Implementing a Mendix low-code application using best practices

Laura Pelkonen

2022 Laurea

Laurea University of Applied Sciences

Implementing a Mendix low-code application using best practices

Laura Pelkonen
Business Information Technology
Bachelor's Thesis
December, 2022

Laurea University of Applied Sciences **Abstract**

Business Information Technology

Bachelor's degree

Laura Pelkonen

Implementing a Mendix low-code application using best practices

Year 2022 Number of pages 37

As digitization increases, low-code development has grown in popularity due to its possibilities for rapid application development. However, adopting a new technology takes time and consideration. Best practices help in the introduction of a new technology as well as to ensure high-quality code. The purpose of this thesis project was to explore best practices to streamline Mendix low-code application development in the client company where Mendix platform was a fairly new technology in use. The research was based on the pre-defined best practices provided by Mendix. In this thesis the Mendix best practices were evaluated through the implementation of an application development project with the agile approach. The goal of the project was to create an extensible low-code application from one feature of an existing high-code solution.

The objectives of the project were met, and the project gave a better understanding of the importance of best practices in application development as well as in adopting a new technology. Best practices helped to create a consistent application and improved team collaboration. The findings of the research are usable also in future low-code projects.

Keywords: low-code development, best practices, Mendix low-code platform

Contents

1 Introduction

Digital transformation has been on the rise for quite some time but sped up during the COVID-19 epidemic. To keep up with digitalization, companies adopt new technologies which requires adaptation and ability to learn new skills. The need to expand digital skill set does not only apply to the future generations, but also to those already in working life (Milanesi, 2020).

The accelerated digitalization poses challenges for the companies in the IT sector. According to Drenik (2022), the traditional application development is no more adaptive enough to today's requirements. As the demand for rapid application development grows, low-code is increasingly seen as a solution to the situation.

However, implementing new technology takes time and does not happen by magic. To be successful, it requires clear alignment and consistent change management (Fahrion, 2022). Applying best practices is a great way to support the introduction of a new technology. In application development, best practices help to create reusable and maintainable code, and to avoid the most common pitfalls. As a result, implementation of best practices tends to increase the developer's development speed and enhance application performance (Mocke 2021). Easily readable and understandable code becomes even more important in low-code development as the teams may be formed from developers of very different skill levels.

Mendix is one of the leading platforms in the low-code market. The Mendix team believes that people's different backgrounds and expertise are a strength and that software development teams should not only consist of professional developers (Mendix, 2022a). To support developers with different qualifications, Mendix platform provides two different development environments as well as various tools to improve team collaboration. In addition, they have also pre-defined best practices to facilitate development work.

This thesis evaluates the importance of best practices in application development and their meaning in introducing new technology. The following section of this chapter opens up more about the project and the objectives of this thesis before diving deeper into the world of low-code, what it is and how the future of low-code looks like. This thesis also discusses more about the Mendix low-code platform and puts the Mendix best practices into practice through an application development project. The outcomes of the research are presented and analysed at the end of the thesis.

1.1 Background

This thesis was created in cooperation with a low-code development team of a Finnish software and service company. At the time of the initiation of the project, Mendix was rather new low-code development platform used by the development team. There were no deeply-rooted customs yet with Mendix development, which slowed down application development and increased the possibility for inconsistent work. This gave a great opportunity to explore the best practices and their benefits in low-code development. Guidelines for Mendix development would be beneficial in streamlining the development process and in ensuring high-quality code. The findings of the research could be utilized later also in other low-code projects and in customer cooperation.

1.2 The project objectives

In this thesis, the best practices of Mendix low-code development are evaluated through implementation of a demo application development project. The goal of the project was to recreate one feature of an existing high-code application with Mendix platform. The existing application is a stock management application which includes several features for different stock management tasks. The feature to be implemented was a rapid inventory tool.

The main purpose of the rapid inventory -feature is to enable making quick quantity changes for products in different stocks. The products, or stock items, can be searched by bar code or in a list. The requirements for the low-code application were:

1) The Mendix application must be later extensible to the other features of the original high-code application.
2) The user interface of the Mendix version should be modern and user-friendly.

2 Low-code development

Even though low-code as a term is fairly new, the idea itself is not. According to Bock and Frank (2021, 735) many low-code products have already existed before the arise of the low-code concept. Now the popularity and recognition of low-code is growing, and even COVID-19 pandemic did not slow down the raise of the low-code trend. In fact, the pandemic seemed to have sped up digital transformation as well as tremendously increase the demand for remote working and therefore benefited low-code. Gartner Inc. estimated an increase of 22,6%, 13,8 billion dollars, on the global low-code development technologies market. (Stamford, 2021)

But what actually is low-code development? Although the use of low-code platforms is growing, the term low-code still seems quite foreign to many. To clarify the low-code

concept, this chapter will discuss more about what is low-code development, its benefits and challenges, as well as the role of a professional developer in low-code. This chapter also presents the future prospects of low-code development.

2.1 What is low-code development?

Low-code development is a form of application development where the goal is to minimize the need of writing custom code and utilize readymade components and other features in the creation of applications. Low-code development makes application development visual as it regularly runs in a model-driven drag-and-drop interface (Mendix, 2022b). Hence application development no longer requires strong coding skills, also a more inexperienced developer can create fully functional applications.

Occasionally low-code gets mixed with no-code due to the similarities of their development methods. No-code development takes place on simple and user-friendly platforms which are suitable also for non-technical people. However, no-code development is more limited, and the customization possibilities are not as broad as in low-code development. Low-code provides more capabilities and enables applications to be extended with custom code.

Traditional high-code development on the other hand gives the most flexible opportunities for application development, but it is also the most time-consuming development method. In addition, high-code has higher demand on experienced developers than low-code development. As Jednaszewski (2022) states, one could say that low-code is somewhere between of no-code and high-code, utilizing the best aspects of both.

2.2 The role of a professional developer in low-code

As application development becomes more accessible for people with no coding skills, some might ponder what the need for professional developers is then. Even though low-code platforms do have built-in features and tools to handle some functionalities, there is still space for professional expertise.

According to Woo (2020, 961) some online application development platforms may have security vulnerabilities. In addition to that, the lack of knowledge in software development makes debugging harder and the product most likely will not work as efficiently as desired. Also, integrations, like API-calls, are often too complex for non-technical people (Jepsen, 2021). These issues still need the support of professionals who know how the application as a whole works and who have the needed knowledge and skills for more complex implementations.

All things considered, instead of decreasing need of professional developers, the case might be more about the transformation of the role of a professional. While in so called high-code

development the technical skills of a coder are essential, in low-code the business consulting skills of an expert are more in demand. Their experience offers irreplaceable expertise on best practices and regulations. (Jepsen, 2021)

2.3 The benefits and challenges of low-code development

The possibility to utilize citizen developers boosted the low-code market as the need for tailored software solutions has increased (Stamford, 2021). Visual modelling makes building applications more understandable for citizen developers, who necessarily do not have previous experience in application development. That increases the number of developers that can participate in modelling and supports the collaboration in cross-functional teams (Mendix, 2022b). As user interfaces and simple logics can be created by less experienced people, the professional developers can focus on more demanding issues (Woo, 2020).

With low-code professional developers can deliver applications faster and with less, or zero, code. Instead of spending time for manually coding basic, repetitive components that are typical to many applications, and figuring out how to maintain the safety of an application, the platform takes care of it and developer can concentrate on configuring and on creating the needed functionalities. Also, the amount of training, that less experienced developers need for being able to work on projects, decreases with low-code development. (Silva, Vieira, Campos, Couto & Ribeiro, 2021)

Another pro in low-code development is undoubtedly the increased development speed and productivity that pre-existing components and building blocks support. Utilizing low-code platforms also tends to reduce costs of the development process as well as maintaining the system later. (Bock & Frank, 2021)

Although low-code development has many benefits in building new applications, it can also be used to update existing software. Den Haan (2022) states that using low-code makes legacy modernization easier. For modernization, low-code can be used to develop applications to be integrated into existing software, or to gradually replace outdated systems.

Adaptability for the needs of different users is a challenge that low-code platforms face. While for a citizen developer sufficiently simple interface and functionalities are essential, professional developers need tools that support solving more challenging tasks. Consequently, it is undoubtedly in the platform's favour the more the platform supports different kind of developers. (Silva et al. 2021)

2.4 The future of low-code

As mentioned before, the popularity of low-code and no-code development is predicted to be increasing. Referring to Wong, J., Iijima, K., Leow, A., Jain, A. and Vincent, P (2021, 1), "by

2025, 70% of new applications developed by enterprises will use low-code or no-code technologies, up from less than 25% in 2020".

However, most likely not all the businesses that adopt low-code technologies will be businesses from IT field. It is predicted in Gartner research, that by the end of 2025, only 50% of new low-code customers will be IT companies (Stamford, 2021). According to Lawton (2021), Shekhar, the program director of the graduate business analytics program at The University of Texas, states that "low-code development skills will become part of all business users' default skill sets, just like spreadsheets and presentations".

In the future enterprises are likely to favour big platform providers due to their wider offerings. When the low-code tools and cloud services come from the same provider, application development become easier and faster. As the future prospects of low-code seem promising, it is no surprise that large cloud service providers Amazon, Microsoft and Google are investing in low-code platforms. (Varerkar, cited in Lawton 2021)

3 Mendix as a low-code platform

Mendix is an enterprise low-code platform that enables development of both web and mobile applications responsively. With two development environments and collaboration tools Mendix supports developers with various set of skills and facilitates teamwork between IT-professionals and business users.

Mendix was founded in 2005 in Netherlands and later in 2018 it joined Siemens family. Their vision is to increase collaboration between IT and business teams, and to enable rapid application development to add more business value to processes. Nowadays over 4 000 companies and over 250 000 developers are using Mendix platform to develop low-code applications. (Mendix, 2022c)

On their website (Figure 1) they give an overview of the platform and demonstration of possible use cases. From the website it is also possible to access the developer portal and Mendix Marketplace, as well as Mendix Docs which provides release notes, guides and how-tos for Mendix development.

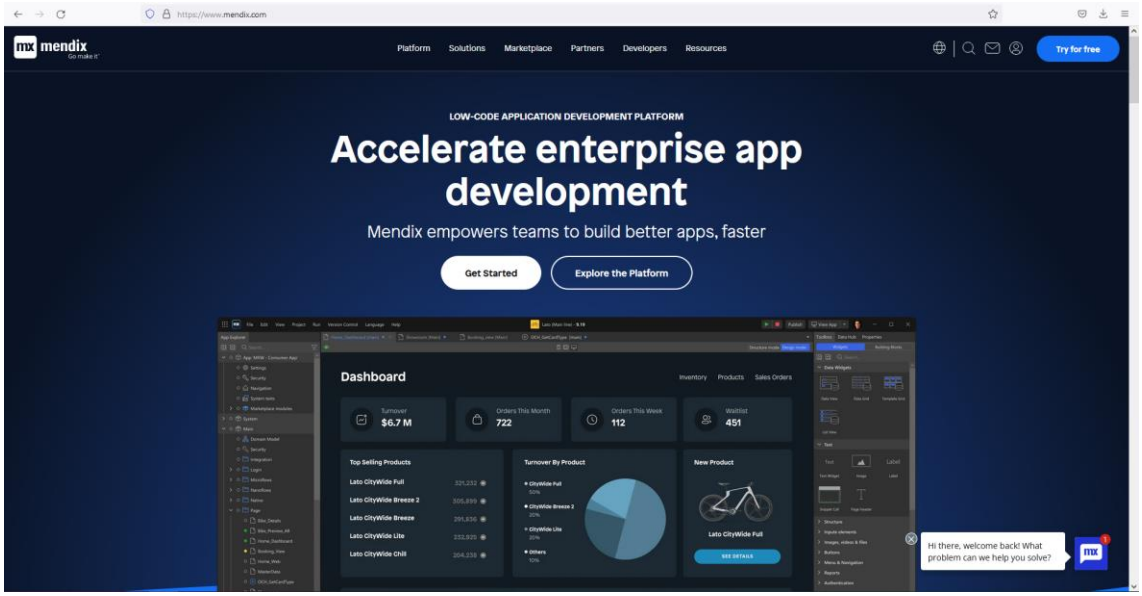

Figure 1: Home page of Mendix's website

Additionally, the website presents examples of several solutions and customer stories. One of the Mendix customers is Amsterdam Airport Schiphol who needed help to facilitate the daily work of their floor managers. Their job at one of the busiest airports of Europe involved a lot of paperwork and being on the phone to report daily issues, as well as using two different computer software. As a solution was developed a Mendix mobile application which enables reporting directly from a mobile device and adding images and the location of the issue to the report. This made the work of floor managers more efficient and freed them for other tasks. (Mendix, 2022d)

3.1 Mendix in the low-code market

For the second time, Gartner chose Mendix as a leader in Magic Quadrant of low-code platforms in their report 2021 (Figure 2). The report evaluates and compares the performance and completeness of vision of several different low-code platforms. (Wong et al. 2021)

In the report, Mendix as a product was overall praised but particularly for its possibilities on UX design, integration support and governance. Mendix has succeeded in building two development environments, which together increase the suitability of the platform for developers of different levels. In addition, viability is mentioned as one of the strengths of Mendix due to its considerable growth, and Siemens' acquisition of Mendix only strengthens the long-term viability. Mendix also does well when it comes to innovation. Mendix has brought the Internet of Things and digital twins as part of the low-code development with MxAssist Performance Bot, which automates code reviewing based on Mendix best practices, and Data Hub -integration technology (Mendix Docs, 2022a). (Wong et al. 2021)

However, the Magic Quadrant of low-code platforms -report also pointed out some weaknesses of each listed platform. For Mendix, the cautions were related to pricing as well as to marketing and geographic strategies. Compared to other low-code platforms, Mendix often has higher price level, and it has less market presence outside of Europe and North America. Though according to the report, Mendix has paid attention to updating the price list and investing in growth in the Asia-Pacific region. (Wong et al. 2021)

Figure 2: Magic Quadrant for Enterprise Low-Code Application Platforms. (Wong et al. 2021)

3.2 Development environments

Mendix has two different integrated development environments which synchronize bidirectionally (Mendix, 2022e). This reduces the silos between business users and professional developers as it provides the possibility to collaborate during the development process (Mendix, 2022f). Figure 3 visualizes the developer target groups of each development

environments. Mendix Studio is the environment for more solution focused users such as business analysts, who benefit from more simple and visual tools. Whereas traditional high-code solutions require professional IT developers, Mendix Studio Pro sets somewhere in between Mendix Studio and manual coding. Studio Pro is a tool that gives great possibilities for rapid application development for a technology focused coder but can be also utilized by a skilled business user.

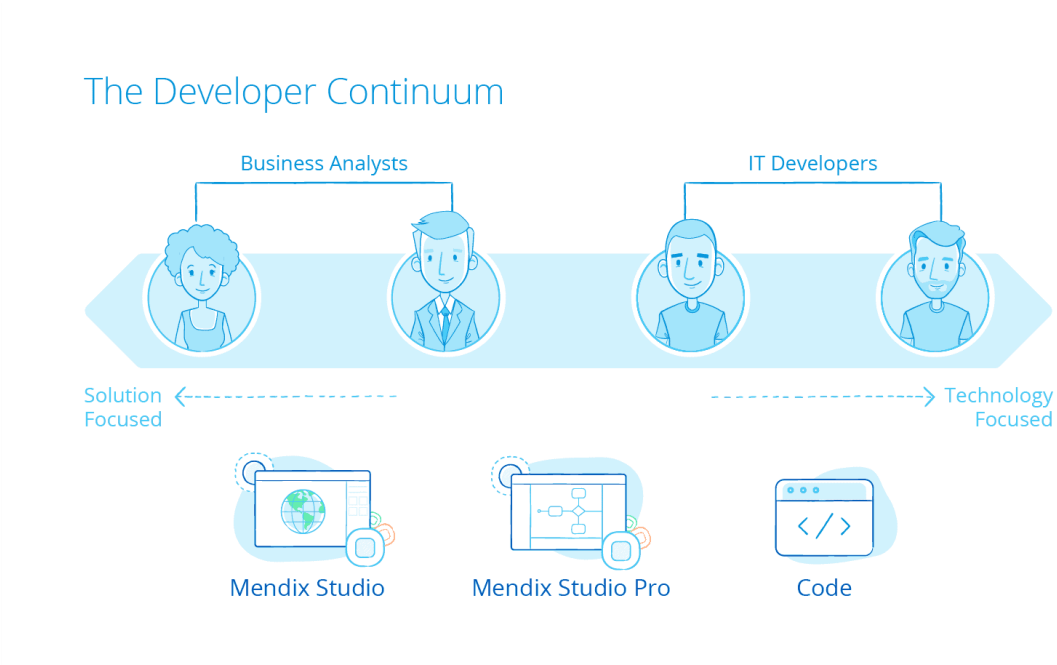

Figure 3: The Developer Continuum in Mendix. (Mendix, 2022g)

3.2.1 Mendix Studio

For business users, like citizen developers and business analysts, Mendix offers browser-based no-code environment Mendix Studio (Figure 3). It is a visual development environment for application modelling and provides the possibility to build applications without previous coding knowledge (Mendix, 2022f). Mendix Studio is so called “what you see is what you get”-environment (WYSIWYG). According to Mendix Docs (2022b), Studio enables to develop user interfaces with drag and drop building blocks and widgets, to create domain models and to visually model interactions and flow control. Figure 4 visualizes the tools and user interface of Mendix Studio no-code development environment.

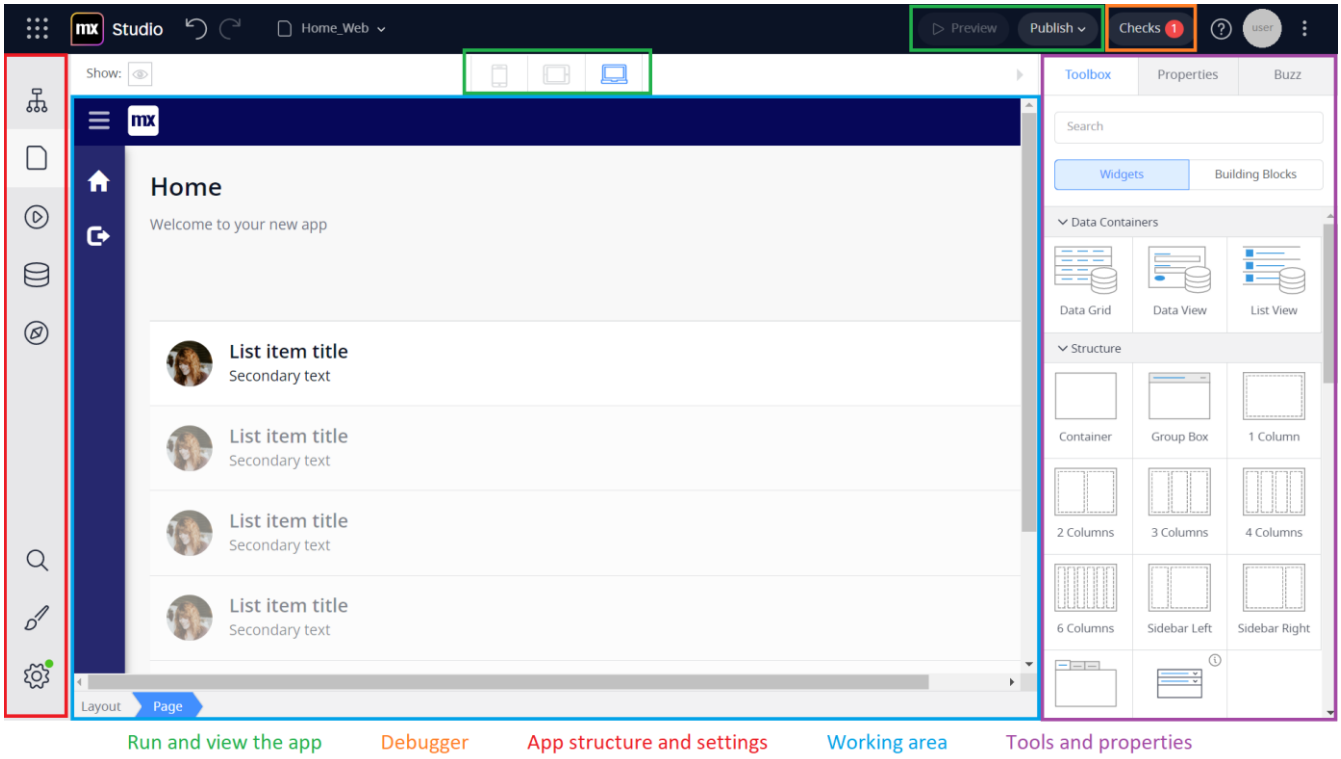

Figure 4: User interface of Mendix Studio no-code development environment.

3.2.2 Mendix Studio Pro

Mendix Studio Pro is a desktop-based low-code environment for professional developers (Figure 3). It is a model-driven environment which has more versatile functionalities than Mendix Studio and therefore enables development of more complex applications. (Mendix, 2019)

Studio Pro extends the functionalities in various ways. It has more advanced modelling tools for page and microflow editing, as well as for domain modelling and configuring integrations. Possibility to manage branch lines facilitates collaboration and that way accelerates application development. Studio Pro has more options for managing application security. Mendix Studio Pro also includes debugger to ease out finding and solving possible issues that occur during the development process. Along with Studio Pro, developers with technical background can extend and customize applications with CSS, Java, and JavaScript. Mendix

Studio Pro also enables application development in offline mode. Figure 5 visualizes the tools and user interface of Mendix Studio Pro low-code development environment. (Mendix, 2022g)

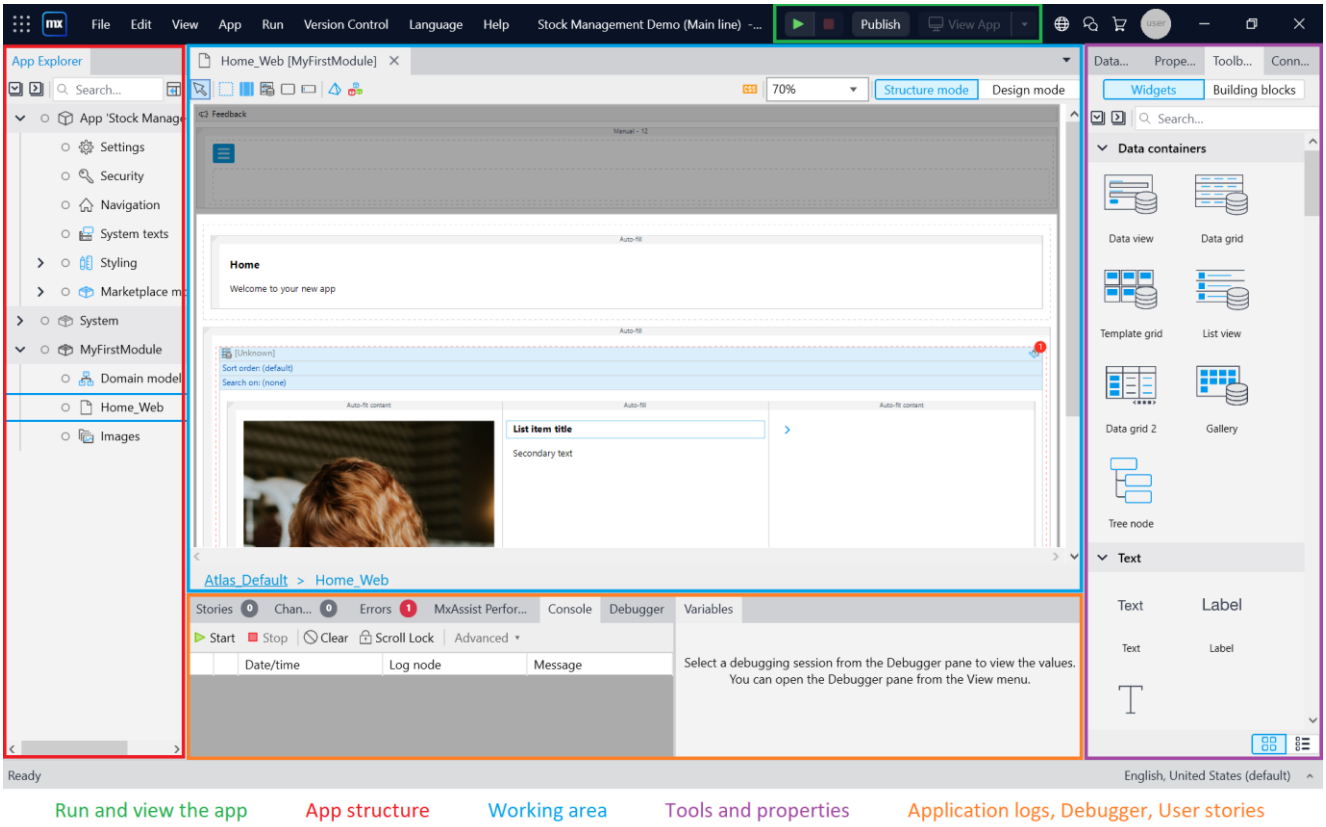

Figure 5: User interface of Mendix Studio Pro low-code development environment.

3.3 Team collaboration

Mendix provides several features for smoother collaboration within a team. The Developer Portal is Mendix's main tool for collaboration and managing projects. In the browser-based portal it is possible to create and manage user epics and user stories, manage team roles, leave feedback, and upload documents. The Developer Portal's Buzz shows the activity in an application project or even in a company. In addition, it allows the user to communicate with others by leaving comments. (Mendix Docs, 2022c)

The Developer Portal also includes team server where all the application's revisions are listed. The version control enables several developers to work simultaneously. Branches can be created from any desired development line and merged back when a feature is ready. Optionally, the version control in Mendix utilizes either Git or Apache Subversion. (Mendix Docs, 2022d)

3.4 Mendix Marketplace

Mendix Marketplace is a marketplace that contains various types of content to be utilized in Mendix application projects. It is fully integrated into both development environments and has wide selection of widgets, connectors, app services, templates, and ready-made solutions, among other things, created by both Mendix and the Mendix community. The marketplace does not only consist of a public marketplace, but also of a private marketplace for internal use within a company. (Mendix, 2022h)

3.5 Best practices for Mendix Application Development

A best practice is a recommended way of doing things to provide good results. Best practices usually represent the most efficient way of accomplishing a task. The best practices for Mendix application development and app performance are presented in Mendix Docs.

Mendix has also developed Mendix Assist to help developers produce high-quality applications. Currently the Assist consists of three different virtual co-developer bots of which MxAssist Performance Bot helps to utilize the Mendix Best Practices in the development of Mendix applications. (Mendix Docs, 2022a)

The best practices can be used as a guidance to ensure easily maintainable and reusable code with improved performance (Mendix Docs, 2022e). This section describes some of the key Mendix best practices.

3.5.1 Naming conventions

Naming conventions are detailed separately for modules, domain models, folders, microflows and for other document types in the Mendix Docs. In general, names should be de descriptive and clear. However, also additional guidance for different documents is provided. (Mendix Docs, 2022e)

Using UpperCamelCase is recommended for naming modules, entities, and entity attributes. Because an entity represents a single object, the name given to it should be singular, like “Stock” instead of “Stocks”. Using abbreviations, underscores or other special characters should be avoided in entity names. Exception for this are the names of entity attributes which are needed only for technical purpose. Those names should begin with an underscore, for example “_ExampleAttribute”. (Mendix Docs, 2022e)

Associations between entities are named automatically by Mendix and usually there is no need to worry about them. However, if there are more than one association between same entities, it is suggested to add an identifier to the names to better understand the use of the association. (Mendix Docs, 2022e)

For some documents there are standard prefixes defined in the Mendix Docs. There are prefixes for different types of microflows, layouts and snippets, enumerations, and integration documents. A prefix tells what kind of event is triggering the feature or what is the purpose of the document. For instance, microflows are generally named in the format "{PREFIX}_{Entity}_{Operation}". Thus, a microflow which creates a list of stocks and is triggered by an action button could be named as ACT_Stock_CreateList. (Mendix Docs, 2022e)

Suffixes, like _New or _Overview, are advised to be used to indicate the use of pages. Home pages on the other hand have a little different kind of recommendations as the name of a home page can be defined by device and user roles that have primary access to the page. (Mendix Docs, 2022e)

3.5.2 Domain models

In Mendix, there are several features that need to be considered while configuring domain models. Some of them may cause performance issues while others might result unexpected behaviour in the application and therefore their use should be avoided. For example, imprudent use of calculated attributes may affect application performance as a calculated attribute will be calculated whenever an object is used even if the attribute would remain unused. Also, event handlers in domain models tend to increase the risk of undesired behaviour. More secure option, than using event handlers, is to manually call a sub-microflow. (Mendix Docs, 2022e)

When creating associations between entities, it is also possible to configure delete behaviour for both entities. Delete behaviour defines how associated entities behave if one of the entities is deleted. This behaviour should be always specified but never fully trusted when deleting lots of data. (Mendix Docs, 2022e)

3.5.3 Microflows and nanoflows

As in application development in general, also in Mendix application development it is desirable to keep component sizes small. In most cases microflows and nanoflows in Mendix applications should not have more than 25 elements like action activities or loops. When needed, splitting microflows and nanoflows to smaller parts is possible with sub-microflows and sub-nanoflows. (Mendix Docs, 2022e)

Documentation and annotations in code are essential part of maintaining code readability and help other developers to better understand the usage of the microflow or nanoflow. Mendix Docs recommends adding a descriptive annotation especially at the beginning of complex flows and parts of a flow. Readability of code can be improved also by making sure that the

lines of the links between elements are not crossing, and by keeping microflows and nanoflows aligned from left to right. (Mendix Docs, 2022e)

A good practice to also implement in microflows is always using microflow error handling for integration and Java calls. Also debug and trace logging should be added to complex microflows. These practices help analysing and solving possible errors later. (Mendix Docs, 2022e)

3.5.4 Performance

In addition to the already mentioned cases, there are also a few other things that should be considered in terms of performance. For instance, instead of committing objects inside loops, it would be preferable to perform the commit of all objects at once outside the loop. Another example for boosting performance is to convert microflows, which do not require network to operate, to nanoflows. (Mendix Docs, 2022f)

3.5.5 App security

To help to prevent security vulnerabilities, Mendix has listed app security best practices to consider in application development.

Setting access rules on entities ensures that data is visible and editable only to authorized users whereas restriction options on widgets are meant to filter irrelevant data instead of securing the data from unauthorized use. Therefore, access rules should always be determined in entities and set separately for each attribute instead of giving permissions by default so that users are assigned only the necessary access. Special attention should be paid with anonymous users. Anonymous users should have access to an application only if there is a valid reason to include them as users. If an anonymous user is given the rights to create objects, the objects should be constrained only to the user who created them. (Mendix Docs, 2022g)

Strong password policy is an easy way to improve application security. With Mendix, a developer does not have to worry about applying strong policy as Mendix has already configured it by default. Accordingly, it is a good practice not to simplify the default password requirements. To improve application security even more, it is suggested to use third party identity providers which can provide more advanced and secure authentication mechanism. (Mendix Docs, 2022g)

Code injections are a security risk that should always be considered in application development. Mendix platform prevents injections in its original components by parameterizing all queries and this way making injections impossible. However, the developer is still responsible for protecting the application from injections for add-ons downloaded from

the Marketplace. Mendix offers some ready-made solutions to prevent injections in HTML content. The Community Commons Java function library, created by Mendix, includes XSSSanitize and HTMLEncode functions to remove malicious code from added HTML and to encode a string to HTML for safe display. Code injections are possible also in database connections which use user inputs. For these it is recommended to use prepared statements and to use expressions to monitor that the input contains only allowed characters. Malicious content might end up to an application also from files uploaded by the users of an application. To avoid this, it is suggested to either manually create a function that handles the risk or download a module from the Mendix Marketplace. (Mendix Docs, 2022g)

4 Implementation of the project

As mentioned in the introduction, the goal of the Mendix low-code application development project was to recreate a rapid inventory tool of an existing high-code stock management application. The main features of the rapid inventory tool are the possibility to make quick stock item quantity changes, and the possibility to search for stock items by bar code scanner or in a list. One of the requirements for the Mendix application was that the low-code application should be later extensible also to the other features of the original stock management application. In addition, the user interface of the Mendix application should be user-friendly and visually appealing.

The project was implemented with agile development mindset. Agile is a project management approach which splits up the project to several sprints for faster delivery. Depending on the project, a sprint can last from one to several weeks although all sprints within the same project are the same length. Sprints repeat the general stages of software development life cycle, from planning and designing to developing, testing, and delivering the end product (Figure 6).

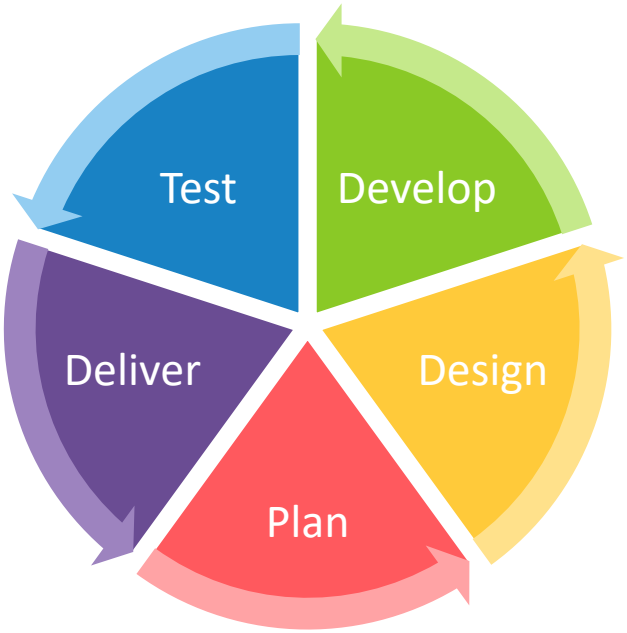

Figure 6: Agile development life cycle.

Early delivery and regular reviews and feedback increase customer satisfaction and product quality. This also makes development process less risky as the potential failures are detected earlier and thus take less time and money. (Layton, M., Ostermiller, S. & Kynaston, D. 2020, 423-425)

Because low-code application development is fast, the duration of a sprint in this project was set to one week. Every sprint started with a sprint planning meeting. The purpose of the meetings was to determine which tasks should be done during the starting sprint. The choices were influenced by available working hours of the team members and the defined order of the user stories in the backlog as well as their workload. During a sprint, daily scrum meetings were held to keep other team members updated about the progress of their tasks and help each other in case any challenges had emerged. After a sprint, there was hold a meeting for presenting the completed work and to get feedback from the product owner.

4.1 Planning

Technical specification took part in the planning phase, where the scope of the project got specified.

4.1.1 Research

Since one of the purposes of the project was to create a low-code version of one feature of an existing application, the first step was to get familiar with the previous implementation. At first the team had to install the development environment to be able to view and test the

rapid inventory functionality through the user interface. After getting to know to the functionality through the user interface, it was time to dive deeper. It was essential to know what kind of data was needed, as the low-code version would be getting its data from the API. The team investigated the data and API call URLs used in the feature and documented the findings in an excel file.

4.1.2 Data modelling

After knowing what data would be needed in the application, it was time to plan the data models to be able to build the application in correct way. This makes building the application smoother.

In Mendix there are two ways to handle data in an application: persistable and non-persistable entities. Persistable objects are stored in the application's database whereas non-persistable data is saved to the temporary memory which lasts for only one user session (Mendix, 2022i). The first question, that needed to be solved, was what data should be saved as persistable and what as non-persistable.

The data, that did not change often, like stocks and stock places, was decided to be retrieved when logging in to the application and to be saved to the application's database as persistable objects. That way the data was easily accessible for later use in the application. Not to forget about the benefits on performance for the user's point of view as part of the data was already retrieved.

However, not all the data was worthwhile to store in the application's database. Frequently changing data, like quantities of stock items, were decided to be retrieved every time that the information of stock items was needed to ensure that the data would be always up to date. This kind of data was saved as non-persistable objects.

4.1.3 User stories and backlog

During this project the team utilized the Mendix's own "Stories"-collaboration dashboard on Mendix Developer Portal for managing backlog. In the planning meeting, user stories were created for all the functionalities of the application. Like Field (2017) recommends in the blog published on MX Blog, the user stories followed the format "As a <user type>, I want to <do something>, so that I <business value>.". To make the development process clear for all the developers, the tasks required for implementing a user story were written down and arranged in order of implementation. The user stories were also categorized with labels and scored based on workload. At the time the project was implemented, the Mendix Developer Portal did not yet have the option to create epics. Therefore, epics were also defined with labels. Finally, the user stories in the backlog were arranged in order of implementation.

4.2 Designing

Considering the usability of the application was one of the key objectives in the project and therefore the team collaborated with an UX/UI-designer. The focus in the design was to streamline the workflow and to enhance the mobile usability. The goal was also to deliver a visually modern application. The UX/UI-design was presented as an interactive Figma prototype design.

4.3 Development

The development phase began with executing the user stories that were included in the minimum viable product (MVP). After finishing the MVP, the outcome of it was presented to the product owner. The MVP fulfilled the basic needs of the application, but to enhance the user experience more, it was decided to add additional features to the application. This meant implementation of the rest of the user stories.

The team used Mendix Studio Pro as the development environment for the project because the development team consisted of only professional developers. The best practices defined by Mendix were utilized in the development work.

4.3.1 Version control

A new branch was created from the main development branch each time when starting to work with a new user story. Having individual branches for each user story secured the main development branch from possible mistakes and errors that might occur while developing a new feature. Usually only one developer worked with one user story but a few times it was necessary to include two developers in the process. In these cases, Mendix Studio Pro's "update branch"-feature allowed the work to be done concurrently. Best practices include always updating a branch to the latest revision before continuing work on the local copy of the branch to avoid and resolve possible conflicts. However, the platform requires updating the branch at the latest before committing new changes to the version control server.

To ensure good quality code, the code was reviewed by another team member each time a user story was completed. Only after the needed changes in the code were made, the sub development branch was merged to the main development branch.

4.3.2 Coding

The entire application was implemented using ready-made components from Mendix and no custom code was written to expand the application. Only for some custom styling was used JavaScript style sheets.

Maintainability as well as smooth navigation were some main considerations when forming the folder structure. When creating the folder structure, the application was first divided into modules based on its main functionality. Next, the files inside a module were grouped in folders by types like microflows, pages and REST resources. This way the files were easily located when needed.

Pages were built mainly based on the UX/UI-design. Some changes and additions were made based on remarks that came up during the development process.

Keeping sizes of microflows and nanoflows compact enables easier reuse and maintainability of the functionalities. The aim during this project was, that one microflow or nanoflow only took care of one function. This was easily accomplished as with Mendix platform it is possible to call sub microflows and sub nanoflows from the main function. For instance, the nanoflow in figure 7 has sub microflow “CWS_GetStockItemList” and sub nanoflow “SUB_SearchStockItem_FilterList”.

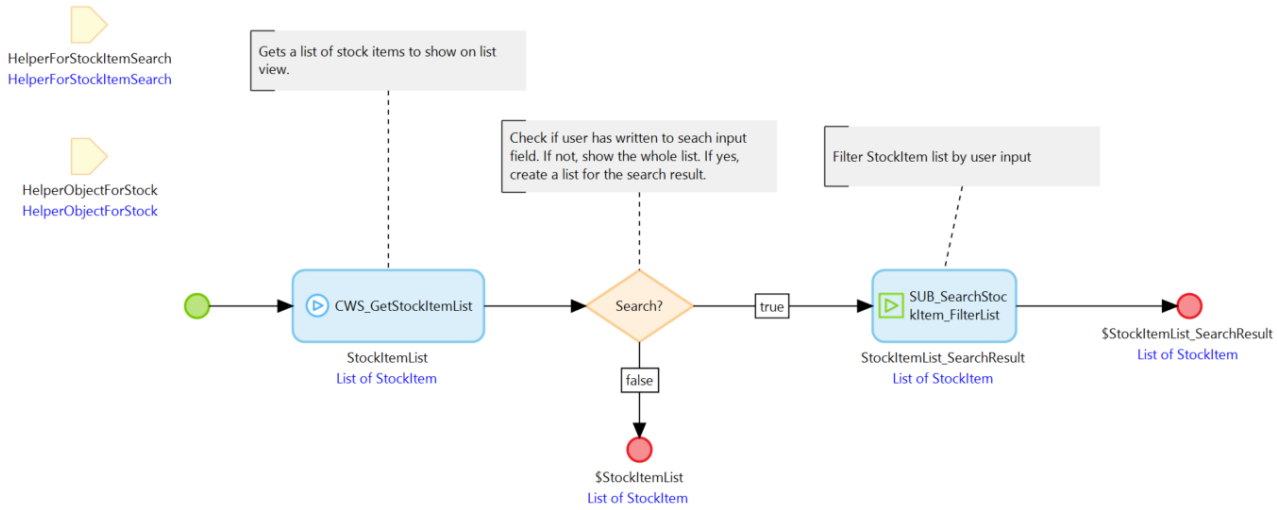

Figure 7: A nanoflow with sub microflow and nanoflow in the Mendix application.

Working in a team emphasized the importance of comments in the code as annotations significantly improve code readability. Even if a functionality of a microflow or nanoflow is clear to the creator, it might be puzzling to another developer. In the project the team followed the recommendations of Mendix’s best practices when commenting the code.

As mentioned, all the needed data for the Mendix application was received from the API. API calls were made for retrieving and submitting stock data as well as for login functionality. The first step in calling to REST service is to create the right JSON structure. In the case of this project, the correct JSON structures were discovered during the research phase. Next step is to create an import mapping which specifies how the JSON structure will be formed as

an entity. After configuration, calling REST services is done in a microflow by adding “Call REST” activity (Figure 8). All the needed details, as REST URL, parameters and HTTP headers, will be defined in the activity. Important part of REST calls is also to configure error handling. If error handling is not configured and an error occurs, the running microflow will be aborted without being able to inspect the details of the error. As figure 8 visualizes, error handling is seen as an orange circle on the chosen activity. In case of an error, the details will be logged to ease out problem detection.

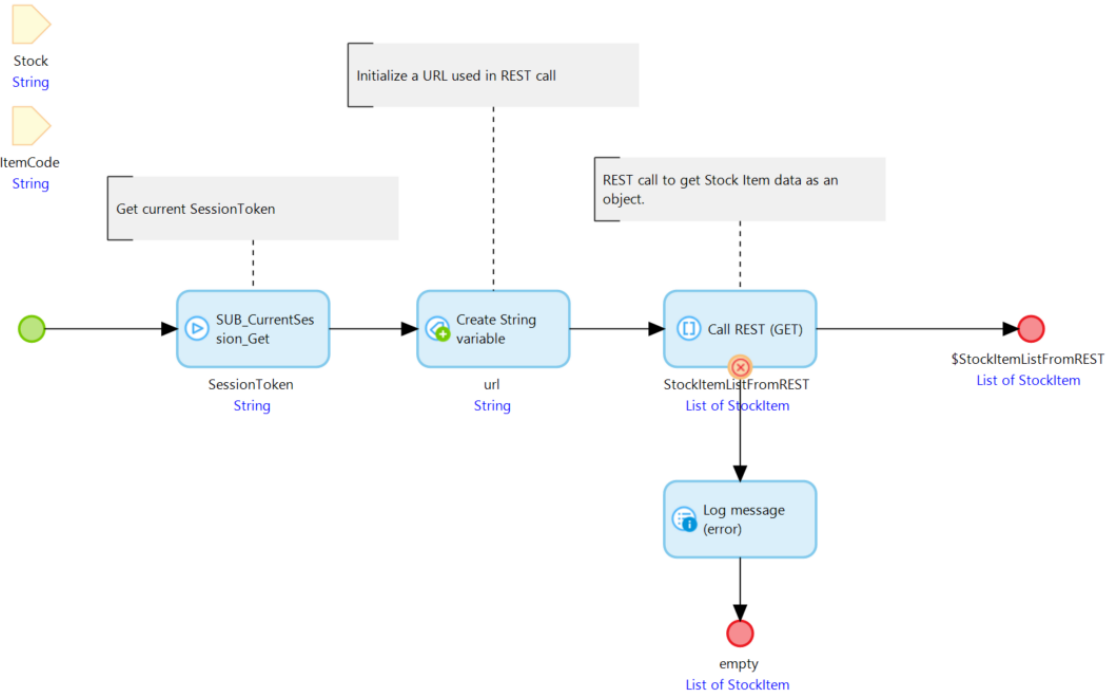

Figure 8: A microflow with REST call in the Mendix application.

4.3.3 App security

Two user groups were defined for the application, administrator and the end user. As recommended by the Mendix best practices, read and write permissions were defined for both user groups separately for each entity attributes. User accesses were also defined in Mendix for both user groups for each page and microflow.

In addition to setting Mendix access rules, validity checks of the API token were added on each page and microflow, which includes API calls, to avoid unauthorized use. If the user's token is not valid, the user is redirected back to the login page.

4.4 Testing

Test plan was created to define which tests are reasonable to carry out to test the MVP application. The plan considered the automated checks administered by Mendix. For instance, Mendix monitors the consistency of the functions and page elements within the application model. The platform also takes care of testing the Mendix components (Mendix, 2022j). In addition, the platform includes built-in debugging tools which reduce the chances for code errors. If the development platform detects a problem, it displays an error message and prevents the application from running.

After all, it was decided that automated unit tests would be created for the most important functionalities of the application and manual testing of the application would be performed through the user interface. For unit testing Mendix provides Application Test Suite (ATS) module which integrates into the platform making unit testing easy for also developers without previous experience of writing test cases (Mendix, 2022j). Due to that, ATS module was decided to be used for unit testing in this project.

4.5 Delivery

As mentioned before, the progress on the development work was presented after every sprint. In these meetings the team received valuable feedback for the development work. After the project was done, there was held a final meeting to present the end product with documentation and a live demo.

The end product is designed to be used on a mobile device. As mentioned, it is implemented with API integrations and gets all the necessary data from the API. The main functionalities of the end product are login, searching stock items by bar code scanner and the possibility to edit stock item quantities. Other features are the possibility to change the default stock location, to browse stock items in a list and a manual search.

In this chapter the end product is presented screen by screen. Logos and other recognizable information have been censored from the figures due to confidentiality.

4.5.1 Login

When opening the application, user lands on the login page (Figure 9). The page has input fields for username and password. The login function makes a POST request to the API and depending on the received answer, either directs the user forward to home page or gives an error message.

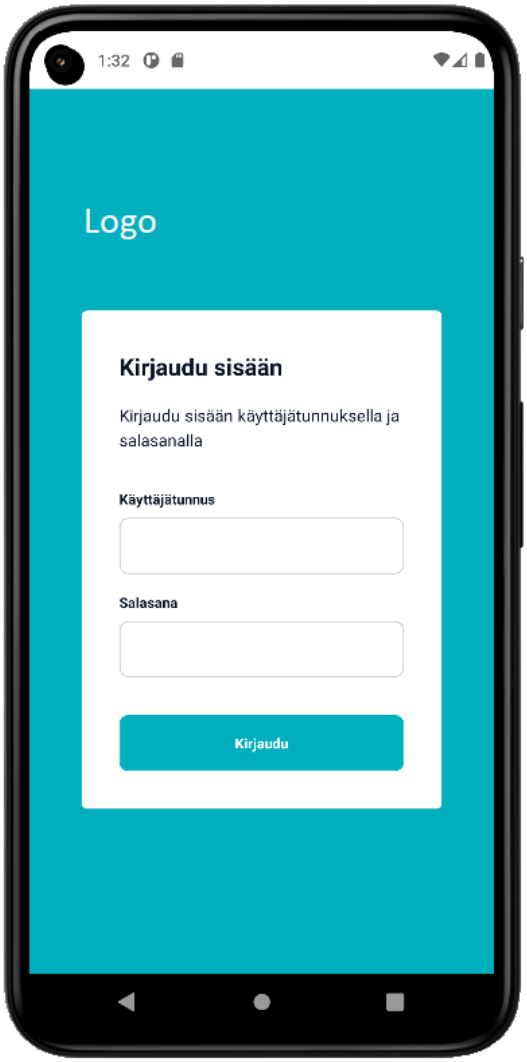

Figure 9: Login page in the Mendix application.

4.5.2 Home page

Logging in directs the user to the application's home page (figure 10) which includes a menu, from which various features of the application can be accessed. Though as mentioned before, only the rapid inventory -feature was created during this project and therefore the other menu items do not lead anywhere.

To make the use of the application as clear and informative as possible, the user’s default stock and location obtained from the login information is displayed right from the front page until the end of the workflow.

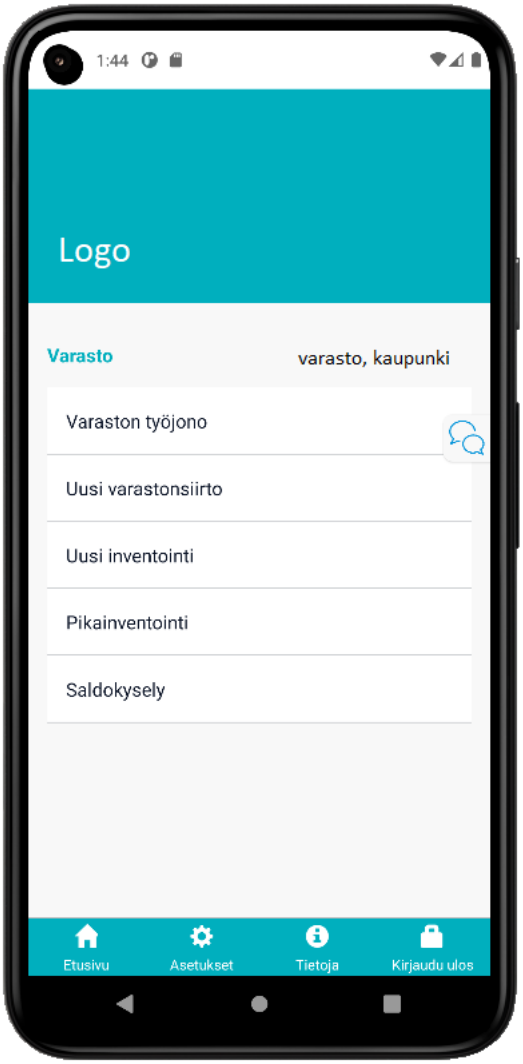

Figure 10: Home page in the Mendix application.

4.5.3 Rapid inventory page

On the rapid inventory page (Figure 11), the aim was to speed up the workflow by highlighting the stock item bar code scanner feature. To improve user-friendliness, the bar code search button “Aloita nimikehaku” was made clearly distinguishable by colouring and placing. Attention was also paid to the sizes of the other buttons on the page to ensure easy use of the application. To improve usability even more, changing the stock and location was made possible also on the rapid inventory page in addition to options page.

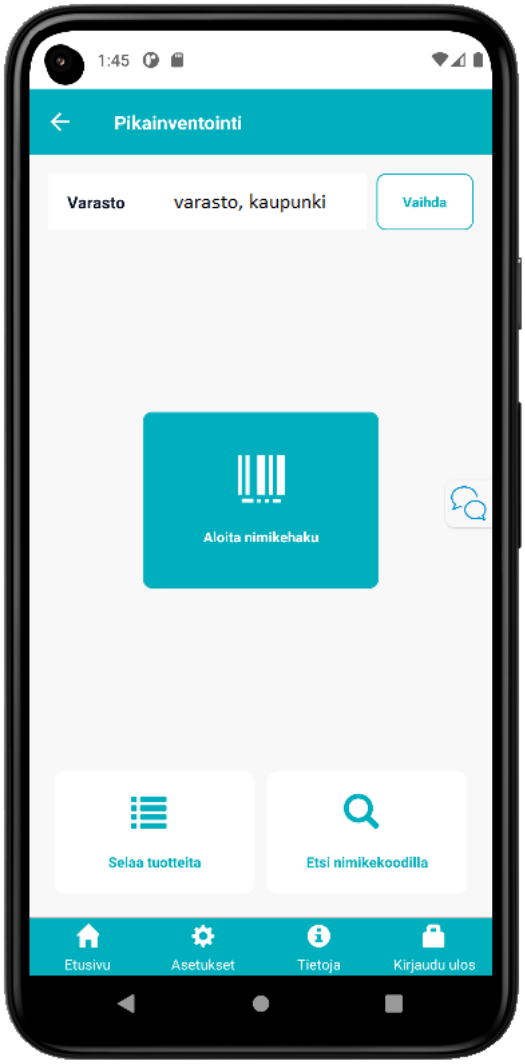

Figure 11: Rapid inventory page in the Mendix application.

4.5.4 Bar code scanner

The bar code scanner (Figure 12) reads a stock item bar code and directs the user to the stock item view if the stock item is available only on one stock place in the current stock. If the stock item is in several stock places, the application shows the user a list of all stock places where the stock item is available.

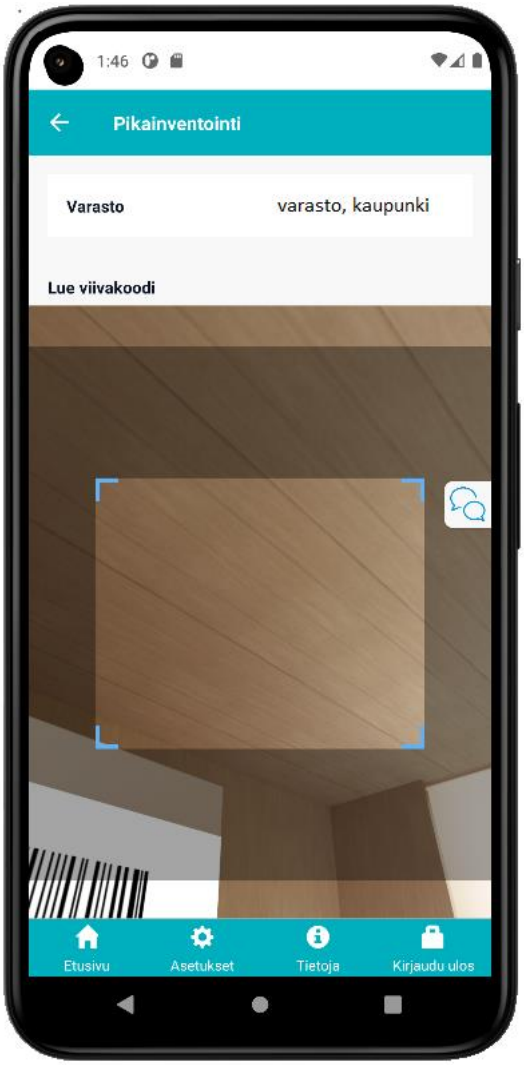

Figure 12: Bar code scanner in the Mendix application.

4.5.5 Stock place list and stock item list pages

In addition to the possibility to search stock items using bar code scanner, it is also possible to browse stock items in a list. First the user is guided to choose a stock place (figure 13) after which a list of stock items in that stock place opens. It is possible to filter products from the lists by user input.

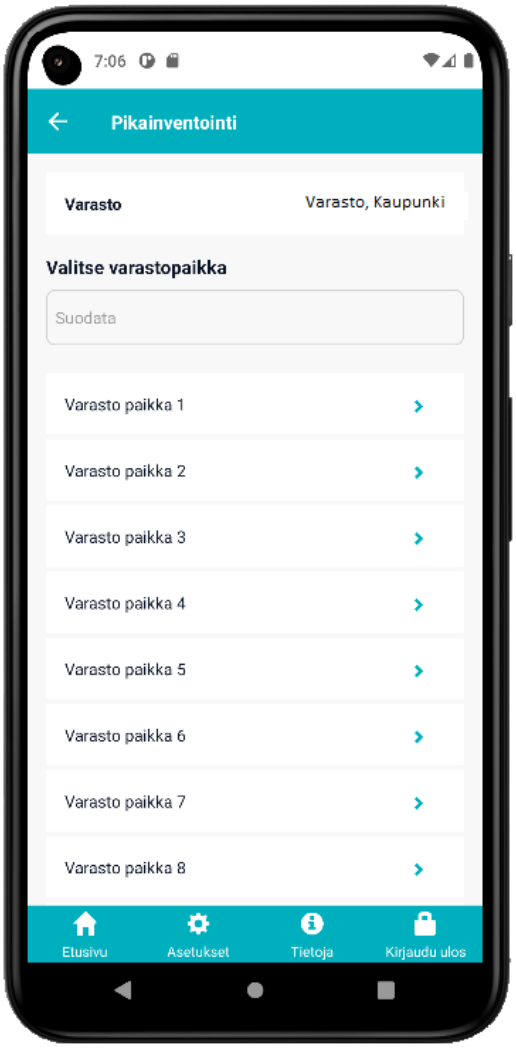

Figure 13: Stock place list in the Mendix application.

4.5.6 Stock item page

On the stock item page (Figure 14) it is possible to view the stock item information and edit the stock item quantity. Other functions on the page include the possibility to change the quantity unit for example from single unit to pallets, and possibilities to change the inventory date and add a note.

The plus and minus buttons were added to enhance usability on this page. With the buttons it is possible to make small quantity changes without having to clear the entire input field and having to write a new amount. To ease out the inventory work, an input field “Muutos” was added to display the difference of the new stock item quantity to the original stock item quantity. This reduces the user's responsibility for remembering and the possibility of making mistakes. In addition, it is possible to only enter the change in quantity. Either way, the quantity amount and quantity change fields are synchronized with each other. Changes made

to one field will also update the amount of the other field. Sometimes users may find themselves in a situation where the amounts get too mixed up and it would be easier to just start over. For these situations, “Nollaa muutos”-button was added to reset the user inputs.

After all desired changes are done, the user can save the changes and search for another stock item either by bar code scanner or in the list. As mobile phones are often used with one hand, the buttons on the stock item page were placed at the bottom of the view to increase the usability more.

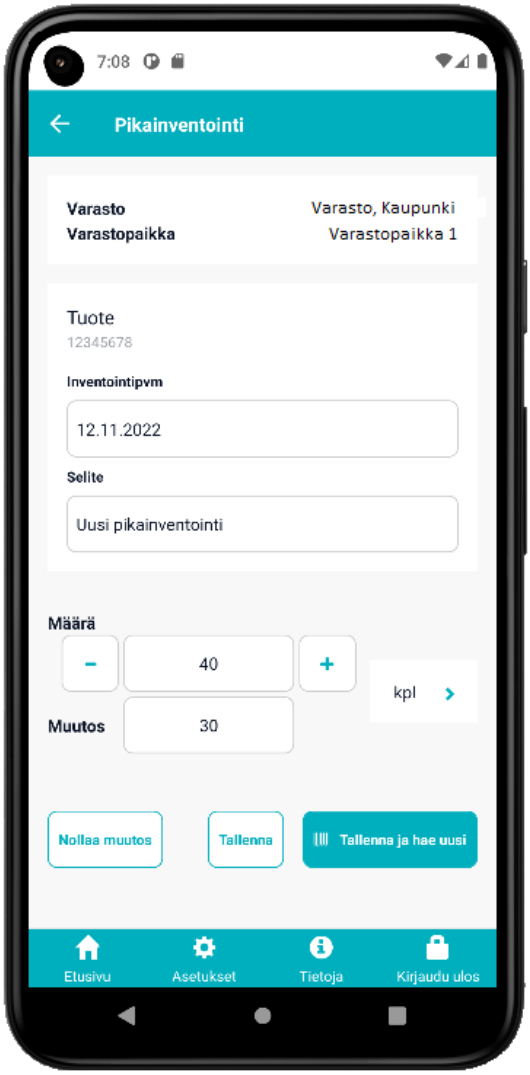

Figure 14: Stock item page in the Mendix application.

4.6 Outcomes of the project

Table 1 summarizes the findings related to the evaluation of best practices through the application development project. These outcomes include best practices that could be considered in Mendix application development. Since Mendix provides comprehensive pre-

defined best practices, there is no need to create own guidelines for practices related to the platform.

Relates to	Best practice	Benefits
Data modelling	Define data models at the beginning of the project. Define rather the data should be saved as persistable or non-persistable entities.	Enhances application performance and helps in the correct construction of the application.
Backlog	Create user stories and tasks for each functionality. Categorize with labels and epics.	Makes sure the team knows what needs to be done.
Version control	Create a new branch for each new feature.	Secures the main development branch from mistakes and errors.
Code reviewing	Code review by another developer after a task is ready.	Ensures good quality code and helps to identify possible mistakes.
Folder structure	Create consistent folder structure	Supports maintainability and extensibility.
Testing plan	Define what is reasonable to test, who should do the testing and which tools to use.	Helps to manage risks and ensures good quality of the product.
Mendix best practices	Get familiarized with best practices defined by Mendix such as component sizes, annotations, naming conventions, error handling, application security, and defining user groups and access rules among other best practices.	Helps to create consistent and secure Mendix application.

Table 1: Best practices for Mendix development

5 Conclusions

One of the Mendix application's requirements was, that it should be extensible for more features later on. Therefore, the application should be reusable and easy to understand. To ensure that, utilization of best practices was obvious. Before the project, the team had basic knowledge of software development best practices, but not much experience in how to implement best practices on the Mendix low-code platform. Because the team had to familiarize themselves with Mendix best practices during the development process, the progress of the project was not as efficient as it could have been.

Nevertheless, the requirements for the application were met. Usability and accessibility were considered in the implementation of the application. An application structured with app modules and a clear folder structure will make it easy to add new features in the future. The Mendix best practices helped for example with creating consistent component names and reminded of the importance of keeping microflows compact for reusability.

Compared to creating a high-code solution, model-driven development with Mendix platform enables much faster progress in projects. Automated processes made application building smoother as it removed the need to create repetitive, basic components commonly used in applications. In addition, the built-in debugging tools saved a lot of time in detecting code errors. Based on this application development project, Mendix Studio Pro platform seems like a great tool for a professional developer. For people with less technical skills low-code opens up opportunities to get more involved in application development, although the thought of a business user creating applications without the help of a professional seems still quite far-fetched.

As already stated in the introduction of this thesis, adopting a new technology is a journey that takes time before getting the best out of all the features and possibilities. Working on this project gave better understanding on the importance of best practices in application development. The best practices answered many questions that came up during the project and improved not only the quality of the code, but also team collaboration.

References

Printed

Bock, A. & Frank, U. 2021. Low-Code Platform. Journal of Business & information systems engineering 2021 vol. 64. Essen: University of Duisburg-Essen, 733-740.

Layton, M., Ostermiller, S. & Kynaston, D. 2020. Agile Project Management for Dummies. Hoboken: John Wiley & Sons, Inc.

Silva, C., Vieira, J., Campos, J., Couto, R. & Ribeiro, A. 2021. Development and Validation of a Descriptive Cognitive Model for Predicting Usability Issues in a Low-Code Development Platform. Journal of Human Factors 2021, vol. 64. 1012-1032.

Wong, J., Iijima, K., Leow, A., Jain, A. & Vincent, P. 2021. Magic Quadrant for Enterprise Low-Code Application Platforms. Gartner.

Woo, M. 2020. The Rise of No/Low Code Software Development – No Experience Needed?. Journal of Engineering 2020 vol. 6. Beijing, 960-961.

Electronic

Den Haan, J. 2022. 5 Benefits of Low-Code Application Development. Posted 13 June. Accessed 22 October 2022. https://www.mendix.com/blog/benefits-low-code-development/

Drenik, G. 2022. Bridging The Gap Between Business And IT: Low-Code Helps Accelerate Innovation and Cultural Change. Accessed 5 November 2022. https://www.forbes.com/sites/garydrenik/2022/07/20/bridging-the-gap-between-business-and-it-low-code-helps-accelerate-innovation-and-cultural-change/

Fahrion, M. 2022. Navigating The Digital Transformation Journey. Accessed 5 November 2022. https://www.forbes.com/sites/forbestechcouncil/2022/10/25/navigating-the-digital-transformation-journey/

Field, D. 2017. Best Practices & Tools for Agile Sprint Planning. Posted 8 August. Accessed 9 October 2022. https://www.mendix.com/blog/agile-tools-within-mendix/

Jednaszewski, M. 2022. What is No-Code vs. Low-Code? Differences, Similarities, and Use Cases. Posted 2 June.agi Accessed 22 October 2022. https://www.mendix.com/blog/understand-no-code-vs-low-code-development-tools

Jepsen, C. 2021. Does Low-Code/No-Code Spell The End For IT Professionals?. Accessed 30 September 2022. https://www.forbes.com/sites/forbestechcouncil/2021/08/11/does-low-codeno-code-spell-the-end-for-it-professionals/

Lawton, G. 2021. What is the future of low-code app development? Accessed 22 October 2022. https://www.techtarget.com/searchsoftwarequality/feature/What-is-the-future-of-low-code-app-development

Mendix. 2022a. The Mendix Vision. Accessed 6 November 2022. https://www.mendix.com/evaluation-guide/mendix-vision/

Mendix. 2022b. The Definitive Guide to Low-Code Development. Accessed 16 September 2022. https://www.mendix.com/low-code-guide/

Mendix. 2022c. Company. Accessed 22 September 2022. https://www.mendix.com/company/

Mendix. 2022d. Schiphol Airport Empowers Workforce with Native Mobile Applications. Accessed 12 November 2022. https://www.mendix.com/customer-stories/in-case-you-missed-it-mendix-native-mobile-empowers-workforce-at-schiphol-airport/

Mendix. 2022e. The Mendix Low-Code Platform. Accessed 22 September 2022. https://www.mendix.com/platform/

Mendix. 2022f. What Is Mendix? Accessed 22 September 2022. https://www.mendix.com/evaluation-guide/what-is-mendix/

Mendix. 2022g. App Development. Accessed 22 September 2022. https://www.mendix.com/evaluation-guide/app-lifecycle/app-development/

Mendix. 2022h. Marketplace. Accessed 17 October 2022. https://www.mendix.com/evaluation-guide/app-lifecycle/app-store/

Mendix. 2022i. Data Storage. Accessed 23 September 2022. https://www.mendix.com/evaluation-guide/app-capabilities/data-storage/

Mendix. 2022j. Test Automation & Quality Assurance. Accessed 11 November 2022. https://www.mendix.com/evaluation-guide/app-lifecycle/test-automation-quality-assurance/

Mendix. 2019. Mendix Announces Studio and Studio Pro; No-Code and Low-Code Visual Development Environments. Accessed 23 September 2022. https://www.mendix.com/press/mendix-announces-studio-and-studio-pro-no-code-and-low-code-visual-development-environments/

Mendix Docs. 2022a. MxAssist Performance Bot. Accessed 19 October 2022. https://docs.mendix.com/refguide/mx-assist-performance-bot/

Mendix Docs. 2022b. Studio 9 Guide. Accessed 22 September 2022. https://docs.mendix.com/studio/

Mendix Docs. 2022c. Developer Portal Guide. Accessed 22 October 2022. https://docs.mendix.com/developerportal/

Mendix Docs. 2022d. Version Control. Accessed 22 October 2022. https://docs.mendix.com/refguide/version-control/

Mendix Docs. 2022e. Implement Mendix Best Practices for Development. Accessed 16 October 2022. https://docs.mendix.com/howto/general/dev-best-practices

Mendix Docs. 2022f. Performance Best Practices. Accessed 17 October 2022. https://docs.mendix.com/refguide/performance-best-practices

Mendix Docs. 2022g. Implement Best Practices for App Security. Accessed 17 October 2022. https://docs.mendix.com/howto/security/best-practices-security/

Milanesi, C. 2020. Digital Transformation And Digital Divide Post COVID-19. Accessed 5 November 2022. https://www.forbes.com/sites/carolinamilanesi/2020/05/11/digital-transformation-and-digital-divide-post-covid-19/?sh=7149899d1656

Mocke, R. 2021. How to develop with best practices. Accessed 5 November 2022. https://medium.com/mendix/how-to-develop-with-best-practices-mendix-how-to-1040f341672

Stamford, C. 2021. Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 23% in 2021. Accessed 18 September 2022. https://www.gartner.com/en/newsroom/vapress-releases/2021-02-15-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-23-percent-in-2021

Figures

Tables