TB141Ic – ICT System Engineering and Rapid Prototyping

Agile software development with Mendix & MVC in Mendix: View - Pages

Jacopo De Stefani, Ph.D

Delft University of Technology, The Netherlands

20/03/2023

Licensing information

Learning objectives and related literature

Learning objectives

- Understand the the different activities in a practical Agile project.
- Create a basic Mendix app
- Implement a basic input mechanism for the user to enter data in the Mendix app
- Implement a basic visualization for the available data in the Mendix app

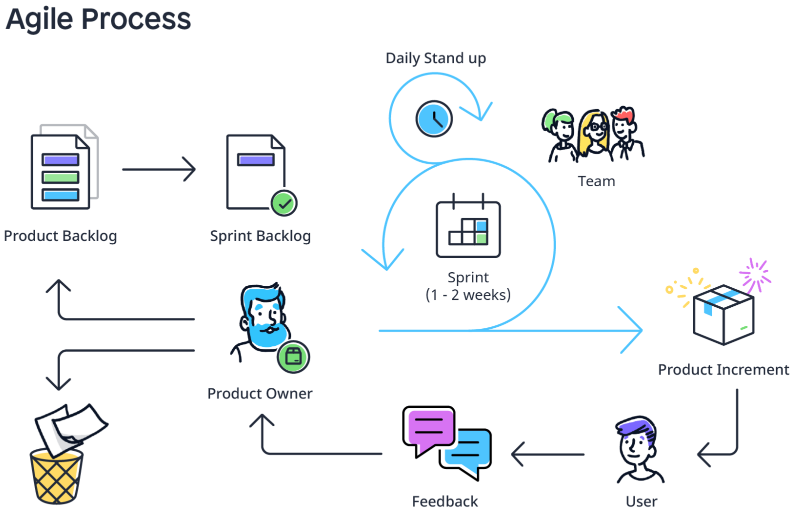
Related literature

- Chapter 3 [1]
- Mendix - Build an App in Mendix Studio Learning Path
- Mendix - Become a Rapid Developer Learning Path

[1] I. Sommerville (2011). "Software engineering 9th Edition". In: *ISBN-10* 137035152, p. 18

Agile project management

Summative Assignment - Methodology - Overview



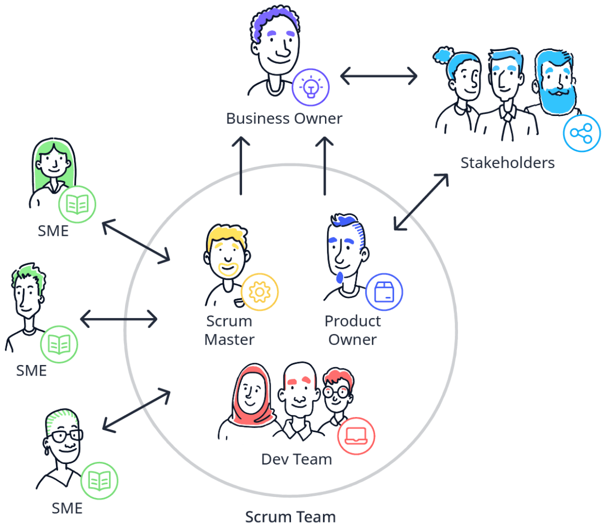
Agile project management

- The principal responsibility of software project managers is to manage the project so that the software is delivered on time and within the planned budget for the project.
- The standard approach to project management is plan-driven. Managers draw up a plan for the project showing what should be delivered, when it should be delivered and who will work on the development of the project deliverables.
- Agile project management requires a different approach, which is adapted to incremental development and the practices used in agile methods.

Scrum

- Scrum is an agile method that focuses on managing iterative development rather than specific agile practices.
- There are three phases in Scrum:
  1. The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.
  2. This is followed by a series of sprint cycles, where each cycle develops an increment of the system.
  3. The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

Summative Assignment - Methodology - Roles



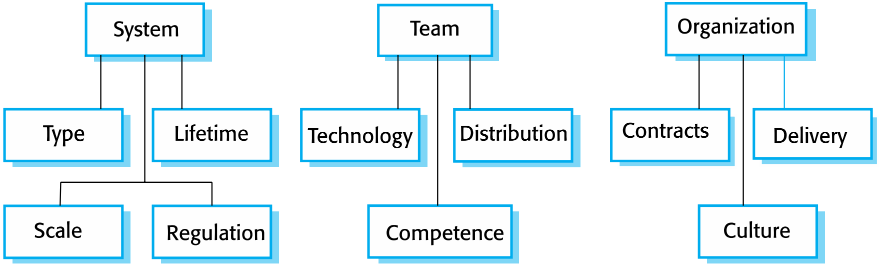
Scrum terminology (a)

Scrum term	Definition
Development team	A self-organizing group of software developers, which should be no more than 7 people. They are responsible for developing the software and other essential project documents.
Potentially shippable product increment	The software increment that is delivered from a sprint. The idea is that this should be 'potentially shippable' which means that it is in a finished state and no further work, such as testing, is needed to incorporate it into the final product. In practice, this is not always achievable.
Product backlog	This is a list of 'to do' items which the Scrum team must tackle. They may be feature definitions for the software, software requirements, user stories or descriptions of supplementary tasks that are needed, such as architecture definition or user documentation.
Product owner	An individual (or possibly a small group) whose job is to identify product features or requirements, prioritize these for development and continuously review the product backlog to ensure that the project continues to meet critical business needs. The Product Owner can be a customer but might also be a product manager in a software company or other stakeholder representative.

Scrum terminology (b)

Scrum term	Definition
Scrum	A daily meeting of the Scrum team that reviews progress and prioritizes work to be done that day. Ideally, this should be a short face-to-face meeting that includes the whole team.
ScrumMaster	The ScrumMaster is responsible for ensuring that the Scrum process is followed and guides the team in the effective use of Scrum. He or she is responsible for interfacing with the rest of the company and for ensuring that the Scrum team is not diverted by outside interference. The Scrum developers are adamant that the ScrumMaster should not be thought of as a project manager. Others, however, may not always find it easy to see the difference.
Sprint	A development iteration. Sprints are usually 2-4 weeks long.
Velocity	An estimate of how much product backlog effort that a team can cover in a single sprint. Understanding a team's velocity helps them estimate what can be covered in a sprint and provides a basis for measuring improving performance.

Scrum sprint cycle



The Scrum sprint cycle

- Sprints are fixed length, normally 2–4 weeks.
- The starting point for planning is the product backlog, which is the list of work to be done on the project.
- The selection phase involves all of the project team who work with the customer to select the features and functionality from the product backlog to be developed during the sprint.

The Sprint cycle

- Once these are agreed, the team organize themselves to develop the software.
- During this stage the team is isolated from the customer and the organization, with all communications channelled through the so-called ‘Scrum master’.
- The role of the Scrum master is to protect the development team from external distractions.
- At the end of the sprint, the work done is reviewed and presented to stakeholders. The next sprint cycle then begins.

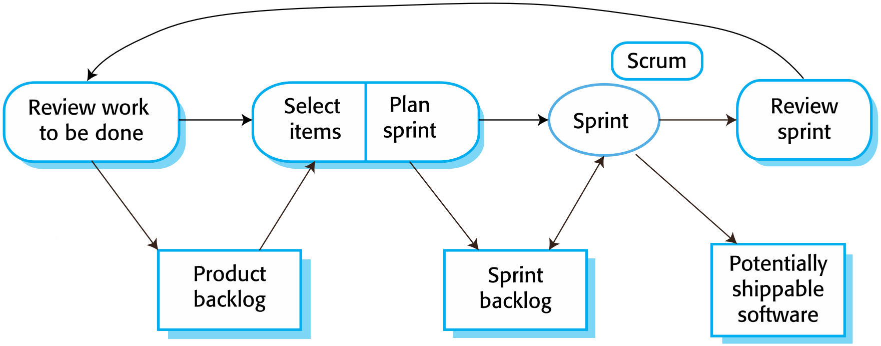
Teamwork in Scrum

- The ‘Scrum master’ is a facilitator who arranges daily meetings, tracks the backlog of work to be done, records decisions, measures progress against the backlog and communicates with customers and management outside of the team.
- The whole team attends short daily meetings (Scrums) where all team members share information, describe their progress since the last meeting, problems that have arisen and what is planned for the following day.
- This means that everyone on the team knows what is going on and, if problems arise, can re-plan short-term work to cope with them.

Scrum benefits

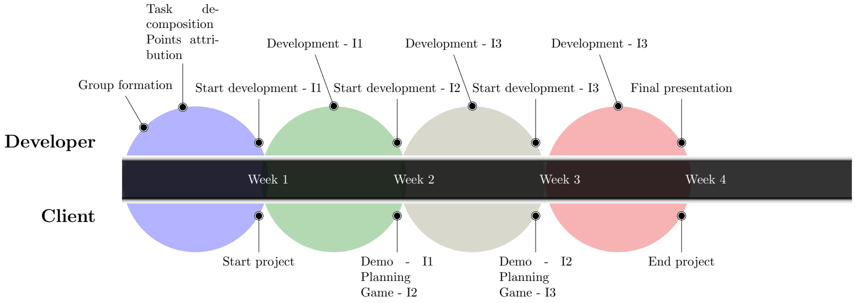
- The product is broken down into a set of manageable and understandable chunks.
- Unstable requirements do not hold up progress.
- The whole team have visibility of everything and consequently team communication is improved.
- Customers see on-time delivery of increments and gain feedback on how the product works.
- Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.

Distributed Scrum



Summative Assignment - Practical Activities in Agile project

Summative Assignment - Overview & Main activities



- User stories decomposition and estimation
- Planning game
- Client Demo

Summative Assignment - Activities - User stories

- User stories represent the functionalities that the **Client** want to be implemented.
- User stories needs to be decomposed in tasks that **Developer** can implement.
- For **each task**, **Developers** needs to estimate the number of points required to complete the task.
    - 1 point ≡ 1 hour of work for a single developer
- Task decomposition should include **all the activities** required to the development of the user story, for instance:
    - Coding
    - Reading documentation
    - Debugging
    - Completing follow-up documentation
    - Additional training
- $Points_{Story} = \sum_i Points_{Task_i}$

Summative Assignment - Activities - User stories - Example

User Story 1: ADD STORE ON PLATFORM

Description : `Businesses` can register on the platform using a form asking at least for the name of their business, their address, their VAT number and an official telephone number. `Admins` should have read/modify access to all this information for debugging/troubleshooting purposes. Both must also be able to modify their information at any time via a profile menu.

Client priority : **2**
Technical Risk Developers :
Points :

- Points ≡ Time needed to develop the user story
- Technical Risk → **1, 2, 3** ≡ **Weak, Medium, Strong**

Summative Assignment - Activities - User stories - Example

User Story 1: ADD STORE ON PLATFORM

Description : `Businesses` can register on the platform using a form asking at least for the name of their business, their address, their VAT number and an official telephone number. `Admins` should have read/modify access to all this information for debugging/troubleshooting purposes. Both must also be able to modify their information at any time via a profile menu.

Client priority : **2**
Technical Risk Developers : **To determine before the meeting**
Points :

- Points ≡ Time needed to develop the user story
- Technical Risk → **1, 2, 3** ≡ **Weak, Medium, Strong**

Summative Assignment - Activities - User stories - Example

User Story 1: ADD STORE ON PLATFORM

Description : `Businesses` can register on the platform using a form asking at least for the name of their business, their address, their VAT number and an official telephone number. `Admins` should have read/modify access to all this information for debugging/troubleshooting purposes. Both must also be able to modify their information at any time via a profile menu.

Client priority : **2**
Technical Risk Developers : **3**
Points : **To determine before the meeting**

- Points ≡ Time needed to develop the user story
- Technical Risk → **1, 2, 3** ≡ **Weak, Medium, Strong**

Summative Assignment - Activities - Planning game

- In the **planning game**, points, technical risk and user priority are used to determine the next stories to be implemented.
- The **Client** selects one or more stories that, ideally:
  - **Maximizes** his priority
  - **Minimizes** the technical risk
  - **Fit** the number of points available by the developers
- $Points_{available} = Hours_{dev,week} \times Weeks \times Developers$
  - $3[\frac{h}{week}]$ in your case per developer
  - $12 - 18\ [h]$ per group according to group size
- **Client** has the final word on planning game, but **Developers** can intervene by adjusting their estimation **before the planning game**.

Summative Assignment - Activities - Demo

- The Demo aims at showcasing the features implemented by the **Developers** to the **Client**.
- In this project:
  - 5-minutes **client** presentation of the functionalities implemented during the previous week.
  - No technical jargon
  - No technical rundown of functionalities
  - Launching the app via a double click
- **Best practice:** Prepare beforehand a scenario to be performed during the demo.

Summative Assignment - Activities - Wrap-up

- **Before the weekly meeting**
  - Estimation/Re-estimation of points and technical risks **for all the user stories**
  - Task decomposition (and corresponding estimation)
  - Complete the follow-up document
  - Upload all of them on **Mendix**

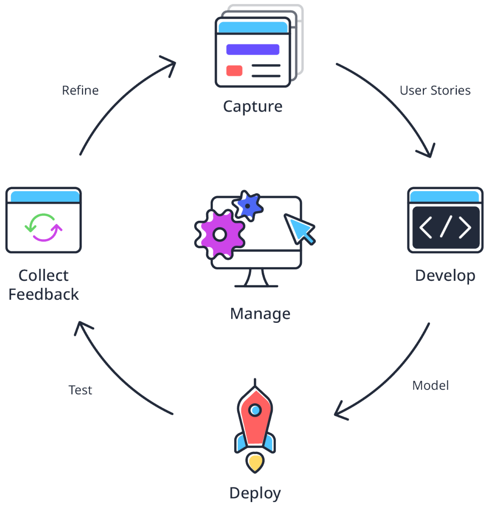
Summative Assignment - Activities - Wrap-up

- **Before the weekly meeting**
  - Estimation/Re-estimation of points and technical risks **for all the user stories**
  - Task decomposition (and corresponding estimation)
  - Complete the follow-up document
  - Upload all of them on **Mendix**
- **During the weekly meeting**
  - Demo
  - Planning game
  - Choice of the user stories **by the Client**

Summative Assignment - Activities - Wrap-up

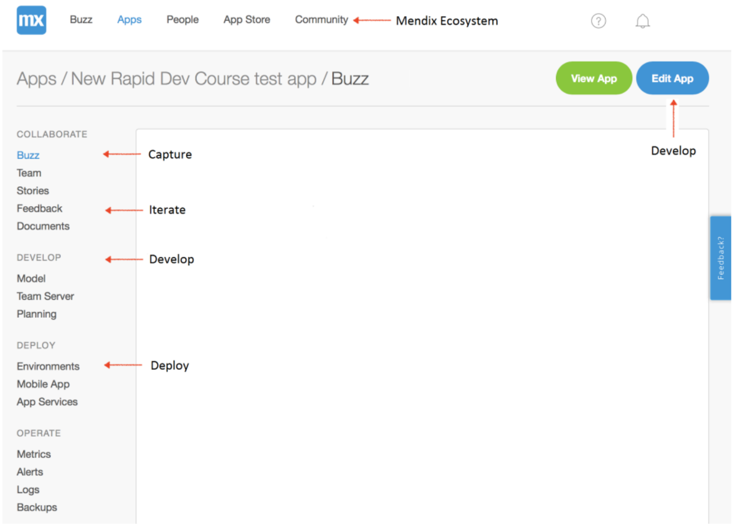
- **Before the weekly meeting**
  - Estimation/Re-estimation of points and technical risks **for all the user stories**
  - Task decomposition (and corresponding estimation)
  - Complete the follow-up document
  - Upload all of them on **Mendix**
- **During the weekly meeting**
  - Demo
  - Planning game
  - Choice of the user stories **by the Client**
- **After the meeting**
  - Implementation task allocation to group members
  - Development
  - Follow up

Summative Assignment - Methodology - Life Cycle



- The template format for the stories (.xls) can be directly imported in Mendix.
- The updated version of the follow-up document should also be uploaded on Mendix.

Summative Assignment - Methodology - Life Cycle



Mendix

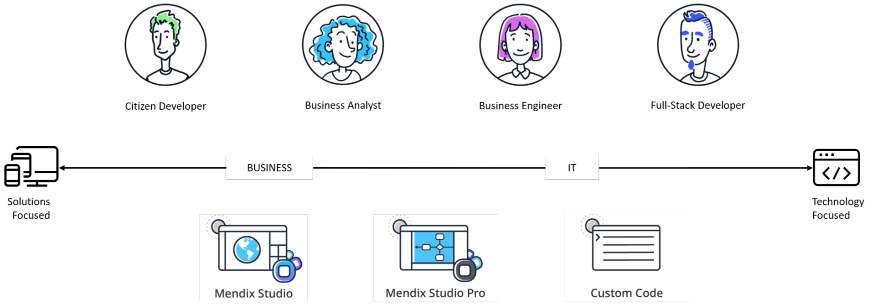
Our Mendix application

- Low-code development platform
- UML-like Diagrams → Code/Data structures
- Visual Sketches → User Interface

Our Mendix application

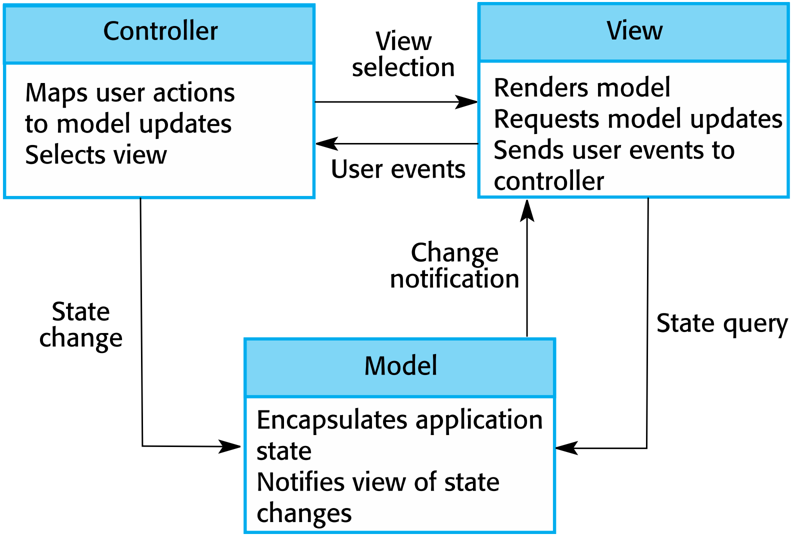
- **Software Development Methodology:** Agile (with SCRUM project management)
- **SW Architecture:** MVC
- **HW Architecture:** Client-Server
- **Programming Language:** Graphical
  - Domain model ≡ Data structures → UML Class-like diagrams
  - Microflows ≡ Code → UML Activity-like diagrams
  - Pages ≡ User Interface → Graphical sketches

Mendix - Studio vs Studio Pro

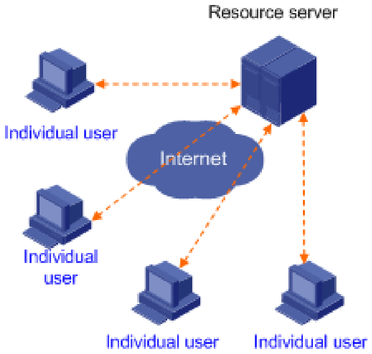


Source: Mendix Academy - Rapid Developer Learning Path

Mendix - MVC


Controller
Maps user actions to model updates
Selects view
View selection
User events
View
Renders model
Requests model updates
Sends user events to controller
Change notification
State change
State query
Model
Encapsulates application state
Notifies view of state changes

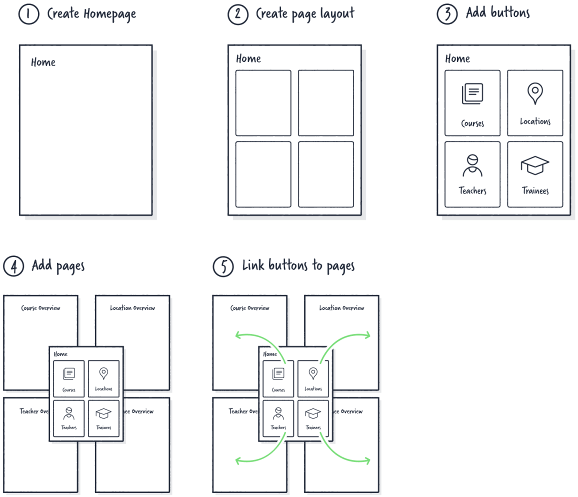
Mendix - Client-Server



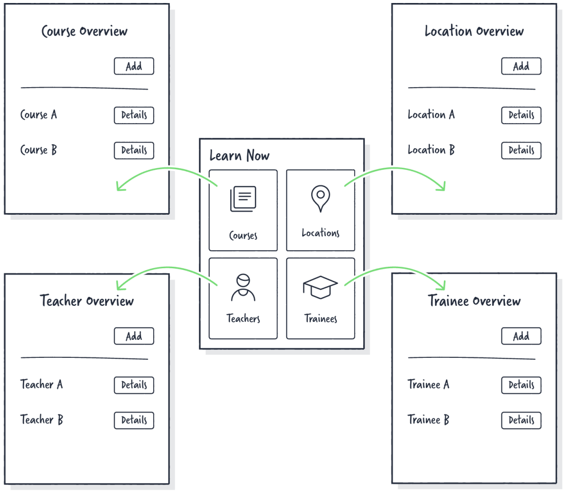
Traditional traffic model

- Two approaches:
  - **Web-app**
  - **Mobile app**
- **Web-app** is a client-server application in which data is stored server side (in a database)
- **Mobile app** is a standalone application in which data is stored locally (in a database)
- In the summative assignment, **web-app** only.

Mendix - UI Planning

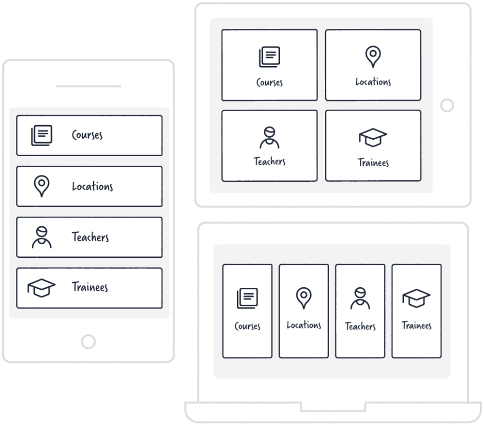


Mendix - Wireframes



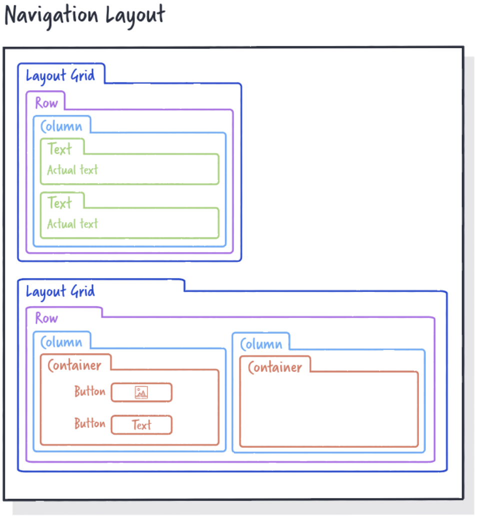
Source: Mendix Academy - Rapid Developer Learning Path

Mendix - Wireframes for different purposes



Source: Mendix Academy - Rapid Developer Learning Path

Mendix - Navigation Layout



Source: Mendix Academy - Rapid Developer Learning Path

Mendix - Development

Now it is your turn to develop...

Mendix - Development - Exercise 1

Register/Login to Mendix and explore the platform. Create a new app from a blank template and explore the Mendix platform.
Can you describe what the role of the following elements are?

- Project Buzz
- Stories
- Team
- Feedback
- Documents
- Team Server
- General Settings

Mendix - Development - Exercise 2

In the app created in **Exercise 1**, create a form to support the loading of the data from **User Story 1**:
Businesses can register on the platform using a form asking at least for the name of their business, their address, their VAT number and an official telephone number.
In order to do so, you need to create a new page, and create the corresponding domain model, through the dedicated Mendix features. Finally, you can proceed to link the form with the domain model.

Mendix - Development - Exercise 3

Adapt the application so that the form created in **Exercise 2**, can be accessible from the homepage.
In order to do so, you need to:

1. Add a button or a card widget
2. Link it to the previously created page.
3. Add the check mark on Create New Object to solve the consistency error.

Mendix - Development - Exercise 4

With the newly found knowledge of **Exercise 2** and **Exercise 3**, create a new page showing an overview of Business Data.
In order to do so, you need to:

1. Create a new page
2. Select a list template
3. Adapt it to display the relevant information.

Mendix - Development - Exercise 5

Finally, add a direct link to the page created in **Exercise 4** on the home page and adapt the navigation bar to have a quick access to all the created pages.