UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Low-code technologies: Comparative analysis of the OutSystems and Mendix platforms

.

Bernardo José Manosalvas Trávez

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación

Quito, 24 de febrero de 2023

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Low-code technologies: Comparative analysis of the OutSystems and Mendix platforms

Bernardo José Manosalvas Trávez

Nombre del profesor, Título académico	**Daniel Riofrio, PhD**

Quito, 24 de febrero de 2023

© DERECHOS DE AUTOR

Nombres y apellidos:	Bernardo José Manosalvas Trávez
Código:	00206757
Cédula de identidad:	1718757519
Lugar y fecha:	Quito, 24 de febrero de 2023

RESUMEN

El objetivo de este trabajo es comparar dos plataformas OutSystems y Mendix, que son basadas en low-code, una tecnología que está en auge y se espera que abarque gran parte del mercado de aplicaciones en el futuro cercano. Esta comparación se consigue planteando una metodología de evaluación para la comparación, que incluye un análisis cuantitativo y cualitativo. En cuanto a lo cuantitativo, se desarrolló una aplicación que implementa los métodos clásicos para manipulación de una base de datos, comparando los tiempos de respuesta de la implementación de cada plataforma en milisegundos. Para lo cualitativo, se utilizó el marco de evaluación de experiencia del desarrollador realizado por Fagerholm y Münch en 2012. Así como se realizó una rúbrica de evaluación con 5 criterios con una escala del uno al cinco para evaluar la parte cognitiva de la experiencia. Y otra rúbrica con 3 criterios con una escala igual para evaluar la parte de afecto y conación. Con el objetivo de puntuar, en base a la experiencia del desarrollador, una implementación de aplicación equivalente en ambas plataformas. Dando como resultado que OutSystems tiene una mejor implementación detrás de las escenas para los tiempos de respuesta de los métodos CRUD. Así como, de igual manera, OutSystems consigue una mejor experiencia de desarrollador en base a las rúbricas de evaluación cualitativa.

Palabras clave: low-code, OutSystems, Mendix, evaluación cuantitativa, evaluación cualitativa, tiempo de respuesta, experiencia del desarrollador

ABSTRACT

The purpose of this work is to compare two platforms, OutSystems and Mendix, that are based on low-code, a technology that is on the rise and is anticipated to dominate a significant portion of the application market in the near future. This comparison is accomplished by proposing a comparative evaluation methodology that combines quantitative and qualitative analysis. Concerning the quantitative, an application was developed that implements the traditional methods for database manipulation, comparing the response times of each platform's implementation in milliseconds. Fagerholm and Münch's (2012) developer experience evaluation framework was used for the qualitative evaluation. In addition to a rubric with five criteria and a scale from one to five for evaluating the cognitive portion of the experience. And another rubric with three criteria and an equal scale for evaluating the affective and conative components. The objective is to score, based on the developer's experience, an application implementation that is equivalent on both platforms. OutSystems had a more efficient implementation for CRUD method response times. Using the provided qualitative evaluation rubrics, OutSystems achieves a higher developer experience-based score based on the developer's rating.

Key words: low-code, OutSystems, Mendix, quantitative evaluation, qualitative evaluation, response time, developer experience

TABLE OF CONTENTS

Introduction......10
State of the art......11
OutSystems......12
Mendix......14
Fagerholm and Münch’s Framework for Developer Experience......16
Proposed Methodology for Comparison......18
Evaluation Methodology......18
Quantitative evaluation......18
Qualitative evaluation......19
Module to implement......26
Implementation......26
Results and Discussion......31
Quantitative......32
Qualitative......32
Conclusions and Future Work......36
References......38
Appendix A: Qualitative evaulation rubric (Cognition)......42
Appendix B: Qualitative evaulation rubric (Affect & Conation)......44

LIST OF TABLES

Table 1. Main research questions……………………...………………………………..……20

Table 2. Database methods response time for each platform…………………..……….…… 32

LIST OF FIGURES

Figure 1. OutSystems Architecture Diagram……………………………...………....…… .13

Figure 2. Mendix Architecture Diagram ………………………………..………....…… .14

Figure 3. Mendix Server Architecture Diagram …………..……………..………....…… .15

Figure 4. Developer experience framework……….…………………….………...…….. 17

Figure 5. App Quantitative Evaluation OutSystems…………………….………...…….. 28

Figure 6. App Quantitative Evaluation Mendix…………………………………...…...… 28

Figure 7. App Qualitative Evaluation OutSystems ……………..…………………...….… 30

Figure 8. App Qualitative Evaluation Mendi*x* …………………………..…………...… 30

INTRODUCTION

The low-code platform market is growing and according to Gartner it is expected that by 2024 it will represent 65% of all application development (Smithson, 2022).

Software development has been in a state of constant evolution over the past few decades, with the demand for distributed applications and web pages increasing. It is difficult to meet this demand due to the complexity of establishing a traditional development environment based on a stack of technologies. Low-code technologies provide development agility and are expanding for this reason. It is advantageous since it reduces response times to meet demand and simplifies rapidly integrating diverse technologies to achieve high productivity (Smithson, 2022).

For businesses to be able to compare and thus improve the platforms with which they have an agreement, it is essential to benchmark the market's leading competitors. In this instance, two platforms will be compared: OutSystems and Mendix. These are among the top three low-code development platforms (Brewster, 2022). Consequently, comparing the best platforms enables businesses to make better decisions based on costs and productivity.

Since the experience of developers working in these development environments have received scant consideration and, although research has proven positive developer experience on low-code platform (Dahlberg, 2020). As far as my knowledge goes, not much study has appeared to have conducted a comparison of the developer experience on two low-code platforms. For this study, an evaluation methodology will be defined based on the framework developed by Fagerholm and Münch in 2012. This is crucial as providing undesired experiences would lessen the advantages of low-code platforms.

In this report, we compare OutSystems and Mendix, two leading low-code platforms. We reviewed documentation and analyzed publicly available data regarding the capabilities of each platform. We analyzed the advantages and disadvantages of each platform. Then, we evaluated the platform's capabilities in specific scenarios to determine their overall utility.

Additionally, developers value performance because it has a direct impact on their experience with the application. If an application is slow, developers are likely to become frustrated and may opt for other alternative platforms.

STATE OF THE ART

Developer experience (DEx) may be characterized as a way to describe how developers feel and think about their job inside their working environments, with the underlying premise that enhancing the developer experience has favorable effects on traits like sustained team and project success (Fagerholm & Münch, 2012). A deeper and more thorough understanding of developers' emotions, perceptions, motivations, and identification with their tasks in their respective project environments will be necessary for new ways of working, in this case low-code technologies (Fagerholm & Münch, 2012).

Low-code platforms (LCP) enable programmers to create software with an intuitive interface and can accelerate the delivery of business applications by decreasing the typical time needed. LCPs can produce fully functional applications without the need to write code, but they may require additional coding for more complex projects. Additionally, certain low-code platforms eliminate the need for specialized skills in areas like security, data management, and infrastructure by bridging and simplifying the gap between them (Sanchis et al, 2019). Furthermore, LCPs can reduce costs associated with installation, training, deployment, and maintenance (Sanchis et al, 2019). A common advantage is the ability for a wider range of individuals to contribute to app development, not just those with coding skills but also those who require excellent governance in order to adhere to standards and regulations.

Low-code platforms like OutSystems and Mendix provide a variety of tools and features that make it easier for developers to build applications in accordance with industry standards. LCPs have a library of pre-built components that follow design patterns and best practices as well as templates with pre-configured components. Often LCPs come with built-in support for compliance requirements to support features like data encryption, access controls, and audit trails that help developer while building industry standard applications.

OutSystems

Founded in 2001, OutSystems is a low-code enterprise application development platform that provides companies with access to resources to create, deploy, and maintain enterprise applications (OutSystems, 2022). OutSystems Achieves ISO 27017 and 27018 Certifications for Cloud Security Compliance which lists controls for a company's information security management system (OutSystems, 2019).

Architecture

The architecture of OutSystems is a layered ecosystem that enables developers to build applications quickly, correctly, and for the future. It includes tools, a repository, builders, processes, and components that simplify difficult integration aspects. The runtime layer offers the option of deployment either in the OutSystems Cloud or on your systems, with the enterprise licensed version (OutSystems 2023). This can be seen in the Figure 1.

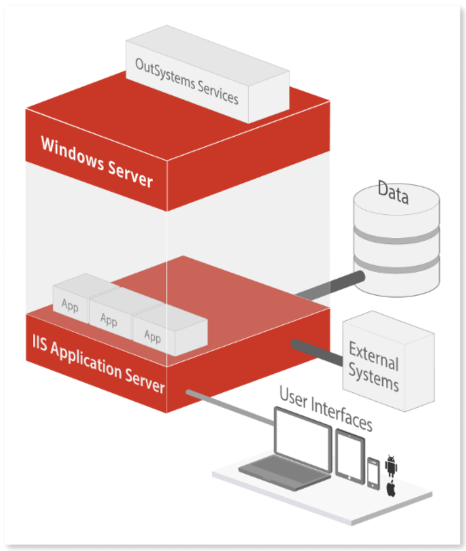


Figure 1. OutSystems Architecture Diagram

Application Server: There are multiple applications running on dedicated application servers in each environment. Microsoft IIS and Windows Server are used for app deployment (OutSystems, 2023).

Database: You can choose to deploy your environment databases on Microsoft SQL Server, Azure SQL Database, or Oracle (OutSystems, 2023).

Platform server: The Platform Server in each environment orchestrates the compilation, deployment, and management of all applications. All application servers that make up an environment have Platform Server installed. (OutSystems, 2023).

Mendix

Founded in the 2000s Mendix is a low-code platform to help software development organizations accelerate the process of creating, managing, and deploying software (Mendix, 2021). According to the ISO 27001 standard, Mendix has implemented an information security management system. (Mendix, 2023).

Architecture

The runtime architecture of Mendix consists of two main components: Clients, and Runtime server as seen in Figure 2.

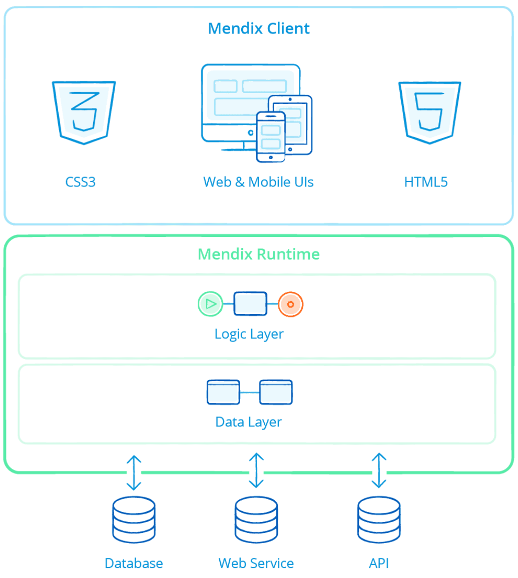


Figure 2. Mendix Architecture Diagram

Server architecture: The Mendix Server architecture consists of multiple components to execute logic, manage data, communicate with the client, and implement security (Mendix, 2023). As seen in Figure 3.

Client architecture: The Mendix Client is responsible for the user interaction and consists of a UI widget layer, a logic layer to execute offline logic, and a data layer for offline storage (Mendix, 2023).

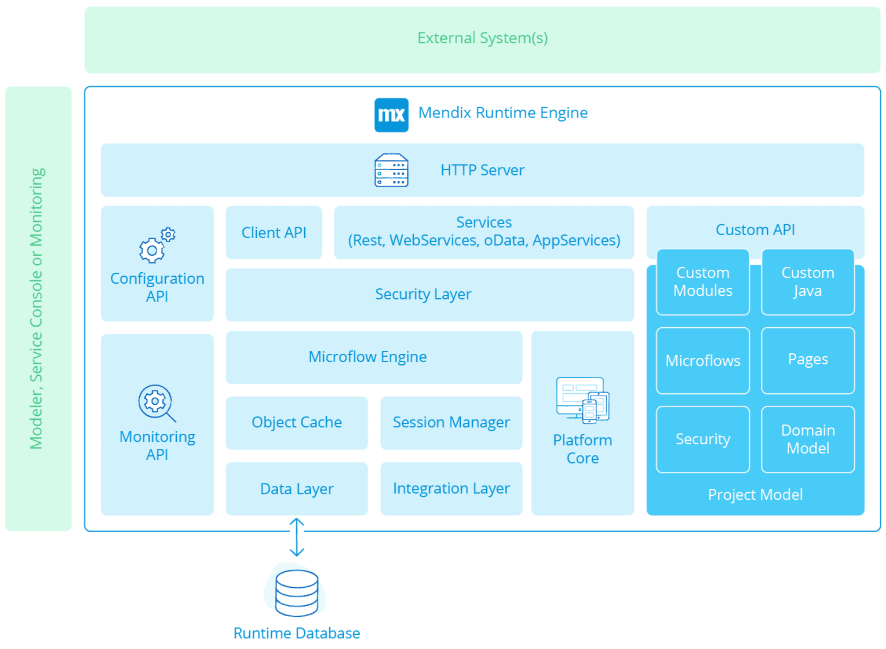


Figure 3. Mendix Server Architecture Diagram

This server uses Cloud Foundry that is based on Amazon Web Services (Mendix, 2023).

ISO 27017 and ISO 27018

ISO 27017 and ISO 27018 are based on ISO 27001 standard, ISO 27017 protects cloud data, it extends to cloud computing providers controls in earlier compliance standards that governed information management and sharing by IT vendors. ISO 27018 specifies cloud data privacy and security. It sets standards for protecting personally identifiable information (PII) in the cloud (International Organization for Standardization, 2022). This relates on how developers could perceive security on the low-code platforms.

Fagerholm and Münch’s Framework for Developer Experience

As research has shown negative experiences may cause mental health issues to developers, hence, it’s important to focus on the activities and experience of the developers, so they don’t take shortcuts or make software of bad quality because of these issues (Graziotin et al., 2017a; Graziotin et al., 2017b).

As the developer's ultimate objective is to produce software, it is crucial to understand how thinking and emotion are translated into deliberate action, and how group work should be methodically arranged to facilitate this. Since low-code platforms break down essential functionality into easy-to-use modules and components that can be reshaped (Dahlberg, 2020). It is important to understand the interactions the developer has, since they are fundamental to understand how they correlate with the experience perceived and how it could be improved (Beecham et al., 2008; Kuusinen et al., 2016).

The framework used for the methodology of this work is based than in psychology, the idea of mind is generally subdivided into cognition (attention, memory, creating and comprehending language, problem-solving, and decision-making), affect (feeling, emotion), and conation (impulse, desire, volition, striving) (Fagerholm & Munch, 2012).

The cognitive dimension includes of aspects that influence the developers' intellectual perception of their development infrastructure. This covers interactions with development tools and software process execution (Fagerholm & Munch, 2012).

The affective dimension is comprised of variables that affect how developers feel about their job. Respect and belonging are social variables that contribute to the development of a sense of safety. This dimension also includes attachment to even work routines (Fagerholm & Munch, 2012).

The conative dimension comprises of aspects that influence how contributors perceive their contribution's worth. Deliberate, planned activity with personal objectives that are appropriately connected with the goals of others is likely to boost a person's feeling of purpose, drive, and commitment, hence favorably affecting DEx (Fagerholm & Munch, 2012).

As the success of software projects depends on humans, it is essential that the platforms' experience be accurate, as the tools and methods can only increase the productivity of trained development teams (DeMarco and Lister, 1999). Because our ability to comprehend data as humans is limited, we maintain an individual mental state of reality with which we interpret new data (Fagerholm and Münch, 2012). Therefore, experience is necessarily subjective. The framework is shown in Figure 4.

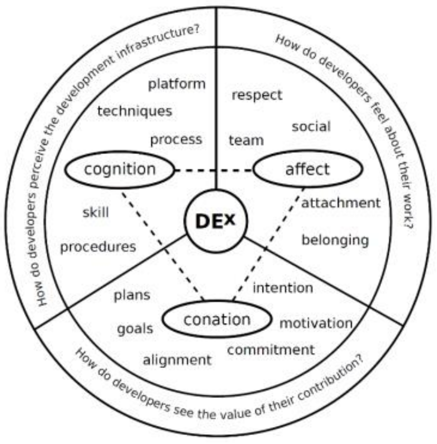

How do developers perceive the development infrastructure?
How do developers feel about their work?
How do developers see the value of their contribution?
platform
techniques
process
cognition
skill
procedures
respect
social
team
affect
attachment
belonging
DEx
plans
goals
alignment
conation
intention
motivation
commitment

Figure 4. Developer Experience framework

PROPOSED METHODOLOGY FOR COMPARISON

To evaluate OutSystems and Mendix a methodology for comparison is proposed. Which includes a quantitative and a qualitative evaluation. The qualitative evaluation is based on 2012’s developer experience framework by Fagerholm and Münch. And contemplates the development of an app that makes the developer submerge in each platform to have a more in-depth experience of development to give they experience based on the proposed qualitative evaluation rubric for each category, to guide the developer. For the quantitative evaluation, an app that implements the core methods of a database will be developed, measuring the response time for each method in both LCPs.

EVALUATION METHODOLOGY

A quantitative evaluation will be conducted obtaining response times as results. To complement the qualitative evaluation. Searching with multiple queries I found no studies that have obtained CRUD response times for neither OutSystems nor Mendix. And a quantitative evaluation based on the framework to answer three research questions with provided evaluation rubrics for each general category.

Quantitative evaluation

With the help of a web crawler, the response times for each method for the two applications will be obtained in milliseconds. An isolated comparison of each platform will be made for each proposed method, thus obtaining quantitative results on whether OutSystems or Mendix better implement the methods based on their respective architecture. The execution time of each isolated database method will be measured 100 times. Next, the arithmetic mean, and standard deviation of the resulting execution times will be determined for each database method. Comparing the execution time of one method against another is an effective way of evaluating the efficiency and effectiveness of a particular method. Having access to reliable data can play a significant role in helping businesses make decisions and improve the way they operate.

Qualitative evaluation

The qualitative evaluation is based on 2012's developer experience framework by Fagerholm and Münch. Given the dimensions of this framework, a research question is posed for each general category and each platform. And a rubric will be provided for each category for developers to better evaluate their experience. A study done by Dahlberg (2020) on low-code platforms and traditional development, with the use of this framework, proved the utility of a binary rubric, (positive experience, or negative experience). In this work my aim is to provide the user with a spectrum of how he feels about the development experience in any given platform. With the obtained results, the questions will be answered by giving the developer experience for both Mendix and OutSystems. This project aims to analyze each platform from the point of view of a developer. In this case, the experience I had developing on each platform.

Category	Research question
Cognition	How do software developers feel about their work on Mendix and OutSystems?
Affect	How do developers perceive their contribution's worth in Mendix and OutSystems?
Conation	How do developers consider the infrastructure for development in Mendix and OutSystems?

Table 1. Main research questions

The first question addresses aspects of how the developer feels about their work, such as a sense of belonging, respect, or attachment to social connections or the work itself (Fagerholm and Münch, 2012).

The second question addresses the conation category and investigates how developers perceive the value of their contribution. This includes motivation, goals, alignment, commitment, plan, and intention (Fagerholm and Münch, 2012).

The third and final question focuses on the cognition category, which describes how developers perceive the infrastructure of the development process. This category includes platform, technique, process, skill, and procedure-related factors (Fagerholm and Münch, 2012).

For the cognition category the following rubric is provided

- Documentation
    - Negative (1)

      There is no documentation. Does not have a community forum.
    - Mostly Negative (2)

The documentation provided is unclear and difficult to follow. Does not have a community forum.

- Lightly Positive (3)

  The documentation is clear and includes what is necessary. Has a community forum, questions are answered sometimes with a satisfactory result.

- Mostly Positive (4)

  The documentation is clear, logical, and easy to follow. Has an active community forum, questions are answered most of the time with a satisfactory result.

- Positive (5)

  The documentation is very clear, logical, easy to follow and provides all possible use cases and issues. Has an active community forum, questions are answered always with a satisfactory result.

- Installation and configuration
  - Negative (1)

    It is complex to install. There are no instructions, or they are unclear.

  - Mostly Negative (2)

    Installation is somewhat difficult, there are minimal instructions.

  - Lightly Positive (3)

    Common installation with quite clear instructions.

  - Mostly Positive (4)

Easy installation, easy to follow instructions.

- Positive (5)

  Step-by-step installation, automatic. Step by step instructions.

- Ability to collaborate
  - Negative (1)

    He has no capacity for teamwork or change control.
  - Mostly Negative (2)

    It has minimal capacity such as file sharing, use of templates, downloading of previously implemented modules.
  - Lightly Positive (3)

    It has the capacity to work simultaneously, warning of potential problems in change control with push and pull operations.
  - Mostly Positive (4)

    It has the ability to work simultaneously and handles change control satisfactorily by presenting a conflict report that handles merge operations.
  - Positive (5)

    It has the ability to work simultaneously, and handles change control in an excellent way, presenting a conflict report that manages merge operations and has version control.
- Entry level
  - Negative (1)

    Thorough programming knowledge is required to start developing.
  - Mostly Negative (2)

Essential programming knowledge is required to start developing.

  - Lightly Positive (3)

    Basic programming knowledge is required to start developing.

  - Mostly Positive (4)

    Minimum programming knowledge is required to start developing.

  - Positive (5)

    No prior programming knowledge is required to develop.

- Reusability of components
  - Negative (1)

    The developed modules and components are not reusable.

  - Mostly Negative (2)

    The developed modules and components can be visualized.

  - Lightly Positive (3)

    The modules and components developed can be reused within the same application.

  - Mostly Positive (4)

    The developed modules and components can be reused within the local environment.

  - Positive (5)

    Developed modules and components can be reused within the cloud environment.

See appendix A, for the evaluation rubric as a table.

For the Affect and Conation categories the following rubric is provided

- Personal-touch

- Negative

  No capacity provided to add personal touch, everything is static.
- Mostly Negative

  Minimal personalization capabilities.
- Lightly Positive

  Basic personalization is provided.
- Mostly Positive

  Most ideas can be implemented.
- Positive

  There is no limit to personalization, the tools provided allow for full customization.

- Problem-solving
  - Negative

    Ease of implementation for templates or modules is hard or they don't exist.
  - Mostly Negative

    Implementation for existing templates or modules requires lots of efforts.
  - Lightly Positive

    Implementations for templates and modules require some changes to their structure.
  - Mostly Positive

    Implementation of modules and templates require minimal changes to their structure.
  - Positive

Implementation for templates and modules require only drag and drop and simple clicks.

- Productivity
    - Negative

      The workflow feels interrupted by unintuitive user interface and knowledge requirements.
    - Mostly Negative

      The user interface doesn’t interrupt the workflow, but it is interrupted by knowledge requirements.
    - Lightly Positive

      The user interface helps with productivity, but knowledge requirements interrupt the workflow.
    - Mostly Positive

      The user interface its optimal for productivity and knowledge requirements rarely interrupt the workflow.
    - Positive

      The user interface its optimal for productivity and there are no interruptions for knowledge requirements.

See Appendix B, for the evaluation rubric as a table.

MODULE TO IMPLEMENT

In order to perform the quantitative evaluation of both platforms, two equivalent applications will be developed on each platform that implement the following database methods.

- CREATE
- READ
- UPDATE
- DELETE

For the qualitative evaluation, another two equivalent applications will be developed These applications will include sing-up, log-in, and log-out functionalities, a world cup table view with sorting capabilities for each column, and pagination for the table.

IMPLEMENTATION

Quantitative evaluation

The implementation of the applications is similar for both platforms. The model used for the database is the following.

Client

- Name: str (50)
- Username: str (50)
- Password: str (50)

Inside the UI we have five buttons, four text fields and one input. The “Load data” button loads the Client with the required information to run each method, ensuring both applications receive the same number of bytes in each method.

The “CREATE” button calls the create method with the following Entity:

- Name: NuevoCliente
- Username: NuevoUsuario
- Password: NuevaClave

The “READ” button calls the read method of the client with id one.

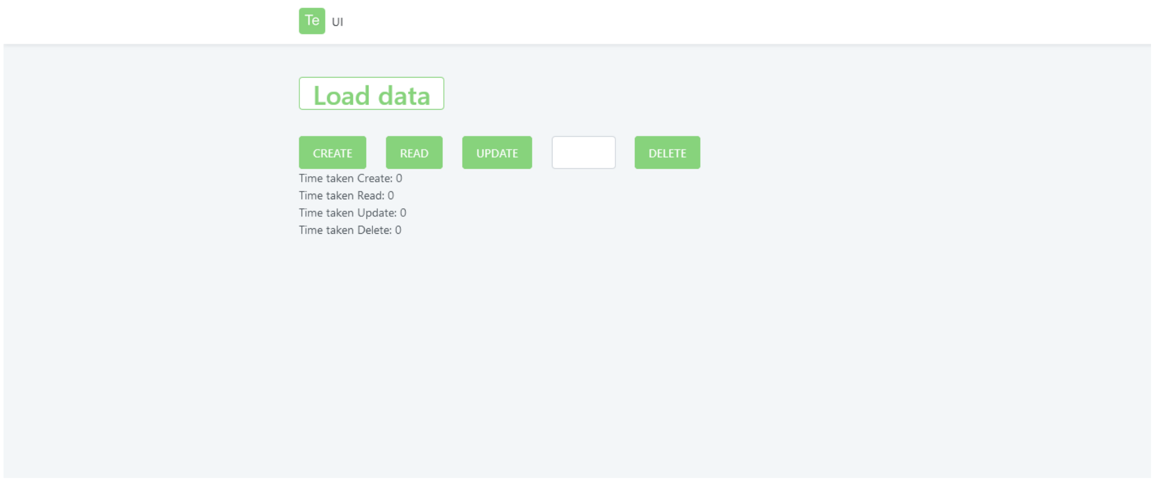
The “UPDATE” button runs the update client action with the following data:

- Name: ClienteActualizado
- Username: UsuarioActualizado
- Password: ClaveActualizada

Finally, the “DELETE” button requests the client with the id provided in the input to be deleted. For this case, the one hundred created Entities were deleted in reverse order.

Each button calls an onClick event, in Mendix this is called a flow, in OutSystems an action. To display the time taken to response to each text field the performance.now() function was used inside a JavaScript called inside a flow in the case of Mendix, and inside an action for OutSystems. Both web pages can be seen in Figure 5 and 6.

To collect all the data a web crawler helped to interact with the desired method button and retrieve the time taken for the text field corresponding to button clicked.



Time taken Create: 0
Time taken Read: 0
Time taken Update: 0
Time taken Delete: 0

Figure 5. App Quantitative Evaluation OutSystems

Figure 6. App Quantitative Evaluation Mendix

Quantitative evaluation

The models used for the database are the following.

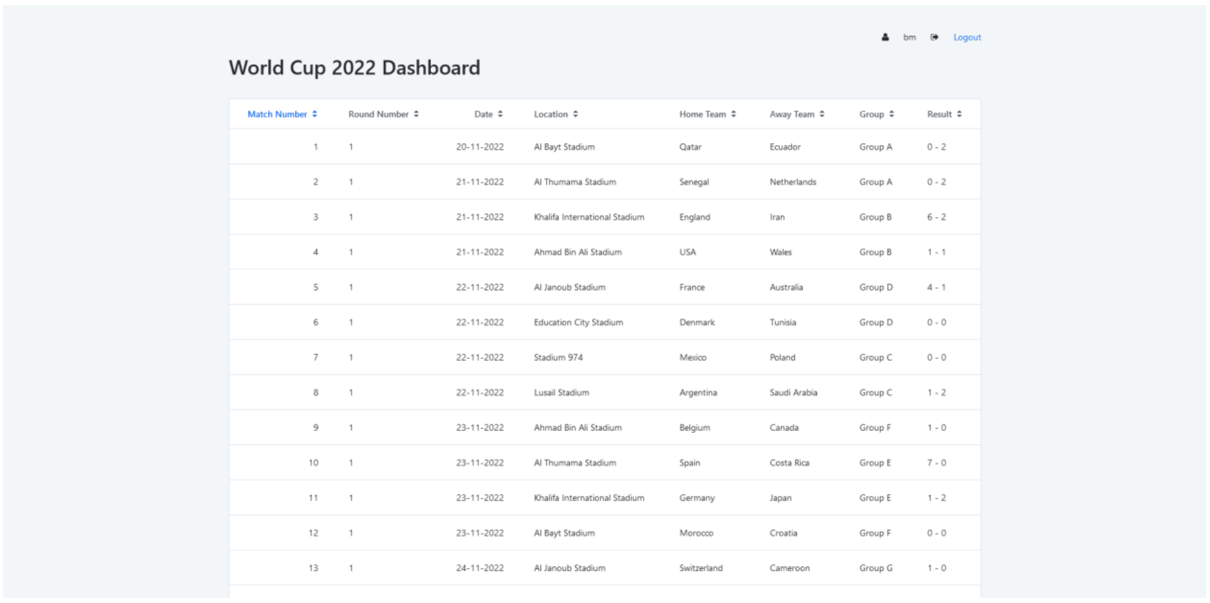
Client

- username: str (50)
- email: str (50)
- password: str (175)
- LoggedIn: Boolean

Fixture

- MatchNumber: Integer
- RoundNumber: str (50)
- Date: date
- Location: str (50)
- HomeTeam: str (50)
- AwayTeam: str (50)
- Group: str (50)
- Result: str (50)

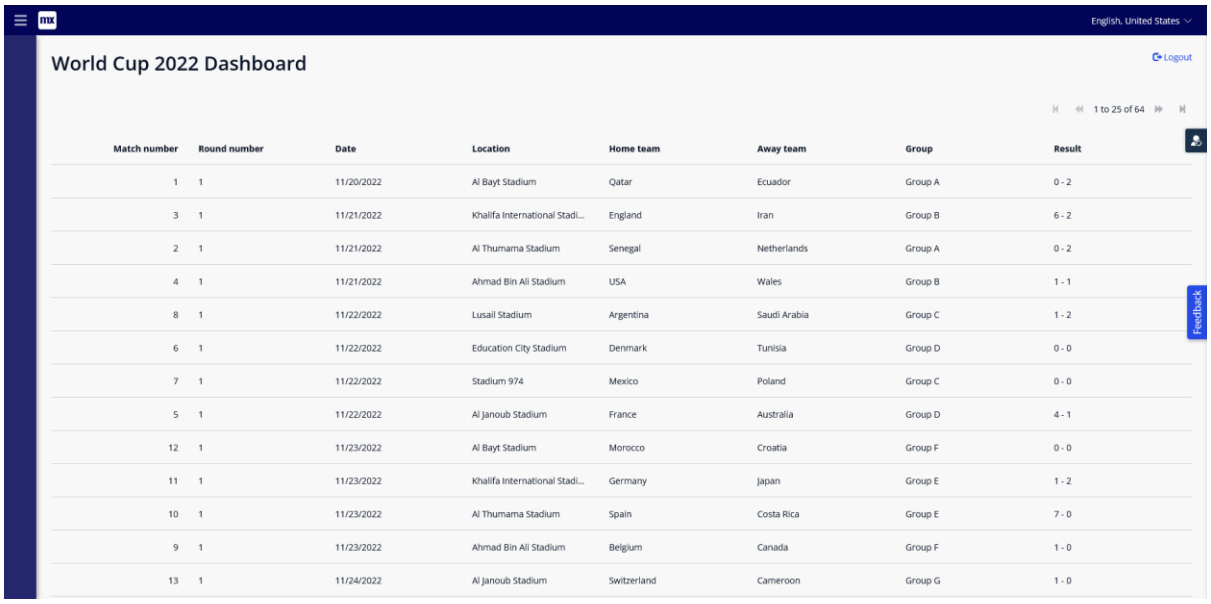
The figures 7 and 8 are the applications implemented, achieving a considerable similarity level with the peculiarities that the pagination of the table in OutSystems it is at the bottom of the table, and in Mendix at the top.



World Cup 2022 Dashboard

Match Number	Round Number	Date	Location	Home Team	Away Team	Group	Result
1	1	20-11-2022	Al Bayt Stadium	Qatar	Ecuador	Group A	0 - 2
2	1	21-11-2022	Al Thumama Stadium	Senegal	Netherlands	Group A	0 - 2
3	1	21-11-2022	Khalifa International Stadium	England	Iran	Group B	6 - 2
4	1	21-11-2022	Ahmad Bin Ali Stadium	USA	Wales	Group B	1 - 1
5	1	22-11-2022	Al Janoub Stadium	France	Australia	Group D	4 - 1
6	1	22-11-2022	Education City Stadium	Denmark	Tunisia	Group D	0 - 0
7	1	22-11-2022	Stadium 974	Mexico	Poland	Group C	0 - 0
8	1	22-11-2022	Lusail Stadium	Argentina	Saudi Arabia	Group C	1 - 2
9	1	23-11-2022	Ahmad Bin Ali Stadium	Belgium	Canada	Group F	1 - 0
10	1	23-11-2022	Al Thumama Stadium	Spain	Costa Rica	Group E	7 - 0
11	1	23-11-2022	Khalifa International Stadium	Germany	Japan	Group E	1 - 2
12	1	23-11-2022	Al Bayt Stadium	Morocco	Croatia	Group F	0 - 0
13	1	24-11-2022	Al Janoub Stadium	Switzerland	Cameroon	Group G	1 - 0

Figure 7. App Qualitative Evaluation OutSystems



World Cup 2022 Dashboard

Match number	Round number	Date	Location	Home team	Away team	Group	Result
1	1	11/20/2022	Al Bayt Stadium	Qatar	Ecuador	Group A	0 - 2
3	1	11/21/2022	Khalifa International Stadi...	England	Iran	Group B	6 - 2
2	1	11/21/2022	Al Thumama Stadium	Senegal	Netherlands	Group A	0 - 2
4	1	11/21/2022	Ahmad Bin Ali Stadium	USA	Wales	Group B	1 - 1
8	1	11/22/2022	Lusail Stadium	Argentina	Saudi Arabia	Group C	1 - 2
6	1	11/22/2022	Education City Stadium	Denmark	Tunisia	Group D	0 - 0
7	1	11/22/2022	Stadium 974	Mexico	Poland	Group C	0 - 0
5	1	11/22/2022	Al Janoub Stadium	France	Australia	Group D	4 - 1
12	1	11/23/2022	Al Bayt Stadium	Morocco	Croatia	Group F	0 - 0
11	1	11/23/2022	Khalifa International Stadi...	Germany	Japan	Group E	1 - 2
10	1	11/23/2022	Al Thumama Stadium	Spain	Costa Rica	Group E	7 - 0
9	1	11/23/2022	Ahmad Bin Ali Stadium	Belgium	Canada	Group F	1 - 0
13	1	11/24/2022	Al Janoub Stadium	Switzerland	Cameroon	Group G	1 - 0

Figure 8. App Qualitative Evaluation Mendix

RESULTS AND DISCUSSION

The results obtained for OutSystems followed the process described previously on the methodology. For Mendix, the results were attained from the Web Console also in milliseconds. With the averaged results for each database method summarized in the Table 2.

Architecture

The following results make sense since OutSystems uses Microsoft's IIS for their server. While Mendix uses their own architecture to handle their data and the server is Cloud Foundry.

Tried implementations

Microflows are strictly a server-side action whereas a JavaScript blocks its executed in the client side. For this reason, a JS block can't be inside a microflow. And an implementation similar to OutSystems was ruled out.

Java action would require implementing additional data models to display information on screen with Mendix, hence negatively affecting the response times by adding them to the data model and then rendering their values on screen. Making a database request essentially run two times, for this reason this was discarded.

Quantitative

The results obtained for OutSystems and Mendix are in the Table 2. The smallest response time values correspond, for every method, to OutSystems.

Method	OutSystems	Mendix
	Time [ms]	Time [ms]
CREATE	107.01 ± 3.49	196.99 ± 5.58
READ	105.98 ± 3.59	197.06 ± 5.52
UPDATE	107.19 ± 3.66	196.95 ± 5.51
DELETE	106.64 ± 3.60	197.36 ± 5.59

Table 2. Database methods response time for each platform.

Qualitative

As a developer who has worked with both OutSystems and Mendix, using the evaluation criteria derived from the framework, I compared and contrasted the two platforms based on my personal experience as a developer. To respond to the three research questions posed, it should be emphasized that I have prior experience with OutSystem.

Research Question 1 (Cognition)

How do software developers feel about their work on Mendix and OutSystems?

Starting with documentation, both OutSystems and Mendix offer extensive documentation that covers all the features of their respective platforms. OutSystems has a comprehensive online documentation portal that includes a wide range of guides, tutorials, and reference materials; with a community forum that answers all of the questions from developers. Mendix also has a rich set of documentation resources, and its community forum answers most of questions from developers. The resources include a developer portal, user guides, and how-to articles. In terms of the quality and comprehensiveness of the documentation and community forum, I would rate OutSystems as *Positive* (5) and Mendix as *Mostly Positive* (4).

When it comes to installation and configuration, both OutSystems and Mendix have straightforward and easy-to-follow instructions. OutSystems offers installation packages for Windows, Mac, and an alternative for Linux users with wine. It provides step-by-step instructions on how to install and configure the platform. Mendix also provides detailed instructions on how to set up and configure the platform. However, it is only available for Windows, to access Mendix's Studio Pro platform in any other OS should be with the use of a virtual machine. In terms of the ease of installation and configuration, I would rate OutSystems as *Positive* (5) and Mendix as *Positive* (5).

In terms of collaboration, both OutSystems and Mendix have robust features that enable teams to work together on development projects with change control. OutSystems has a built-in collaboration feature that allows team members to share their work, assign tasks, and collaborate on projects in real time. It manages version control and presents the user with a user-friendly UI that manages merge operations. Mendix also offers a range of collaboration tools, including version control, code review, and team management features with change control. However, in my opinion, Mendix does not have a user-friendly UI to manage merge conflicts like OutSystem does. I would rate OutSystems as *Positive* (5) and Mendix as *Mostly Positive* (4).

In terms of entry level, both OutSystems and Mendix are relatively easy to learn and use, making them suitable for developers with a range of skill levels. OutSystems has a user-friendly interface and a wide range of tutorials and learning resources to help developers get up to speed with the platform. Mendix also has a straightforward interface and offers a range of resources to help developers get started, including a comprehensive developer portal and a range of learning materials. In terms of programming knowledge, both require minimum programing knowledge such as HTML, CSS, and JS for more advanced features. I would rate OutSystems as *Mostly Positive* (4) and Mendix as *Mostly Positive* (4).

Finally, both OutSystems and Mendix offer a range of reusable components that can be easily integrated into applications. OutSystems has a library of pre-built components that can be easily added to projects, saving developers time and effort, maximizing the development time. Mendix also offers a range of reusable components, including widgets, templates, and libraries that can be easily integrated into projects. In terms of the availability and reusability of components, I would rate OutSystems as *Positive* (5) and Mendix as *Positive* (5).

As a developer, I found working with the OutSystems platform to be more enjoyable. Since my interactions with the provided resources felt more natural, I did not have to spend as much time searching for answers when necessary. In other words, I felt more productive developing in OutSystems.

Research Question 2 (Affect)

How do developers perceive their contribution's worth in Mendix and OutSystems?

Personal-Touch was the most notable distinction between the platforms for me. I discovered that I could implement almost any concept in OutSystems. In Mendix, I felt less like that, somewhat limited. I would rate OutSystems as *Positive* (5) and Mendix as *Mostly Positive* (4).

In terms of problem-solving, I felt that both OutSystems and Mendix provided adequate tools. To solve my problems, the built-in modules and templates needed only minor modifications. I would rate OutSystems as *Mostly Positive* (4) and Mendix as *Mostly Positive* (4).

Finally, both platforms are highly productive, with intuitive user interfaces that interrupt my workflow infrequently. Mendix was also effective, but I must admit that OutSystems had a more aesthetically pleasing user interface.. I would rate OutSystems as *Positive* (5) and Mendix as *Mostly Positive* (4).

With OutSystems, I felt that my work was mine, but with Mendix, I had mixed feelings. I accomplished my goals, though with some constraints. Despite the limitations, I valued my work contribution in Mendix more.

Research Question 3 (Conation)

How do developers consider the infrastructure for development in Mendix and OutSystems?

For the Personal-touch, I had difficulty implementing additional functionality to templates in Mendix, and particularly integrating JavaScript into the flow of logic. In fact, I was unable to. This was because Mendix's microflow runs on the server and a JavaScript Block (in Mendix) on the client side. This is how the architecture of Mendix works, the server runs the logic and the client is responsible for the user interaction with an offline logic layer. OutSystems, on the other hand, didn't pose any complications with its infrastructure. I would rate OutSystems as Positive (5) and Mendix as Mostly Positive (4).

For the Problem-solving category, I encountered a similar issue. Some template functionalities required an entity to be declared in order to use them as variables. In contrast, OutSystems provides local variables for each new page. I would rate OutSystems as Positive (5) and Mendix as Mostly Positive (4).

Finally, in Productivity, for the reasons mentioned above, Mendix did interrupt my workflow with prior knowledge requirements. While the interruption caused by OutSystems was rare. I would rate OutSystems as Mostly Positive (4) and Mendix as Lightly Positive (3).

With their respective architectures, both platforms simplify the process of developing a functional website. However, I believe OutSystem's infrastructure impacts the developer's workflow less.

CONCLUSIONS AND FUTURE WORK

In conclusion, based on my experience with both platforms, I am more productive developing on OutSystems because the user interface is more intuitive and the overall developer experience is superior. However, it is important to note that I have previously received training in OutSystems, and this could be a bias to consider. Despite this, it would be interesting to evaluate if the proposed rubrics helps reduce this bias. This can be done by evaluating more developers who had not received training for either platform as well as provide them with the same amount of training in each LCP, ensuring a no bias approach. Also having two teams of developers with the same amount of experience in each LCP, could be an option. Based on the proposed methodology and the framework developed by Fagerholm and Munch, we could draw stronger conclusions about whether OutSystems provides a better developer experience than Mendix. The rubric is an additional tool that provides a range of how a developer could feel rather than giving a binary approach. It could also be a useful instrument for companies to evaluate their developers and determine whether or not their partner platform offers a pleasant DEx.

To better asses the results of the quantitative evaluation it must be mentioned the problems I had with trying to run the code in the same environment, specifically trying to extract the generated code for each database method to compare the lines of code versus the time taken. A more robust quantitative evaluation would probably be to run locally each application. Mendix does provides a run locally option, but OutSystem does not in its free version, the enterprise level license would allow for this. Finally, the quantitative evaluation could be completely rethought, for example: Discarding interactions with the UI, not displaying the response times on screen. To evaluate just the actions called after the buttons are pressed for each method (This would solve the issue the Java action had). Use the Web Console to obtain response times for both platforms. With the help of the Network Tab. Since it helps to inspect the network requests giving information like Status, Method, Domain, File, Initiator, transferred (data in bytes), size (of data in bytes), and time to complete in milliseconds.

REFERENCES

Beecham, S., Baddoo, N., Hall, T., Robinson, H., & Sharp, H. (2008). Motivation in Software Engineering: A systematic literature review. *Information and Software Technology*, *50*(9–10), 860–878. https://doi.org/10.1016/j.infsof.2007.09.004

Brewster, C. (2022). 9 Best Low-Code Platforms to Use in 2022 | Trio Developers. Retrieved 13 September 2022, from https://www.trio.dev/blog/low-code-platforms

Brown University. (2022). Designing Grading Rubrics | Sheridan Center. Retrieved 13 October 2022, from https://www.brown.edu/sheridan/teaching-learning-resources/teaching-resources/course-design/classroom-assessment/grading-criteria/designing-rubrics

Bock, A., & Frank, U. (2021). In Search of the Essence of Low-Code: An Exploratory Study of Seven Development Platforms. Essen: University of Duisburg-Essen.

Dahlberg, D. (2020). Developer Experience of a Low-Code Platform: An exploratory study. Umeå: Umeå University.

DeMarco and T. Lister, "Programmer performance and the effects of the workplace," in Proceedings of the 8th international conference on Software engineering, ser. ICSE '85. Los Alamitos, CA, USA: IEEE Computer Society Press,1985, pp. 268–272.

Fagerholm, F., & Munch, J. (2012). Developer experience: Concept and definition. *2012* International Conference on Software and System Process (ICSSP), 73–77. https://doi.org/10.1109/ICSSP.2012.6225984

Frank, U., Maier, P., & Bock, A. (2021). Low code platforms: Promises, concepts, and prospects. A comparative study of ten systems. Essen: University Duisburg-Essen, Institute for Computer Science and Business Information Systems.

Graziotin, D., Fagerholm, F., Wang, X., & Abrahamsson, P. (2017a). Consequences of Unhappiness While Developing Software. *2017 IEEE/ACM 2nd International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, 42–47. https://doi.org/10.1109/SEmotion.2017.5

Graziotin, D., Fagerholm, F., Wang, X., & Abrahamsson, P. (2017b). Unhappy Developers: Bad for Themselves, Bad for Process, and Bad for Software Product. *2017* IEEE/ACM 39th International Conference on Software Engineering Companion *(ICSE-C)*, 362– 364. https://doi.org/10.1109/ICSE-C.2017.104

International Organization for Standarization. (2022). *ISO/IEC 27001:2022 (Standard No. 27017 and 27018).* Retrieved from https://www.iso.org/obp/ui/#iso:std:iso-iec:27001:ed-3:v1:en

Khorram, Faezeh & Mottu, Jean-Marie & Sunyé, Gerson. 2020. "Challenges & opportunities in low-code testing". 10.1145/3417990.3420204.

Lichtenthäler, R., Böhm,, S., Manner, J., & Winzinger, S. (2022). A Use Case-based Investigation of Low-Code Development Platforms. Bamberg: University of Bamberg.

Luo, Y., Liang, P., Wang, C., Shahin, M., & Zhan, J. (2021). *Characteristics and Challenges of Low-Code Development: The Practitioners' Perspective*. Wuhan: Wuhan University.

Mendix. (2021). Why was Mendix founded? Mendix. Retrieved from https://www.mendix.com/evaluation-guide/why-founded/#2-how-did-it-all-start

Mendix. (2022). Runtime architecture. Mendix Retrieved from https://www.mendix.com/evaluation-guide/enterprise-capabilities/runtime-architecture/

Mendix. (2023). Organization & Compliance. Mendix. Retrieved from https://www.mendix.com/evaluation-guide/enterprise-capabilities/organization-compliance/

OutSystems. (2019). OutSystems Achieves ISO 27017 and 27018 Certifications for Cloud Security Compliance. OutSystems. Retrieved from https://www.outsystems.com/news/cloud-security-certifications/

OutSystems. (2022). It began with a vision. OutSystems. Retrieved from https://www.outsystems.com/evaluation-guide/it-began-with-a-vision/

OutSystems. (2023). Infrastructure architecture and deployment options. OutSystems. Retrieved from https://success.outsystems.com/documentation/11/setup_and_maintain_your_outsystems_infrastructure/setting_up_outsystems/possible_setups_for_an_outsystems_infrastructure/infrastructure_architecture_and_deployment_options/

Sanchis, R., García-Perales, Ó., Fraile, F., & Poler, R. (2019). Low-Code as Enabler of Digital Transformation in Manufacturing Industry. *Applied Sciences*, *10*(1), 12. https://doi.org/10.3390/app10010012

Smithson, R. 2022. “As demand for enterprise apps booms, agility can be found in low-code solutions.” Retrieved 9 September 2022, from https://www.developer-tech.com/news/2022/may/05/as-demand-enterprise-apps-booms-agility-low-code-solutions/

Trendowicz and Münch, “Factors influencing software development productivity – state-of-the-art and industrial experiences,” Advances in computers, vol. 77, pp. 185 241, 2009.

Waszkowski, R. (2019). Low-code platform for automating business processes in manufacturing. *IFAC-PapersOnLine*, *52*(10), 376–381. https://doi.org/10.1016/j.ifacol.2019.10.060

Yajing Luo, Peng Liang, Chong Wang, Mojtaba Shahin, Jing Zhan. 2021. “Characteristics and Challenges of Low-Code Development: The Practitioners Perspective.” In ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM ’21), October 11–15, 2021, Bari, Italy. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3475716.3475782

APPENDIX A: QUALITATIVE EVALUATION RUBRIC (COGNITION)

Evaluation criteria					
Category	Negative (1)	Mostly Negative (2)	Lightly Positive (3)	Mostly Positive (4)	Positive (5)
Documentation	There is no documentation. Does not have a community forum.	The documentation provided is unclear and difficult to follow. Does not have a community forum	The documentation is clear and includes what is necessary. Has a community forum, questions are answered sometimes with a satisfactory result.	The documentation is clear, logical, and easy to follow. Has an active community forum, questions are answered most of the time with a satisfactory result.	The documentation is noticeably clear, logical, easy to follow and provides all possible use cases and issues. Has an active community forum, questions are answered always with a satisfactory result.
Installation and configuration	It is complex to install. There are no instructions, or they are unclear.	Installation is somewhat difficult, there are minimal instructions.	Common installation with quite clear instructions.	Easy installation, easy to follow instructions	Step-by-step installation, automatic. Step by step instructions.
Ability to collaborate	He has no capacity for teamwork or change control.	It has minimal capacity such as file sharing, use of templates, downloading of previously implemented modules.	It has the capacity to work simultaneously, warning of potential problems in change control with push and pull operations.	It has the ability to work simultaneously, and handles change control satisfactorily by presenting a conflict report that manages merge operations	It has the ability to work simultaneously, and handles change control in an excellent way, presenting a conflict report that handles merge operations and has version control
Entry level	Thorough programming knowledge is required to start developing.	Essential programming knowledge is required to start developing.	Basic programming knowledge is required to start developing.	Minimum programming knowledge is required to start developing.	No prior programming knowledge is required to develop.

Evaluation criteria					
Category	Negative (1)	Mostly Negative (2)	Lightly Positive (3)	Mostly Positive (4)	Positive (5)
Reusability of components	The developed modules and components are not reusable.	The developed modules and components can be visualized.	The modules and components developed can be reused within the same application.	The developed modules and components can be reused within the local environment.	Developed modules and components can be reused within the cloud environment.

APPENDIX B: QUALITATIVE EVALUATION RUBRIC (AFFECT & CONATION)

Evaluation criteria					
Category	Negative (1)	Mostly Negative (2)	Lightly Positive (3)	Mostly Positive (4)	Positive (5)
Personal-touch	No capacity provided to add personal touch, everything is static.	Minimal personalization capabilities.	Basic personalization is provided..	Most ideas can be implemented.	There is no limit to personalization, the tools provided allow for full customization.
Problem-solving	Ease of implementation for templates or modules is hard or they don't exist.	Implementation for existing templates or modules requires lots of efforts.	Implementations for templates and modules require some changes to their structure.	Implementation of modules and templates require minimal changes to their structure.	Implementation for templates and modules require only drag and drop and simple clicks.
Productivity	The workflow feels interrupted by unintuitive user interface and knowledge requirements.	The user interface doesn't interrupt the workflow, but it is interrupted by knowledge requirements.	The user interface helps with productivity, but knowledge requirements interrupt the workflow.	The user interface its optimal for productivity and knowledge requirements rarely interrupt the workflow.	The user interface its optimal for productivity and there are no interruptions for knowledge requirements.