntax is designed to be as simple-to-read and easy-to-write as possible. A Markdown-formatted document should be released as plain text, without appearing to have been labeled with tags or instruction formatting. (DaringFireball 2002.) They believe that the update to the Markdown syntax will greatly simplify the contribution of the group and make the docs better over time, according to NestJS's official announcement.

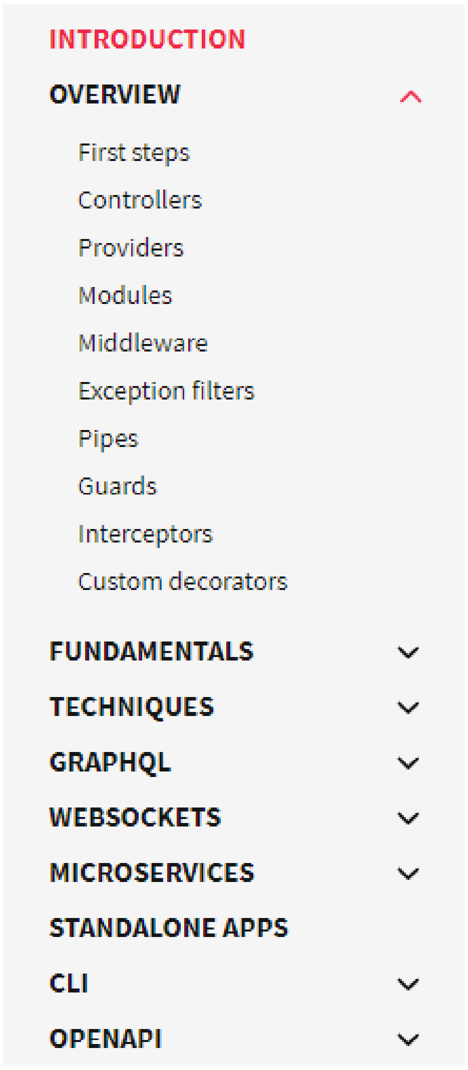


Figure 12. NestJS documentation structure (NestJS Documentation 2020)

However, according to Centizen Nationwide (2019), while Nest has excellent integrations with other frameworks, the documentation is very limited and does not include additional problems that may arise. Besides, there seems to be a lack of examples in the documentation. Although very helpful and in-depth in a variety of topics, some sections are not well described (with some going as far as only stated as possible) and leave developers to find out on their own. Therefore, lack of documentation is a drawback of NestJS. (Kalaitzis 2019).

4.5 Community

An active community around the technologies that developers use for the project application can be an important factor that encourages coders' venture's success. This will speed up the production of software, make it error-proof, connect developers to the best experts, and even help to promote the company's goods and services. (Ciszewski 2018.)

Although NestJS is one of Node.js's most rapidly growing frameworks, relative to others, it still has a lack of community support, meaning fewer resources and less comprehensive documentation. In consequence, this makes debugging on Stack Overflow or any developer's forum trickier. (Parody 2019.)

4.6 Learning curve

A learning curve represents the cost-output connection over a particular period, typically to reflect an employee or worker's repetitive task. Psychologist Hermann Ebbinghaus first defined the "learning curve" in 1885 and used it to calculate manufacturing performance and predicted expenses. In short, the learning curve is the rate of improvement of an individual in gaining new abilities or skills. (Kagan 2020.)

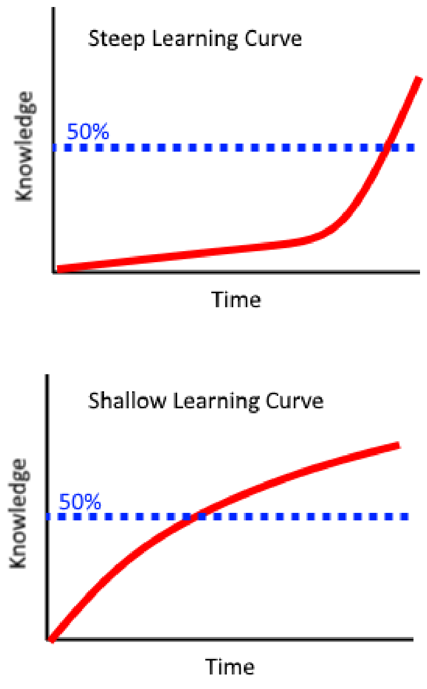


Figure 13. Example of a steep and shallow learning curve (Brix 2016)

As we can see from the figure above, a "steep learning curve" means the knowledge takes a long time to learn while a "shallow learning curve" represents a quick learning process.

In terms of NestJS, it is written in TypeScript and its architecture is heavily inspired by Angular. When coding with NestJS, JavaScript developers who are not familiar with TypeScript or Angular philosophy such as Decorator, Dependency Injection, or many more concepts may run into problems. Also, developers may need to learn more about the Node.js database integration library or ORM to run the database at a higher level of abstraction, such as Sequelize or TypeORM. Therefore, the steep learning curve for new developers is a handicap for NestJS. (Rahman 2019.)

4.7 Summary of the findings

This section includes the NestJS pros and cons summary list that was collected in the above literature review. The validity of this theory will also be tested by experienced developers via an in-depth interview process in chapter 7.

4.7.1 Advantages of NestJS

Architecture:

- It is a highly opinionated framework
- NestJS applications are written in Typescript
- It has a solid Angular-like architecture
- It offers a command-line interface (CLI) which let developers easily scaffold the project

Performance:

- It has high responsiveness in term of performance
- It is highly scalable

Testing:

- Its automated testing starter increases the simplicity of testing

4.7.2 Disadvantages of NestJS

Documentation:

- Lack of documentation

Community:

- Lack of community support

Learning curve:

- It has a steep learning curve for new developers

5 RESEARCH PROCESS

This chapter explains the research process of the thesis through different phases in practice. Moreover, the chapter describes in detail how the mentioned research methods in chapter 2 were applied to the study.

The case company chosen project for the research was the student management system (SMS) backend project. The SMS backend team consisted of 5 coders (including the author) working with the NestJS framework. The author conducted in-depth interviews with the remaining 4 teammates.

5.1 Interview question plan

According to Carolyn Boyce and Palena Neale (2006, 5), questions should be open-ended rather than closed-ended while performing an in-depth interview so that respondents can speak freely. "For instance, rather than asking" Do you know about services at the clinic?", the author could ask "Please describe the facilities of the clinic". Therefore, there were overall 8 open-ended questions to be developed to answer 6 viewpoints.

Firstly, the "architecture" viewpoint was answered through a guided conversation about interviewees' opinions on 4 structure features: Opinionated framework, Typescript, Angular-like architecture, NestCLI. Secondly, the "testing" viewpoint was addressed via one request and one question. The first request was "Please describe the testing facilities of this project". The purpose of this was to have an overall view of the developer's NestJS project testing process. The next question was meant to find out the interviewees' ideas about NestJS testing. Thirdly, the "performance" viewpoint was answered by 2 questions. The first question was "How do you think about NestJS's responsiveness to this project compared to the old one?". This question aimed to measure the responsiveness performance of NestJS with another Node.js framework. The second question was to determine the scalability of NestJS when applying to their backend project. Finally, the driven discussions on "documentation", "community" and "learning curve" were conducted to collect developers' opinions through their real-life experience with these last 3 viewpoints. Each in-depth interview was anticipated to last 60 minutes. The table below shows the detailed plan of the interview process.

Table 2. Detailed interview plan

Research criteria	Interview questions/conversation	Expected time to answer
Architecture	- Let us talk about your opinion on some key architecture features of NestJS: Opinionated framework, TypeScript, Angular-like architecture, and CLI	15 minutes
Testing	- Please describe the testing facilities of this project. - Could you give an example of the impact of NestJS testing facilities on debugging?	15 minutes
Performance	- How do you think about NestJS's responsiveness to this project compared to the old one? - With a large, always ongoing project like SMS, what is your opinion on the scalability of NestJS when applying to the project?	15 minutes
Documentation	- What is your opinion on the NestJS documentation?	5 minutes
Community	- What do you think about the NestJS community?	5 minutes
Learning curve	- How would you describe the learning curve of NestJS, especially for newcomers?	5 minutes
Total		**60 minutes**

5.2 Interview data collection plan

In terms of the videotaping interviews process, a native screen recording application on Windows (the author's computer operating system) was applied to record quality audio and video of the process.

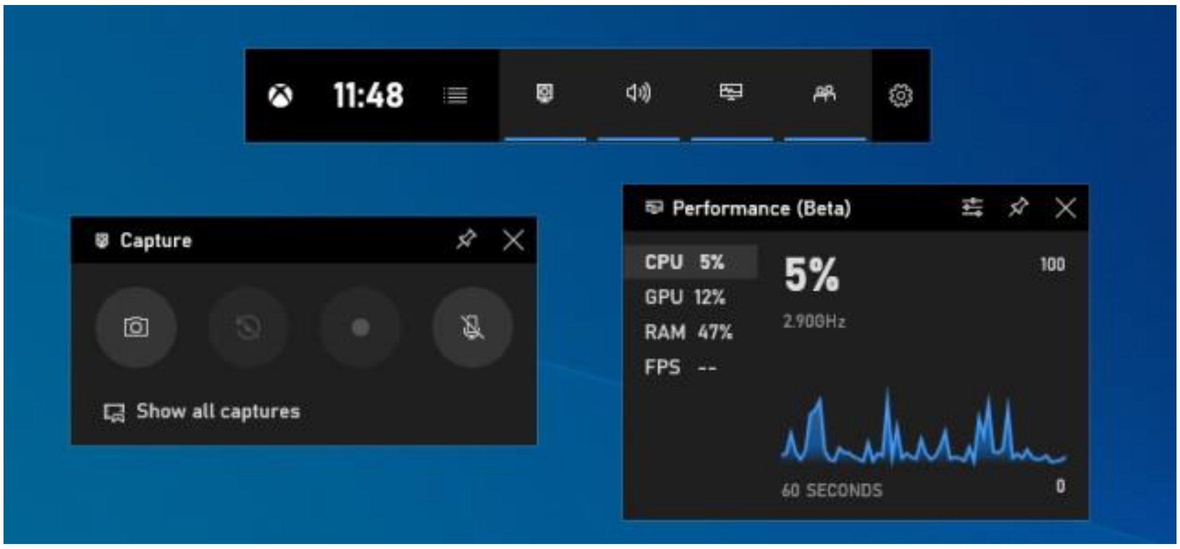


Figure 14. Windows screen recording tool (Wyciślik-Wilson 2020)

After recording the interviews of 4 developers, the transcription of verbal data was performed to convert spoken words into readable texts. The manual transcription method was selected for the translation. This process was after the interview process (not along with the interview process) to assure the highest quality of the transcription. According to Indianscribes (2018), the number one rule of verbal transcription is to capture every word of the conversation. Therefore, the author listened to each of the interview recordings at least 2 times before transcribing word by word the conversation to minimize the mistakes. The collection of data will be presented in the next chapter.

5.3 Interview data analysis plan

Regarding the research, sentiment analysis was utilized to assess the opinions of 4 developers on each of the NesJS pros and cons theory gathered from chapter 4.

According to Thematic (2020), the sentiment analysis is conducted by comparing some specific words in each sentence with the sentiment word dictionary to determine the sentence's opinion. For example, some of the positive words are great, delicious while positive words could be terrible, expensive, etc. Furthermore, the SentiWords sentiment dictionary is chosen for this analysis because of its popular, high coverage resource containing roughly 155.000 English words (Guerini 2020).

The main research approach of the thesis to answer the research question is the deductive approach. Therefore, after determining the opinions, the comparison between the assembled theory of chapter 4 and the real-life data will be presented to examine the validity of the literature review hypothesis and produce the final result of the research.

6 COLLECTION OF DATA

All of the four interviewees have been working with NestJS for at least 6 months and SMS is their first backend project using NestJS. Regarding the high responsiveness data collection, since SMS's NestJS also uses Express.js underneath, it is not sensible to measure this NestJS version and the old Express.js version. Therefore, the NestJS benefit of high responsiveness is not answered by the developers.

6.1 Developer 1's interview

Developer 1 has one year of experience in web application development. In terms of the SMS project, he acts as a technical leader. He is in charge of reviewing other's codebase, helping with the application logic, developing new features, and fixing bugs.

Regarding NestJS being a highly opinionated framework, he mentioned:

> *I like it as it is highly opinionated, vanilla Express.js has far too many structure design options which can lead to developer's disagreement. SMS project is the first project that I applied an opinionated framework to, and the folder structure is very helpful since developers only need to follow the guided structure to create the NestJS project. (Developer 1)*

For the Typescript feature question, he said:

> *I enjoy writing NestJS in TypeScript. The types were fairly straightforward and helped with code completion, avoid silly mistakes, etc. With normal JavaScript, developers could only see the error when running the application, but with TypeScript in the SMS project, there would be the error notification right away when we are coding, so that is a big plus. (Developer 1)*

Talking about NestJS provides a solid Angular-like architecture, he pointed out that:

> *As much as many folks seem to hate on Angular, I think the highly opinionated OOP structure works well in the background where modularity is key. Every module contains a controller and a service, which supports each file to do one job and makes the code split nicely. This is called "separation of concerns" in software development. (Developer 1)*

Besides, he also answered that the NestCLI feature helped him a lot during development. His thought about NestCLI is that:

> *NestCLI is a nice tool to avoid writing a ton of boilerplate, which nobody likes. I enjoy having this tool. This helped the SMS project developers to focus on the actual codebase rather than spending time manually scaffold the project which ends in the same result as the NestCLI does. (Developer 1)*

Concerning Nest automated testing facilities and its impact on application debugging, he stated that:

> *Our unit and e2e automated tests completely killed the need for using postman (a platform for testing API requests) as well as, to a certain extent, having to click around with the frontend. SMS frontend teams often complained that we have bugs in our code for example, when they are trying to add a new student to a course. Running these kinds of testing automation helps us to show them that the problem is not from the backend but their incorrect payload (for instance, not putting the correct start or stop dates on the inputs) (Developer 1)*

When discussing the technical performance, he explained that high responsiveness and scalability are the advantages that NestJS bring to us:

> *As the architecture is already in place, the SMS application is definitely able to scale. That because we were forced to write the code in a very specific way by Nest, it was hard to make hacky, shortcut solutions that prevent a backend from being able to get bigger. (Developer 1)*

Developer 1 said that the documentation is pretty readable and well-done. Furthermore, regarding the NestJS community, he mentioned:

> *It seems to be fairly active – I join a Telegram (a messaging application) community that constantly has new posts. I did not really discover one in Finland though, so finding help here is a bit challenging. I once was stuck with a simple structure error for a day without getting any existing solution on the Internet. (Developer 1)*

Finally, when chatting about the learning curve of NestJS, he stated that:

> *The learning curve is fairly steep given the amount of boilerplate, intense OOP design, Java-like structure. It took me a few weeks to learn about these concepts before implementing NestJS to the SMS project (Developer 1)*

6.2 Developer 2's interview

Developer 2 is the developer and main tester of the SMS project. He has 1 year of experience in building web applications. SMS project is the first time that he acts as a backend developer. His jobs are creating automation tests, reviewing other's codebase, developing new features, and fixing bugs. In terms of the interview, Developer 2 went through all of the interview questions and he also gave thorough responses about NestJS testing. Specifically, information about the SMS backend unit and end to end testing/debugging.

In terms of the high opinion of the NestJS framework, he mentioned that this helped the team a lot while developing the SMS from scratch:

> *Being highly opinionated NestJS follows the design paradigm convention over configuration. Also, while working with NestJS, it strongly guides us to use certain tools and to code in a certain way. Furthermore, with a large project like SMS, a consistent structure is a crucial aspect to maintain the code. (Developer 2)*

Talking about the combination of NestJS and TypeScript, developer 2 said that TypeScript is the obvious choice for him, because of the strict static typing. This makes the code more robust and bugs free. Furthermore, he said that even the NestJS documentation examples are written in TypeScript so this must be a big benefit of NestJS.

Regarding the Angular-like architecture, he had a similar thought as Developer 1:

> *Personally, I like the architecture of NestJS. It opens up some powerful patterns. Since it is very modular, once you create a module, you can just plug-and-play this in every other NestJS service you are building. Other fundamental Angular concepts like decorators or dependency injection are also fully utilized in NestJS architecture (Developer 2)*

Concerning the NestCLI, he said that this is a very powerful and convenient tool for creating services and controllers on-the-go without any hustle.

Furthermore, Developer 2 believed that this is a significant advantage of NestJS because of its diversity. Firstly, NestJS provides unit testing for testing small chunks of logic in controllers and services. Secondly, end to end testing is also offered for testing the whole application's workflow from beginning to end. However, he acknowledged that it required a lot of boilerplate to run end to end tests. Developer 2 also mentioned:

> *NestJS automated testing proves to be very crucial when it comes to debugging your code. It gives you the ability to debug your code in a very efficient way. For example, instead of using tools like Postman which could be very time-consuming, you could immediately test your API endpoint through the integration tests of the SMS project. Furthermore, the automation tests which run every time we publish the*

> *new codebase to the server make it very simple to debug human errors. (Developer 2)*

When discussing the scalability of a large, always ongoing like SMS project, Developer 2 commented:

> *In my opinion, NestJS is well suited for a large project like SMS. it can be easily scaled to further functionalities because of its logical groupings of functionality. This means, each of the features has its module and it is painless to add new functions to each module. (Developer 2)*

About NestJS documentation, he stated that:

> *NestJs has unclear documentation at least in my opinion. Sometimes I have some trouble while reading on how to use some new features. (Developer 2)*

Also, Developer 2 shared an opinion about the NestJS community:

> *NestJS community is small since NestJS is new as compared to other frameworks but growing quite fast. Help is available online sometimes, but it is still difficult to find the correct information. (Developer 2)*

Finally, about the learning curve, his thoughts are:

> *I would say for beginners, the learning curve is a bit steep since it takes time to understand the workflow and the boilerplate. For people with Angular background, it would be easier since they are already familiar with the concepts. (Developer 2)*

6.3 Developer 3’s interview

Developer 3 acts as a developer of the SMS project. She has less than one year of experience in the software development industry. The time that Developer 3 spent on the SMS project is less than others since she switched to another project while we were developing the SMS, so most of her answers are briefer than others. However, she provided a decent answer on NestJS scalability.

Firstly, when discussing NestJS being an opinionated framework, Developer 3 shared that:

> *Being an opinionated framework, it provides a good architecture for the project and helps in fast development. When working in the SMS team with different developers, an opinionated framework is important to maintain code consistency across the whole project. (Developer 3)*

Secondly, about the TypeScript built-in of NestJS, she replied that it helps the developers a lot to identify the types of methods when the application gets bigger and more complicated.

Some of her thoughts about the Angular-like architecture are:

> *Angular-like architecture boosts the understanding and the reusability of the application to the next level. In the SMS project, I find it easier to read and understand the code from other developers in this kind of architecture. (Developer 3)*

In terms of the Nest CLI, Developer 3 stated that it is a go-to choice to instantly initialize, develop, and maintain the Nest applications. Furthermore, she said that Nest CLI also presents best-practice architectural patterns to encourage developers to create well-structured applications.

Regarding the testing facilities of the SMS project, she pointed out that:

> *SMS project has unit and end to end testing facilities. The automation helps increase coverage and provides a quicker response to developers. Also, the error messages are easier to identify than in the Express.js application. (Developer 3)*

Also, about the scalability of this subject, some of her thoughts are:

> *Scalability is effortless since NestJS has the architecture of object-oriented programming. I would say highly scalable is one of the NestJS best features that I have experienced so far. (Developer 3)*

In terms of the documentation, she gave an opinion:

> *I think that their documentation is not mature enough and there is still much room for improvement. For example, I would like to see small tutorials on different topics, this would help a lot for new users. (Developer 3)*

When talking about the limitation of the NestJS community, Developer 3 stated that:

> *Yes, it is small compared to other popular frameworks. Sometimes, it is quite difficult to find the solution for basic issues on the Internet. (Developer 3)*

Lastly, when describing the learning curve of NestJS for the newcomers, she gave a comment that it certainly depends on their background. For example, it would be painless for someone coming from Angular or Java, but it would become tricky for people with a JavaScript background.

6.4 Developer 4's interview

Developer 4 is the taskmaster, as well as the developer of the SMS project. He has 2 years of experience in developing websites and web applications. His everyday jobs are assigning tasks to other developers, keeping track of the workflow, reviewing other's codebase, developing new features, fixing bugs.

Firstly, Developer 4 shared that he preferred the high opinion architecture of the NestJS framework a lot since he wanted to work with a known structure, best-practices project and there is no reason to re-invent a brand-new structure.

Some of Developer 4 views about the TypeScript built-in feature are:

> *I do not like the fact that the NestJS application is written in TypeScript since this makes the typing system becomes overly complicated and hard to use properly. Furthermore, TypeScript requires compilation, while JavaScript does not, and this increases the application build time. (Developer 4)*

Furthermore, since Angular architecture is built around the TypeScript language, he shared that:

> *In my opinion, I think Angular architecture is nice but maybe the NestJS team relies too heavily on the Angular system. It takes lots of time and effort to learn about different concepts and manage this kind of architecture. (Developer 4)*

When discussing Nest CLI, Developer 4 said that it is a nice tool to automatically build up the desired folder structure through only some commands from the terminal.

Regarding NestJS testing facilities, he said:

> *Yes, automated testing helped us a lot in debugging human errors. NestJS testing facilities offered us in the SMS project an easier way to integrate with other testing frameworks like Jest or Supertest and provides the dependency injection system for mocking components. (Developer 4)*

In terms of the high scalability, Developer 4 shared the opinion that SMS is surely able to scale because of the well-designed architecture. He also said that SMS started as a small project and has been scaling up ever since.

About the lack of documentation, he pointed out that:

> *During the first few weeks of the SMS project, I had a hard time reading and understanding this document. Perhaps, they can add more step-by-step tutorials on this. (Developer 4)*

Also, Developer 4 stated that it is quite hard to find a solution post on Stack Overflow (a developers' online community website) about NestJS's wide variety of issues.

Finally, regarding the learning curve of the framework, he said:

> *As I mentioned, Angular-like architecture makes it difficult for new learners to adapt to NestJS in a short period. This architecture is not suitable for newcomers who want to quickly build a simple backend server. I will choose Express.js instead when it comes to the learning curve. (Developer 4)*

7 STUDY AND ANALYSIS

In this chapter, all data collected in the in-depth interview is evaluated with the theory gathered from the literature review in chapter 4. Sentiment analysis is utilized to analyze the thoughts of 4 developers toward each NestJS pros and cons theory.

7.1 Conducting sentiment analysis

The analyzing process contains 3 steps for each phrase: Extract important words which are mostly adjectives, compare those words to the SentiWords sentiment dictionary and determine the opinion of that phrase. Some of the sentiment analysis examples are:

> *As I mentioned, Angular-like architecture makes it difficult for new learners to adapt to NestJS in a short period. This architecture is not suitable for newcomers who want to quickly build a simple backend server. (Developer 4)*

Some of the keywords to determine the feeling of Developer 4 on the above phrases are “difficult”, “not suitable”, etc. According to the SentiWords sentiment dictionary, these words are listed as negative comments. Therefore, this phrase’s sentiment is mostly negative.

> *NestCLI is a nice tool to avoid writing a ton of boilerplate, which nobody likes. I enjoy having this tool. This helped the SMS project developers to focus on the actual codebase rather than spending time manually scaffold the project which ends in the same result as the NestCLI does. (Developer 1)*

Important words to define the sentiment of Developer 1’s opinion in this phrase are: “nice”, “enjoy”, “help”, etc. As a result, this statement’s attitude is primarily positive according to the sentiment dictionary.

> *It seems to be fairly active – I join a Telegram (a messaging application) community that constantly has new posts. I did not really discover one in Finland though, so finding help here is a bit challenging. (Developer 1)*

Some of the crucial words to assess the above statement are “fairly”, “not really”, “a bit”, “challenging”. According to the Sentiwords dictionary, most of the words in these phrases are neutral. Thus, Developer 1’s opinion on this issue is neutral.

7.2 Comparing the literature review theory (chapter 4) and real-life interview data (chapter 6)

In this sub-chapter, data collected from the interview process (chapter 6) is compared and analyzed to testify to the NestJS pros and cons theory list gathered from the literature review in chapter 4.

In table 2 contains a summary of the reported pros and cons of NestJS in the Literature Review chapter with respondents' opinions in the in-depth interview chapter. Responses that have the most positive words are approved opinion - labeled with the symbol: ☑. Disapproved opinions with most negative words are notified with the symbol: ✕. Neutral decisions with most neutral words or equal positive and negative words are marked with the symbol: 💢. Finally, unanswered issues are labeled with the symbol: —.

Table 3. Sentiment analysis summary

	Viewpoints	Issues	Dev 1	Dev 2	Dev 3	Dev 4
Pros	Architecture	Highly opinionated framework	☑	☑	☑	☑
		Written in Typescript	☑	☑	☑	✕
		Solid Angular-like architecture	☑	☑	☑	💢
		Nest CLI	☑	☑	☑	☑
	Testing	Automated testing starter	☑	☑	☑	☑
	Performance	High responsiveness	—	—	—	—
		High scalability	☑	☑	☑	☑
Cons	Documentation	Lack of documentation	✕	☑	☑	☑
	Community	Lack of community support	💢	☑	💢	☑
	Learning curve	Steep learning curve	☑	☑	💢	☑

Approved: ☑, Disapproved: ✕, Neutral: 💢, Unanswered: —

7.3 Comparing the NestJS advantages between literature review theory (chapter 4) and real-life interview data (chapter 6)

Except the high responsiveness benefit is not answered due to the measurement difficulty, most of the reported benefits are approved.

Firstly, the benefit of NestJS being a high opinionated framework is confirmed in all of the cases.

Secondly, apart from Developer 4, who mentioned TypeScript made the typing system became too complicated and hard to use, the others verified that they enjoyed writing NestJS applications in this language.

Thirdly, the solid Angular-like architecture advantage is approved by 3 developers. These three confirmed that the Angular structure helped them a lot in making the code more understandable and reusable. Besides, Developer 4 gave a neutral comment about this type of architecture.

Fourthly, all of the interviewees approved of listing NestCLI as a big plus of NestJS when scaffolding the project.

Fifthly, about the NestJS automated testing facilities, all four developers confirmed that the SMS project received significant assistance from this. NestJS testing starter support and give developers the ability to debug the code in a very effective way.

Finally, every interviewee certified that NestJS users can scale up their application to further functionalities with ease and comfort.

Table 4. Summary of theory and interview data comparison in NestJS advantages

NestJS advantages	Developers confirmation
It is a highly opinionated framework	4 developers approved
NestJS applications are written in Typescript	3 developers approved
It has a solid Angular-like architecture	3 developers approved, 1 was neutral
It offers a command-line interface (CLI) which let developers easily scaffold the project	4 developers approved
It is highly scalable	4 developers approved
Its automated testing starter increases the simplicity of testing	4 developers approved

7.4 Comparing the NestJS disadvantages in literature review theory (chapter 4) and real-life interview data (chapter 6)

Although the reviewed drawbacks from the literature findings are generally approved, there are some mixed comments about these.

Firstly, Developer 2, 3, and 4 confirmed listing the lack of documentation as an obstacle. They mentioned that the NestJS team is not doing great with the documents and there is room for improvement such as small tutorials on each topic, etc. In contrast, Developer 1 stated that the documentation is pretty readable and clear.

Secondly, about the lack of community support, Developer 1 and 3 gave a neutral comment and stated that although it is fairly active, it is still very small compared to other mature frameworks. Besides, the other two interviewees had the same opinion as to the author's findings.

Finally, apart from the neutral point of view of Developer 3, other interviewees agreed that NestJS has a pretty steep learning curve for beginners because of its Angular-like architecture.

Table 5. Summary of theory and interview data comparison in NestJS disadvantages

NestJS disadvantages	Developers confirmation
Lack of documentation	3 developers approved, 1 disapproved
Lack of community support	2 developers approved, 2 was neutral
It has a steep learning curve for new developers	3 developers approved, 1 was neutral

7.5 Conclusion

This sub-chapter concludes the answer to the research question after the analysis of theoretical research and case study based on 6 viewpoints: Architecture, performance, testing, documentation, community, learning curve.

The main research question is:

- What are the pros and cons of using the NestJS framework for a backend project?

Based on the research and interview data with four experienced developers' comparison, it can be wrapped up that NestJS has several advantages and disadvantages when applying to a backend project.

7.5.1 NestJS's advantages

All the benefits of NestJS are from the technical point of view such as architecture, performance, testing issues.

- It is a highly opinionated framework
- NestJS applications are written in Typescript
- It has a solid Angular-like architecture
- It offers a command-line interface (CLI) which let developers easily scaffold the project
- It is highly scalable
- Its automated testing starter increases the simplicity of testing

7.5.2 NestJS's disadvantages:

When it comes to the drawback, the study revealed 3 mains issues around the NestJS documentation, community support, and learning curve.

- Lack of documentation
- Lack of community support
- It has a steep learning curve for new developers

8 SUMMARY

In summary, with the rise in a wide variety of different Node.js backend frameworks over the past few years, it becomes crucial for software development companies to choose a well-designed, suitable framework for the web application project.

The goal of the thesis is to study the advantages and disadvantages when applying NestJS – a popular Node.js framework to a backend project. To achieve the research's goal, a theoretical background, literature review, along with research that provided the data from in-depth interviews with NestJS experienced developers were presented. This data then was compared with the gathered theory to give out the finest result for the research question. As a result, the benefits that NestJS brings to the web development process come from its technical aspect like opinionated framework, great scalability, solid Angular-like architecture. Moreover, NestJS offers awesome built-in features like TypeScript language support, user-friendly NestCLI, and an automated testing starter. In contrast, the drawbacks of the framework are from other aspects like steep learning curve, and the lack of framework documentation/community support. From these findings, it can be concluded that NestJS is a great Node.js backend framework choice for companies in terms of technical aspects, but there are some issues with the learning curve, and documentation and community support. Limitations, reliability and validity, and suggestions for further study are also introduced in this chapter.

8.1 Limitations

There are there limitations in the research. The first limitation is the restriction of the case study project and company. Integrify Oy is a small IT start-up company with below 10 developers and the SMS project is the first and fairly new backend project using the NestJS framework in the company. Furthermore, the second limitation is the lack of academic references. Most of the resources in the thesis are digital documents that might not be completely reliable. Nevertheless, the NestJS framework is a pretty new subject and there are hardly any subject-related books on the market. Finally, the actual interview time is slightly shorter than the estimated time from the interview plan.

8.2 Reliability and validity

According to Leung (2015), consistency is the nature of reliability in a qualitative study.

In terms of the study, the data collection is based on the previous reports, findings of many legitimate sources, and the interview process with four experienced NestJS developers. Therefore, the research is reliable.

With qualitative research type, validity refers to the relevance of the methods, procedures, and information. This means the research objectives, the choice of methodology, the data collection, and analysis should be appropriate with the results and conclusion. (Leung 2015.)

Regarding the thesis, the topic is a software development subject, the case study company is a technology start-up that has experience with this software development service and the final results are the advantages and disadvantages when applying this to a technical project. Consequently, this paper is valid.

8.3 Suggestions for future studies

The fact that NestJS is still growing rapidly leads to certain subjects for further study as follow:

- The benefits that NestJS brings to IT companies compared to other non-Node.js backend frameworks
- The pros and cons of NestJS microservice architectural style of development
- Promoting NestJS best practices for better team development

LIST OF REFERENCES

Written References

Hudson Borges, Marco Tulio Valente. 2018. What is in a GitHub Star? Understanding Repository Starring Practices. The Journal of Systems and Software, 112-129.

Carolyn Boyce, Palena Neale. 2006. CONDUCTING IN-DEPTH INTERVIEWS. Watertown: Pathfinder International.

Chakraborty, Bhattacharyya, Bag, Hassanien. 2019. Sentiment Analysis on a Set of Movie Reviews Using Deep Learning Techniques. Social Network Analytics, 127-147.

Creswell, J. W. 2014. Research Design: Qualitative, Quantitative, and Mixed-Method Approaches – 4th edition. Sage Publications.

Gacenga, T. C.-S.-G. 2012. A proposal and evaluation of a design method in design science research. Electronic Journal on Business Research Methods, 89-100.

Leung, L. 2015. Validity, reliability, and generalizability in qualitative research. J Family Med Prim Care, 324-327.

Sbalchiero, S. 2017. In-Depth Interviews. The Blackwell Encyclopedia of Sociology, 1-3.

Electronic Sources

AltexSoft. 2018. Software Documentation Types and Best Practices [accessed 30 September 2020]. Available at: https://blog.prototypr.io/software-documentation-types-and-best-practices-1726ca595c7f

Arora. 2020. 10 Best NodeJS Frameworks for Developers [accessed 18 September 2020]. Available at: https://hackr.io/blog/nodejs-frameworks

Aston. 2015. A brief history of JavaScript [accessed 30 August 2020]. Available at: https://medium.com/@_benaston/lesson-1a-the-history-of-javascript-8c1ce3bffb17

Blondin, A. 2019. Choose the NodeJS Framework that Best Suits Your Needs [accessed 1 Septemer 2020]. Available at: https://blog.theodo.com/2019/03/choose-best-nodejs-framework/

Bouchefra. 2019. Introduction to Nest.js for Angular Developers [accessed 23 August 2020]. Available at: https://www.sitepoint.com/introduction-to-nest-js-for-angular-developers/

Brix. 2016. Why is a steep learning curve called 'steep' if it means you are making little progress over relatively much time/effort? [accessed 10 October 2020]. Available at: https://www.quora.com/Why-is-a-steep-learning-curve-called-steep-if-it-means-youre-making-little-progress-over-relatively-much-time-effort

Centizen Nationwide. 2019. Is Nest JS the next big thing? [accessed 12 September 2020] .Available at: https://medium.com/@centizennationwide/is-nest-js-the-next-big-thing-2b5413608612

Cleveroad. 2017. How to choose the best Node.js framework: Express.js, Koa.js, or Sails.js [accessed 10 September 2020]. Available at: https://www.cleveroad.com/blog/the-best-node-js-framework-for-your-project--express-js--koa-js-or-sails-js

Ciszewski, B. 2018. Node.js Community Support - How Your Web Application Can Benefit from Node.js [accessed 16 August 2020]. Available at: https://www.netguru.com/blog/how-the-node.js-developer-community-can-help-your-app-succeed#:~:text=An%20active%20community%20around%20the,promote%20your%20product%20and%20services.

Connie U. Smith, Lloyd G. Williams. 2001. Introduction to Software Performance Engineering [accessed 12 October 2020]. Available at: https://www.informit.com/articles/article.aspx?p=24009

DaringFireball. 2002. Markdown: Syntax [accessed 12 October 2020]. Available at: https://daringfireball.net/projects/markdown/

Demchenko, M. 2020. The Best NodeJS Frameworks [accessed 16 September 2020]. Available at: https://ncube.com/blog/the-best-nodejs-frameworks

DeMichele, T. 2018. Deductive, Inductive, and Abductive Reasoning Explained [accessed 20 August 2020]. Available at: http://factmyth.com/deductive-inductive-and-abductive-reasoning-explained/

Earl, J. 2019. What is NestJS and should I use it? [accessed 3 August 2020]. Available at: https://medium.com/@jtearl188/what-is-nest-js-and-should-i-use-it-b71c7646926b

F.Adam, M. 2017. The History and Impact of Node.js [accessed 12 September 2020]. Available at: https://nixa.ca/blog/the-history-and-impact-of-nodejs/

Ferguson, N. 2020. What is the difference between frontend and backend Web Development? [accessed 5 September 2020]. Available at: https://careerfoundry.com/en/blog/web-development/whats-the-difference-between-frontend-and-backend/

FIPP. 2019. Global Digital Subscriptions Snapshot [accessed 16 September 2020]. Available at: https://www2.fipp.com/l/685373/2019-10-23/28vb8

Geer, Z. 2019. Learn Express.js fundamentals with hands-on [accessed 16 September 2020]. Available at: https://medium.com/edureka/expressjs-tutorial-795ad6e65ab3#:~:text=Express.js%20is%20a%20fast,overshadowing%20the%20Node.js%20features.&text=Features%20of%20Express.js

Guerini. 2020. SentiWords [accessed 20 October 2020]. Available at: http://www.marcoguerini.eu/resources-software/100-sentiwords

Hat, R. 2020. What is an application architecture? [accessed 5 September 2020]. Available at: https://www.redhat.com/en/topics/cloud-native-apps/what-is-an-application-architecture

Hayani. 2018. JavaScript ES6 — write less, do more [accessed 16 August 2020]. Available at: https://www.freecodecamp.org/news/write-less-do-more-with-javascript-es6-5fd4a8e50ee2/#:~:text=JavaScript%20ES6%20brings%20new%20syntax,destruction%2C%20Modules%E2%80%A6%20and%20more.

Henderson, M. 2019. What is NodeJS and Why You need to learn it? [accessed 16 August 2020]. Available at: https://medium.com/@michaelhenderson/what-is-nodejs-and-why-you-need-to-learn-it-f0760ba9a76a

Indianscribes. 2019. 4 rules of verbatim transcription [accessed 20 October 2020]. Available at: https://www.indianscribes.com/4-rules-of-verbatim-transcription/

Johnston, J. 2020. A beginner’s guide to web application development (2020) [accessed 16 August 2020]. Available at: https://www.budibase.com/blog/web-application-development/

Jung, D. 2019. Node.js Backend Frameworks [accessed 17 August 2020]. Available at: https://medium.com/@danielmjung/node-js-backend-frameworks-ce35a1781ecc#:~:text=Node%20is%20a%20popular%20choice,manager%20(such%20as%20NPM).

Kagan, J. 2020. Learning Curve [accessed 10 September 2020]. Available at: https://www.investopedia.com/terms/l/learning-curve.asp

Kalaitzis, D. 2019. NestJS - The good, the bad, and the ugly [accessed 5 September 2020]. Available at: https://medium.com/@dimosthenisK/nestjs-the-good-the-bad-and-the-ugly-bc39ba32eb52

Kiss. 2020. Exploring NestJS - Nest’s module system [accessed 5 September 2020]. Available at: https://medium.com/javascript-in-plain-english/exploring-nestjs-nests-module-system-88c6d7ad0970

Kreuzer. 2019. Step up your game, start using Nest! [accessed 5 September 2020]. Available at: https://medium.com/@kevinkreuzer/step-up-your-game-start-using-nest-36674f732565#:~:text=NestJS%20is%20a%20fully%2Dfeatured,coupled%2C%20and%20easily%20maintainable%20applications.

McHenry. 2020. Why You Should Use NestJS for Your Next Project [accessed 5 September 2020]. Available at: https://codeburst.io/why-you-should-use-nestjs-for-your-next-project-6a0f6c993be

Nations, D. 2020. What Is a Web Application? [accessed 16 August 2020]. Available at: https://www.lifewire.com/what-is-a-web-application-3486637

NestJS Documentation. 2020. Documentation [accessed 5 September 2020]. Available at: https://docs.nestjs.com/

Oluyemi, O. K. 2018. 10 Node Frameworks to Use in 2019 [accessed 20 August 2020]. Available at: https://scotch.io/bar-talk/10-node-frameworks-to-use-in-2019#:~:text=Benefits%20of%20Node%20frameworks,-Node.&text=js%20allows%20you%20to%20write,same%20coding%20pattern%20all%20through.

Oreofe, O. 2019. NestJS Framework Series: A gentle Introduction [accessed 7 September 2020]. Available at: https://medium.com/@oreofeolurin/nestjs-framework-series-a-gentle-introduction-7d8d2b7ca89d#:~:text=An%20introduction%20to%20building%20scalable%20NodeJS%20apps%20with%20NestJS&text=Scaling%20the%20app%20makes%20it,add%20more%20lines%20of%20codes.

Peters, S. 2019. A Brief History of JavaScript: from Netscape to Frameworks [accessed 16 August 2020]. Available at: https://blog.bitsrc.io/a-brief-history-of-javascript-from-netscape-to-frameworks-74bf4774eeef

Rahman. 2019. Why I choose NestJS over other Node JS frameworks [accessed 8 September 2020]. Available at: https://medium.com/monstar-lab-bangladesh-engineering/why-i-choose-nestjs-over-other-node-js-frameworks-6cdbd083ae67

Samanta. 2019. Nest js Tutorial Series — Part 3: Providers, Services & Dependency Injection [accessed 8 September 2020]. Available at:

https://medium.com/@kaushiksamanta23/nest-js-tutorial-series-part-3-providers-services-dependency-injection-a093f647ce2e

Sayantini. 2020. What is Software Testing? All you need to know about methods and testing [accessed 20 September 2020]. Available at: https://www.edureka.co/blog/what-is-software-testing/

Skowronski. 2019. Opinionated or Not: Choosing the right framework for the job [accessed 18 September 2020]. Available at: https://dev.to/heroku/opinionated-or-not-choosing-the-right-framework-for-the-job-4e9f

Stack Overflow. 2020. Developer Survey Results [accessed 26 August 2020]. Available at: https://insights.stackoverflow.com/survey/2019#technology

Starling. 2019. Selecting the right Node.js framework for your app [accessed 3 September 2020]. Available at: https://www.techwell.com/techwell-insights/2019/11/selecting-right-nodejs-framework-your-app

RisingStarsJS. 2020. 2019 JavaScript Rising Stars [accessed 3 September 2020]. Available at: https://risingstars.js.org/2019/en/

Rouse. 2019. Web application (Web app) [accessed 19 August 2020]. Available at: https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app

Tachintha. 2020. Choose the best JavaScript framework for your server-side development [accessed 18 September 2020]. Available at: https://codeburst.io/choose-the-best-javascript-framework-for-your-server-side-development-28951fd87d35

Thematic. 2020. Complete Guide to Sentiment Analysis [accessed 18 October 2020]. Available at: https://getthematic.com/insights/sentiment-analysis/

Vision Exault. 2020. Web development process and planning [accessed 8 August 2020]. Available at: http://www.visionexalt.com/web-application-development.php

Wallace Foundation. Conducting In-depth Interviews [accessed 8 September 2020]. Available at: https://www.wallacefoundation.org/knowledge-center/Documents/Workbook-E-Indepth-Interviews.pdf

Worthy. 2019. Automated or Manual Transcription Service: Which Is Better? [accessed 20 October 2020]. Available at: https://www.gmrtranscription.com/blog/automated-or-manual-transcription-service-which-is-better#:~:text=Manual%20transcription%20is%20the%20process,accuracy%2C%20speed%2C%20and%20timeliness.

Wyciślik-Wilson. 2020. How to find and use the free screen recorder tool in Windows 10 [accessed 20 October 2020]. Available at: https://betanews.com/2020/01/20/windows-10-screen-record-xbox-game-bar/

Yadav. 2020. Getting started with NestJS [accessed 8 September 2020]. Available at: https://medium.com/better-programming/getting-started-with-nestjs-a4e8b0b09db4

Yadav. 2020. Introduction to NestJS Services [accessed 12 September 2020]. Available at: https://medium.com/better-programming/introduction-to-nestjs-services-2a7c9a629da9