Production Systems and Information Engineering
Volume 11 (3), pp. 64-84
doi: https://doi.org/10.32968/psaie.2023.3.5

INTRODUCTION TO MENDIX

HERMANS MARC
University of Miskolc, Hungary
Institute of Logistics
marc.philip.hermans@student.uni-miskolc.hu

PÉTER MILEFF
University of Miskolc, Hungary
Institute of Informatics
mileff@iit uni-miskolc.hu

Abstract. Mendix is a visual model-driven development platform that enables developers to create web and mobile applications without extensive coding knowledge. By using a low-code or no-code approach, Mendix aims to increase productivity and speed up time to market for application development. With Mendix, developers can create applications using a range of visual tools and widgets, reducing the need for manual coding. As a result, developers can focus on designing and implementing application logic, rather than on the technical details of programming languages and syntax. The use of visual tools and a low-code approach can also reduce development time and costs, allowing organizations to quickly and efficiently create applications that meet their specific needs.

Keywords: low-code, no-code, application development.

1. Introduction

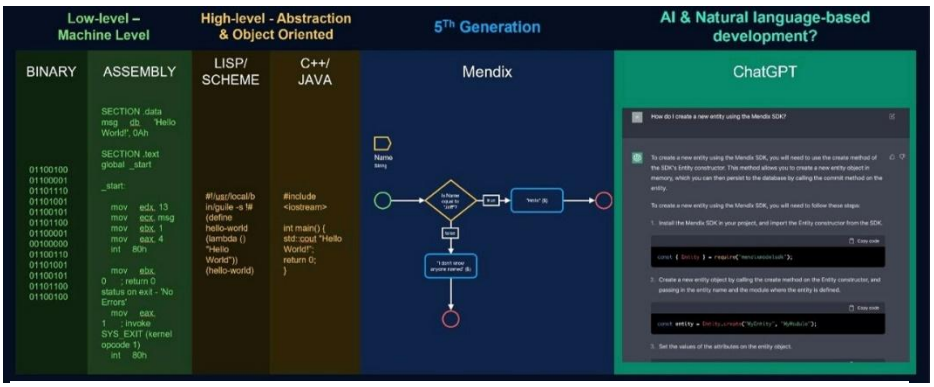
The use of Excel spreadsheets in managing logistics processes has created a tangled data jungle for many organizations. These spreadsheets, while familiar and easy to use, present numerous challenges such as version control issues, data loss, and inefficient workflows. The reliance on manual data entry and copying and pasting between systems further exacerbates these problems, resulting in significant costs and lost revenue due to inefficiencies. Additionally, miscommunications between business and IT often lead to frustrated users resorting to building their own spreadsheet solutions.

To address these challenges and bring order to the chaos, companies like Siemens Italy [7] have turned to Mendix, a powerful low-code development platform. Mendix offers an all-in-one solution for automating business processes and migrating away from the spreadsheet jungle. One of the key features of Mendix is the Workflow editor, which simplifies the development of complex data structures and allows for visualizing and modifying business process logic. This enables domain experts, who have a deep understanding of logistics processes, to actively

participate in the development process.
Another advantage of Mendix is its reusability capabilities. Many logistics processes share common components, such as forms, documents, and notifications. With Mendix, these components can be built once and reused across multiple applications, ensuring consistency and saving time and effort. This reusability also allows for the incorporation of corporate features and security best practices, providing a unified and secure user experience.
By leveraging Mendix, organizations like Siemens Italy can cut through the dense spreadsheet overgrowth and create streamlined, efficient, and scalable applications. The platform fosters inclusive collaboration between business and IT, accelerating the development process and driving production momentum. With Mendix, companies can transition from manual and error-prone spreadsheet-based processes to digitized and automated applications, ultimately transforming the logistics landscape and overcoming the challenges of the data jungle.

2. Low-code and the history of coding

At the lowest level of abstraction, programming was done using binary code, which consists of ones and zeroes that directly represent machine instructions. This was followed by assembly language, which uses mnemonics to represent machine instructions and provides a higher level of abstraction. High-level programming languages such as C++, Java, and LISP were then developed to provide even higher levels of abstraction, making it easier for developers to write code by providing more expressive syntax, built-in data structures, and other features.
These languages also introduced concepts such as object-oriented programming and functional programming, which further expanded the range of programming paradigms. In recent years, there has been a trend towards visual programming and low-code development platforms, which provide an even higher level of abstraction and enable developers to create applications without writing much code. Mendix is a good example of such a platform. These platforms allow developers to focus on the business logic and user experience of their applications, while the platform handles the underlying technical details.



1. Figure: *History of coding [Source: Mendix, mi is az és mire jó? PLM Felhasználói Konferencia 2023, Gárdony]*

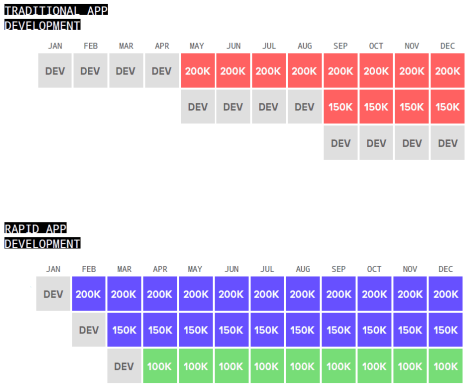
As artificial intelligence and natural language processing technologies continue to improve, we may see further developments in coding and programming languages that enable more natural and intuitive ways of programming. For example, chatbots like ChatGPT could be used to help developers write code by interpreting natural language commands and generating code automatically. While this technology is still in its early stages, it has the potential to further revolutionize the way we approach programming and development.

3. Low-Code and the business value [6]

Assessing the value of a low-code platform is crucial when considering a shift in software development methodology. It is important to have an understanding of the potential return on investment from adopting a low-code platform. However, quantifying low-code value can be challenging despite its straightforward concept.
The low-code value proposition of Mendix is clear: it can accelerate time-to-market for applications, resulting in faster returns on investment. With Mendix, solutions can be developed up to four times faster than traditional methods, allowing businesses to go live sooner and start realizing value sooner. This time-to-market advantage can be seen in examples such as Zurich Insurance, who built their underwriting application in just 12 weeks with Mendix, resulting in £280,000 in annual operational efficiency savings. Similarly, Suez developed their e-commerce portal in just three months, delivering £500,000 in additional business in its first three months. While it can be challenging to quantify the value of low-code platforms, the time-to-market advantage of Mendix is a clear benefit for businesses looking to build software solutions quickly and efficiently.

2. Figure: *Accelerate time-to-market [Source: Mendix Blog]*

The ability to release products faster with low-code development creates a virtuous circle of value, where more value can be delivered in a shorter amount of time, leading to the ability to build even more solutions. For example, delivering a solution three months sooner not only adds three months of business value but also creates an additional three months to develop more solutions. With low-code, organizations can build more solutions, adding value that would not be possible with traditional methods. The benefits of low-code application development are demonstrated by examples such as NC State University's lab management solution, which was built in just six months and delivers $1M in new revenue per year.
Low code offers benefits beyond just faster time-to-market and increased business value, it can also improve IT productivity. Technical debt can be reduced with low code by using reusable components, collaboration-focused IDEs, and easy integration with other systems. This can result in reduced capital expenditures and operating expenditures, as well as decreased development and support time, which can significantly impact cost savings. For example, a Mendix customer was able to shrink the time spent on product approvals from 34 hours to just 4 hours with low-code development, resulting in significant cost savings. Overall, low code can increase IT efficiency and enable developers to focus on new opportunities.



3. Figure: *More value, faster [Source: Mendix Blog]*

4. Welcome to Mendix

Mendix is a low-code software development platform that enables businesses to build web and mobile applications quickly and efficiently. Utilizing an automation-based approach, Mendix accelerates the lifecycle development of applications and streamlines process flows and workflows for business applications. The platform's scalability and speed make it accessible to both business users and professional developers, allowing for efficient application building. This paper explores how Mendix developers utilized the object-oriented programming (OOP) and the declarative programming paradigm (DPP) to achieve their goals, and provides an overview of the platform's main features and capabilities.

4. Figure: *Mendix Studio Lato Application [Source: Mendix.com]*

As can be seen in Figure 4, Mendix allows developers to build applications without having to write a lot of code from scratch. Instead, they can use the platform's visual modeling tools to design the data model, user interface, and business logic of their application. This enables developers to focus on the application's functionality and user experience, rather than on the technical details of building and maintaining an application.

The platform also provides a range of pre-built integrations with third-party services and systems, including databases, APIs, and enterprise systems like Salesforce and SAP. This allows developers to easily integrate their applications with other systems and services, and to take advantage of existing functionality and data.

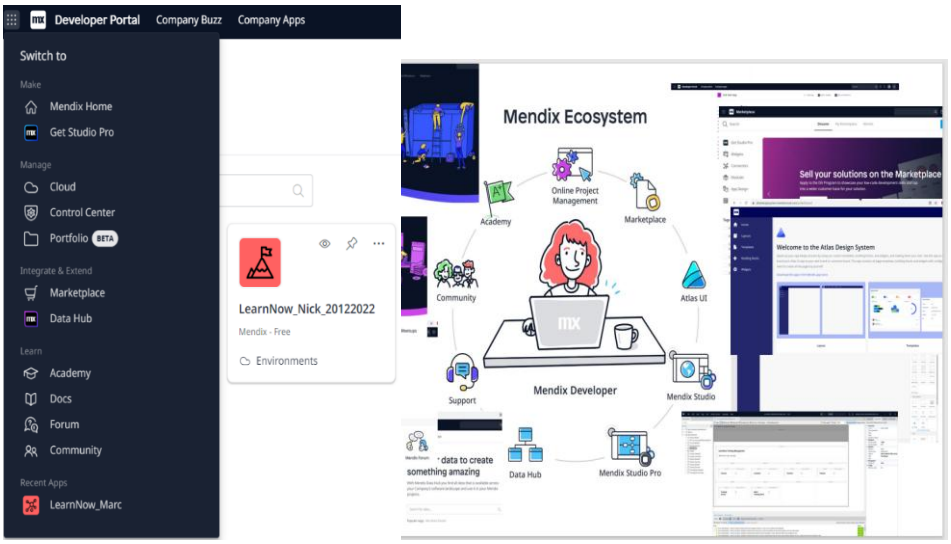
Mendix offers a range of deployment options, including cloud-based hosting, on-premises deployment, and hybrid options that combine cloud and on-premises hosting. The platform also includes tools for testing, monitoring, and managing applications in production, as well as collaboration features that enable teams to work together on the same project.

Type of Deployment	Last Updated
Mendix Cloud	May 4th, 2023
Mendix for Private Cloud	May 3rd, 2023
SAP Business Technology Platform (SAP BTP)	November 17th, 2022
Other Deployment Options	October 26th, 2020

5. Figure: *Mendix deployment methods*
[Source: Mendix Docs Release notes]

5. Mendix Developer Portal

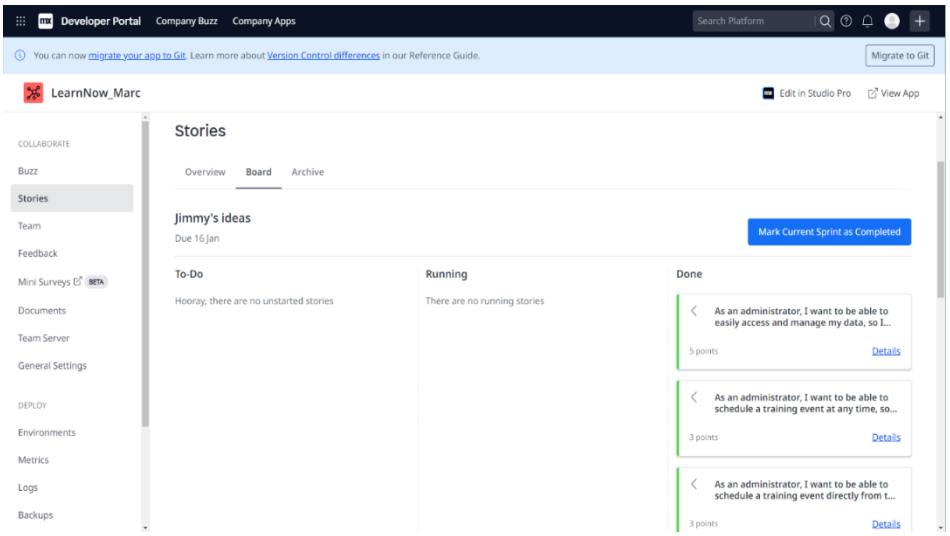
The Developer Portal is one of the key components of the Mendix Platform. After Registering, the user is redirected to the home page of this portal. In the Developer Portal, Mendix developers can collaborate, deploy, and manage their apps, and manage their company and users. The Developer Portal also offers open, well-defined APIs, enabling third-party developers to integrate their own widgets and plugins.



6. Figure: *Developer Portal General Overview*

The Mendix developer portal offers several key features, including the Mendix Home button, which takes you to the portal's main page. You can acquire the user

interface for building Mendix applications, called Studio Pro, by clicking the Get Studio Pro button. Additionally, you can access the marketplace to search for 3rd party plugins, which range from free to paid depending on company regulations. The Data Hub allows you to centralize all your data sources for your applications. Finally, the Learn section provides access to the main documentation on the platform as well as on Studio Pro, the Mendix Academy, and, depending on company regulations, the Mendix forums and community.
After selecting an application, Mendix presents two primary sections: the Collaborate and the Deploy sections. The Collaborate section contains several topics in order of importance, starting with the Teams section. Here, we can build our team and invite all stakeholders to the project, assigning them following roles: SCRUM Master, Application Operator, Business Engineer, End-User, Product Owner.
The Stories section is the most ingenious part of the developer portal where we can define the required functionality of our application in the form of short stories. These stories are initially collected in a backlog, and the SCRUM Master assigns them to the engineers. Each team member can view their open tasks, the task they are currently working on, and report back once they have completed the task. This allows them to continuously monitor the progress of the application and ensure its maturity.
In the Document settings, we have the option to create documents that will be available alongside the application. On one side, we can hold descriptions of how the application is built, and on the other side, we can collect documentation on how to use the application for end-users.

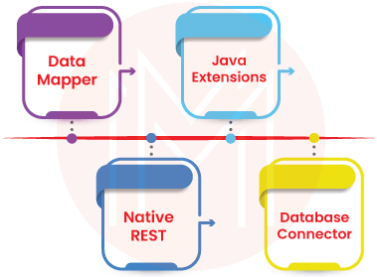


7. Figure: *Developer Portal - COLLABORATE- Stories [Source: own]*

In the General Settings, we can set things like the cloud environment to deploy to, access management, and register needed public API keys. We can also manage the project and view the history of the application's evolution.

Mendix utilizes four connectors to establish communication with external applications. These connectors are as follows:

- Data Mapper: This connector facilitates the integration of JSON and XML messages with Mendix domain models, and vice versa, without requiring developers to possess knowledge of data structures and services.
- Native REST: It enables developers to invoke REST and SOAP APIs from within Mendix microflows.
- Java Extensions: By integrating Mendix models with Java code, this connector turns into a powerful tool, allowing Java actions to become Native actions in Mendix microflows.
- Database Connector: This connector allows Mendix models to be linked to third-party databases without the need for code.



8. Figure: *Mendix Connectors [Source: mindmajix.com]*

6. Mendix Studio Pro

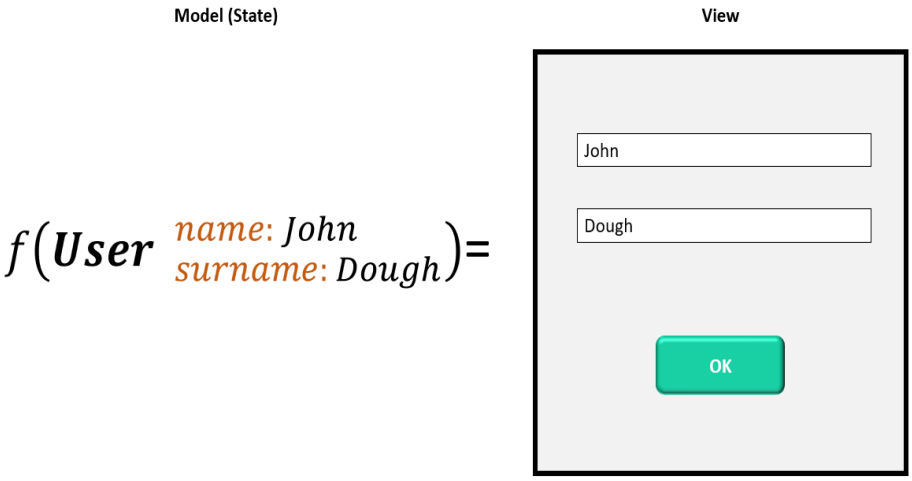
While building my initial application, two programming paradigms became evident. The first is the Object Oriented Programming (OOP) paradigm, which forms the foundation for the structures in Mendix, especially the usage of Microflows and Nanoflows. The second paradigm, which wasn't as prominent, was the Declarative Programming Paradigm, in the form of the domain model. Before moving forward, it's important to briefly explain both paradigms.

6.1. Declarative Programming Paradigm (DDP)

Declarative programming is a programming paradigm that expresses the logic of a computation without describing its control flow. [1] In other words, it focuses on describing what the program should do, rather than how it should do it. In a declarative program, the programmer specifies the properties that the output should have, and the computer determines how to compute the output based on those specifications. This is in contrast to imperative programming, where the programmer specifies each step of the computation and how it should be performed.

Declarative programming is often used in domains such as database programming, where SQL is a declarative language that describes the properties of the data to be retrieved or modified. It is also used in functional programming languages like Haskell and Lisp, where the programmer specifies functions to transform data, but

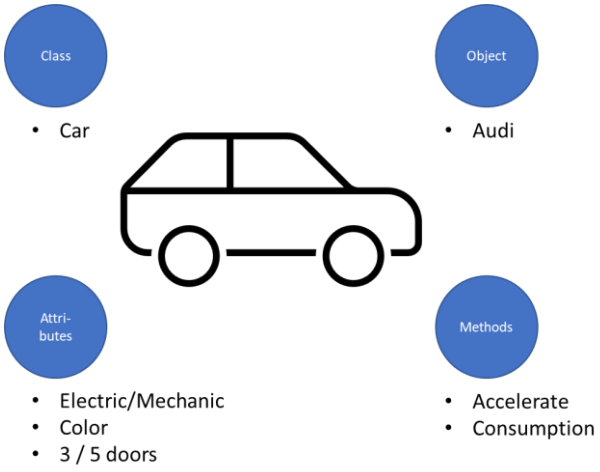
does not specify how those transformations are implemented.
One advantage of declarative programming is that it allows the programmer to focus on the problem domain rather than the implementation details. This can make the code easier to read, understand, and maintain. Additionally, declarative programs are often more concise than imperative programs, which can make them easier to write and debug.



9. Figure: *Declarative programming [Source: own]*

6.2. Object Oriented Programming Paradigm (OOP)

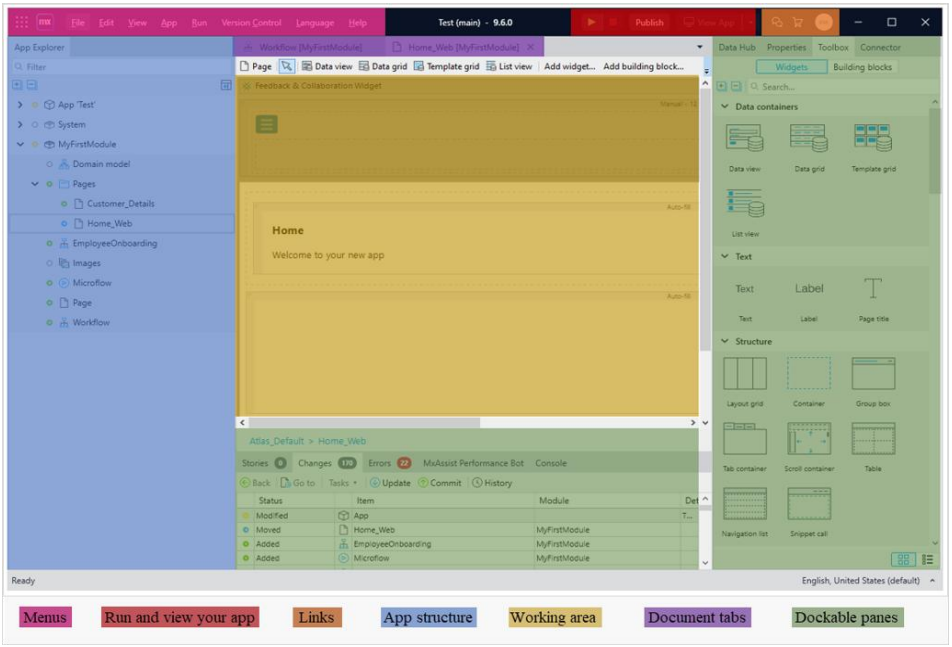
Object-oriented programming (OOP) is a programming paradigm that organizes code into objects that encapsulate data and behavior. Objects are instances of classes, which define the structure and behavior of the objects. [2] In OOP, the programmer thinks in terms of objects and their interactions, rather than procedures or functions. The basic idea is to model real-world entities or concepts as objects, which can have attributes (data) and methods (functions) that operate on that data. For example, consider a program that models a bank account. The account can be represented as an object with attributes such as account number, balance, and owner name, and methods such as deposit, withdraw, and check balance.
In OOP, classes provide a blueprint for creating objects. They define the properties and behavior of the objects, such as what data they can store and what operations they can perform. Objects can then be created from these classes, and each object can have its own unique data and state.



10. Figure: *Object Oriented Programming [Source: own]*

One of the key features of OOP is inheritance, which allows classes to inherit properties and behavior from other classes. This enables the creation of complex hierarchies of classes and objects, with common attributes and behavior inherited from higher-level classes. Another important feature of OOP is polymorphism, which allows objects of different classes to be treated as if they were of the same type. This enables code to be written in a more generic way, making it easier to reuse and maintain.

6.3. Studio Pro Interface



11. Figure: *Studio Pro Interface [Source: Mendix Docs]*

The top bar of Studio Pro contains various menus, such as Switch-to, Edit, View, and Version Control, each containing items for performing different actions like creating deployment packages, setting preferences, or viewing the Errors pane. To deploy your app, you can use the Publish or play (Run locally) buttons. Clicking the View App button allows you to view your deployed app. In the upper-right corner of Studio Pro, you can find links to the Developer Portal and Marketplace. Your profile picture appears next to these links if you are signed in. Clicking on your profile picture will display a drop-down menu with your full name, email, as well as links to your user profile, My Apps screen, and signing out option.
The App Explorer displays the complete structure of your app, consisting of individual files and settings grouped in folders and modules. The working area is the current document tab you're working in, with its own settings and save state. You can have multiple tabs open, reorder them, and view them side by side. Dockable panes can be positioned around the working area and contain various elements and settings, such as a list of errors or a toolbox.
At the bottom of the Studio Pro main window pane is the status bar, displaying the current status of Studio Pro and the currently selected language. If you have set up multiple languages in your app, you can change the currently selected language by clicking here.

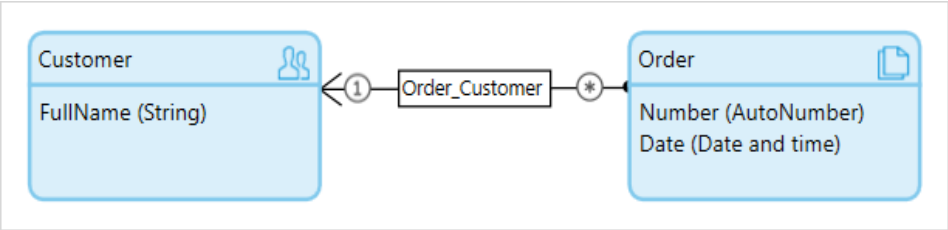
6.4. Building an Application

6.4.1. Selecting an application template

After identifying the problem or challenge to be addressed and understanding the application's requirements and objectives, one can leverage an application template to expedite development. Alternatively, the "Start from Scratch" template can be loaded to begin from the ground up. Application templates come with pre-defined entities, pages, and microflows, establishing a fundamental structure for constructing a specific type of application. Mendix provides a range of application templates for different use cases, including project management, HR management, and e-commerce.

6.4.2. Creating a Domain Model

The domain model in Mendix is an essential part of the development platform and is utilized to establish the data structure of the application. It is a graphical representation of the entities, attributes, and relationships that constitute the data model of the application. The domain model plays a crucial role in the Mendix platform and is employed to define the data structures, validations, and associations between various types of data across the application. It is recommended to develop this model at the beginning and then extend it as needed during the development process. However, it is not mandatory, as it is also possible to create domain entities on-the-fly, one may skip this step and start with the user interface.



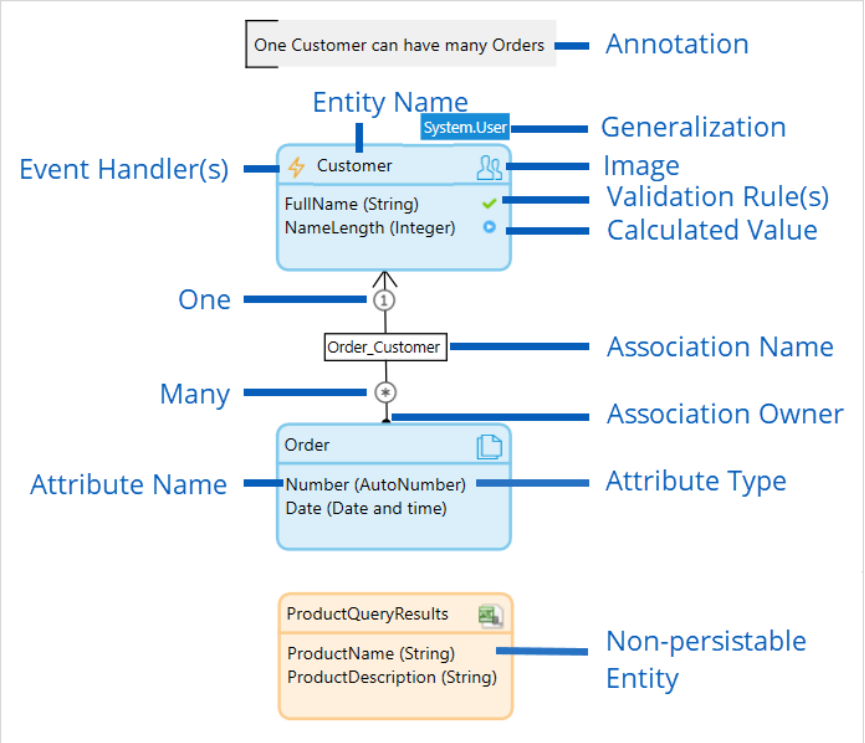
12. Figure: *Two entities with an n/1 association*

The domain model in Mendix is designed to be flexible and extensible, allowing developers to define their own custom entities, attributes, and relationships to match the specific needs of their application. It is also designed to be easy to use, with a visual editor that allows developers to define the data model using a drag-and-drop interface.

At the highest level, the domain model consists of entities, which represent the different types of data that the application will use. Entities can have attributes, which define the properties of the data, such as its name, description, and data type. Entities can also have associations, which define the relationships between different entities.

Entities in Mendix can be categorized into persistent and non-persistent entities. Persistent entities are stored in the database and can be accessed and modified using the platform's built-in data storage and retrieval mechanisms. Non-persistent

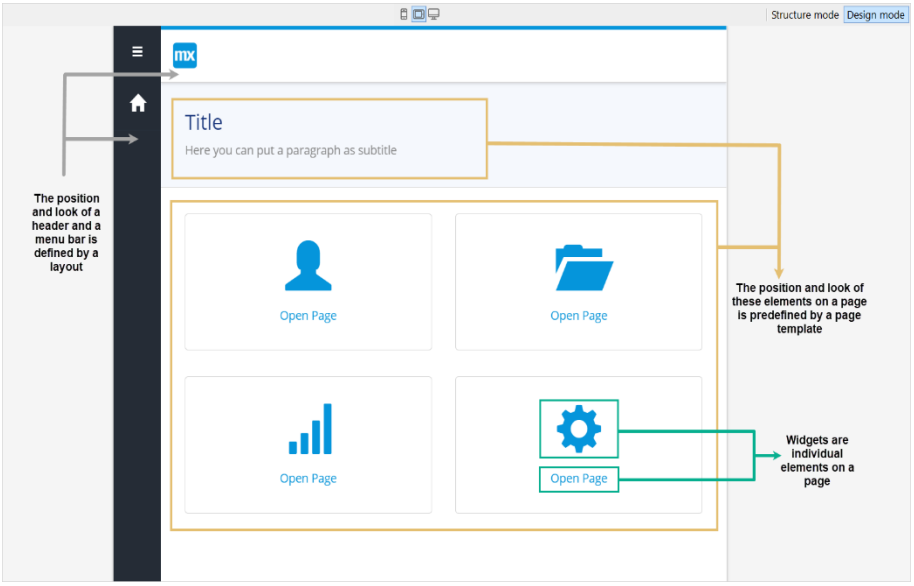
entities, on the other hand, are not stored in the database and are used for temporary data that does not need to be persisted between application sessions.
Attributes in Mendix can be of many different types, including strings, numbers, dates, times, and more. They can also have default values, validation rules, and other properties that help to define the behavior of the attribute.
Associations in Mendix are used to define the relationships between entities. Associations can be one-to-one, one-to-many, or many-to-many. They can also have roles, which define the direction of the relationship and the cardinality of the association.
In addition to entities, attributes, and associations, the domain model in Mendix also includes validation rules, which are used to ensure the integrity and consistency of the data in the application. Validation rules can be defined on entities and attributes and can be used to enforce data constraints, such as ensuring that a certain attribute is unique or that a certain value is within a certain range.
The domain model is a versatile tool that enables developers to define the structure and behavior of data in a Mendix application easily. It is designed to be user-friendly and allows developers to modify the data model quickly to meet the specific requirements of their application. This flexibility highlights that the data domain in Mendix embodies the declarative programming paradigm, where the programmer specifies what the program should do without detailing how it should be done. By defining the entities, attributes, and relationships between data, the developer declaratively defines the data structure, and Mendix automatically generates the code that implements the data model. In contrast, object-oriented programming (OOP) focuses on defining the behavior of objects in the system and how they interact with each other. While OOP can be used in Mendix for building custom widgets and defining microflows, the domain model is an example of declarative programming.



13. Figure: *Domain model that defines customers and orders [Source: Mendix Docs]*

6.4.3. Creating User Interface for the Application

The pages structure in Mendix is another key element of creating a web or mobile application. In Mendix, a page represents a single screen or view within the application, and is used to display and interact with data.



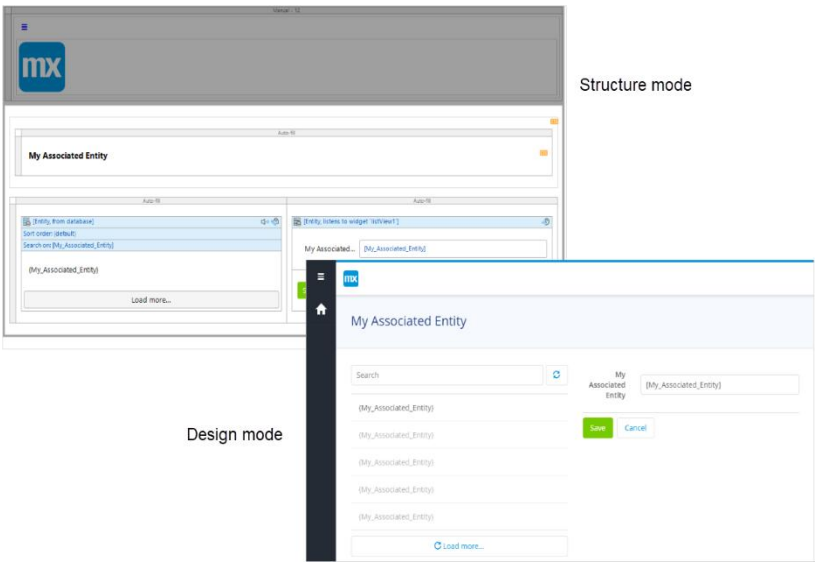
14. Figure: *Page layout [Source: Mendix Docs]*

The composition of a page in Mendix is based on a collection of widgets, which are visual components utilized to display data or enable user interaction. Widgets are categorized into the following groups:

- Data containers that form the core of building forms in Mendix, used for viewing and modifying data in the application
- Text widgets that display textual information to the user
- Structure widgets that can contain other widgets
- Input elements that make it possible to show and edit attribute and association values
- Images, videos, and files that enable working with corresponding media types
- Buttons, which trigger actions
- Menus and navigation, which allow users to navigate throughout the application
- Reports, which aggregate data and present it in tables or charts
- Authentication widgets, which facilitate the user verification process, such as password and login text boxes.

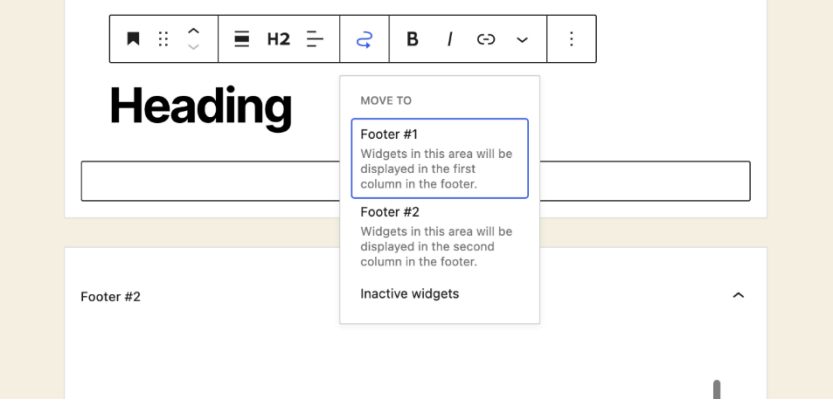
Each page in Mendix is associated with a specific entity in the domain model, which represents the data that will be displayed on the page. This allows developers to create pages that are specific to the data and functionality required for each screen within the application.

The looks and feels of each page can be defined in the Design mode, in the Structure mode the widgets and the connections to the data are visible. Designing each page request the user to jump several times between modes, so an optimal result can be reached.



15. Figure: *Design and Structure mode for Mendix Page Design [Source: Mendix*

In addition to widgets, pages can also contain microflows, which are used to perform business logic and data processing within the context of the page. For example, a microflow might be used to retrieve data from an external system and display it on the page, or to validate user input before saving it to the database.
The structure of pages in Mendix is similar in some ways to the structure of pages in WordPress, a popular content management system for creating websites. In WordPress, pages are used to create static content that is displayed on the website, such as a home page, about page, or contact page. WordPress pages are also structured using a set of visual components, called blocks, which can be added to the page to display different types of content.



16. Figure: *Blocked based widget editor from Wordpress [Source: Wordpress.org]*

However, there are some key differences between the pages structure in Mendix and WordPress. For one, the pages in Mendix are dynamic, meaning they can be used to display and interact with data in real-time, whereas WordPress pages are typically static and do not change based on user input. Additionally, the widgets and microflows in Mendix are highly customizable and can be tailored to the specific needs of the application, whereas WordPress blocks are more limited in their functionality and are designed primarily for displaying static content.

6.4.4. Defining Logic with Microflows and Nanoflows

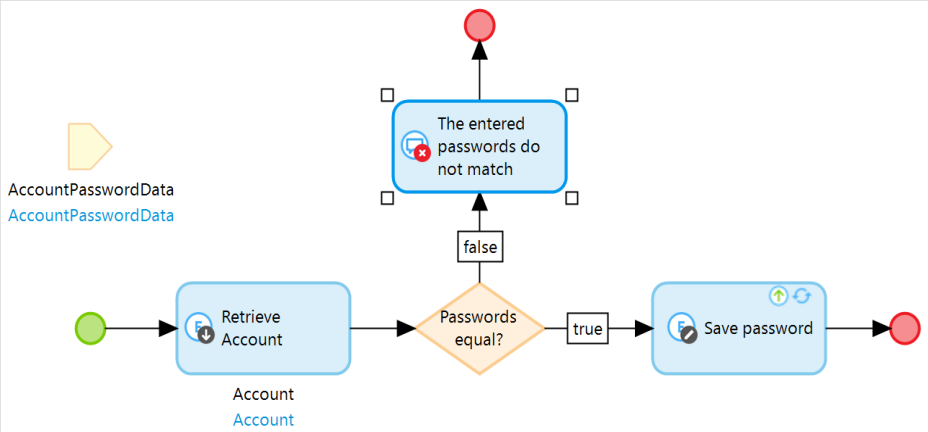
Microflows and nanoflows provide a visual way to express the logic of your application, including actions like object creation and updating, page display, and decision-making. This approach replaces traditional textual program code with a more visual representation.

While microflows are translated to Java and run on the runtime server, they cannot be used in offline apps. Nanoflows, on the other hand, are translated to JavaScript and run directly on the browser/device, making them available for offline use. Additionally, most of the actions in nanoflows run directly on the device, resulting in a faster execution for logic that does not require server access.

Microflows automate business processes and enforce business rules by performing various operations, including validation rules, data transformation, and integration with external systems. They are designed to be flexible and extensible, making it possible to create custom logic tailored to the specific needs of the application.

A microflow typically consists of a sequence of actions triggered by an event. These actions can be simple, like setting a value or sending an email, or more complex, such as retrieving data from external systems and processing it. Microflows can be triggered by a wide range of events such as a button click, a data change, or a timer.

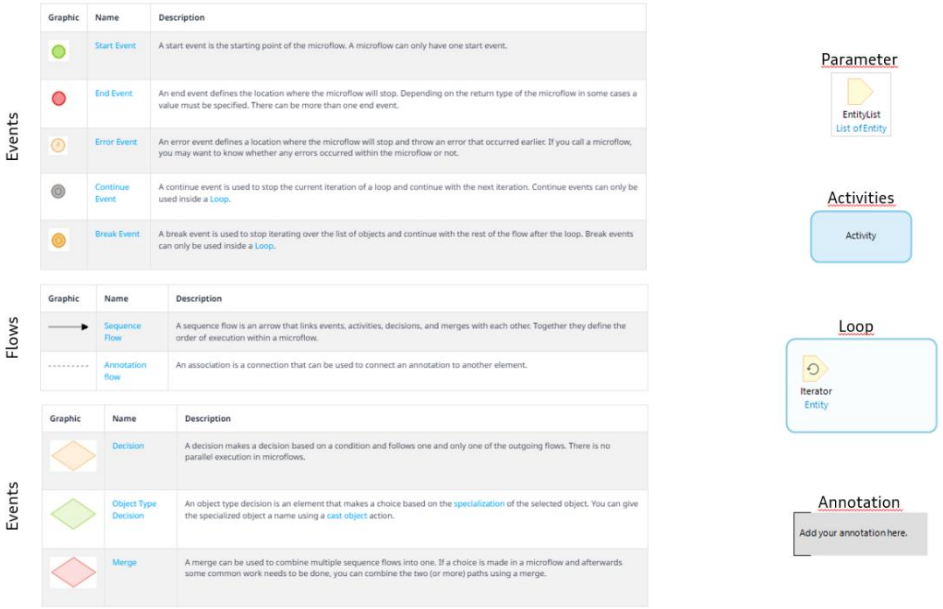
Microflows also enforce security by limiting access to certain actions based on user roles or permissions. They can be created using a drag-and-drop interface that allows developers to add actions, conditions, and loops to the flow. Additionally, developers can enhance microflows with custom Java code to create complex logic not possible with the built-in actions.



17. Figure: *'Retrieve Account' Microflow [Source: Mendix Docs]*

A microflow in Mendix is made up of various elements, which can be categorized into different types based on their function. The following categories are used to group these elements:

- Events: These represent the starting and ending points of a microflow, as well as special operations within a loop.
- Flows: These form the connections between elements in the microflow.
- Decisions: These elements deal with making choices and merging different paths within the microflow.
- Activities: These are the actions that are executed within the microflow, such as creating or updating objects, showing pages, and sending emails.
- Loop: This element is used to iterate over a list of objects in the microflow.
- Parameter: This represents data that serves as input for the microflow.
- Annotation: This element can be used to add comments to the microflow for documentation purposes.



18. Figure: *Standardized graphical notation of all components [Source: own]*

The various elements of microflows in Mendix offer developers the essential components to construct intricate and tailored business logic for their applications. The graphical representation of microflows follows the Business Process Model and Notation (BPMN), which is a universally recognized graphical notation used for illustrating business workflows.

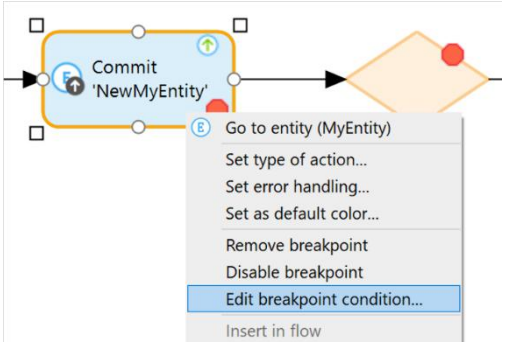
The built-in consistency checker in Mendix Studio Pro ensures that any obvious errors in the application being built are detected, reducing the number of technical errors during runtime. However, the tool cannot detect functional errors, which require manual checking. Fortunately, Mendix provides a range of tools that make the manual checking process easier. For instance, if a functional error occurs in a microflow or nanoflow, it can be debugged using the debugger.

Microflows in Mendix represent the principles of Object-Oriented Programming (OOP) through the use of objects, properties, and actions. In OOP, objects are instances of a class that have attributes and methods that define their behavior. Similarly, in Mendix, objects are instances of entities that have attributes and associations that define their properties and relationships.

Microflows in Mendix allow developers to create actions that can manipulate the properties and associations of objects. This is similar to the methods that can be called on objects in OOP. Microflows also allow for the creation of custom actions, which can be reused across different microflows, just like how methods can be reused across different objects in OOP.

Furthermore, microflows can also make use of inheritance, polymorphism, and encapsulation, which are core concepts of OOP. Inheritance allows for the creation of specialized microflows based on a general microflow, while polymorphism allows for the use of different microflows interchangeably based on their inputs and outputs. Encapsulation allows for the hiding of implementation details, ensuring

that microflows can be modified without affecting the rest of the application.



19. Figure: *Setting condition of a breakpoint*
[Source: Mendix Docs]

7. Conclusion

In conclusion, Mendix is a powerful low-code development platform that allows developers to create complex applications quickly and efficiently. With its visual modeling tools, microflows, and nanoflows, developers can easily express the logic of their applications without writing extensive code. The platform provides a wide range of widgets and elements that enable developers to build customized and scalable applications. While Mendix Studio Pro has a built-in consistency checker that helps prevent technical errors, functional errors still require manual validation. However, the platform's debugging tools, such as the debugger, can assist in identifying and fixing any issues that may arise. Overall, Mendix is a versatile platform that enables developers to create sophisticated applications that meet the specific needs of their business.

Declaration of competing interest

The author declares that he has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

I thank Dr. Mileff Péter for his suggestions on the topics of no code and low code programming, Dr. Tamás Péter and Mr. Molnár Zsolt for the continued support. And my father, Mr. Hermans John for his support and the elaborated proofreading.

References

[1] Twain Taylor (2022). The basics of working with declarative programming languages. TechTarget|App Architecture. https://www.techtarget.com/searchapparchitecture/tip/The-basics-of-working-with-declarative-programming-languages

[2] J.L. Hodges (2019). Software Engineering from Scratch. Jason Lee Hodges. https://doi.org/10.1007/978-1-4842-5206-2_10

[3] Madhuri Yerukala (2023). What is Mendix? MindMajix. https://mindmajix.com/mendix-tutorial

[4] Mendix Docs (2023) Studio Pro 9 Guide. Mendix Technology BV https://docs.mendix.com/refguide9/

[5] Mendix Academy (2022). Rapid Developer Certification course material. Mendix Technology BV

[6] A. van Oosten (2021). Achieve New Levels of Business Value with Low Code. Mendix Technology BV. https://www.mendix.com/blog/achieve-new-levels-of-business-value-with-low-code/

[7] A. Verweg (2021). Solving the Process Problem: How Workflow Works. Mendix Technology BV. https://www.mendix.com/blog/solving-the-process-problem-how-workflow-works/