Experimental Evaluation of Low Code development, Java Swing and JavaScript programming

André Calçada
Department of Systems and Computer Engineering, Polytechnic of Coimbra, Coimbra Institute of Engineering (ISEC), Rua Pedro Nunes, 3030-199 Coimbra, +351239790200
a21250156@isec.pt

Jorge Bernardino
Department of Systems and Computer Engineering, Polytechnic of Coimbra, Coimbra Institute of Engineering (ISEC), Rua Pedro Nunes, 3030-199 Coimbra, +351239790200, Centre for Informatics and Systems of the University of Coimbra (CISUC), Polo II, Pinhal de Marrocos, 3030-290 Coimbra, +351239790000
jorge@isec.pt

ABSTRACT

Low Code is a technology that has been gaining popularity over the years, due to its potential and simplicity. But so far there has not been an experimental evaluation with other programming methods. This paper aims to introduce Low Code development technology and compare it with Java Swing programming and manual development with HTML (HyperText Markup Language), CSS (Cascading Style Sheet), and JavaScript. These technologies are compared using the following metrics: development time, execution time, and the number of written code lines. In this evaluation, two applications are implemented, a simple calculator, and a text editor, developed in all technologies. It is concluded that it is faster to develop applications in Low Code but in terms of execution time, these are usually slower. Although the Low Code development is still at a somewhat embryonic stage which leads to some bugs and errors, Low Code development is better in general than Java Swing programming, and somewhat similar to manual programming with HTML, CSS, and JavaScript. Another benefit is that Low Code generates HTML and CSS automatically.

CCS CONCEPTS

• **General and reference** → Cross-computing tools and techniques; Evaluation; Cross-computing tools and techniques; Experimentation; • **Software and its engineering** → Software notations and tools; General programming languages; Language features.

KEYWORDS

Java Swing, Low Code, JavaScript

ACM Reference Format:
André Calçada and Jorge Bernardino. 2022. Experimental Evaluation of Low Code development, Java Swing and JavaScript programming. In *International Database Engineered Applications Symposium (IDEAS'22), August 22–24, 2022, Budapest, Hungary.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3548785.3548792

IDEAS'22, August 22–24, 2022, Budapest, Hungary

ACM ISBN 978-1-4503-9709-4/22/08…$15.00
https://doi.org/10.1145/3548785.3548792

1 INTRODUCTION

Low Code development is a technology that facilitates programming by diminishing the handwritten code and allowing non-programmers to build and have a more active presence in the development process of the application. Low-code is a collection of tools that enables developers to avoid hand-coding and reduces the development effort of having an application ready for production.

There are many ways of developing an application but when it comes to programming, users sometimes have difficulties in choosing the language and the type of programming. For Low Code development, Neptune Planet 9 is used, a Low Code Development Platform (LCDP), for Java Swing programming NetBeans IDE (Integrated Development Environment), and for manual development with HTML, CSS, and JavaScript Visual Studio Code is applied.

Java is one of the most recognized and used programming languages, and Swing is a Java toolkit that allows programmers to easily build a User Interface (UI) for their Java applications. JavaScript is a lightweight, interpreted, object-oriented programming language mostly used to program web pages.

These technologies are chosen, Java Swing and JavaScript to compare with Low Code because Java Swing has a similar way of making the UI but runs on the operating system, unlike Low Code applications that run on a browser. JavaScript is chosen because it is the same type of programming that LCDPs use, but the UI development is manual.

These technologies can use database systems as external services, where the communication with the database is made through an API (Application Programming Interface) or another communication protocol, where a request is sent to the database and an answer is given in response. Depending on the database it is possible to manage the database through these requests, facilitating the development work.

In this paper, Low Code development, Java Swing programming, and manual development with HTML, CSS, and JavaScript are compared, by developing two applications in all types of programming, which has not been done before according to [1]. The platforms chosen to develop the apps are based on our experience with these types of platforms and because they are free of charge. The objective of this study is to help users to know what type of programming to choose through the comparative evaluation of the developed applications.

The rest of this paper is structured as follows. Section 2 introduces background concepts and the methodology applied. Section 3

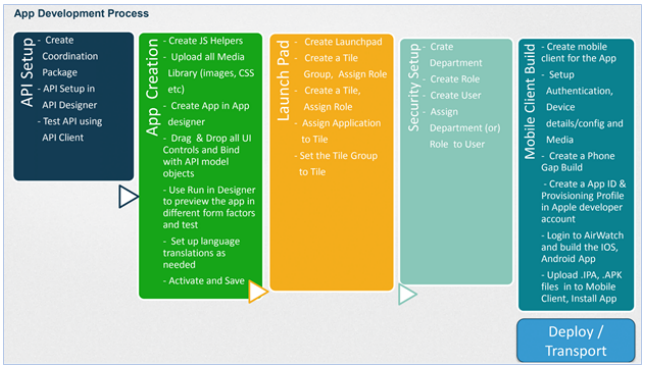

Figure 1: Low code app development process. Source: [6].

exhibits the results. Section 4 presents the discussion of the results. Section 5 describes the open problems. Finally, section 6 presents the main conclusions and future work.

2 BACKGROUND

This section introduces Low Code development, LCDPs, Java Swing, HTML, CSS, JavaScript, Node.js, IDEs, and the methodology used in the experiments.

2.1 What is Low Code?

The term 'Low Code' was introduced by Forrester market research company in June 2014, where these platforms are described as extraordinarily disruptive [2].

Low Code applications are developed using model-driven engineering principles and taking advantage of cloud infrastructures, automatic code generation, declarative, high level, and graphical abstractions to develop entirely functioning applications, meaning that these applications are mostly made through drag and drop of objects [3].

2.2 Low Code Development Platform

LCDPs emerged in the early 2000s helping development teams work faster. The Low Code Development Platform market started back in 2011, speeding up the development and maintenance processes.

An LCDP is set on a cloud or locally [3], allowing the development of Low Code applications using minimal code writing. Its objective is to give, to different types of users, the possibility to create applications in an easy, simple, and fast way [4].

Each LCDP has its programming language, such as Java, JavaScript, Python, and others [2]. LCDPs allow the development of distinct types of applications such as web apps and mobile apps [5].

The development of a Low Code application has the following processes: API Setup, App Creation, LaunchPad, Security Setup, Mobile Client Build, and Deploy/Transport, as shown in Figure 1.

API setup is a process where the developer creates APIs, which can be made in an LCDP or imported. The App Creation process is where the developer creates the app/module. In this process, LaunchPad is used to transform a module or various modules into a suitable application. The Security setup process is where is set up who has access to the application or which parts of it. Mobile Client Build process is where the application is prepared to become a mobile app, and at last Deploy. It should be mentioned that not all of these processes are required to implement an application, which depends on the type of application that is developed [6].

2.3 Java Swing

Java is an object-oriented language, and Swing is a widget toolkit GUI (Graphic User Interface). Swing started to be developed in 1996, supports multithreading, and allows Java programmers to easily create a UI for an application.

Some of the Swing components are labels (text), text areas, buttons, tables, frames (application page), combo box, scroll pane (object to scroll page), file chooser, (to get a file for reading or to save), menus, toolbars, and others [7].

2.4 HTML

HTML is the standard markup language for documents displayed in a web browser, defining their meaning and structure. There are also other technologies used to describe a web page's appearance (CSS) or behaviour (JavaScript). "Hypertext" refers to links that allow users to create, store, and view text, connecting web pages directly so that "travelling" from one to another is quicker. Links are a fundamental characteristic of the Web.

HTML uses "markup" to distinguish text, images, and other content for display in a Web browser. An HTML element is detached from other text in a document by "tags", which consist of the element name surrounded by "<" and ">". HTML markup includes special "elements" such as <head>, <title>, <p>, <button>, and many others. The name of an element inside a tag is not case-sensitive, that is, it can be written in uppercase, lowercase, or a combination. For example, the <title> tag can be written as <Title>, <tItle>, or in any other style [8].

2.5 Cascading Style Sheet (CSS)

CSS is a stylesheet language for describing the presentation of elements in a HTML or XML (eXtensible Markup Language) document, including XML dialects such as SVG (Scalable Vector Graphics), MathML (Mathematical Markup Language) or XHTML (eXtensible HyperText Markup Language). CSS describes colours, layout, and fonts of Web pages, allowing to adapt the presentation to different types of devices. CSS is independent of HTML and can be used with any XML-based markup language.

CSS is one of the core languages of the web and is standardized across Web browsers according to W3C specifications [9]. Formerly, the development of various parts of CSS specification was synchronous, which allowed the versioning of the latest recommendations. There are new versions of CSS such as CSS1, CSS2.1, and CSS3. However, CSS4 has never been released.

Since CSS3, the scope of the specification increased considerably with CSS modules differing significantly. Therefore, it became more efficient to develop and release recommendations separately per module. Currently W3C, as an alternative of versioning the CSS specification, takes periodically a snapshot of the latest stable state of the CSS specification [10], [11].

2.6 JavaScript

JavaScript is a dynamic and lightweight scripting language, and it has broad participation in website and web application services. JavaScript has become one of the most widely used languages for Web development, however, many non-browser environments also use it, such as Node.js, Apache CouchDB, and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, which is used in web pages interface design, creating cookies, mobile apps, games, and so on.

JavaScript should not be confused with the Java programming language. The two programming languages have very different syntax, semantics, and uses [12]. The main difference between JavaScript and Java is that JavaScript code is written completely in text and needs only to be interpreted, while Java, on the other hand, must be compiled.

2.7 Node.js

Node.js is an increasingly popular event-driven architecture, open-source, cross-platform, back-end JavaScript runtime environment, widely used in server-side and desktop applications. Node.js executes JavaScript code outside a web browser and provides an effective asynchronous programming model. In Node.js, time-consuming IO operations, e.g., file access operations, can be delegated as asynchronous tasks, running in the dedicated threads. Thus, Node.js applications are not blocked by these time-consuming IO operations.

Node.js provides an effective asynchronous event-driven programming model and supports asynchronous tasks allowing developers to use JavaScript to write command-line tools and produce dynamic web page content before the page is sent to the user's browser. Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server-side and client-side scripts. These design choices aim to optimize throughput and scalability in web applications with many input/output operations [13].

2.8 Integrated Development Environment (IDE)

IDEs provide a convenient standalone solution that supports developers during various phases of software development and are designed to include all programming tasks in one application. One of the main benefits of an IDE is that they offer a central interface with the tools that a developer needs, including the following [14]:

- Code editor: Designed for writing and editing source code, these editors are distinguished from text editors because their function is to simplify and enhance the process of writing and editing code for developers.
- Compiler: Compilers transform source code that is written in a human-readable language into a machine-readable language.
- Debugger: Debuggers are used during tests and can help developers debug their application code.
- Build automation tools: These tools help developers to automate common developer tasks to save time.

Additionally, some IDEs may also contain [14]:

- Class browser: Used to study and reference properties of an object-oriented class hierarchy.
- Object browser: Used to inspect objects instantiated in a running application program.
- Class hierarchy diagram: This allows developers to visualize the structure of object-oriented programming code.

An IDE can be a stand-alone application, despite the fact it could be also included as part of one or more compatible applications [14].

2.9 Methodology

To compare Low Code development, Java Swing, and JavaScript programming, two applications are developed in all technologies. These applications are developed and tested on a computer with an Intel i5-8250U CPU, Windows 10, 512 GB SSD, and 8 GB of RAM (Random Access Memory).

The first application is a calculator with the basic math operations, sum, subtraction, multiplication, and division. The second is a simple text editor, like a simple notepad. In the implementations of these applications, the following metrics are assessed: development time, in minutes; execution runtime, in milliseconds (ms), this is the time that the application takes to set up the UI; the number of written code lines; and the operations execution time, in milliseconds. The execution runtime and operations execution time that is considered is the average of five executions of the application. To ensure that the results are viable, in JavaScript and Low Code, since an LCDP and its applications run on a browser, Brave browser, introduced in subsection 2.9.3 is used in incognito mode to get the execution time the browser cache is cleared before each run. Also, in Java Swing, before each run, CachemanXP program is executed to clean the RAM.

The calculator has a text area to show the input and results, and alongside the basic operation buttons, mentioned above, it has the numbers, from 0 to 9, the equal, the delete, the decimal point ".", and the clear buttons. The calculator doesn't give any errors, when inserting two operations it must do the first operation and with its result do the second (when introduced 5+2*10, the first operation is computed, 5+2=7, and then the second operation, 7*10=70), this may lead to mathematical miscalculation but it's not important for the evaluation. When there is an operation character in the text area, and another is inserted the first one must be replaced (if there is have "2+" in the text area and a "-" is entered, the result is "2-"). To get a valid operation, a number must be entered followed by an operation and another number and then press the equal button to get the result or add another operation.

The text editor is a simple text area with two buttons, one to load and the other to save the text. The text editor must give the user the possibility to choose where to save or load the file. The load and save functionalities must only allow .txt files.

The Low Code applications are implemented using Neptune Planet 9 LCDP, version 2.3.1, which will be introduced in subsection 2.9.1. In the Java Swing applications, NetBeans IDE is used, version 12.5, which is described in subsection 2.9.2.

The JavaScript applications are implemented with Visual Studio Code, version 1.63.2, which is presented in subsection 2.9.4. It

should be noted that in pure JavaScript, it is not possible to let the user choose where to save a file, because it is always saved in the Downloads folder [15]. These platforms are chosen considering our knowledge about these types of platforms. NetBeans is chosen because it is one of the most user-friendly IDE, however, it does not support JavaScript. Due to this fact, with JavaScript, Visual Studio Code is chosen, which is one of the best IDEs, based on their usage and popularity [16].

The description of tests execution is presented in Tables 1 and 2 for the calculator and text editor, respectively. The tests are based on the following operations:

- "Add number" operation includes adding a number and the "." to the text area.
- "Add operation" adds a mathematical operator like "+" to the text area.
- "Add operation (S)" is the same as "Add operation". However, when there is already an operation character this is substituted, for example, when there is "1+" in the text area and a "-" is introduced, the result is "1-".
- "Add operation (R)" is the same as "Add operation". However, when there is already a valid equation its result is calculated. For example, "1+1+" turns "2+".
- "Delete" operation deletes a character of the text area.
- "Delete (N)" operation is when the text area is empty and the delete button is pressed.
- "Clear" operation clears the whole text area.
- "Result" operation calculates the result of the equation in the text area.
- "Result (N)" is the same as the result operation but when the equation is not valid, for example when there is a "1+" in the text area the equation is not altered, the value "1+", in the text area stays unchanged.
- "Load" operation loads the text from a .txt file to the text area
- "Save" operation saves the text in the text area into an existing .txt file. This is not possible to evaluate on JavaScript text editor, because when trying to save on an existing file, the browser creates a new file by adding "(1)" to the new file name.
- "Save (N)" operation saves the text in the text area into a new .txt file.

Some observations: 1) The calculator tests start with the calculator clear of values. 2) In the calculator tests the time to select a button is not considered, only the time of the operation. 3) The text editor tests do not take into consideration the time to write a text, and the same text is used for all tests. 4) In the text editor tests the time to select a file is not considered, only the time to save/load a file.

2.9.1 Neptune Planet 9. Neptune Planet 9 is an LCDP that uses as core technologies HTML, and CSS, and uses Node.js as the programming language. Its architecture is shown in Figure 2 and Figure 3.

The development of applications is done in the App Designer component, where data and resources from the Store, ODATA, Media Library, API Designer, and Server Scripts are used. Table Definition is used to define data types, which is not always necessary, as these types can be automatically imported from an external database. The LaunchPad serves as a Modules portal, where each developed module is added, which is possible with the use of Tiles that allows navigation for each module. Tiles are organized into groups through Tiles Groups. Users are configured in the Users component and can be created in the LCDP or obtained externally. In the LaunchPad each user has access to the modules depending on their role. Finally, the LaunchPad can run on the Web or in an APK (Android Package) that can be generated in the Mobile Client component for mobile devices [6].

Modules can be created from a workflow using the Workflow Designer component and the Theme Designer component can be used to create a predefined theme for the entire LaunchPad to visually enhance it [6].

2.9.2 NetBeans IDE.. NetBeans is a free IDE where a programmer can develop applications in languages like Java, C, C++, PHP, and others. This IDE supports many platforms such as Windows, Linux, Solaris, and macOS, and it supports many types of API services. NetBeans like many other IDEs allows a programmer to develop many kinds of applications, from a plain text editor to a complex web app [17], [18].

2.9.3 Brave Browser. Brave is a free and open-source web browser developed by Brave Software, Inc. based on the Chromium web browser. Brave's popularity is on the increase, driven by privacy-by-default functionality, which automatically blocks online advertisements and website trackers. Brave is developed upon the open-source Chromium browser project which promotes faster and safer browsing. As the project is open source, Brave can make use of the code for their product, adding additional features on top. Privacy features include ad-blocking, antitracking functionality, and cryptocurrency offerings [19].

2.9.4 Visual Studio Code. Visual Studio Code is a cross-platform editor implemented by Microsoft for Windows, Linux, and macOS. In 2016, Visual Studio Code has progressed from the public preview stage and was released to the Web. Then, it has quickly become one of the top editors in terms of the popularity.

Visual Studio Code is a very powerful code-focused development environment expressly designed to make it easier to write web, mobile, and cloud applications using languages that are available to different development platforms and to support the application development lifecycle with a built-in debugger and integrated support for the popular Git version control engine [20].

3 EXPERIMENTAL RESULTS

This section presents the experimental results of the developed applications.

Figures 4, 5 and 6 presents the user interface of the calculator using Java Swing, Low Code, and JavaScript, respectively.

Figures 7, 8 and 9, presents the user interface of the text editor using Java Swing, Low Code, and JavaScript, respectively.

Figure 10 presents the text editor example of the *fileChooser* window for load operation.

Table 3 presents the results of the development of each application concerning development time, execution runtime, and handwritten code lines. In Tables 3, 4, and 5, Java Swing is referred to as JSW, Low Code as LC, and JavaScript as JSC.

Table 1: Calculator test execution

Operation	Description
Add number	1. Click on a number (random) 2. Read time of operation (1)
Add operation	1. Click on an operation (random) 2. Read time of operation (1)
Add operation (S)	1. Click on an operation (random) 2 Click on another operation (random) 2. Read time of operation (2)
Add operation (R)	1. Click on a number (random) 2. Click on an operation (random) 3. Click on a number (random) 4. Click on an operation (random) 5. Read time of operation (4)
Delete	1. Click on a number/s or operation/s (random) 2. Click on delete 3. Read time of operation (2)
Delete (N)	1. Click on delete 2. Read time of operation (1)
Clear	1. Click on a number/s or operation/s (random) 2. Click on clear 3. Read time of operation (2)
Result	1. Click on a number (random) 2. Click on an operation (random) 3. Click on a number (random) 4. Click on equal 5. Read time of operation (4)
Result (N)	1. Click on a number/s or operation/s (random) 2. Click on equal 3. Read time of operation (2)

Table 2: Text editor test execution

Operation	Description
Load	1. Click on Load 2. Select a file 3. Read time of operation (2)
Save	1. Write the text 2. Click on save 3. Select an existing file 4. Read time of operation (3)
Save (N)	1. Write the text 2. Click on save 3. Choose file and name location 4. Read time of operation (3)

Table 4 presents the average runtime of each operation of the calculator, in milliseconds.

Table 5 presents the runtime for each operation of the text editor in milliseconds.

4 DISCUSSION OF THE EXPERIMENTAL RESULTS

This section presents the discussion of the results of the previous section. To discuss these results it must be considered that Low Code and manual programming with HTML, CSS, and JavaScript are similar except Low Code generates HTML and CSS automatically.

4.1 Discussion of the Results: Comparing Low Code with Java Swing and JavaScript

As shown in Table 3 Java Swing and JavaScript applications have a better performance but at a bigger cost in terms of development time and hand-written code, compared to Low Code applications. Nevertheless, in a deeper analysis:

- In terms of development time, developing in Low Code is in average, 1.74 times faster than programming in Java Swing and 1.10 times faster than programming in JavaScript:
  - ◯ In the Low Code calculator, development is 1.73 times faster than Java Swing.
  - ◯ In the Low Code text editor, the development is 1.75 times faster than Java Swing.
  - ◯ In the Low Code calculator, development is 1.06 times faster than JavaScript.

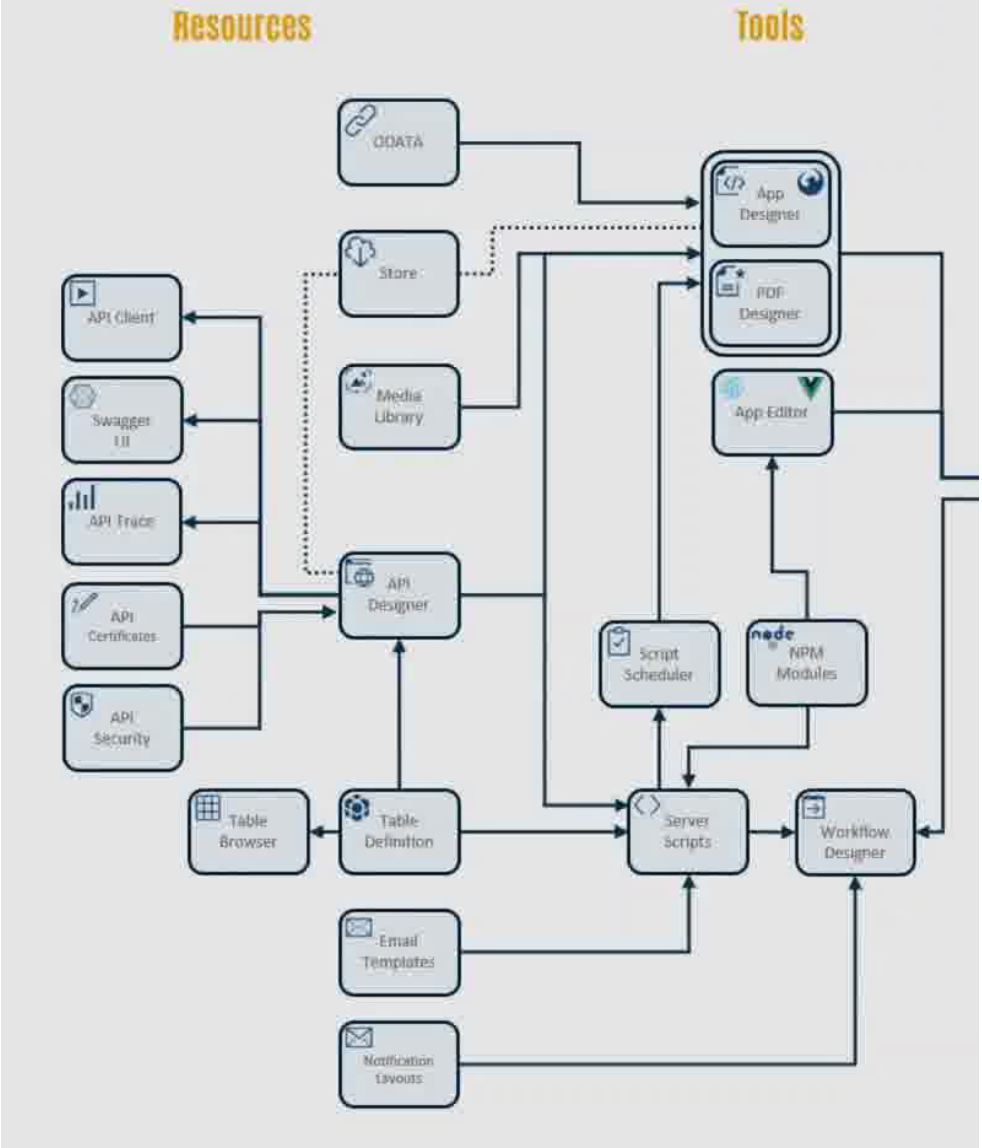

Figure 2: Neptune Planet 9 architecture, resources and tools. Source: [6].

Table 3: Results of the experiments

Application / Property	Development time (minutes)	Execution runtime (ms)	Hand-written code lines
Calculator (JSW)	57	156.34	72
Calculator (LC)	33	1082.60	64
Calculator (JSC)	35	37.40	106
Text Editor (JSW)	14	790.63	39
Text Editor (LC)	8	1494.60	12
Text Editor (JSC)	10	33.00	27

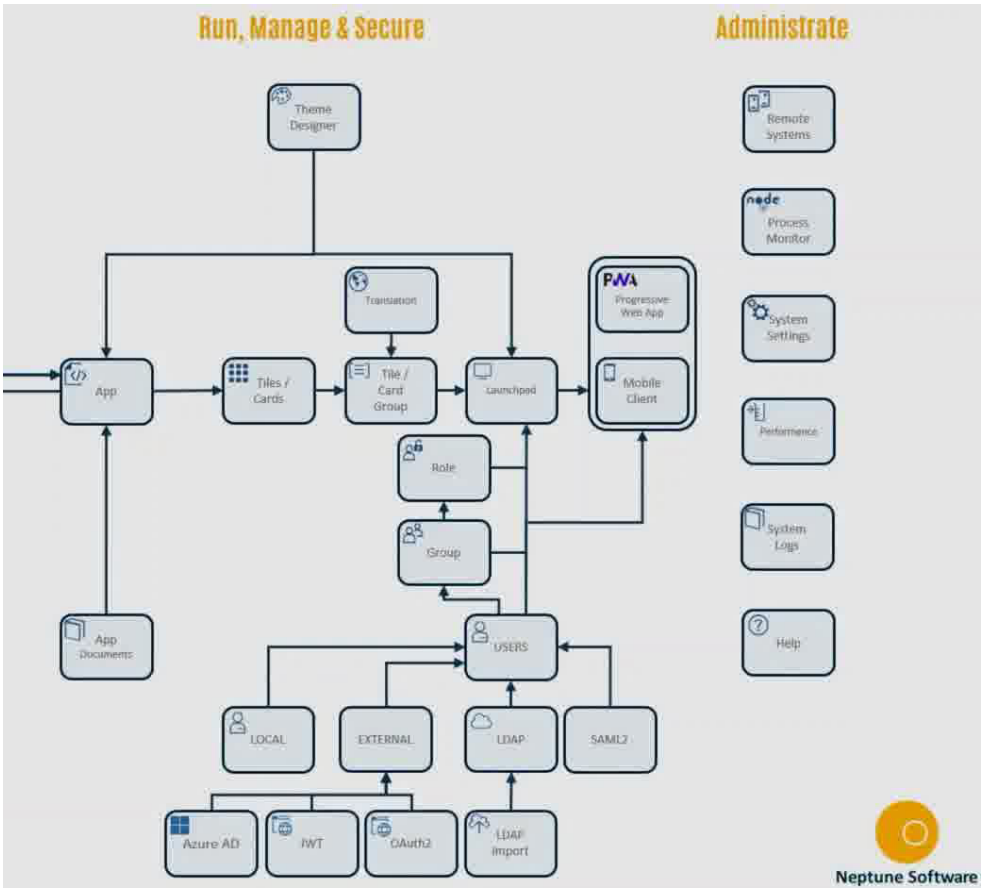

Figure 3: Neptune Planet 9 architecture, run, manage & secure and administrate. Source: [6].

Table 4: Results of the calculator operations in ms using JSW, LC, and JSC

Operation	Calculator (JSW)	Calculator (LC)	Calculator (JSC)
Add number	3.14	0.56	0.14
Add operation	3.25	0.52	0.16
Add operation (S)	1.05	0.74	0.06
Add operation (R)	1.78	0.80	0.48
Delete	0.42	0.26	0.10
Delete (N)	0.10	0.44	0.12
Clear	0.10	0.38	0.22
Result	1.26	0.22	0.60
Result (N)	0.21	0.44	0.16

Table 5: Runtime results for the text editor (in ms) using JSW, LC, and JSC

Operation	Text editor (JSW)	Text editor (LC)	Text editor (JSC)
Load	37.33	0.30	0.34
Save	5.66	0.92	Not possible
Save (N)	7.13	0.86	1.02

- In the Low Code text editor, the development is 1.25 times faster than JavaScript.

- In terms of execution runtime, Java Swing applications are in average, 2.72 times faster and JavaScript applications 36.61 times faster, when compared to Low Code applications:
  - In Java Swing, the calculator, runtime is 6.92 times faster than Low Code.
  - In Java Swing, the text editor runtime is 1.89 times faster than Low Code.

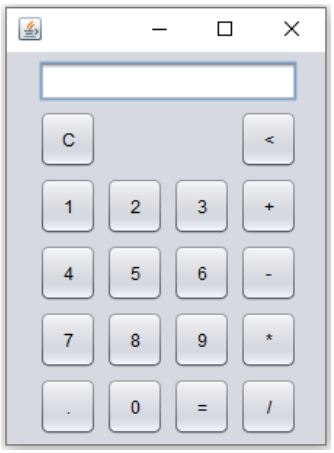

Figure 4: Java Swing calculator based on NetBeans.

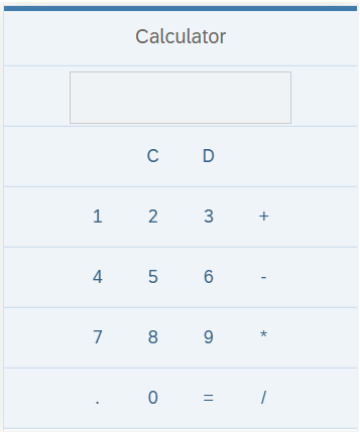

Figure 5: Low Code calculator based on Neptune P9 and Brave browser.

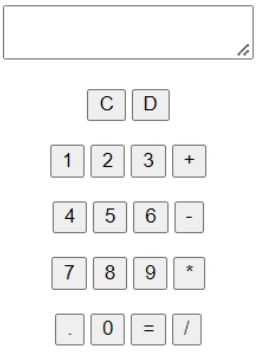

Figure 6: JavaScript calculator based on Visual Studio Code and Brave browser.

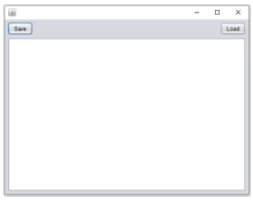

Figure 7: Java Swing text editor based on NetBeans.

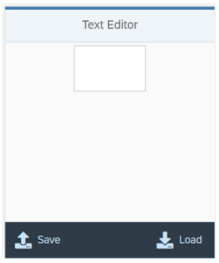

Figure 8: Low Code text editor using Neptune P9 and Brave browser.

Figure 9: JavaScript text editor using Visual Studio Code and Brave.

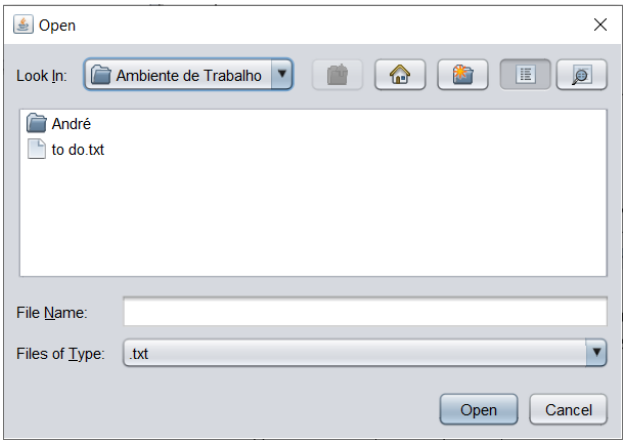

Figure 10: Load operation *fileChooser* using NetBeans.

- ○ In JavaScript, the calculator, runtime is 28.95 times faster than Low Code.
- ○ In JavaScript, the text editor runtime is 45.29 times faster than Low Code.

- In terms of hand-written code lines, Low Code applications have in average, 2.18 times less code, compared to Java Swing and 1.75 times less code, compared to JavaScript:
  - ◯ The low Code calculator has 1.13 times fewer code lines than Java Swing.
  - ◯ The low Code text editor has 3.25 times fewer code lines than Java Swing.
  - ◯ The low Code calculator has 1.66 times fewer code lines than JavaScript.
  - ◯ The low Code text editor has 2.25 times fewer code lines than JavaScript.

As presented in Table 4 Low Code calculator has a better performance in terms of operation execution runtime, compared to the Java Swing calculator, and it is similar to the JavaScript calculator. Considering all the operations:

- In average the calculator operations are 2.14 times faster in JavaScript than in Low Code and 2.59 times faster in Low Code than in Java Swing.
- Java Swing calculator is faster in three operations "Delete (N)", "Clear" and "Result (N)". This is because Java is usually faster and, in this case, there is only one line of code to execute (in "Delete (N)" there's an if, in "Clear" there's a set value, in "Result (N) there's an if), and the difference, compared to Low Code, is 0.34, 0.28, and 0.23, respectively. In each operation there exists a negligible difference, thus considering that it is much slower in other operations.

As shown in Table 5, the Low Code text editor has a better performance in terms of operation execution runtime. For example:

- "Load" operation is 124.43 times faster in Low Code than in Java Swing, and 1.13(3) times faster than JavaScript.
- "Save" operation is 6.15 times faster in Low Code than in Java Swing.
- "Save (N)" operation is 8.20 times faster in Low Code than in Java Swing and 1.19 times faster than in JavaScript.

The Low Code text editor is in average of execution runtime of operations, 24.10 times faster than the Java Swing text editor, and 1.17 times faster than the JavaScript text editor. Low Code and JavaScript are faster than Java Swing in this case because unlike JavaScript, Java Swing needs to understand what operating system it is running on to make a system call, JavaScript does not need that because it is managed by the browser [21].

Comparing only Java Swing and JavaScript depends on the application's complexity and nature and it is presented in the next subsection.

4.2 Discussion of the Results: Java Swing with JavaScript

As presented in Table 3, JavaScript applications have better performance and take less time to develop than Java Swing but need more handwritten code lines. Making a deeper analysis:

- In terms of development time, developing in JavaScript is, in average, 1.58 times faster than programming in Java Swing.
  - ◯ In the JavaScript calculator, development is 1.63 times faster.
  - ◯ In the JavaScript text editor, the development is 1.40 times faster.
- In terms of execution runtime, JavaScript applications are, in average, 13.45 times faster than Java Swing applications.
  - ◯ In JavaScript, the calculator, runtime is 4.18 times faster.
  - ◯ In JavaScript, the text editor runtime is 33.96 times faster.
- In terms of hand-written code lines, Java Swing applications have, in average, 1.20 times less code, compared to JavaScript.
  - ◯ In the Java Swing calculator, the number of code lines is 1.47 times less.
  - ◯ In the Java Swing text editor, the number of code lines is 1.44 times less.

Table 6: Operations average for all experiments

Operation	Java Swing	Low Code	JavaScript
Code lines	55.50	38.00	66.50
Execution runtime	473.49	1288.60	35.20
Development time	34.00	20.50	22.50
Operations runtime	5.12	0.54	0.310

As shown in Table 4, the JavaScript calculator has a better performance in terms of operations execution runtime, compared to the Java Swing calculator. Considering all the calculator operations:

- In average the calculator operations in JavaScript are 5.54 times faster than in Java Swing.
- Java Swing calculator is only faster in two operations "Clear and "Delete (N)" and the difference is 0.02 and 0.12 milliseconds, respectively, which is a negligible difference, considering that Java Swing is much slower in other operations. JavaScript calculator is 5.54 times faster than Java Swing calculator in average of runtime execution of all operations.

As presented in Table 5 JavaScript text editor has a better performance in terms of operation execution runtime. Making a deeper analysis:

- "Load" operation is 109.79 times faster in JavaScript than in Java Swing.
- "Save" operation is not comparable because it cannot be tested in the JavaScript text editor.
- "Save (N)" operation is 6.99 times faster in JavaScript than in Java Swing.

The JavaScript text editor is 32.69 times faster than the Java Swing text editor considering the average execution runtime of all operations.

4.3 Summary of all Results

Table 6 presents the average of the operations, and Table 7 the standard deviation for all the previous results taking into consideration the number of code lines, execution runtime (ms), development time (minutes), and operations runtime (ms). These results are obtained from Table 3 and from Tables 4 and 5.

Table 7: The standard deviation for all experiments

Operation	Java Swing	Low Code	JavaScript
Code lines	16.50	26.00	39.50
Execution runtime	317.15	206.00	2.2
Development time	20.00	12.50	12.50
Operations runtime	9.95	0.23	0.28

With the results of Tables 6 and 7, the differences between these technologies can be seen more clearly:

- Low Code is 9.48 times faster than Java Swing and 1.74 times slower than JavaScript, in terms of operation runtime.
- JavaScript is 16.52 times faster than Java Swing, in terms of operation runtime.
- JavaScript has the lowest standard deviation compared to the other technologies, except in the number of code lines, where Java Swing has the lowest standard deviation.

Note that the values comparing these technologies may vary due to the application's complexity and its nature.

5 OPEN PROBLEMS

This section presents the problems addressed in this paper. The comparison of Low Code, Java Swing, and Java Script technologies can be addressed in many ways. In this paper, each technology was compared by looking at some of its application's metrics, like execution runtime. This practical approach gives developers a perspective of how these technologies can help them in their job.

As described in [1], there is no comparison of Low Code with other technologies, which leaves space for this type of research. This gap was filled with our research, but there is still significant work to be done. We identify the following open research problems:

- Comparison with other Low Code technologies.
- Using Low Code with more complex applications including database development.
- Comparing the frontend UI, backend logic, and data store, to be developed using Low Code technologies.
- Using machine learning in Low Code vs machine learning in other technologies.
- Research of Low Code Development Platforms usage for communication, human behaviour, and decision-making.

6 CONCLUSIONS AND FUTURE WORK

Low Code development is a technology that speeds up the process of deploying an application version to the production environment. Low Code facilitates programming by diminishing the handwritten code and allowing non-programmers to build applications. These technologies may also simplify the work of database developers.

The Low Code development, Java Swing, and JavaScript programming have been experimentally evaluated and it can be concluded that Low Code applications are valuable when what is important is the development time and writing code. However, their runtime execution for the setup of the application is slower than Java Swing and JavaScript. Low Code and JavaScript are faster at executing most of the operations than Java Swing which leads to the conclusion that Low Code and JavaScript applications have a better performance. Their performance in terms of operation execution time is very similar, which occurs because Node.js is based on JavaScript and the applications of these technologies run on the same environment, a browser.

Despite the advantages of Low Code, it must be taken into consideration that Low Code has a big learning curve. It should also be noted that the compared technologies run in different environments, Java Swing runs on the operating system, and the other technologies run on a browser, which directly impacts performance.

It can be concluded that JavaScript is the best programming method in terms of execution runtime, although it may be the one with a larger number of code lines, depending on the applications.

As future work is intended to develop a study with other technologies, like .Net, and with more complex applications, such as a web app that manages users, so there can have a thorough comparison.

REFERENCES

[1] N. Prinz, C. Rentrop, and M. Huber, "Low-Code Development Platforms – A Literature Review," AMCIS 2021 Proceedings, 2021.
[2] Y. Luo, P. Liang, C. Wang, M. Shahin, and J. Zhan, "Characteristics and Challenges of Low-Code Development," 2021, doi: 10.1145/3475716.3475782.
[3] A. Sahay, A. Indamutsa, D. Di Ruscio, and A. Pierantonio, "Supporting the understanding and comparison of low-code development platforms," Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020, 2020, doi: 10.1109/SEAA51224.2020.00036.
[4] C. Silva, J. Vieira, J. C. Campos, R. Couto, and A. N. Ribeiro, "Development and Validation of a Descriptive Cognitive Model for Predicting Usability Issues in a Low-Code Development Platform," Human Factors, vol. 63, no. 6, 2021, doi: 10.1177/0018720820920429.
[5] "No-code/low-code: why should you be paying attention." https://venturebeat.com/2021/02/14/no-code-low-code-why-you-should-be-paying-attention/ (accessed Nov. 04, 2021).
[6] "Neptune." https://www.neptune-software.com/ (accessed Feb. 12, 2020).
[7] B. Cole, R. Eckstein, J. Elliott, M. Loy, D. Wood, and O. ' Reilly, "JavaTM Swing, 2nd Edition," 2002.
[8] "HTML: HyperText Markup Language | MDN." https://developer.mozilla.org/en-US/docs/Web/HTML (accessed Mar. 11, 2022).
[9] "Cascading Style Sheets." https://www.w3.org/Style/CSS/#specs (accessed Jan. 18, 2022).
[10] "CSS: Cascading Style Sheets | MDN." https://developer.mozilla.org/en-US/docs/Web/CSS (accessed Jan. 18, 2022).
[11] H. Wium. Lie and Bert. Bos, "Cascading style sheets: designing for the Web," p. 396, 1999.
[12] "JavaScript | MDN." https://developer.mozilla.org/en-US/docs/Web/JavaScript (accessed Jan. 18, 2022).
[13] "Node.js." https://nodejs.org/en/ (accessed Jan. 31, 2022).
[14] "What is an IDE or Integrated Development Environment?" https://www.veracode.com/security/integrated-development-environment (accessed Nov. 30, 2021).
[15] "Mozilla | MDN - downloads." https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/downloads/download (accessed Jan. 24, 2022).
[16] "TOP IDE index." https://pypl.github.io/IDE.html (accessed Nov. 30, 2021).
[17] Tim Boudreau, Jesse Glick, Simeon Greene, Vaughn Spurlin, and Jack Woehr, NetBeans: The Definitive Guide: Developing, Debugging, and Deploying Java Code, 1st ed. O'REILLY, 2002.
[18] "Apache NetBeans." https://netbeans.apache.org// (accessed Nov. 18, 2021).
[19] "Secure, Fast & Private Web Browser with Adblocker | Brave Browser." https://brave.com/ (accessed Dec. 30, 2021).
[20] "Visual Studio Code - Code Editing. Redefined." https://code.visualstudio.com/ (accessed Jan. 13, 2022).
[21] "System Calls in Operating System - javatpoint." https://www.javatpoint.com/system-calls-in-operating-system (accessed Feb. 16, 2022).