# The training will start at 9:00

mx mendix

# Intermediate Developer

mendix

# Introduce yourself

Where do you work?

How long have you worked with Mendix?


What do you hope to learn in this course?

# Agenda

1 The SCRUM methodology applied

2 Laying out the basics

3 Branding

4 Adding requests

5 Approving requests

6 Notifications

7 Translation

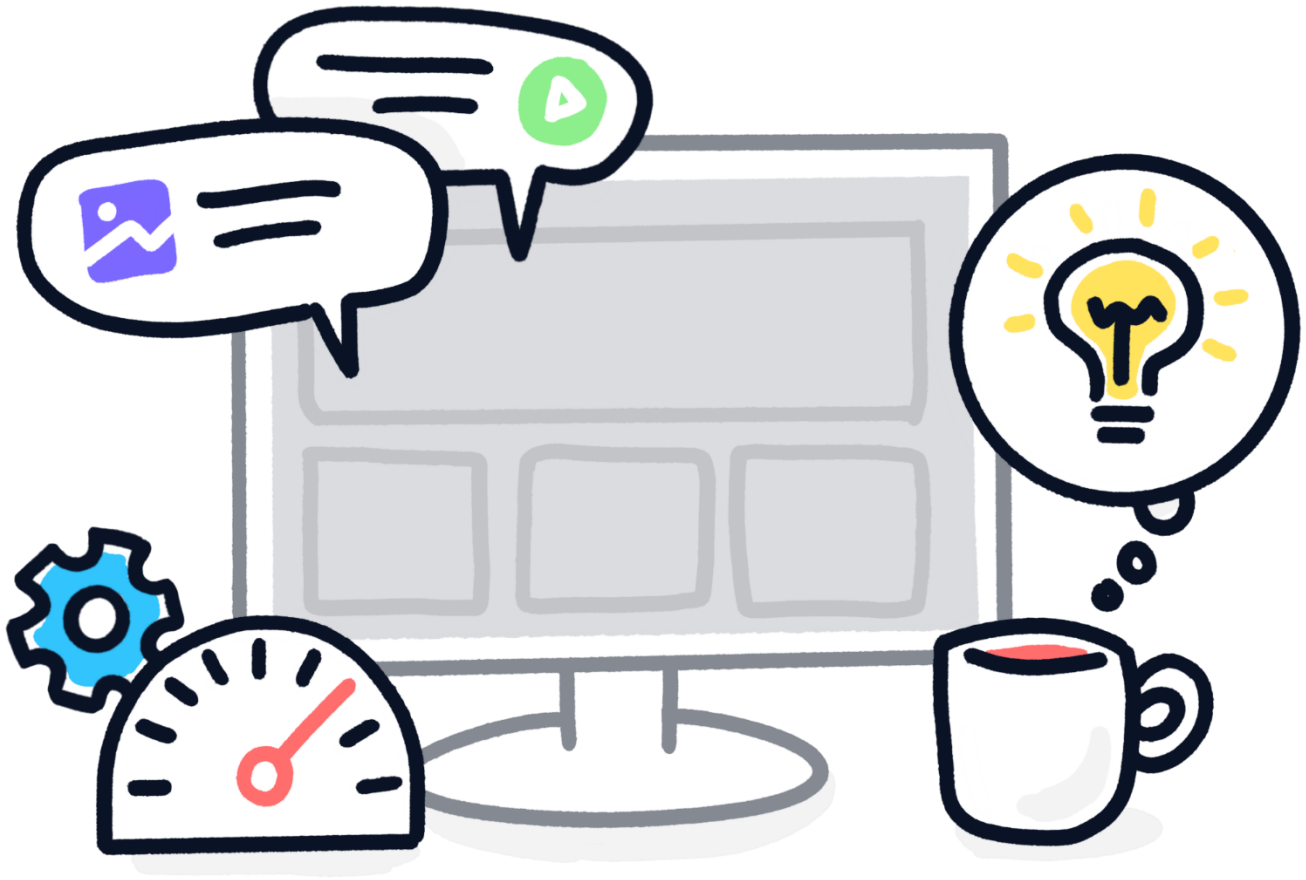
8 Conclusion

mx mendix

# The SCRUM methodology applied

# Learning goals

- What responsibilities there are in software development.
- Who is accountable for those responsibilities.
- Which events are used in the Scrum methodology.



# The Five values of Scrum

Focus

Courage

Openness

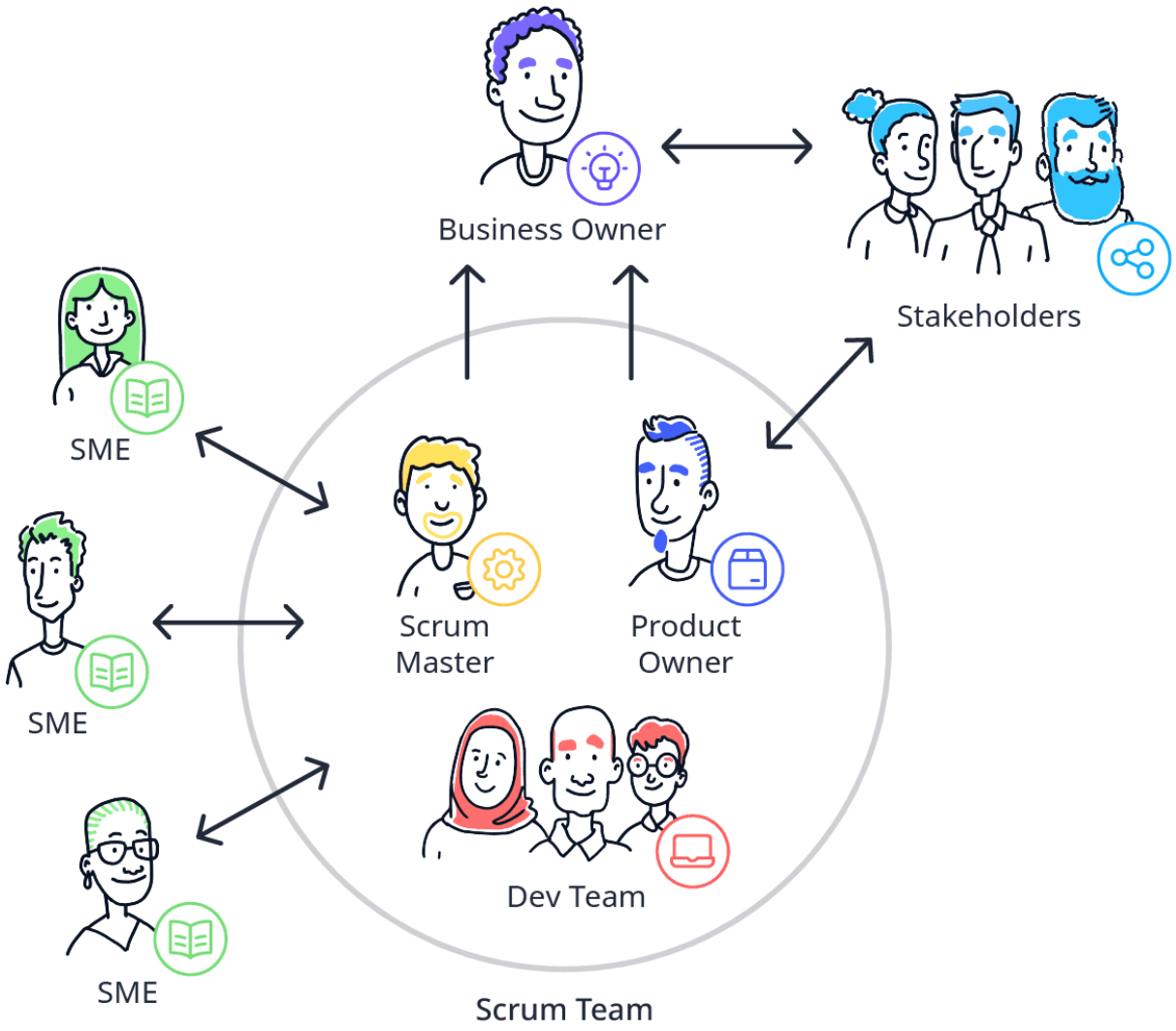
Commitment

Respect

# Team Size & Location

- Teams of 3
  - Big enough to build awesome features
  - Small enough for easy collaboration

- Recommended to work in single place (scrum room)
- Direct and quick communication is key
- Offshoring more complicated



# Sprint 0

- Preparation period of one or two weeks
- Get familiar with:
  - overall picture
  - organization
  - stakeholders & SMEs
  - availability of team and stakeholders/SMEs
- Have official kick-off meeting
- Product Backlog Refinement meeting

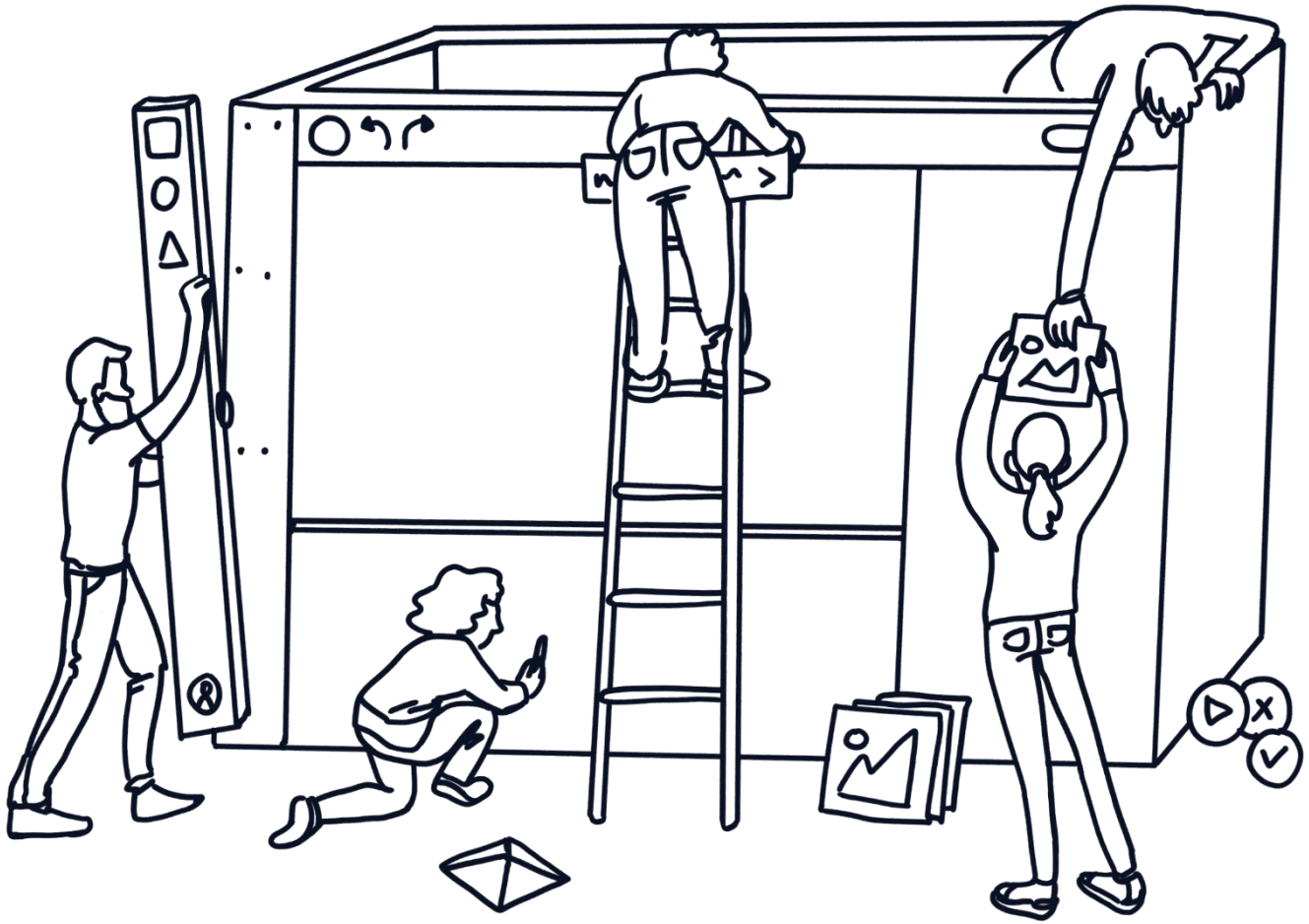
# Our Use Case: North Sea Shipbuilding

*Andrea is the CEO of North Sea Shipbuilding. She wants her employees to have an easy way of requesting time off. It's also important that managers can approve or reject those requests, and that they can see who has time off when.*

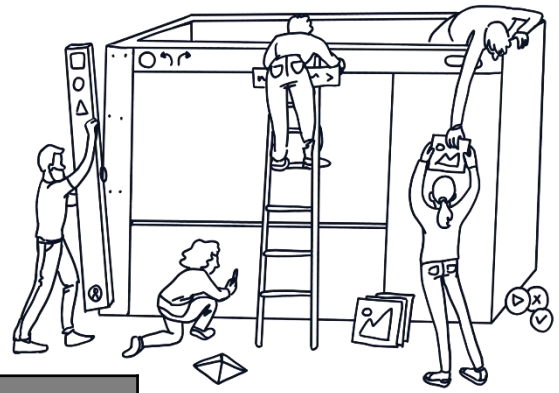
# SCRUM events

Sprints contain and consist of:

- Sprint planning
- Daily Scrums
- Development work
- Sprint review
- Sprint retrospective.

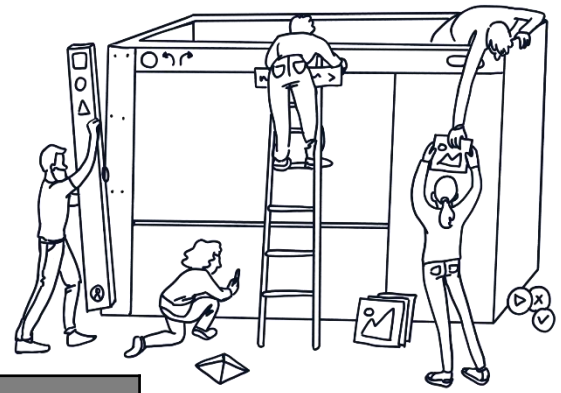


# SCRUM events



| Event | Inspection | Adaptation |
|---|---|---|
| Sprint Planning | • Product Backlog<br>• Increment<br>• Actions of the last Sprint<br>• Definition of Done | • Sprint Goal<br>• Forecast<br>• Sprint Backlog |
| Daily Scrum | • Progress towards Sprint Goal | • Sprint Planning<br>• Sprint Backlog |
| Sprint Review | • Increment<br>• Sprint<br>• Product Backlog | • Product Backlog<br>• Change Planning<br>• Release |
| Sprint Retrospective | • Work Method/ Process<br>• Engineers practitioners<br>• Definition of Done | • Actionable committed improvements |

# SCRUM events

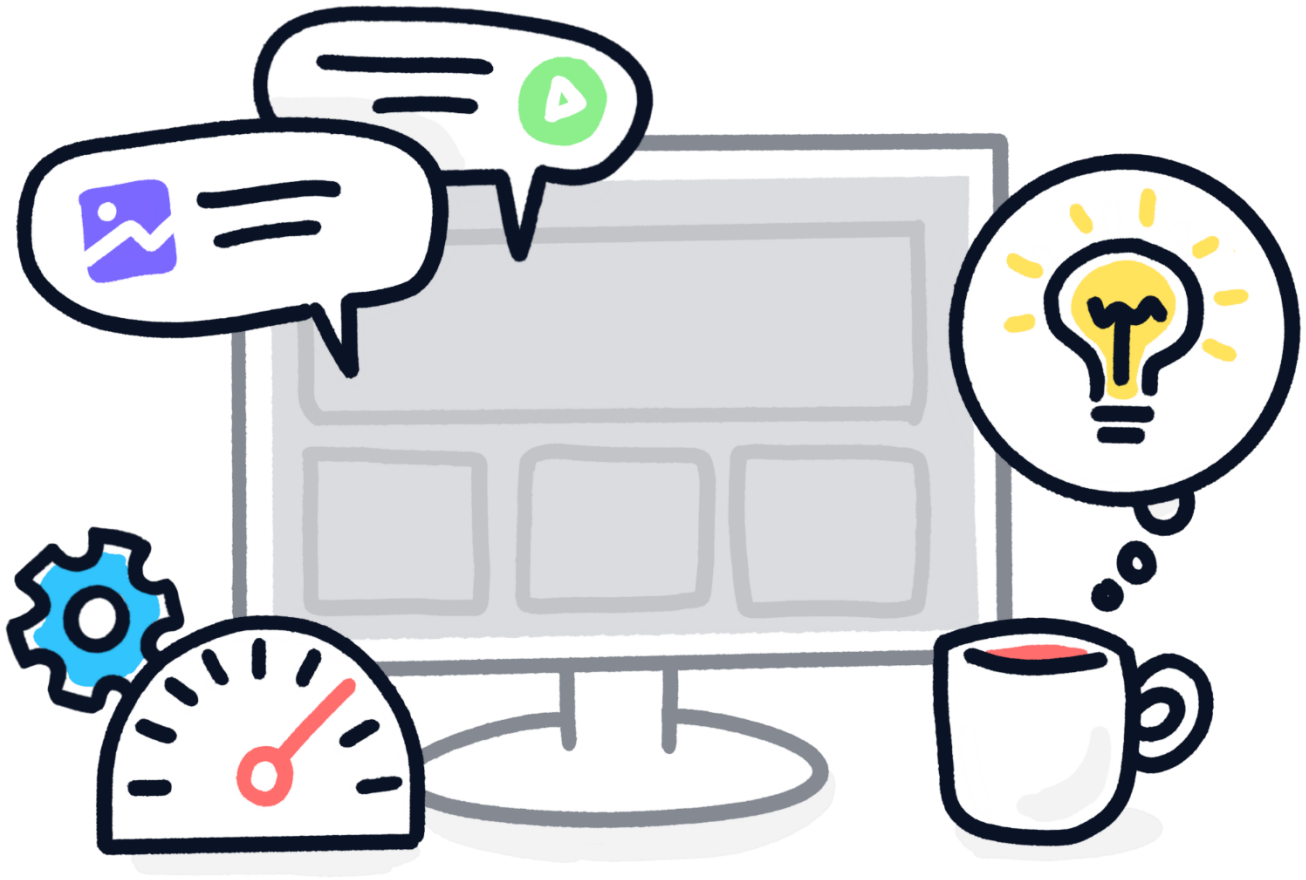


| Event | 4 Weeks | 3 Weeks | 2 Weeks | 1 Week |
|---|---|---|---|---|
| Sprint Planning | 8 hrs | 6 hrs | 4 hrs | 2 hrs |
| Daily Scrum | 15 min daily | 15 min daily | 15 min daily | 15 min daily |
| Sprint Review | 4 hrs | 3 hrs | 2 hrs | 1 hr |
| Sprint Retrospective | 3 hrs | 2.25 hrs | 1.5 hrs | 0.75 hr |

# Laying out the basics

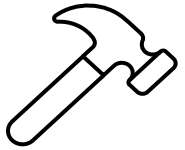
# Learning goals

- Import modules from the AppStore
- Extending AppStore modules
- Layout with Flexboxing
- XPath basics



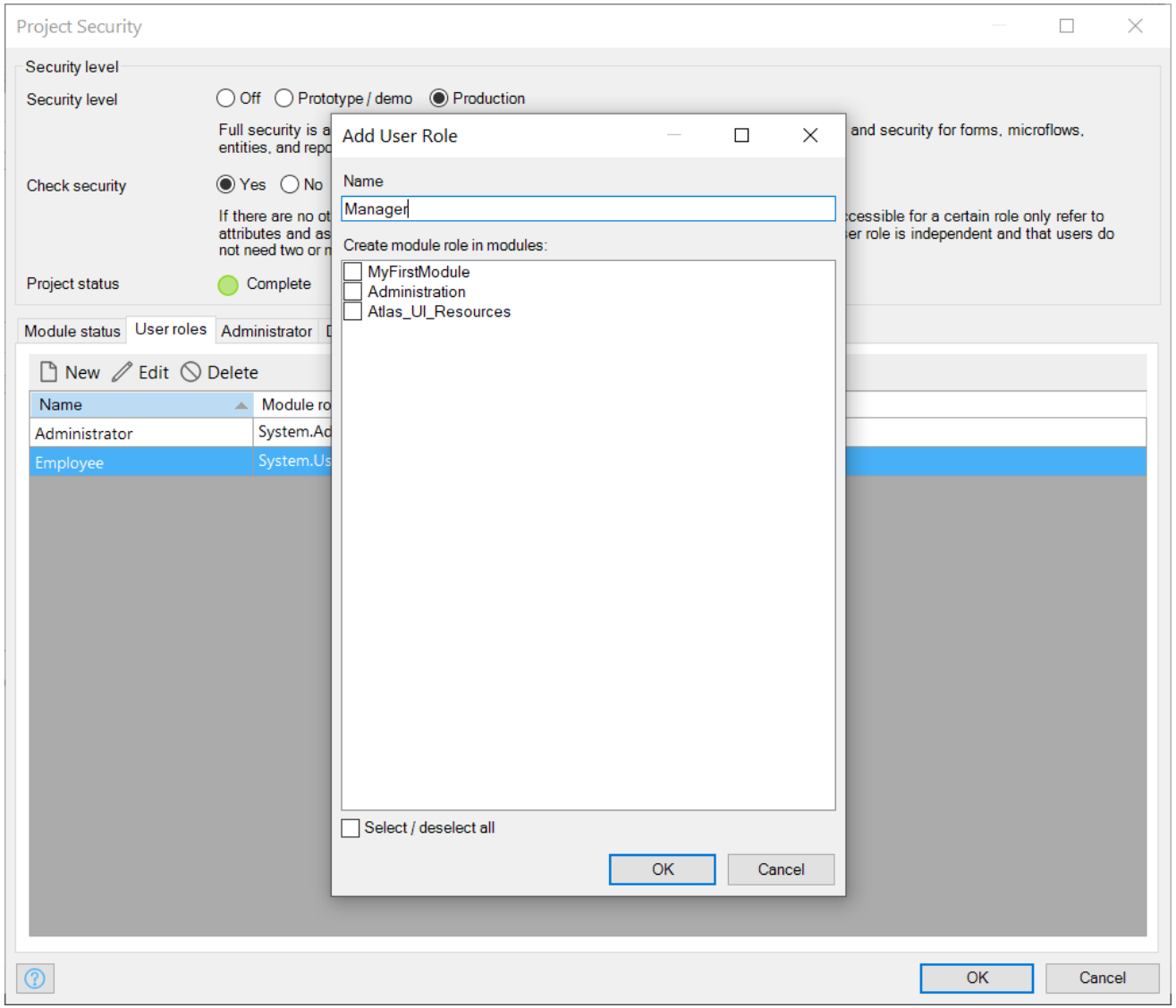
# User story

*As an authorized user I want to be able to add a profile picture so people can easily identify me*



# Create the app!

- Name: **Vacation Tracking**
- Theme: **Blank app**
- Security level: **Production**
- User roles: **Administrator, Employee, Manager**
- Make sure the **Manager** role is a user in the Administration module



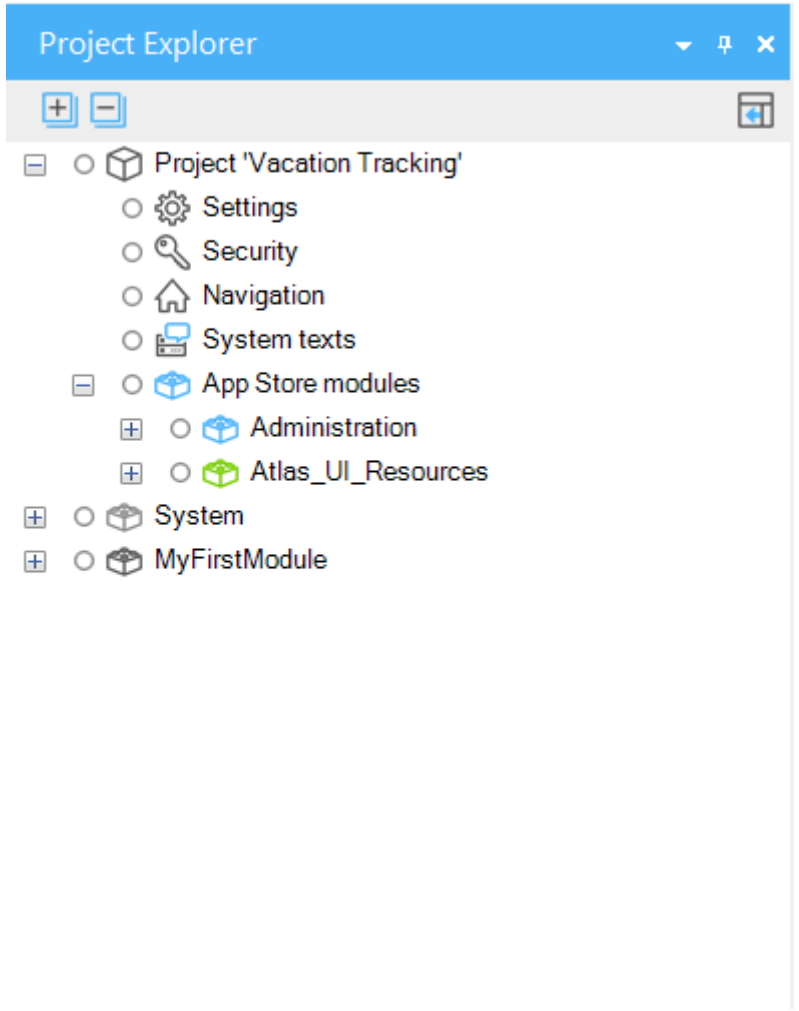

mx mendix

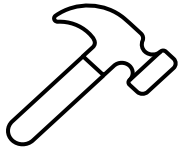
# Appstore modules

- Two modules are sometimes changed
  - Administration
  - Atlas_UI_Resources
- All other modules are always extended

Use extension because:

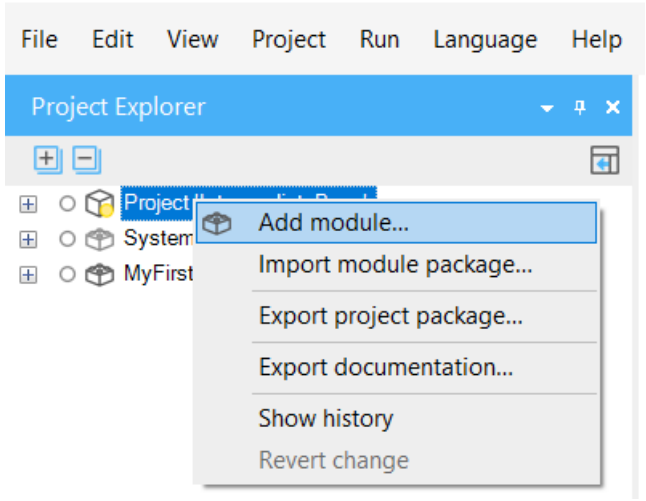
- Any changes are overwritten when you update
- Preferred way to add functionality is extending





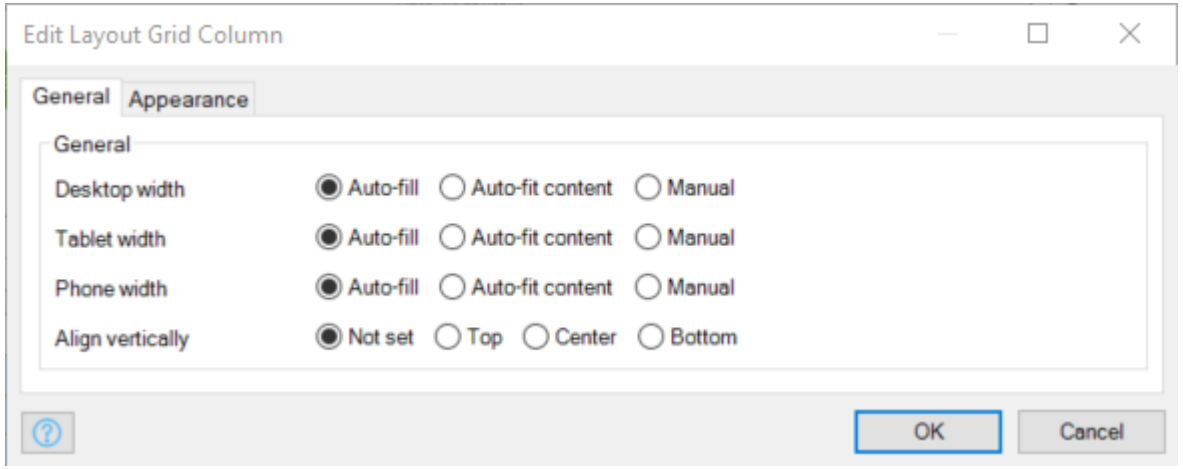
# Create new GeneralExtentions module

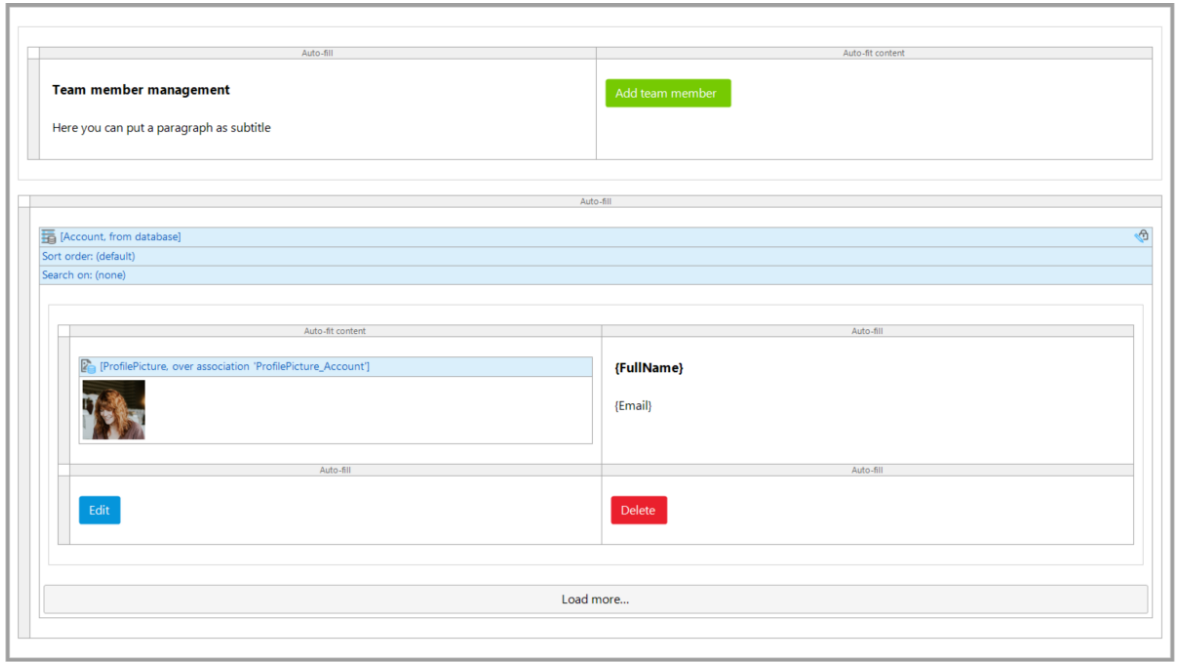
- Right-click on the Project node
- Select **Add Module…** to create a new module
- Give it the name **GeneralExtentions**
- Add two module roles to your **GeneralExtentions**
  - Administrator
  - User
- Link the new module roles to the user roles in project security



# Layout Grid

- Layout grid is based on Flexboxing
- Supports unlimited number of columns
- Three options can be selected
  - Auto-fill: Divide available space over all auto-fill columns
  - Auto-fit content: Take as much space as content requires
  - Manual: revert to old layout grid

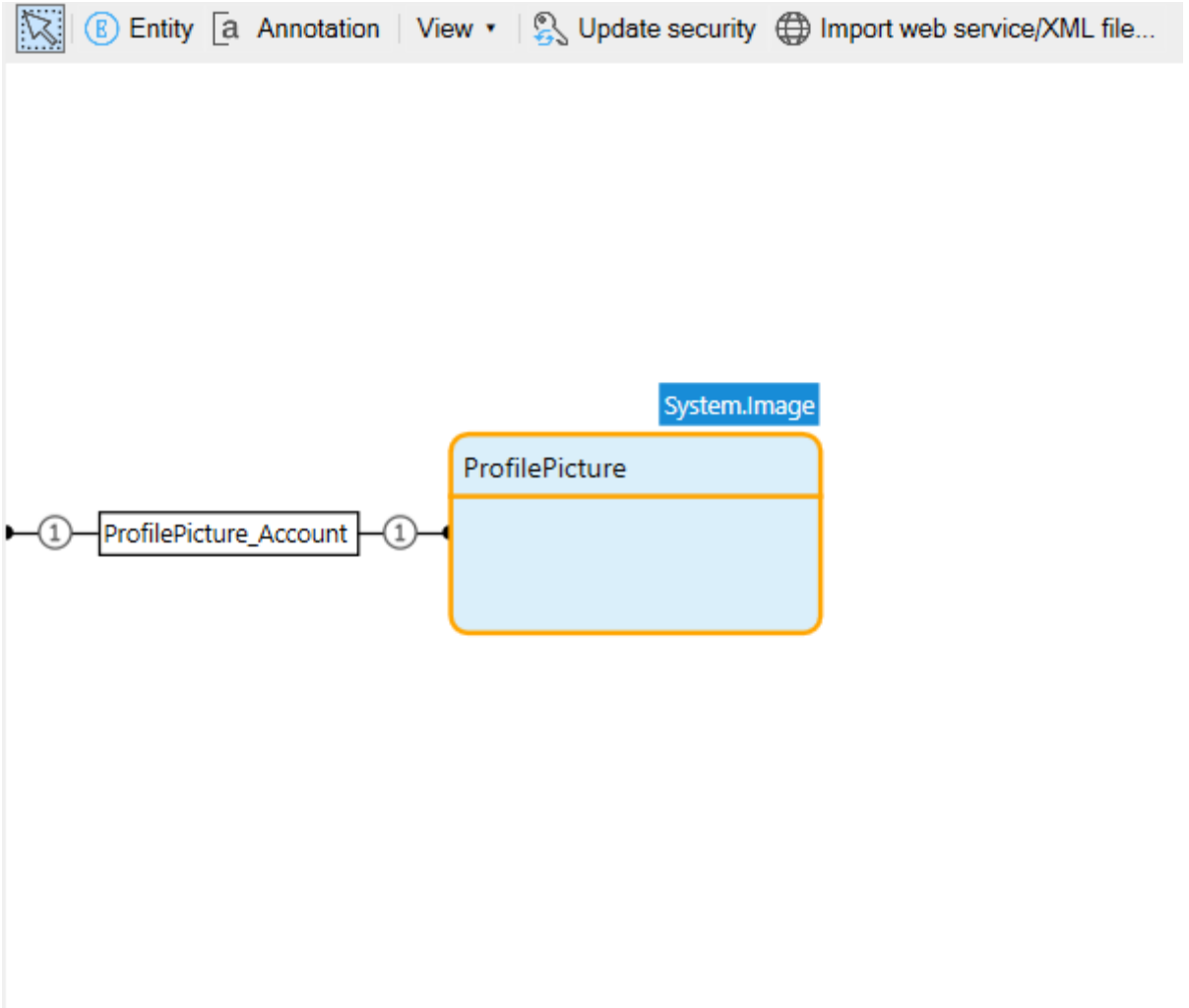


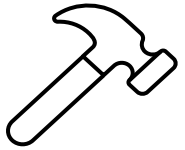

mx mendix

# Cross module associations

- System & Administration module
- MyFirstModule
- Big app = multiple modules?



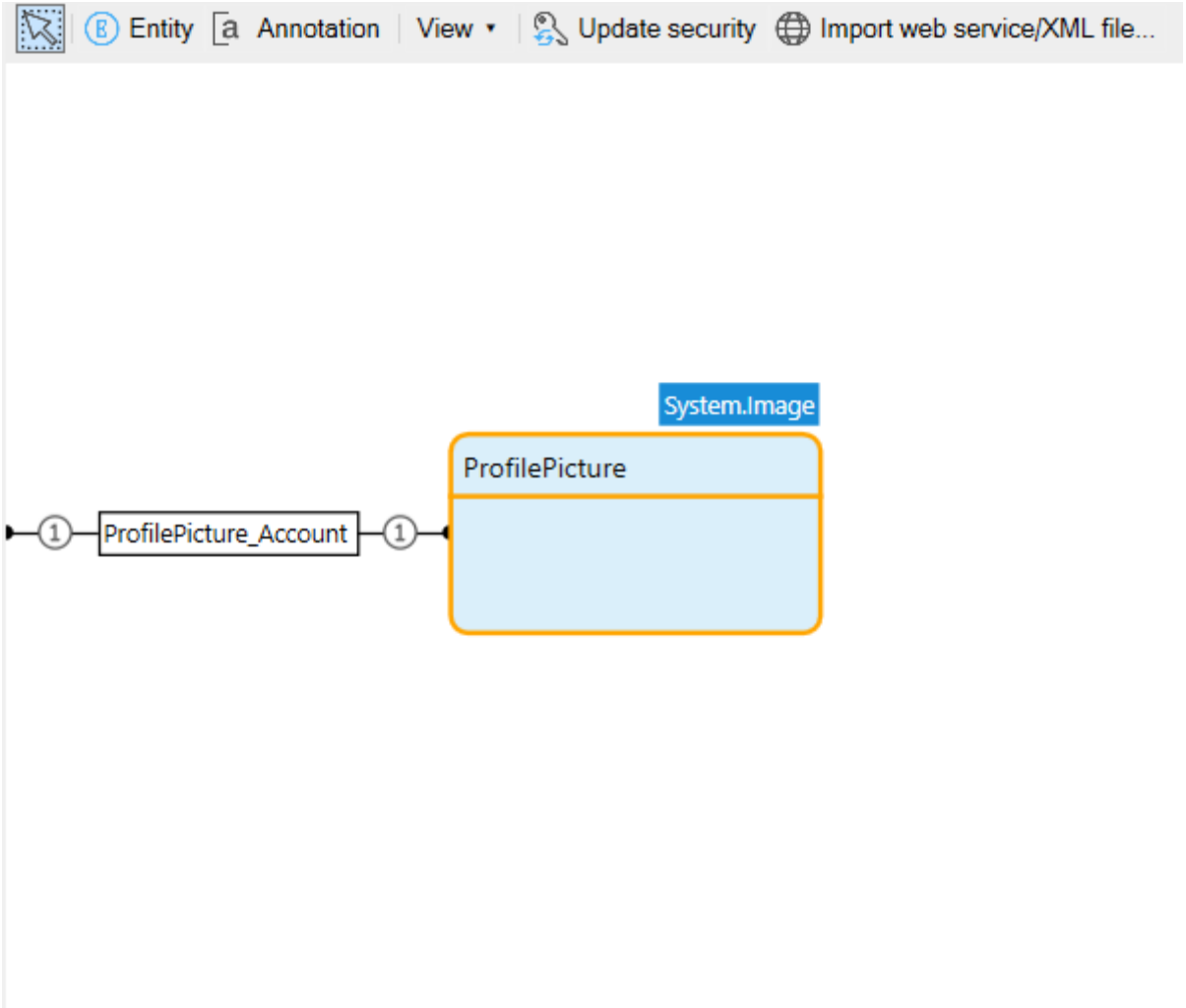

mx mendix

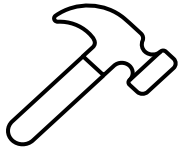


# Managing Accounts

- Add an entity to the **GeneralExtentions** module called **ProfilePicture**
  - 1-1 association with **Account** entity
  - Generalizes from **Image** entity

- Fix security errors

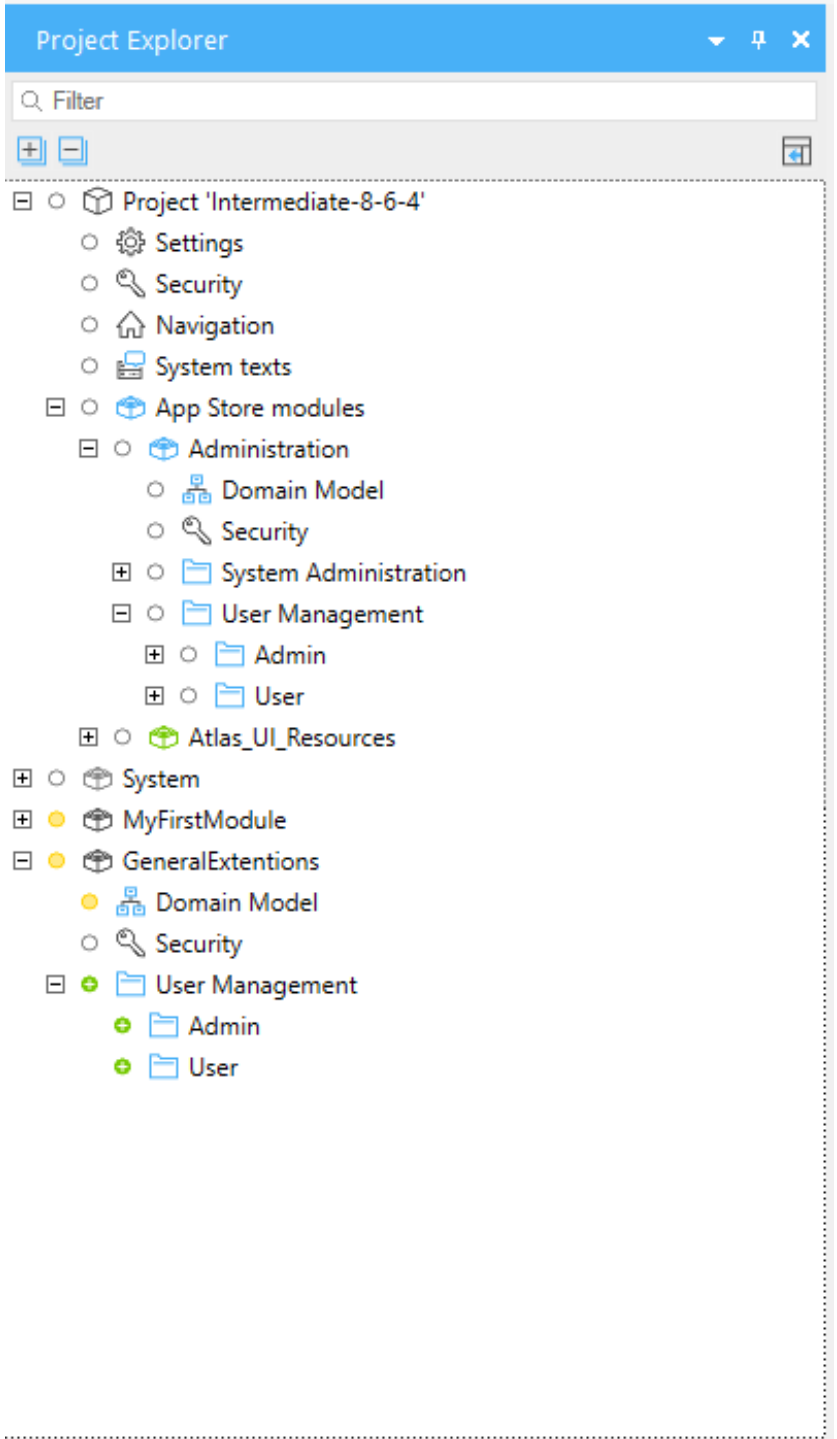




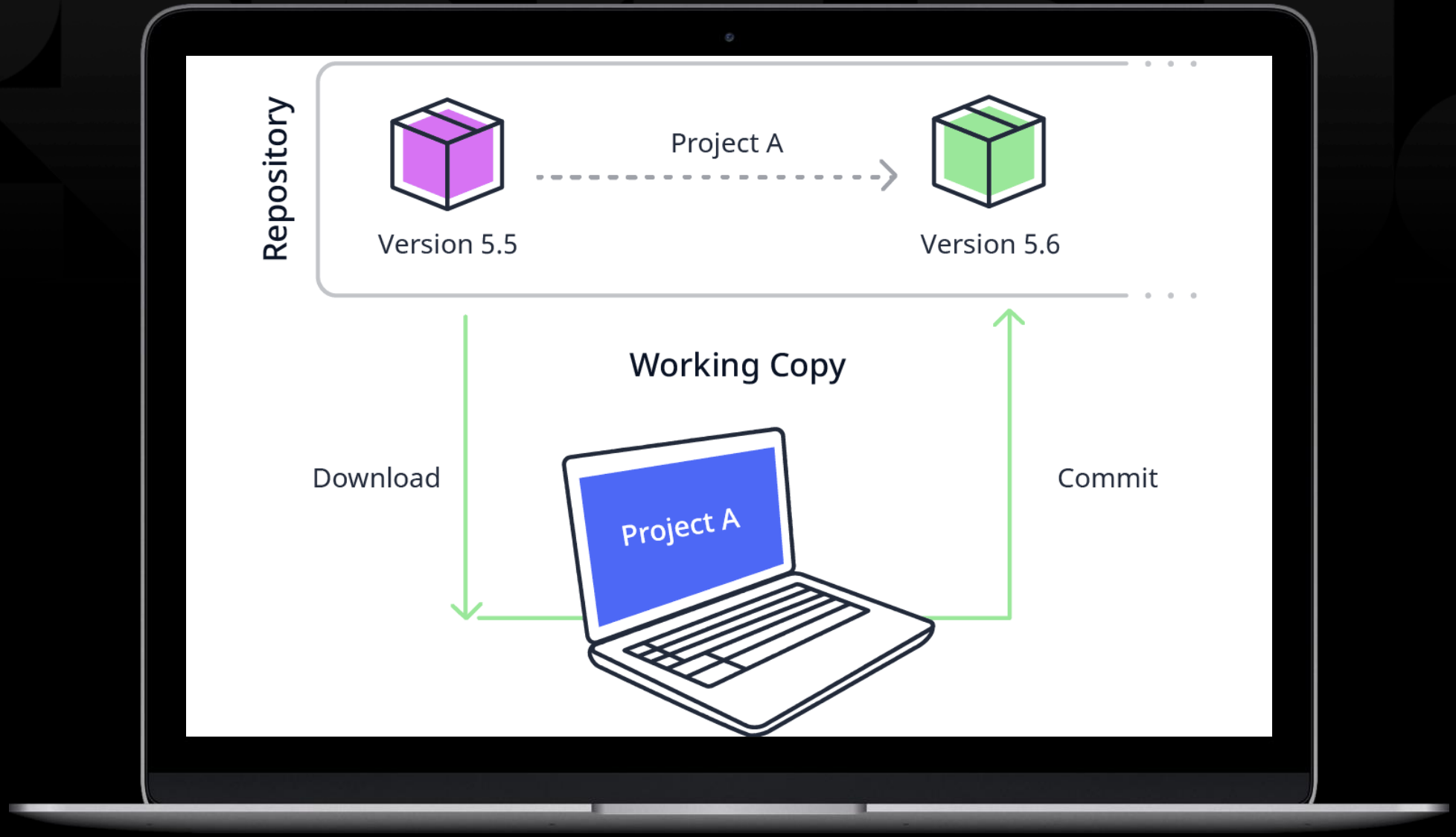
# Organize your project

It is important to keep your project organized so let's duplicate the **Administration** folder structure to get into the habit.

- Create a folder named **User Management** in your general extensions module.
- Create a folder named **Admin** in your **User Management** folder.
- Create a folder named **User** in your **User Management** folder.
- If a folder gets too full, separate Microflows and Pages into individual folders within each

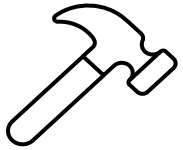


# Commit your work



# User story

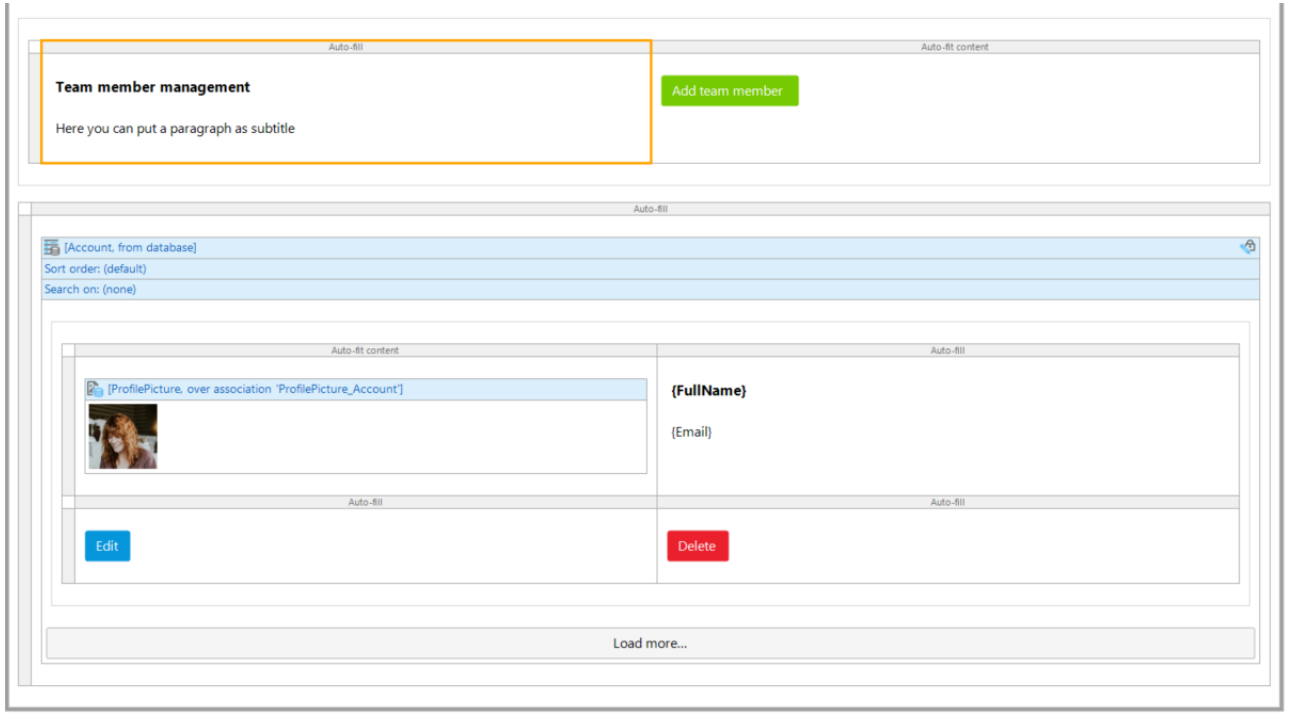
*As an administrator I want to manage accounts using a more modern interface*



# Managing Accounts

- Build new Administrator functionality in the **GeneralExtentions** module
  - New **Account_Overview** page:
    - Layout: **Atlas_Topbar**
    - Template: **List Blocks**
  - Match the design
  - Extend existing New, Edit and Delete buttons. Choose between microflow and default behavior

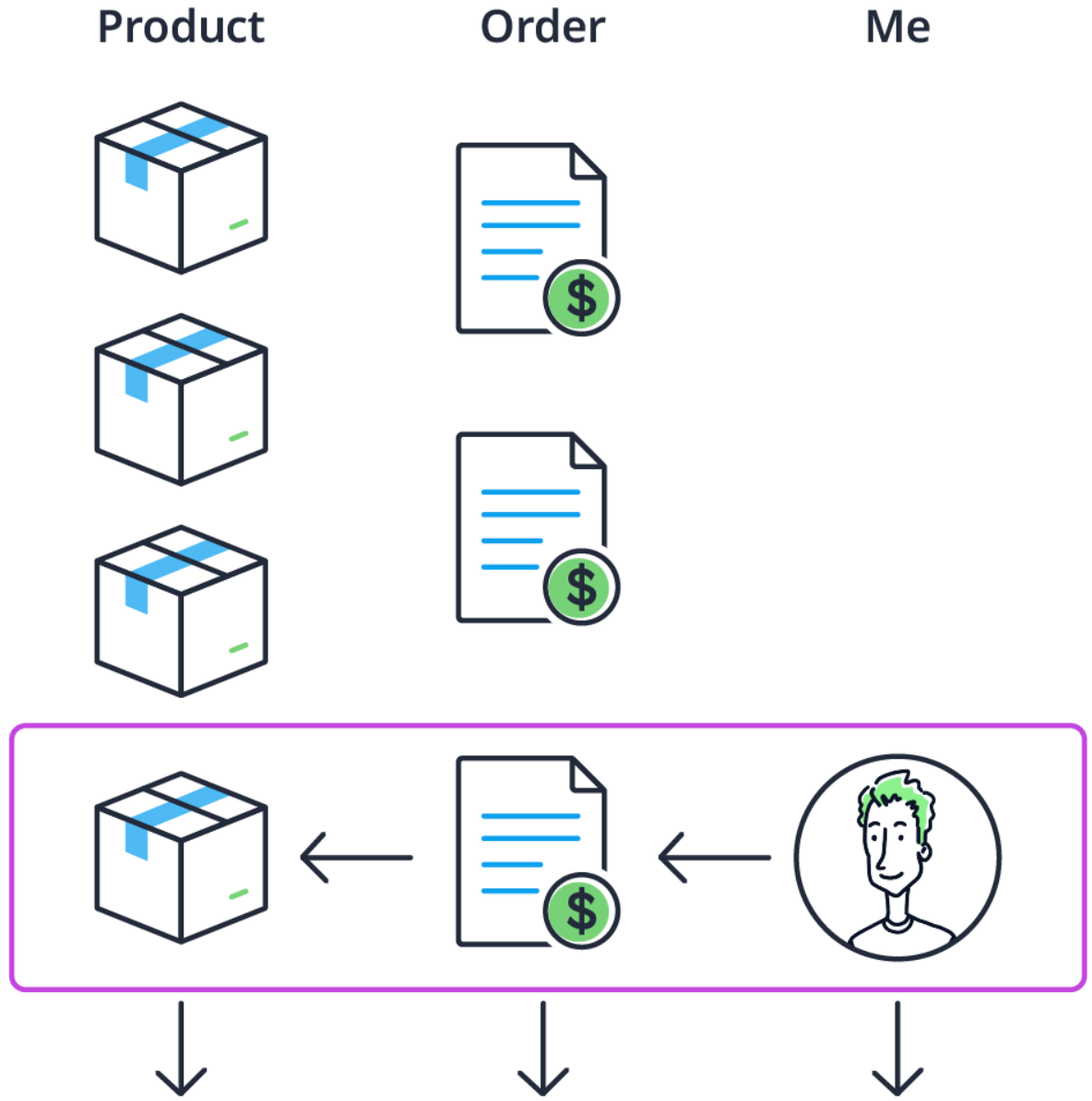
- Add the page to the navigation menu




mx mendix

# Constraining data with XPath

- Constrain data using entities, attributes, assocations & variables
- SQL or OQL are similar syntaxes
- Tokens (//, [], /, ())



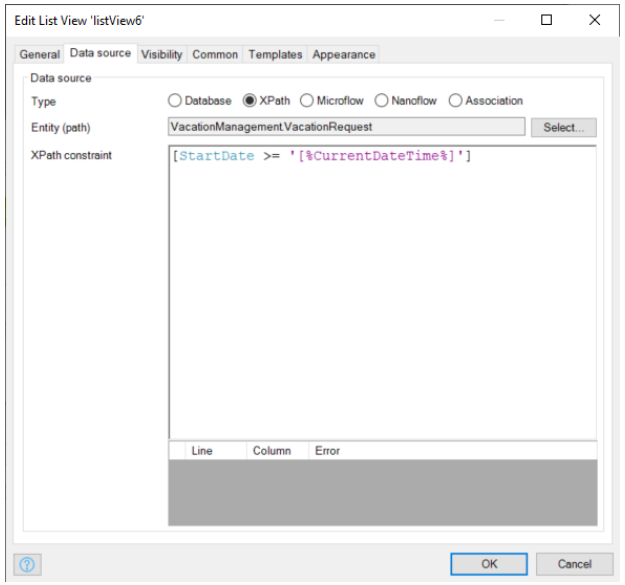
# Where do we use XPath?

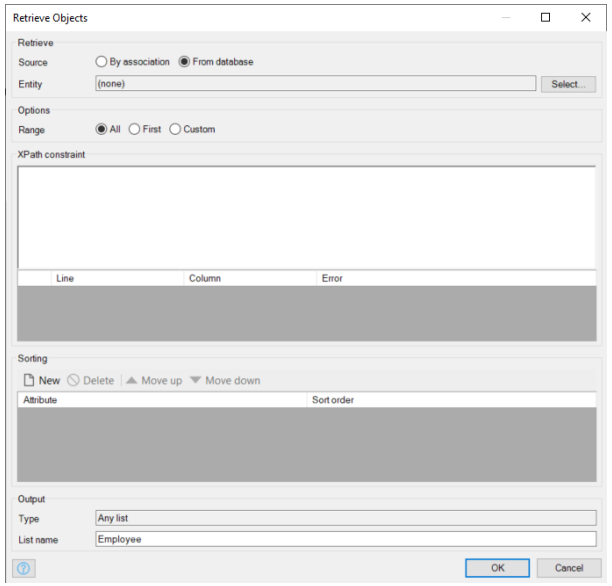
- XPaths on pages, to filter data or selectable objects
- XPaths in microflows, to define queries to the database
- XPaths in security

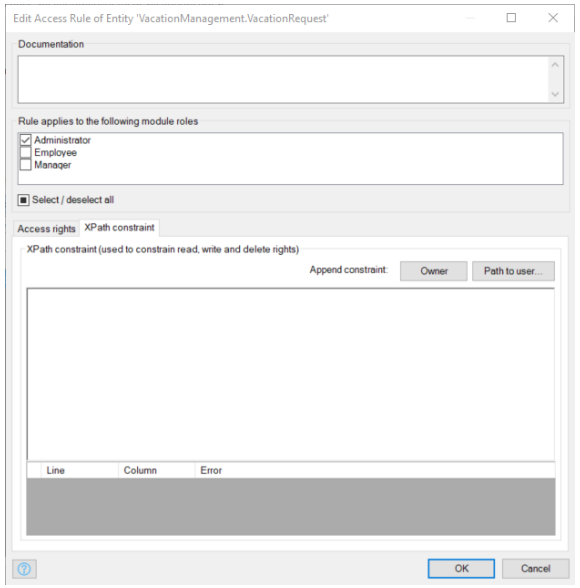
//Product/[Product_Order/Order/Order_Person = Me]

Added by Studio Pro

XPath expression you write in Studio Pro



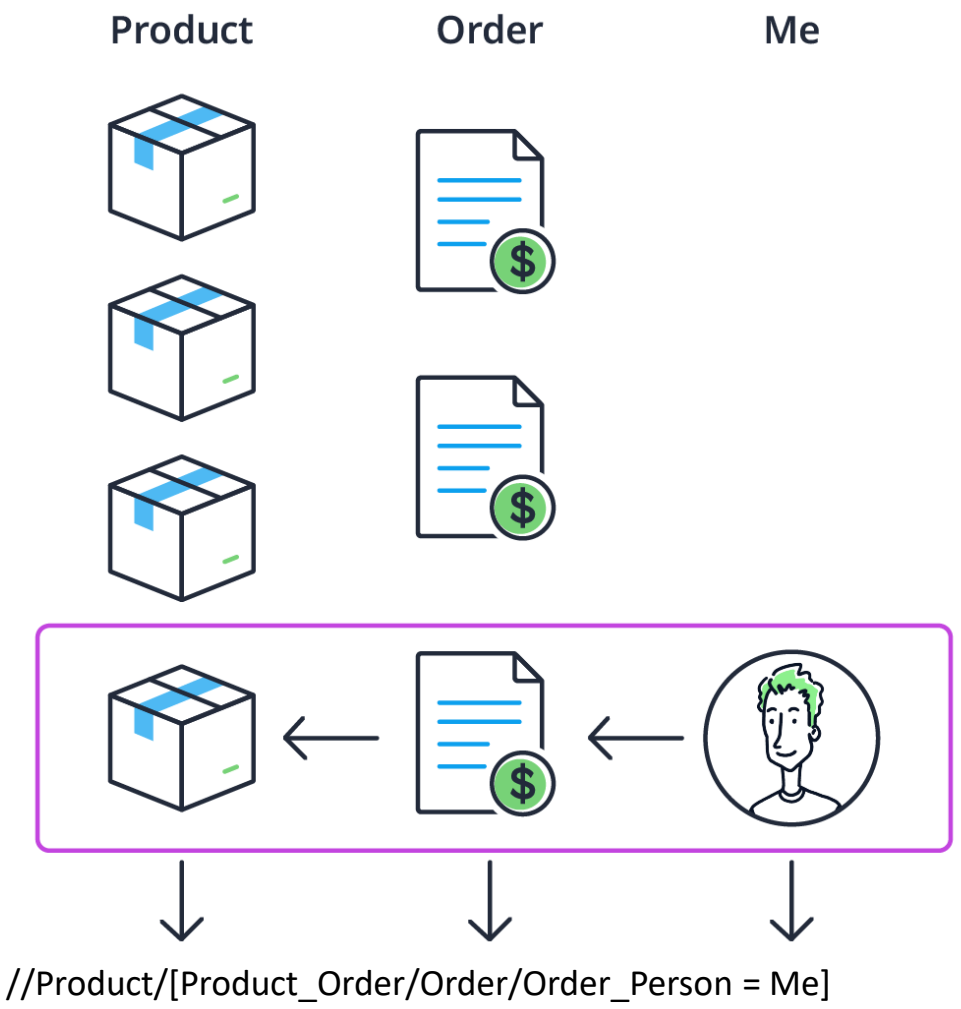

mx mendix

# How to structure an XPath

- Entities, attributes & associations

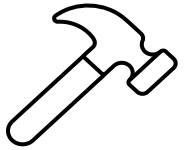
**How can Mendix help you?**

- Use CTRL + Space
- Know your domain model
- Well-organized domain model (apply naming conventions)
- Split your screen



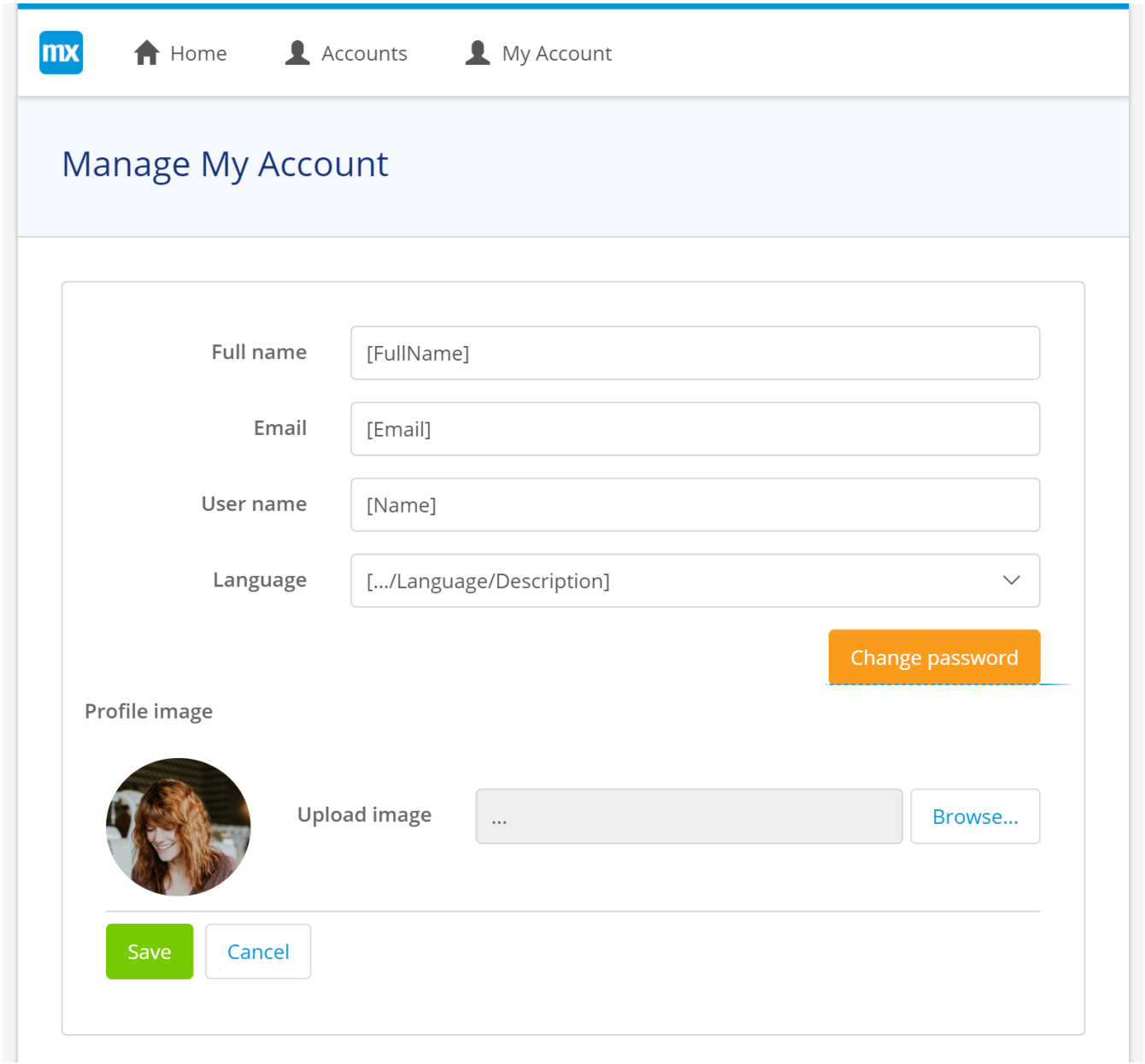
# Using simple constraints

- Constrain data using Enumeration values
- Constrain data using string comparison
- Constrain using Mendix System Variables
  - Current Object
  - Current User
  - Other system variables (ie current session, datetime related)

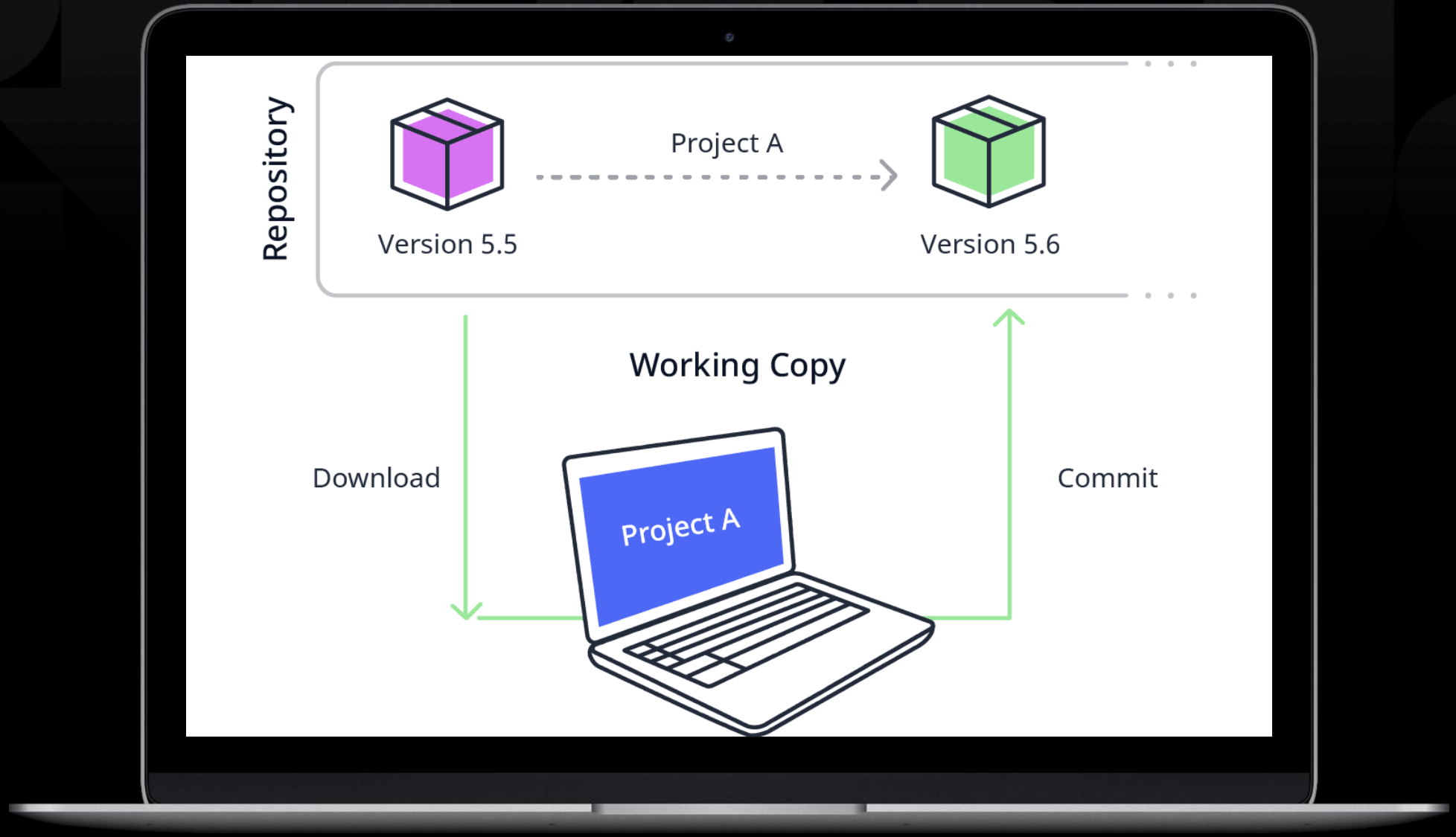


# Managing My Account

- Build Manage My Account functionality
  - Use or copy the ManageMyAccount microflow
  - Add to navigation
  - Create a new Manage My Account page
  - Make sure that only users can edit their profile picture and the rest can only view
- Add the page to the navigation menu
- Fix security errors
- Deploy & view



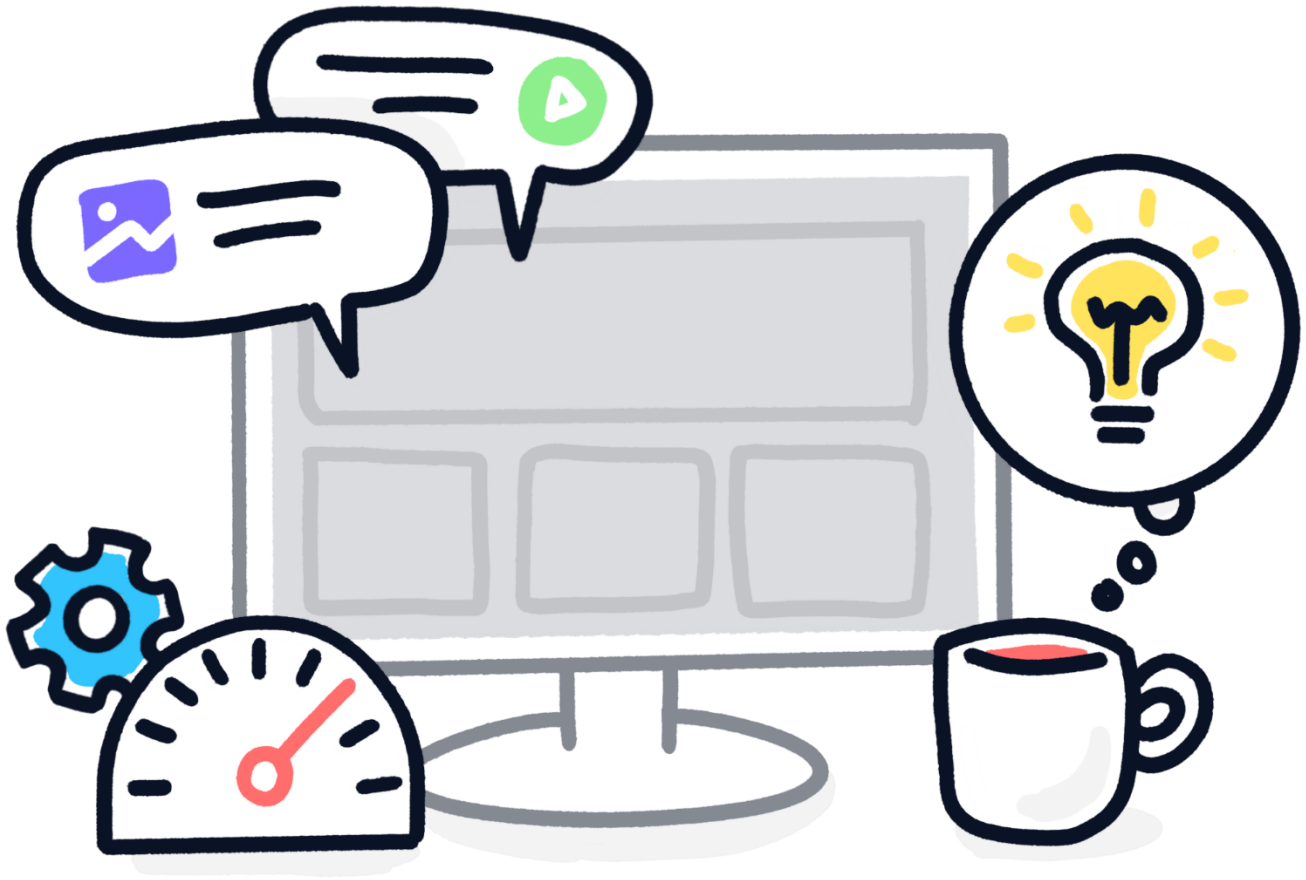
# Commit your work



# Branding

# Learning goals

- Stylesheet (Custom styling) – adding a custom theme to the app
- Custom Navigation Layout – including a sign out button
- Custom login page – where employees can also sign up for the app



# Custom Styling: 3 options

1

Use the **Theme Editor** in **Mendix Studio**

2

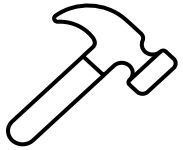
Use **design properties** to change styling in **Studio Pro**

3

Change **Sass variables**, to change existing styling or add new styling, with **Sass** or **CSS**

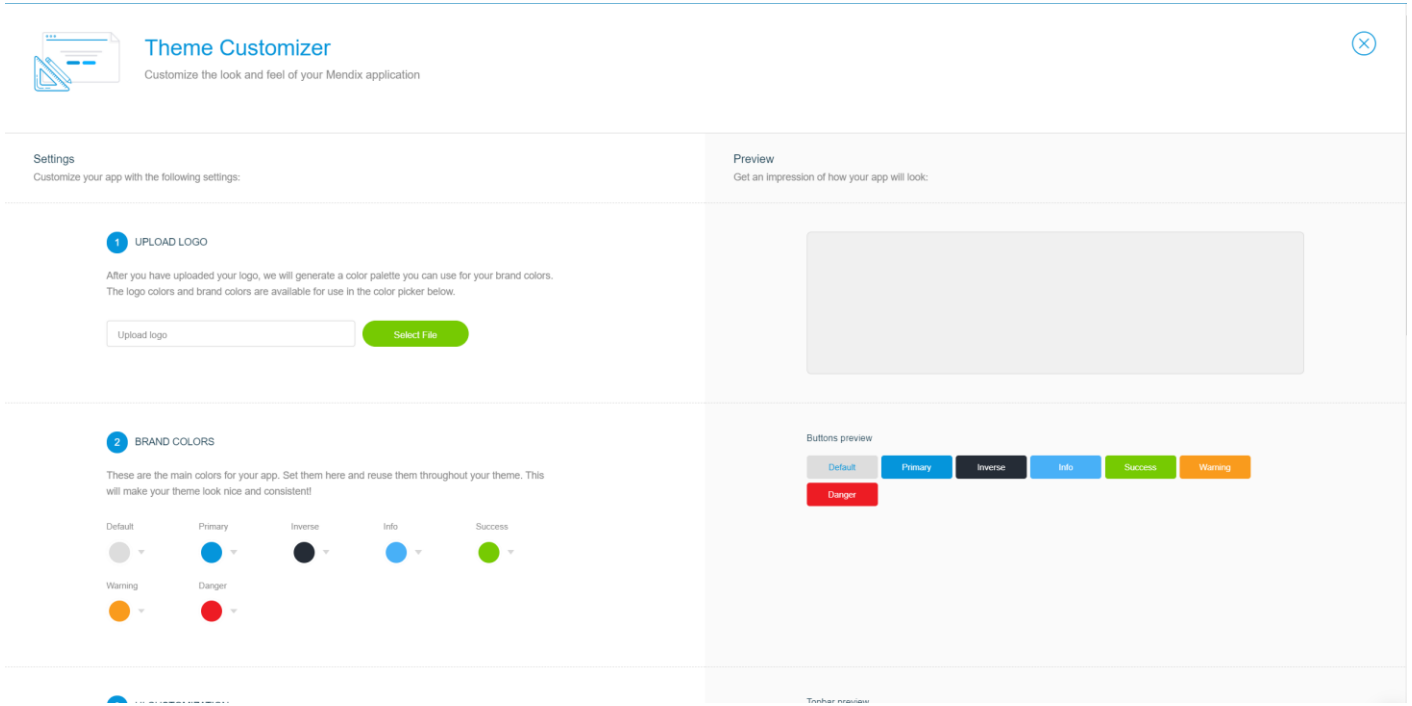
# User story

*As a company I want my logo in the app so all my employees know that this is our app*



# Adding custom styling

- Open the app in Mendix Studio
- Use Mendix Studio to upload the logo
- Go back to Mendix Studio Pro and Update

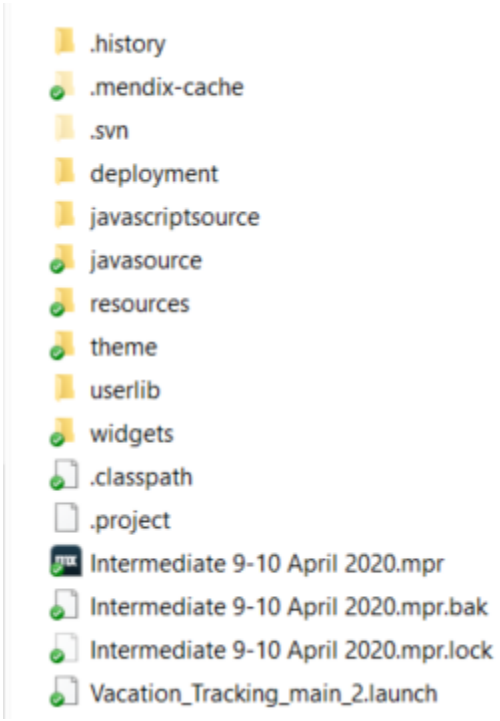



mx mendix

# The Project Directory

**Best Practices**

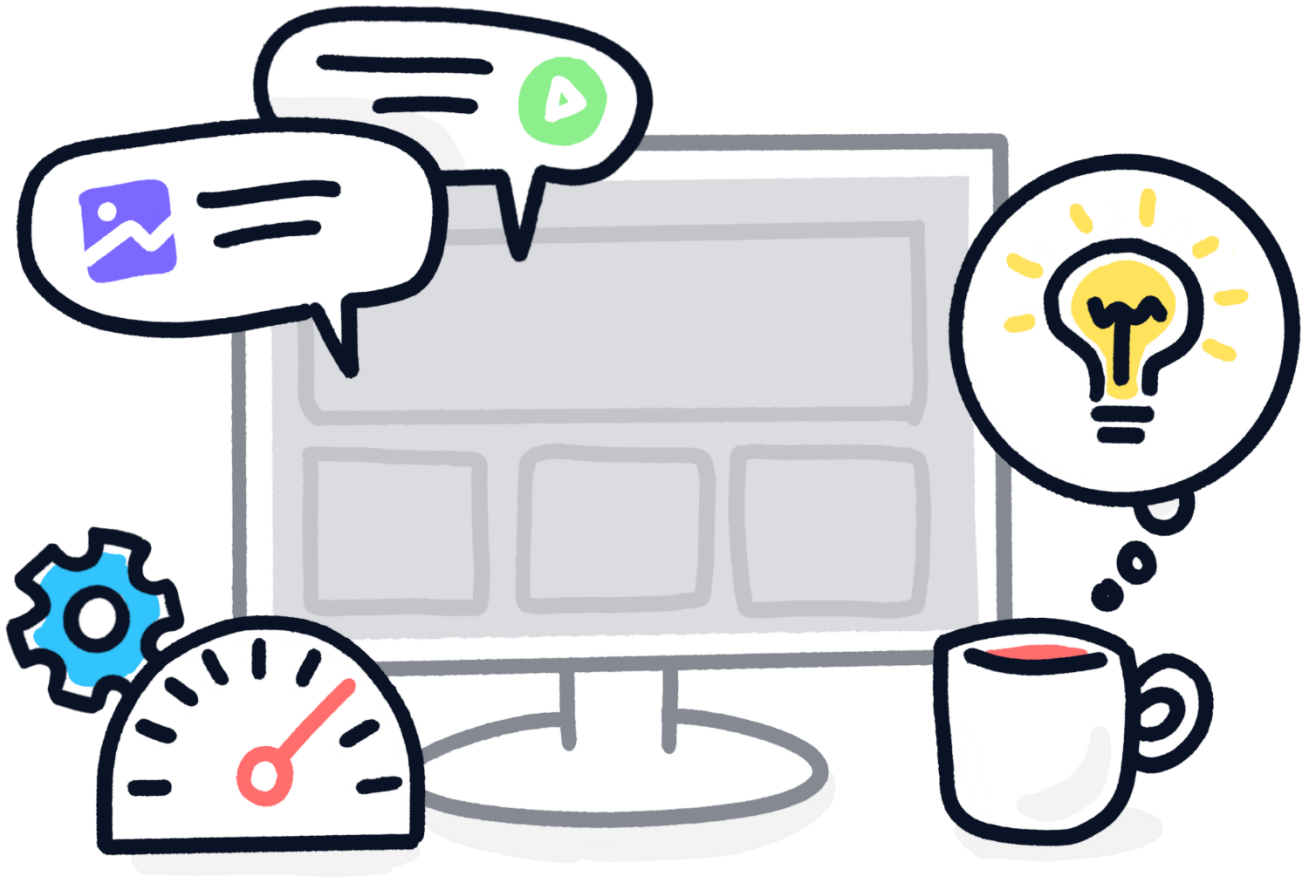
- Keep your Project Directory clean!
- Know what you import! (when importing external module, several files could be added to **userlib** and **resources** folder, good to know which ones those are)
- This is because you need to manually remove those files, when removing the external module again. Always do this, to prevent future conflicts!
- When using version management, on commit also check changes to project directory (to prevent accidental changes)



# Anonymous users

# Learning goals

- What is an anonymous user in Mendix
- What are the benefits and drawbacks
- How do you use an anonymous user in Mendix



# Introduction

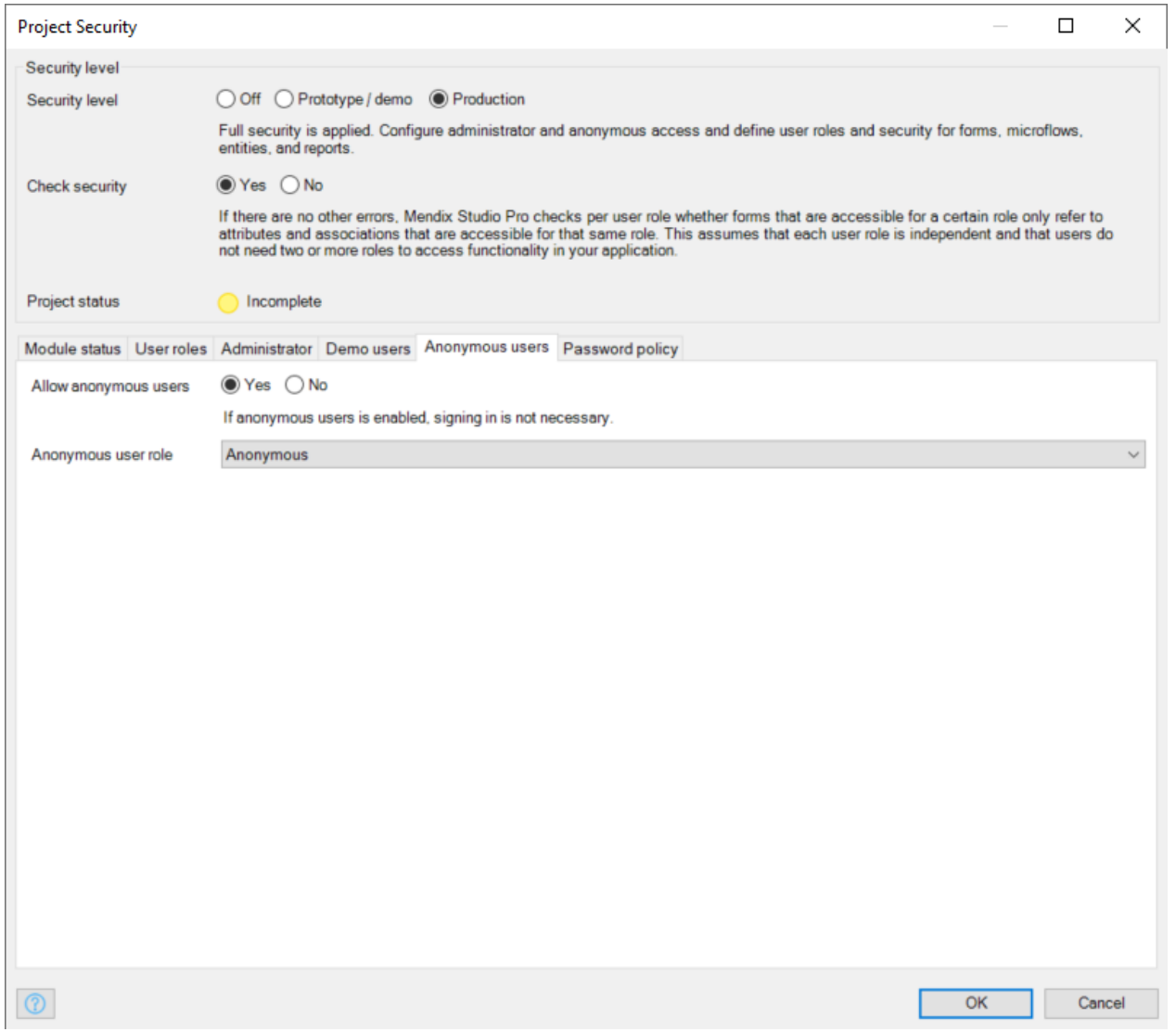
Anonymous users allow people without accounts to have access to a Mendix app

Benefits

- Allows for landing page with dynamic data
- Allows for a custom login page

Drawbacks

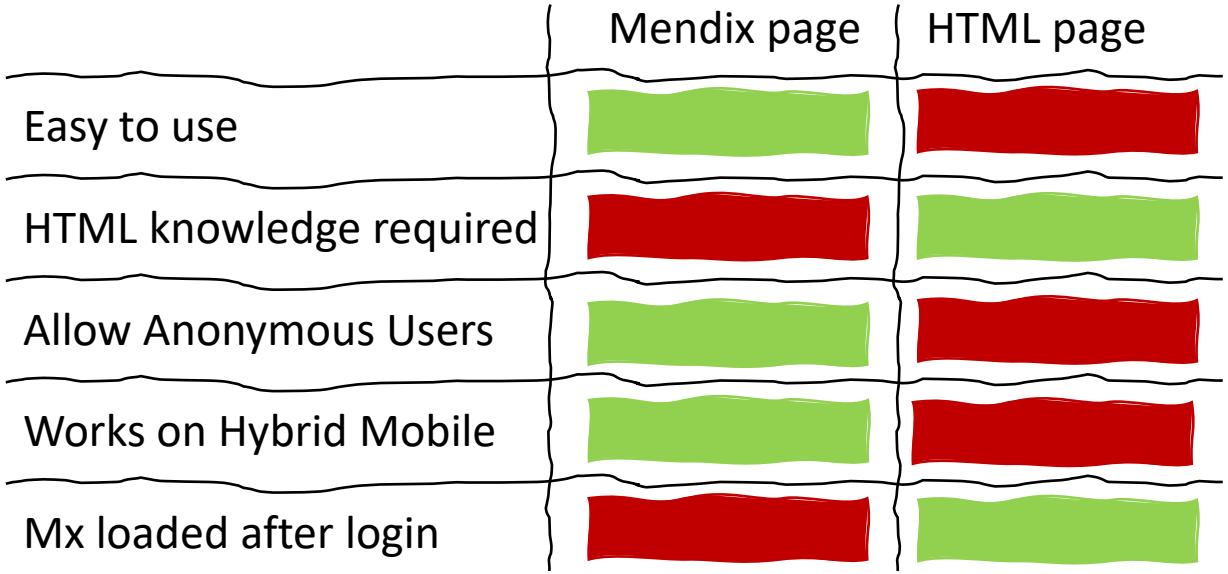
- Potential security risk




mendix

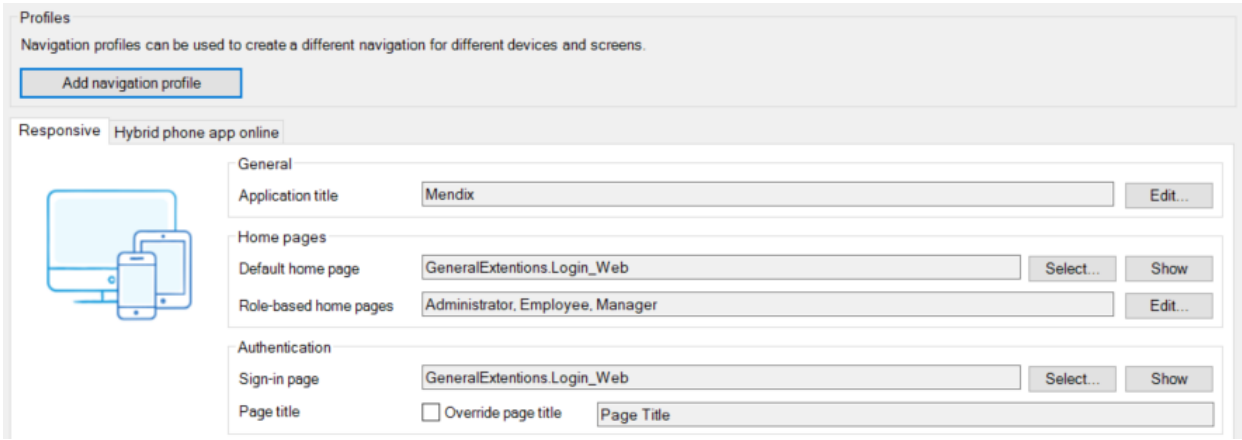
# Custom login page

- Anonymous users requires a custom login page
  - Can be a Mendix page
  - Can be a custom HTML page
- Custom Mendix login page allows for
  - Access to Mendix pages for anonymous users
- Custom HTML page allows for
  - Advance integration scenario’s (e.g. SSO)



# Custom login page in Mendix

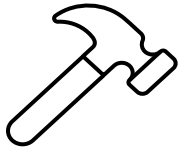
- Default home page is shown to all unauthenticated users
- All other user roles need a role-based homepage
- A sign-in page has to be defined




mx mendix

# User story

*As an administrator I want to allow users to sign up for the system on their own so I do not have to create all the user accounts myself*



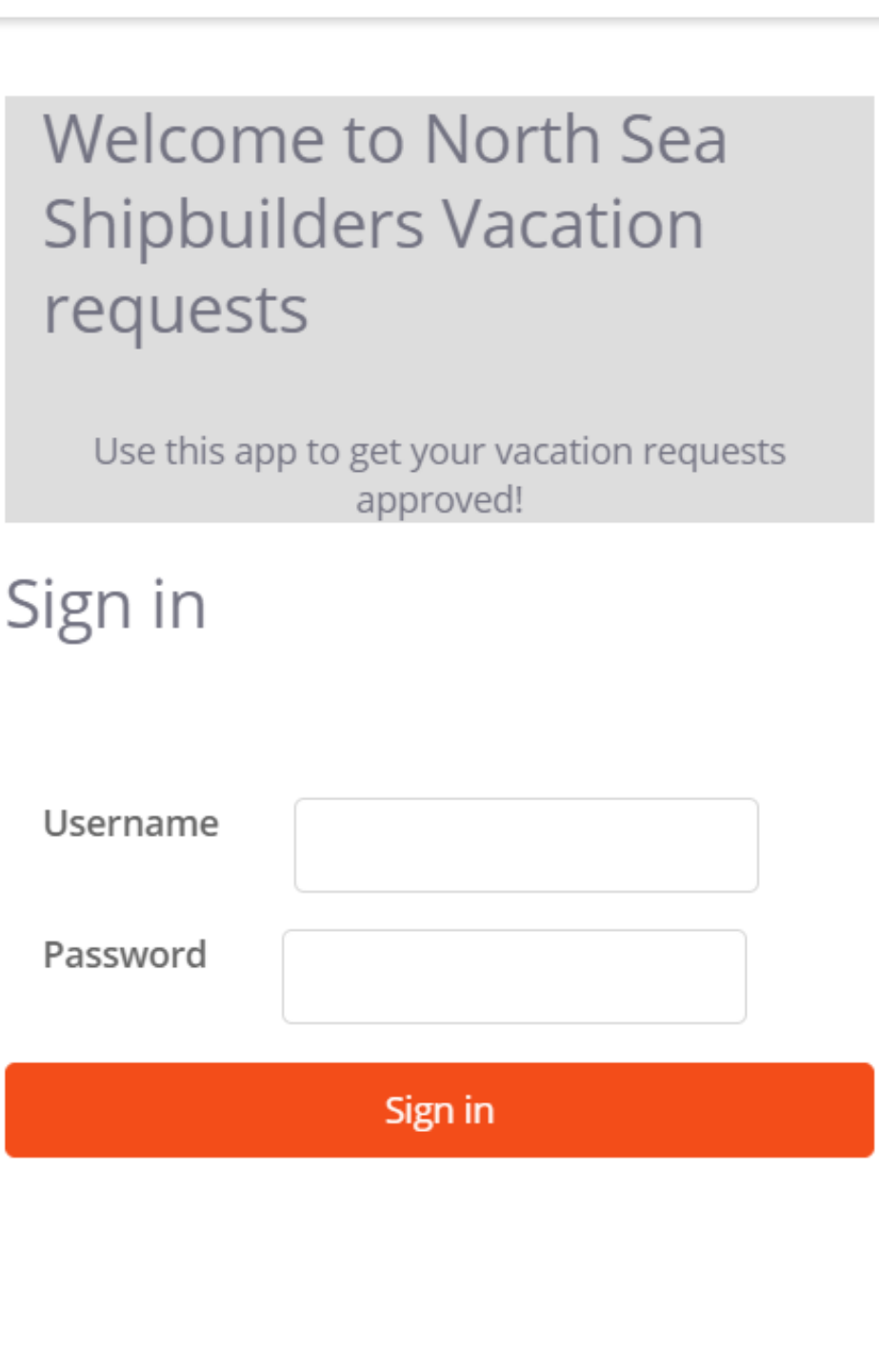
# Create custom login page

- Create **Anonymous** user role & corresponding module role in **GeneralExtentions** module
- Allow Anonymous users

- Create new login page

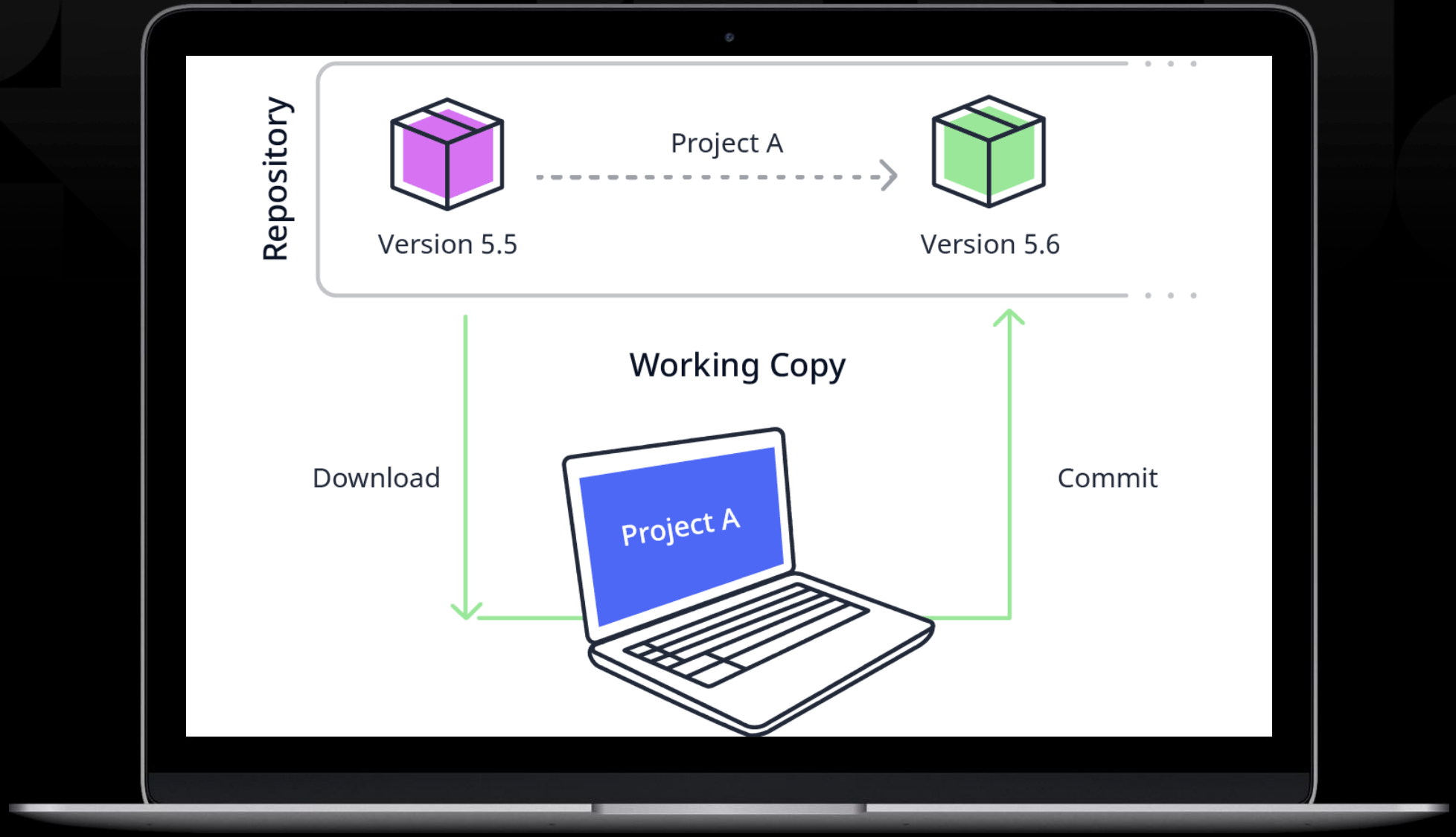
- Set the login page as:
  - Default homepage
  - Sign-in page
- Set up rolebased homepages for Administrator, Manager & Employee

- Build out custom login page to match this design




mx mendix

# Commit your work

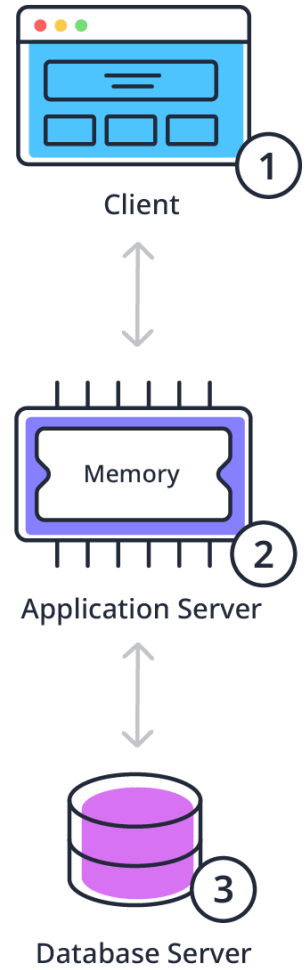


# User story

*As a user I want to be able to sign up for the app so I do not have to go to IT for an account*

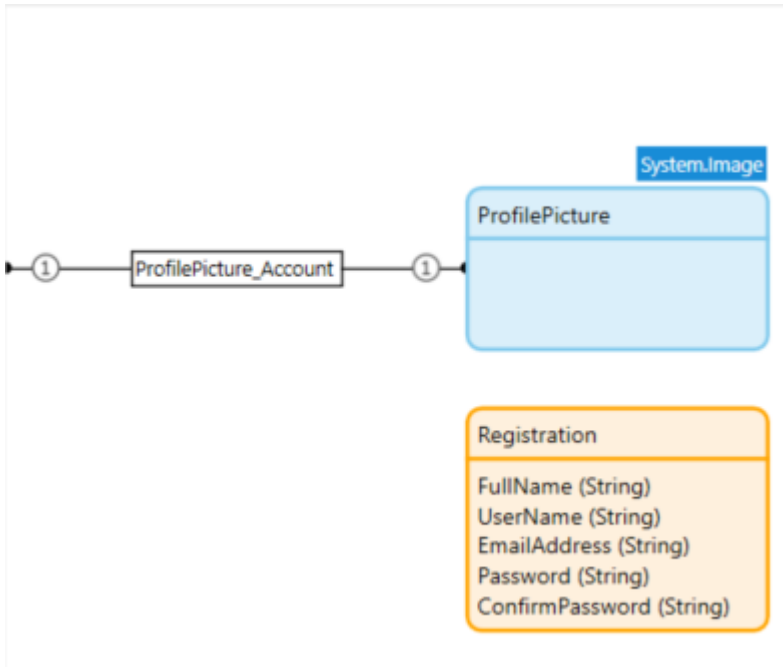
# Database vs Memory

- Persistable and non-persistable
- Transient objects
- Non-persistable assocations (1-1 not possible)
- Other limitations
  - No AutoNumber
  - No validation rules on Domain Model
  - No indexes
- Why use them?



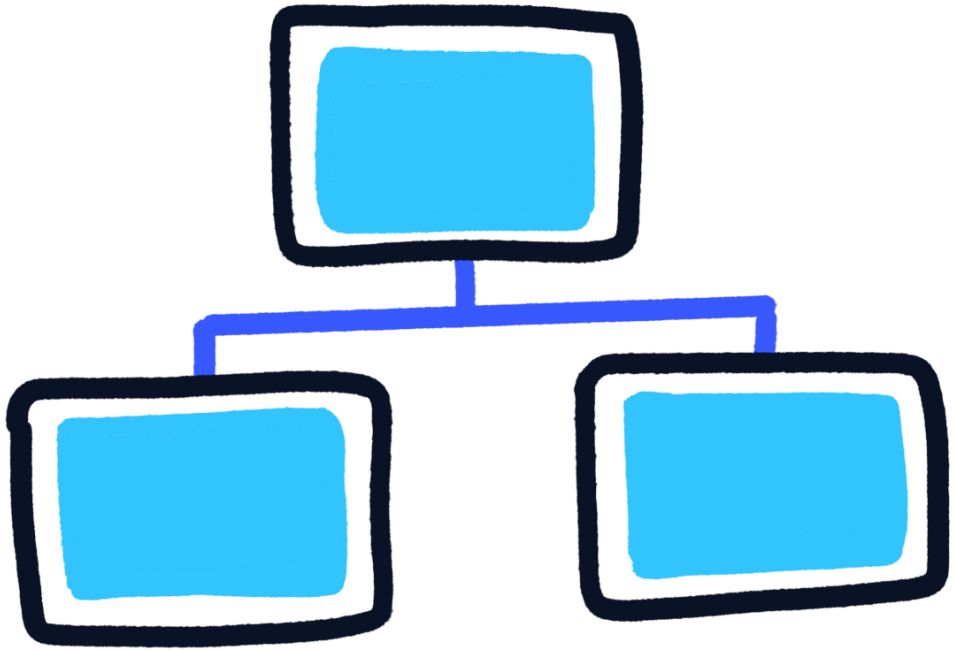
# Allow Anonymous users to sign up

- Add a *Sign up here* section to the login page.
- Extend the Domain Model with a new **Registration** entity
- Create new popup page **Registration_New**
- Make the **Sign up here!** button create a new Registration object and open the page.
- Set up access rights for Anonymous user (Create & Read, Write)

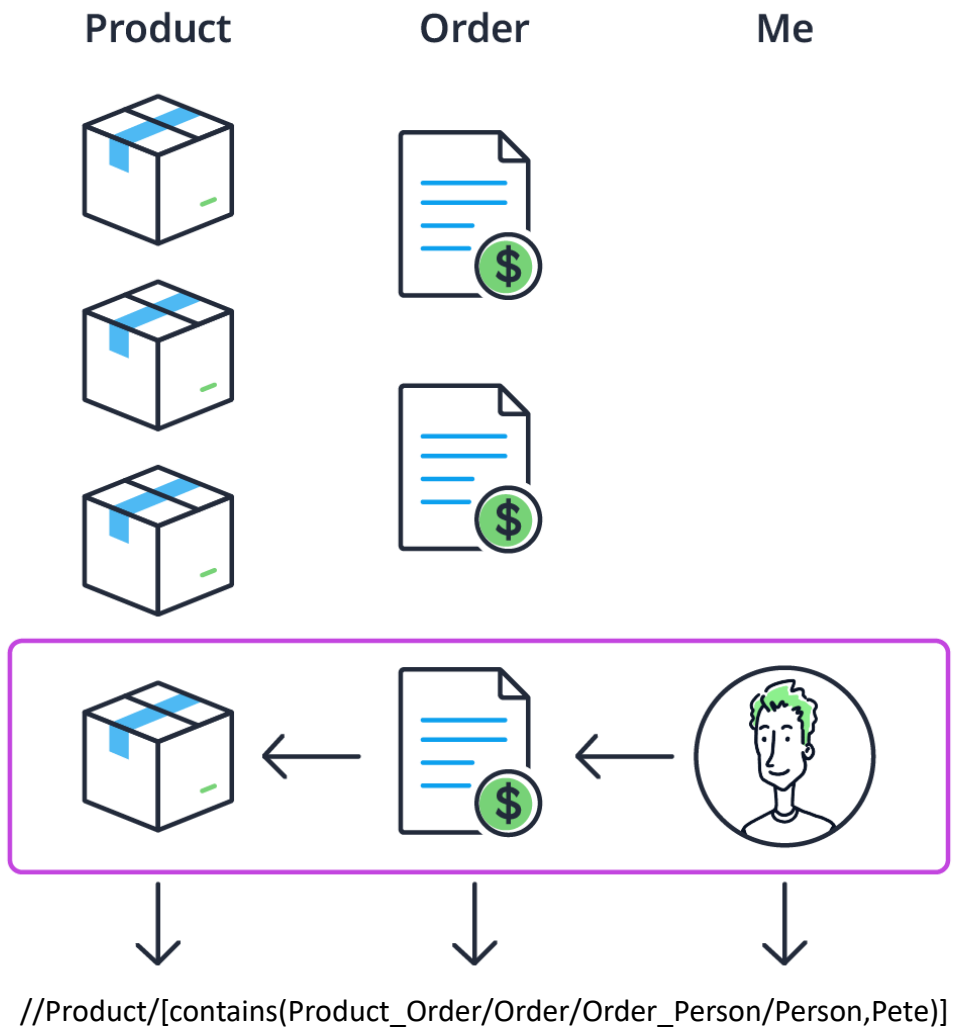
# Using more complex constraints

- XPath Keywords & System Variables
  - NULL/empty
  - System variables
  - Time-related variables
- XPath Constraint Functions



# What is an XPath function?

- **String functions:**
  - contains(*{path}*, *{string}*)
  - string-length(*{path}*)
  - ends-with(*{path}* , *{string}*)
- **DateTime functions:**
  - year-from-dateTime(*{path}*)
  - weekday-from-dateTime(*{path}*)
- **Boolean functions**:
  - true()
  - false()

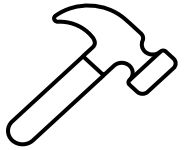


# What is the Userrole System Variable?

- Automatically created when you create a **User Role** in **Project Security**

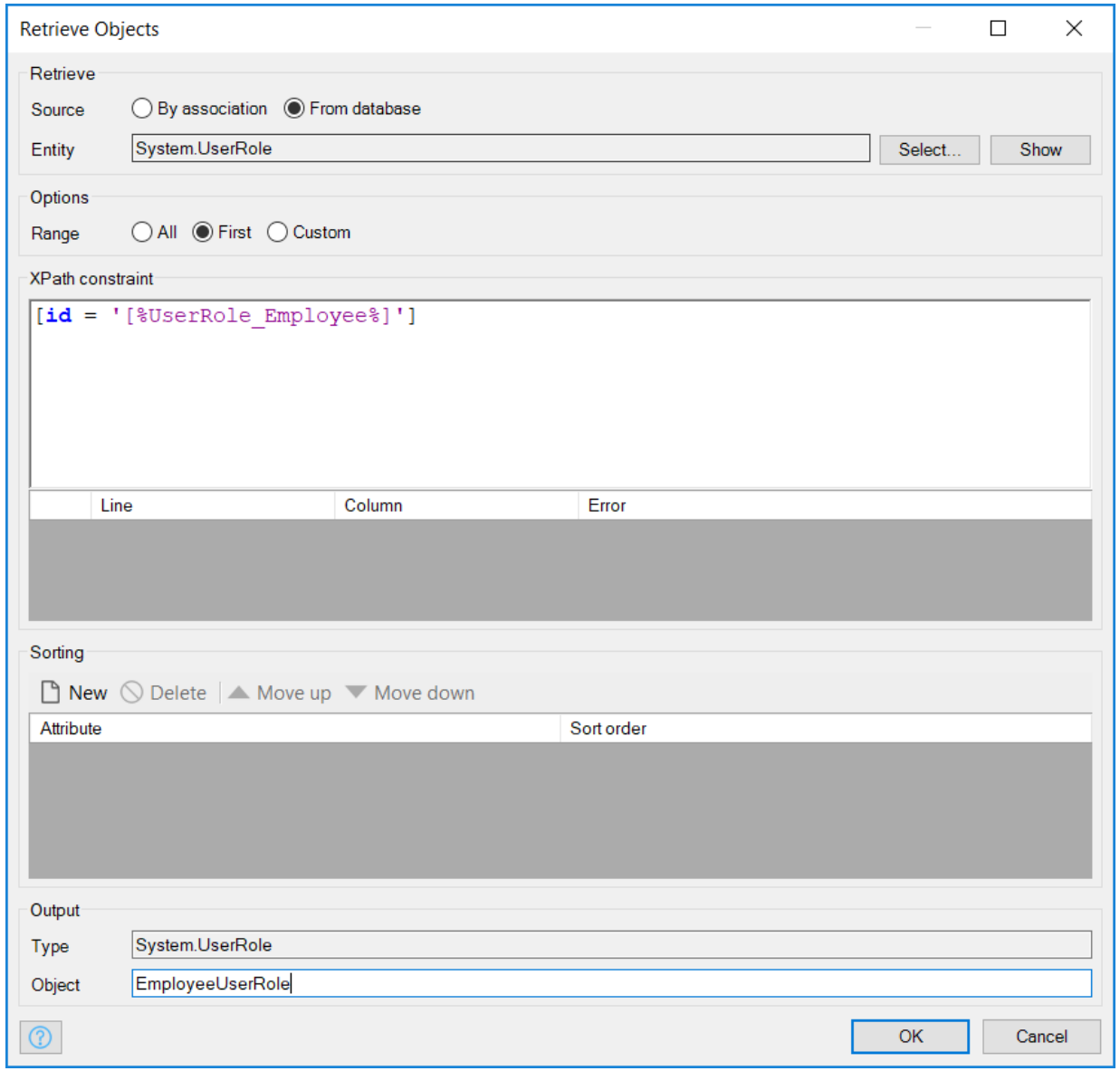
- Contains name of the **User Role:** '[%UserRole_Employee%]’

- Appears in auto-complete when XPath ends in System module association System.UserRoles



# Create Account for Anonymous user

- Make **Create my Account (Save)** button **Call a microflow**
- Create new microflow, and give Anonymous user access to it
- In the microflow:
  - Make sure all fields are filled out
  - Make sure the passwords match
  - Create a new **Account** object and pass in the data from the **Registration** object
  - Set the UserRole to Employee
  - Commit and close the page



# Sub microflows

- Improve readability and maintainability
- Analyze usage (dependencies and input parameters)
- Sub Microflows and security (parent MF dictates access)

**Notes on submicroflows:**

- Impossible to extract:
  - start events
  - end events
  - input parameters
- Don’t overdo it!
- Changing primitive input parameters

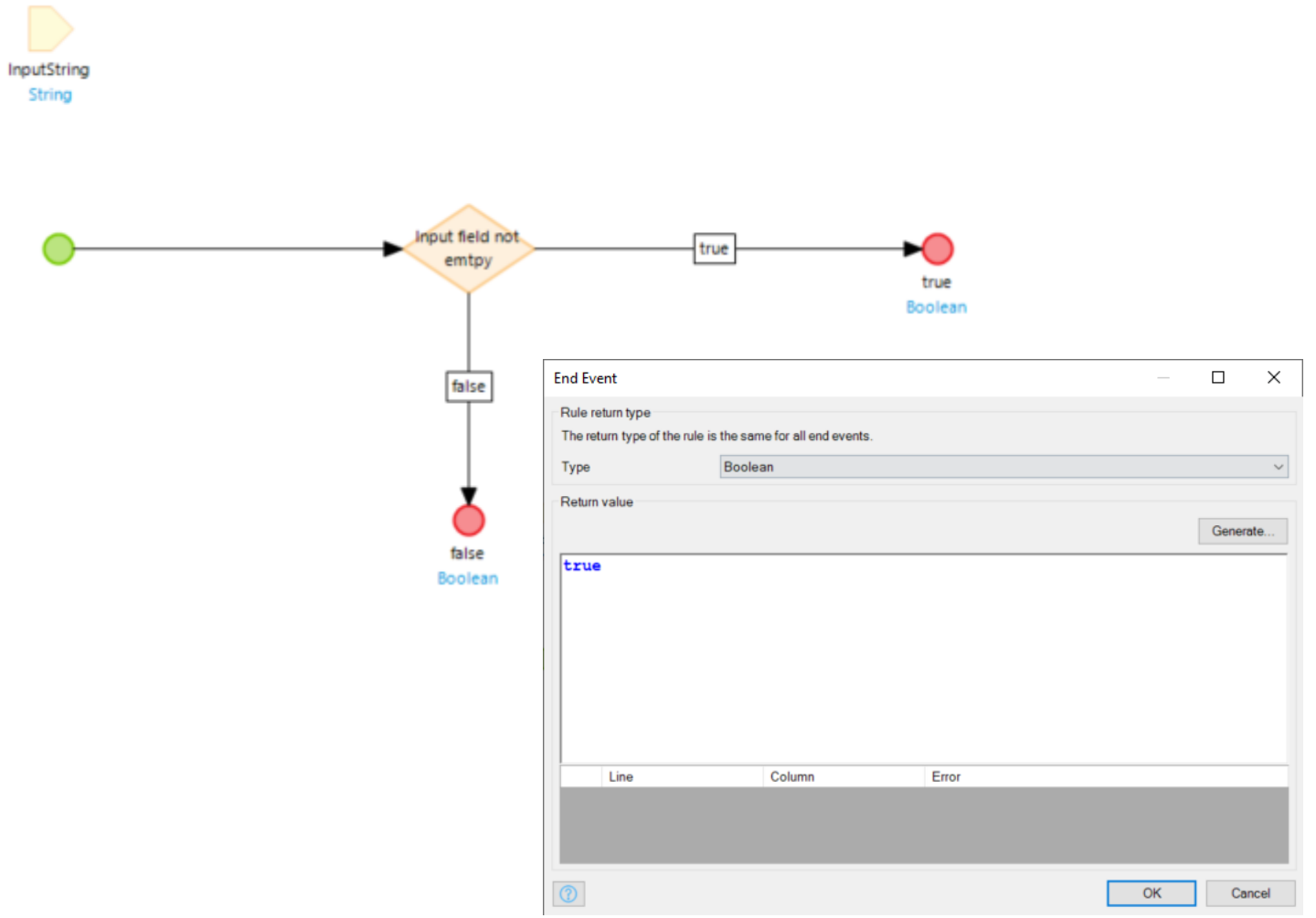
# Validating string inputs

- **$InputString != empty** doesn't cover it!
- **empty** is not the same value as **''**
- Correct way of checking empty string:

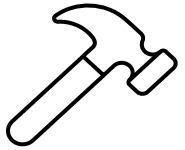
```
$InputString != empty
and
trim($InputString) != ''
```

# Rules

- Similar to a microflow
- Evaluate data
- Can only be used in a decision
- Should always return a Boolean
- Cannot change data
- Cannot interact with the client
- Cannot call webservices




mx mendix



# Extract submicroflow

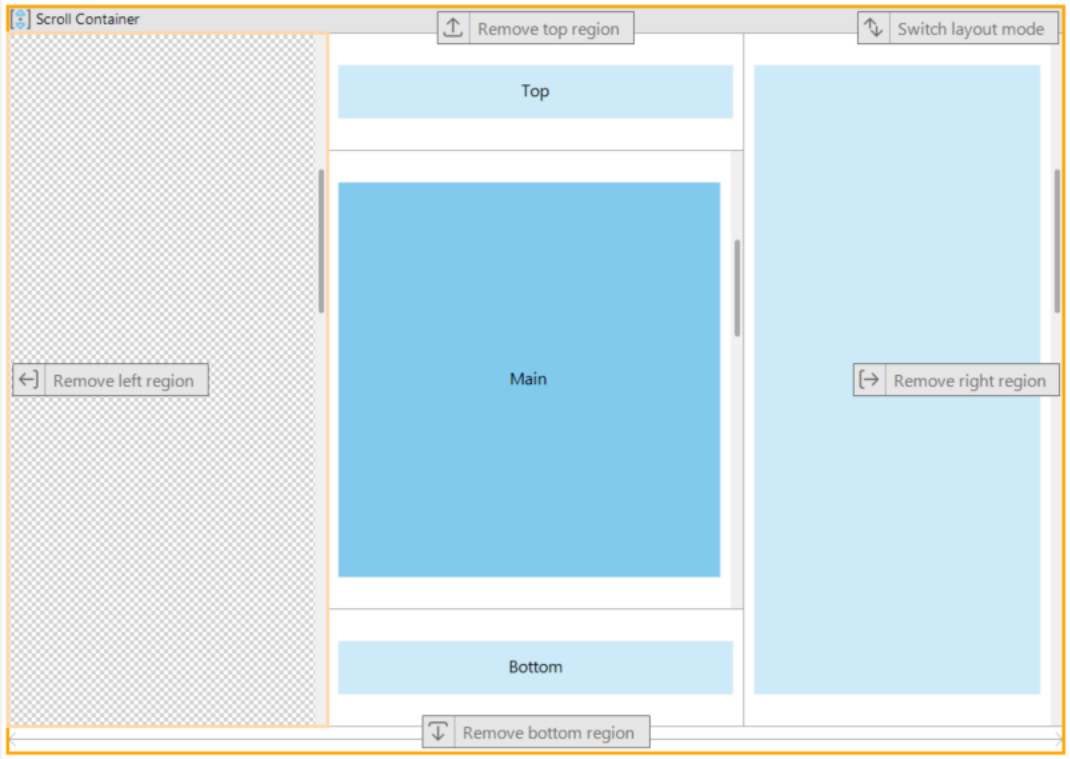
- Extract validation logic as a submicroflow
- Create reusable rule to check for empty strings
- Use it in the validation microflow
- Deploy and test your functionality

# Layouts

- What are layouts?
- Why do we use them?
- Default options
  - Atlas Default
  - Atlas Topbar
  - Popup

# Layouts: Scroll container

- Divide the layout in individual regions
- Set size of these regions
- Set toggle mode of these regions
- **Contents** → present on every page that uses that layout
- **Placeholders** → where the page content will be placed



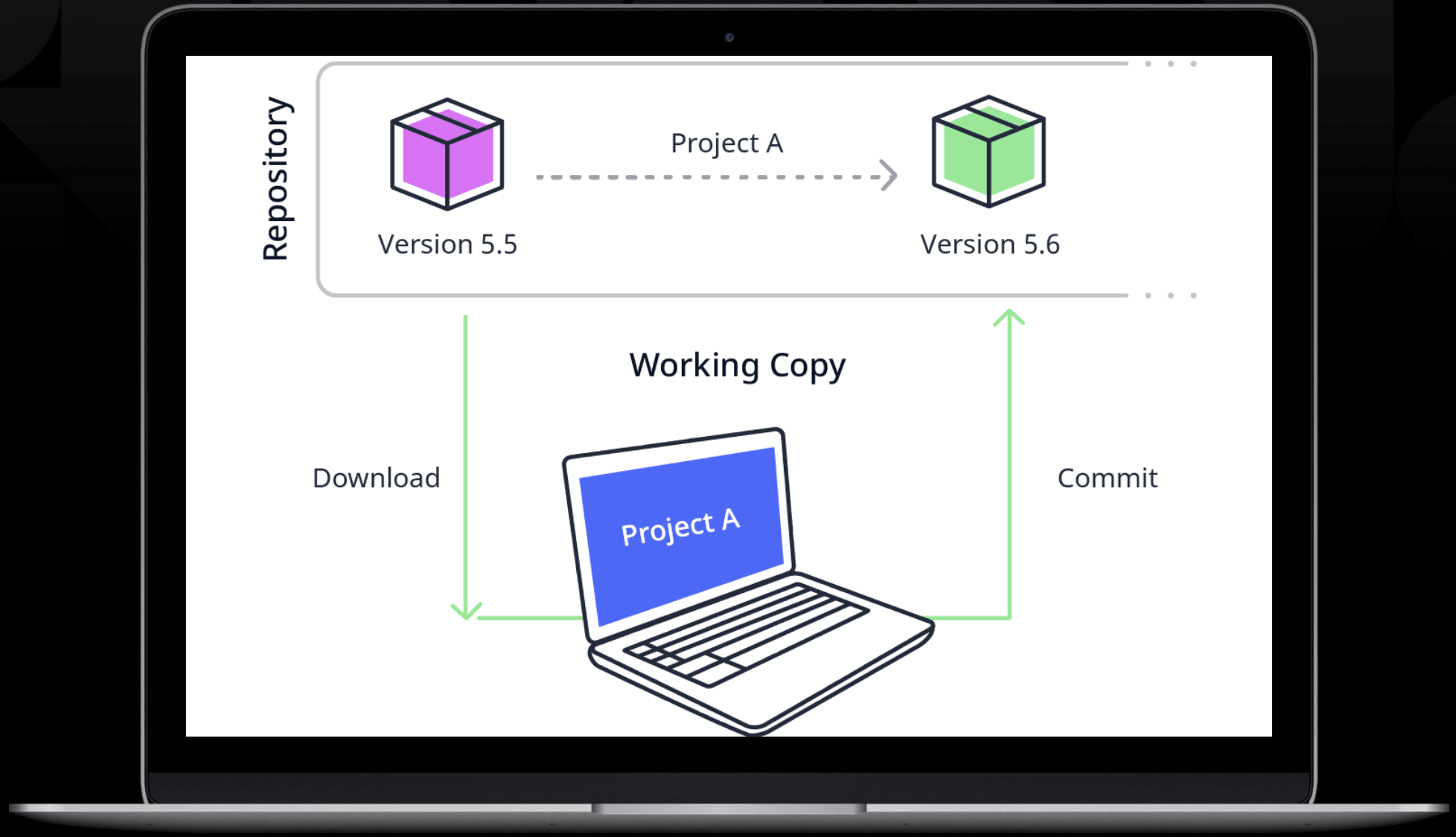

mx mendix

# Layouts: Master layout

- Layouts can be based on other layouts
- Use placeholders in Master to create specialized configuration
- Don't forget to add new placeholders for the page!

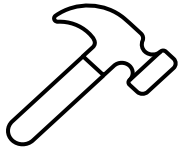
- DRY (Don’t Repeat Yourself)
- Increase productivity & maintainability
- Best practice: maximum 3 levels

# Commit your work



# User story

*As a company I want to allow users to log in to our app without exposing functionality to the world*



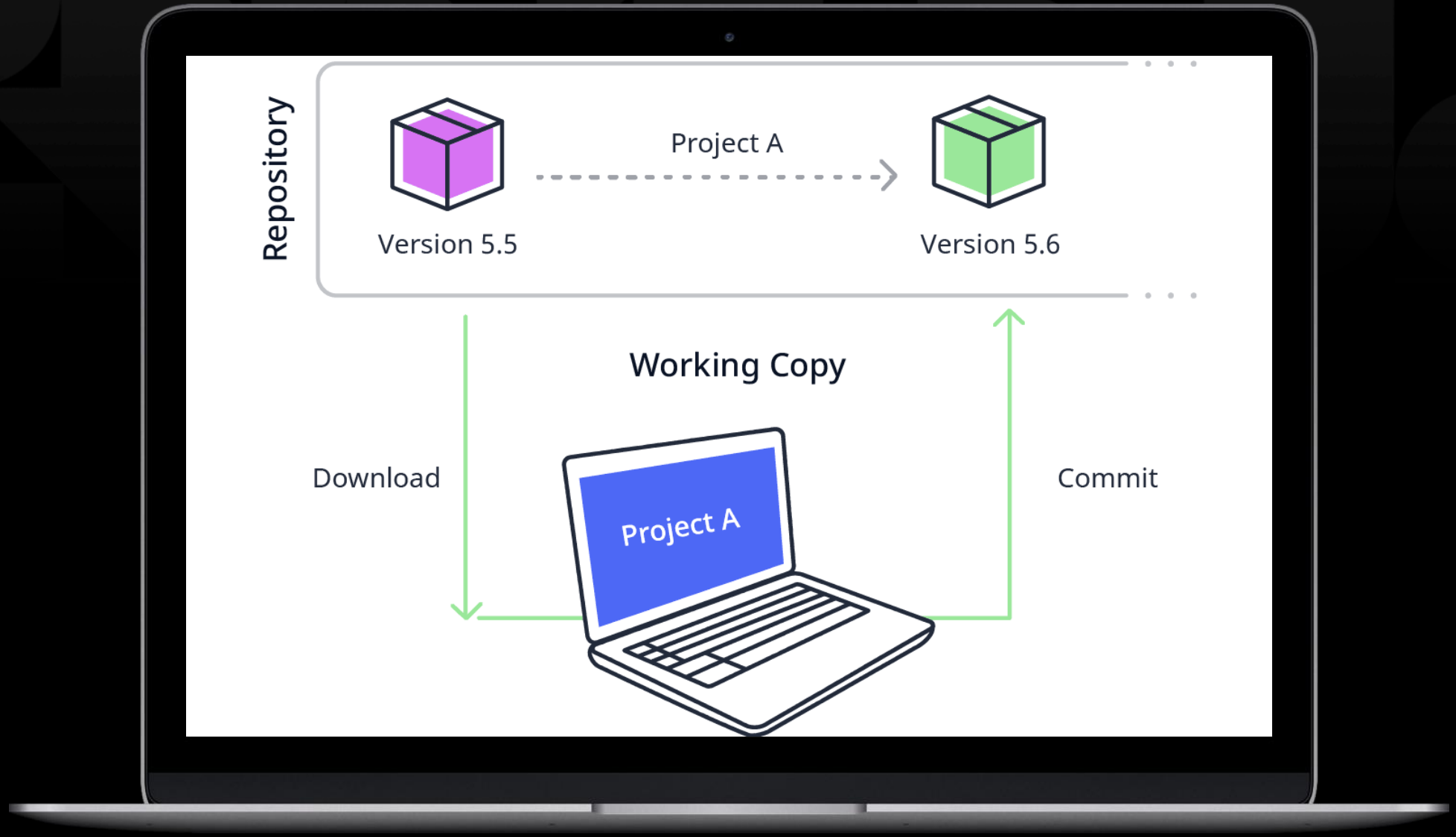
# Using and creating navigation layouts

- For the **Home** page, change the navigation layout to **Atlas_TopBar**

- For the custom login page, create a new navigation layout **Atlas_Login**
  - The navigation layout should only contain the **Main** placeholder
  - Place this navigation layout in:

    *App Store Modules > Atlas_UI_Resources > _Layouts > Responsive*

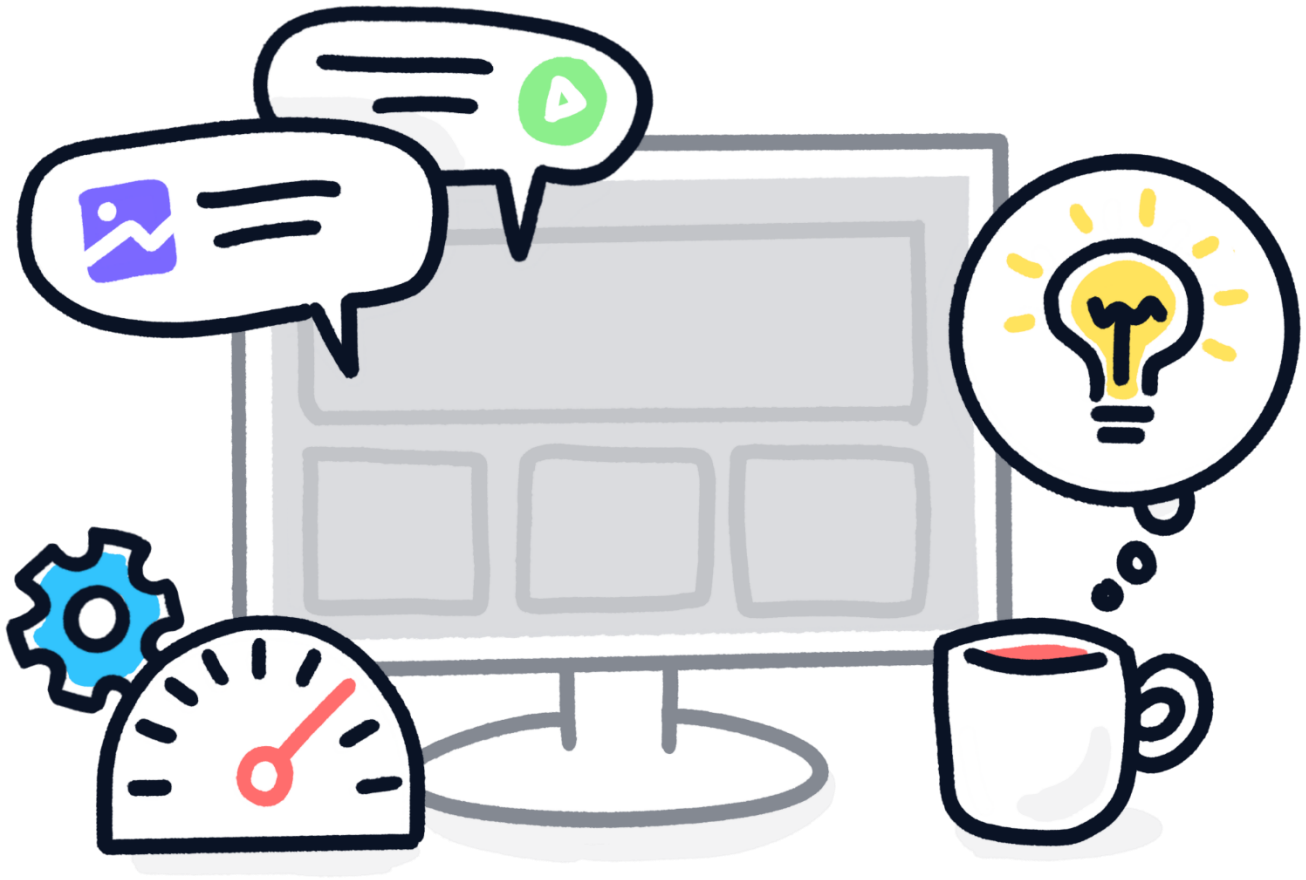
# Commit your work



# Adding requests

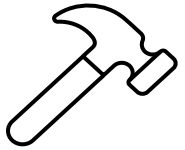
# Learning goals

- How to use enumerations in Mendix
- Why you want to create multiple associations between two entities
- What XPath is and how you can apply it
- How to create a wizard in Mendix
- How to calculate the total number of requested hours
- How to exclude weekends



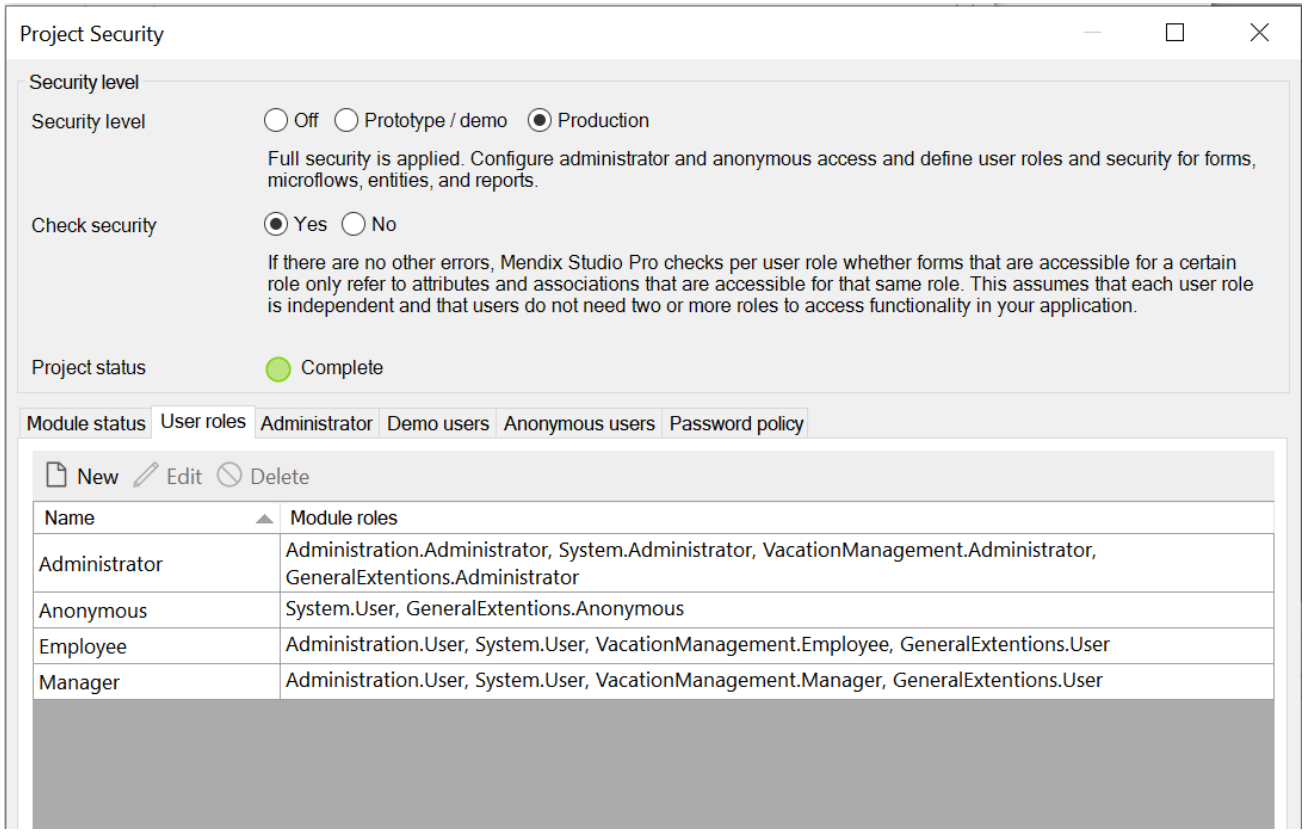
# User story

*As a user I want to add vacation requests so I can go on vacation*



# Configure MyFirstModule

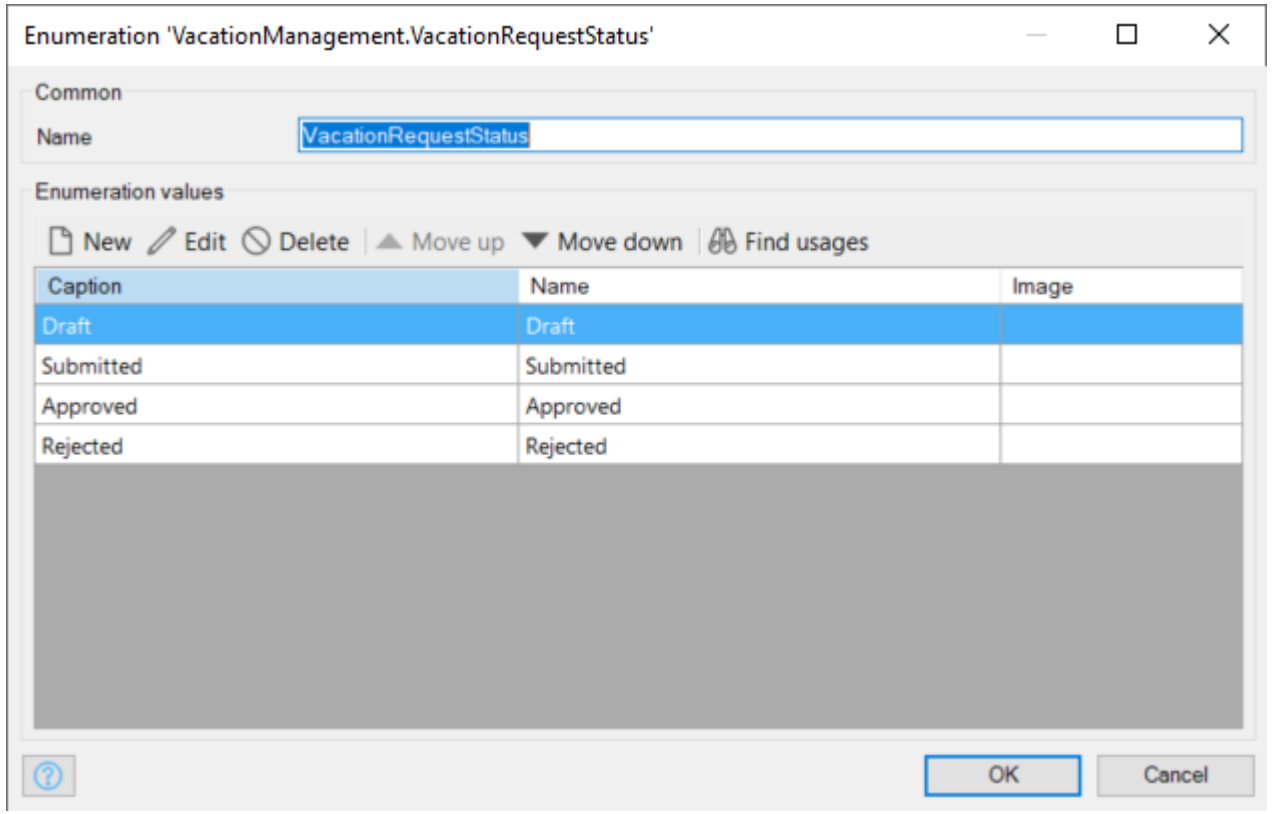
- Rename **MyFirstModule** to **VacationManagement**
- Create the following module roles:
  - Employee
  - Manager
- Change the User module role to Administrator
- Connect the module roles to the corresponding user roles
- Give every module role access to the Home page.




mx mendix

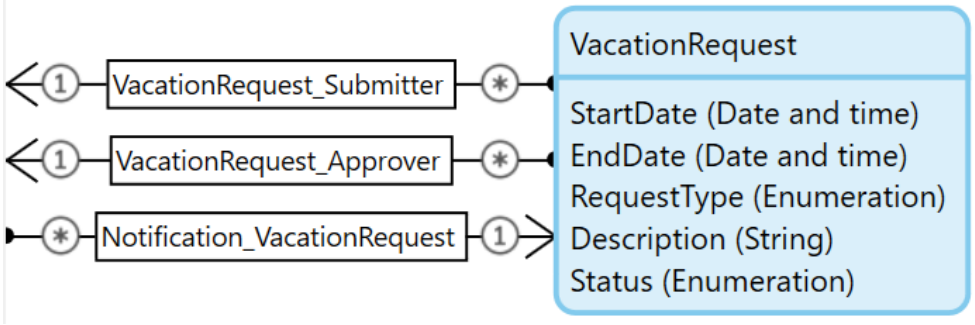
# What is an Enumeration Value?

- Pre-defined list of string values
- Can be selected in a dropdown or from radio button
- Can be used to constrain data



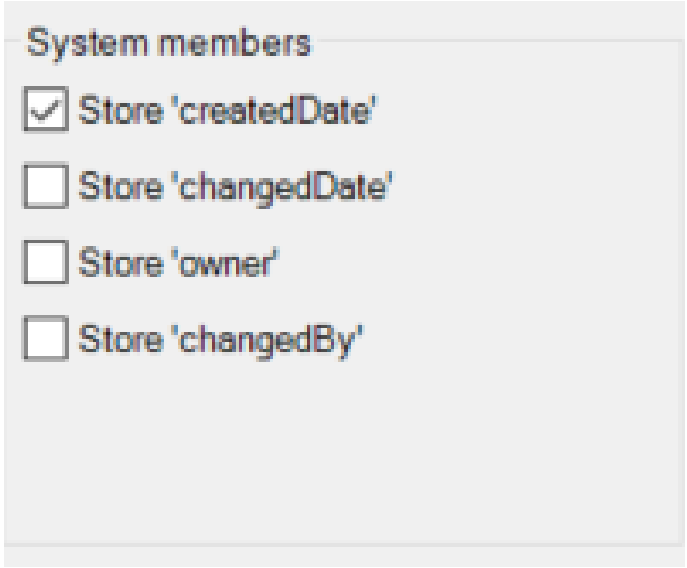
# Multiple associations between entities

- Different relations between instances of entities
- How does the user want to see and manipulate data?



# System Members

- createdDate
- changedDate
- owner
- changedBy

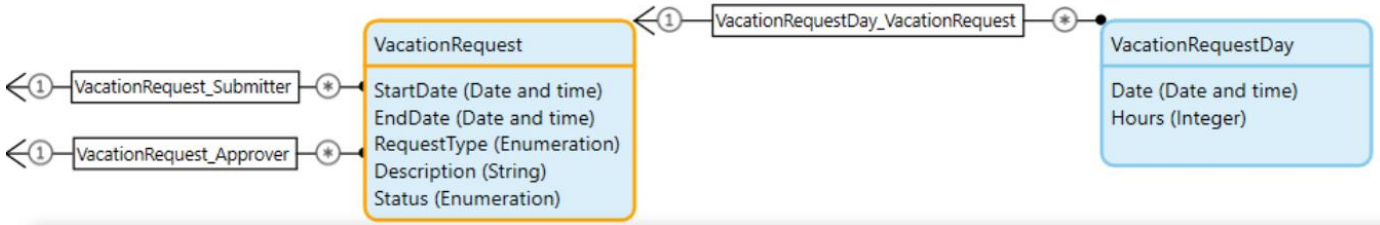


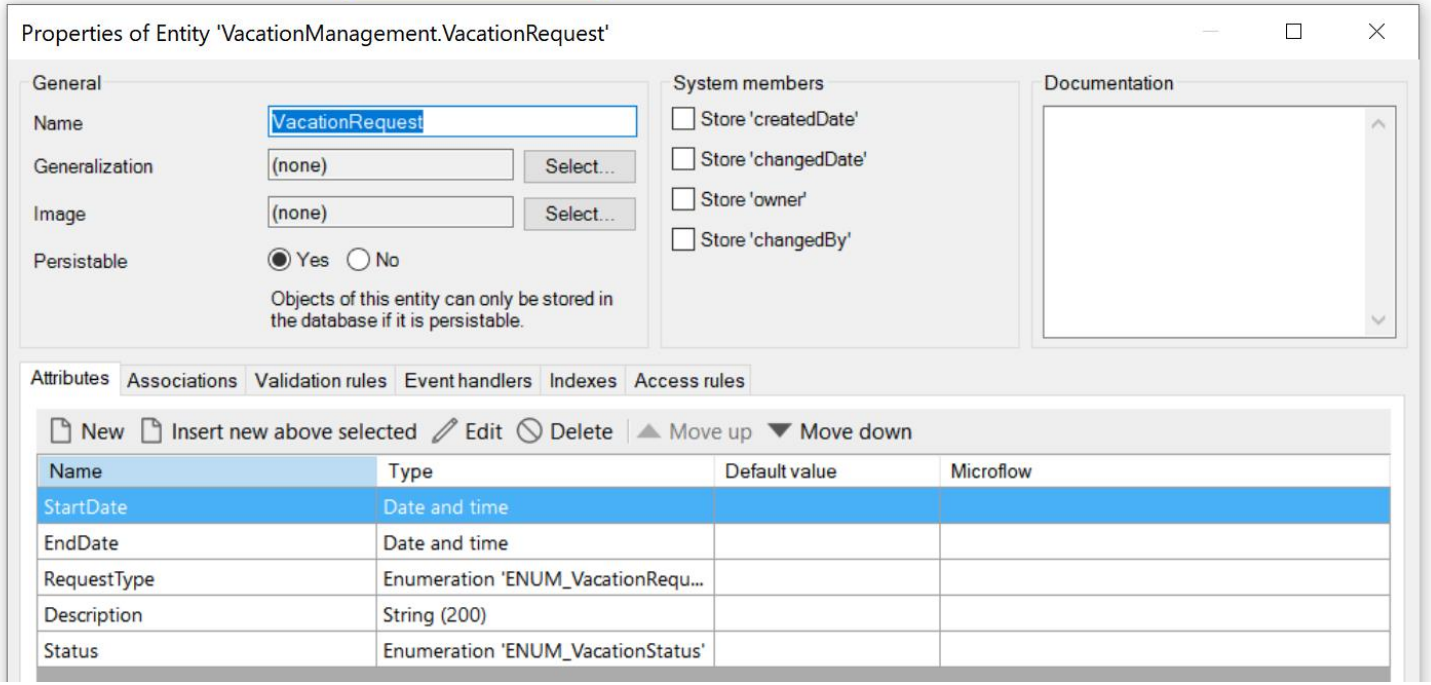
# How do we store a vacation request?

- Start-date and end-date
- Type of request
- Status of the request
- Description
- Number of hours
  - Number of hours per day
  - We need a VacationRequestDay entity!

# Adding requests: extend Domain Model

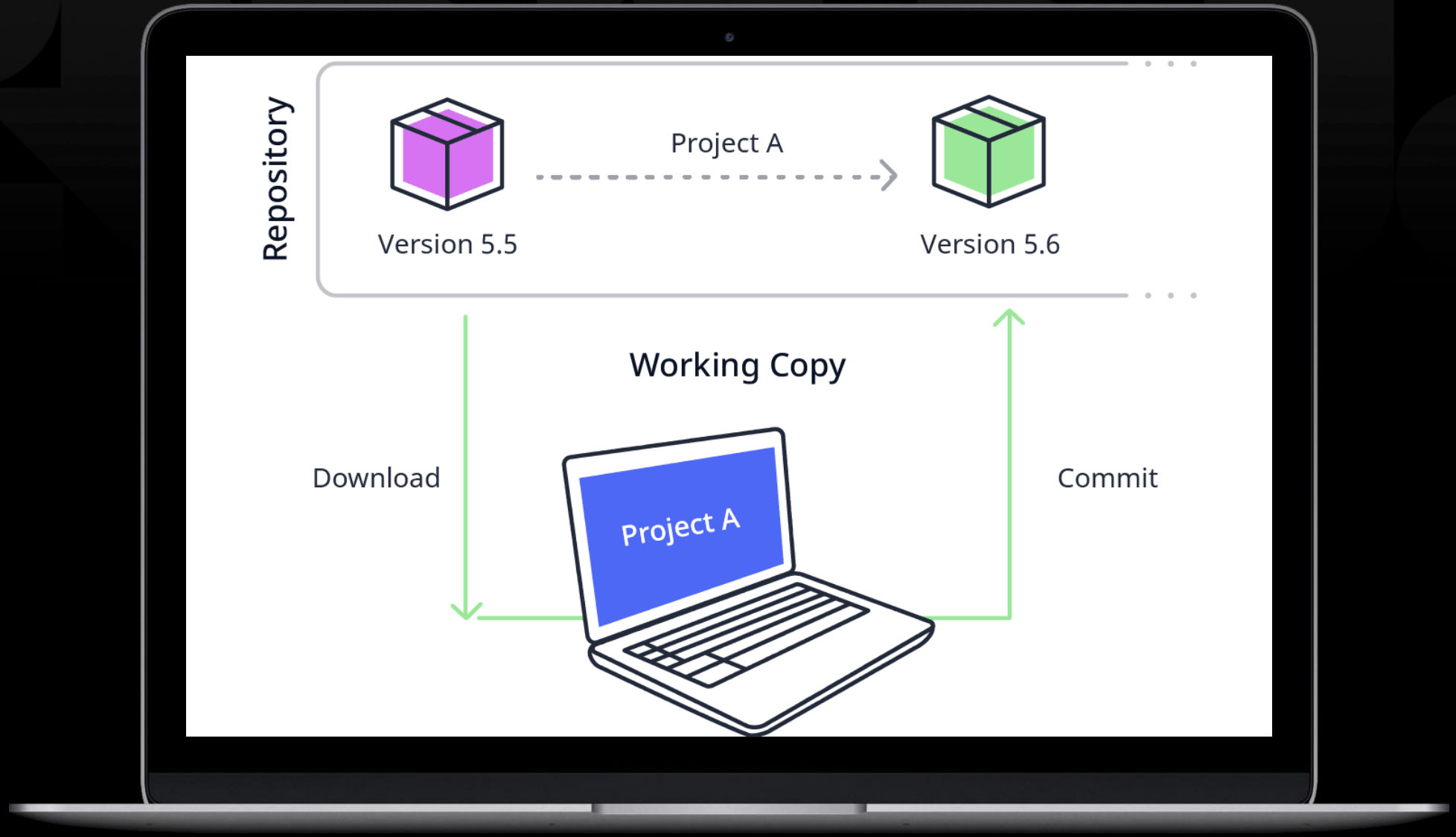
- Add **VacationRequest** entity
  - Add attributes
  - Store **'createdDate'** system member
- Add **VacationRequestType** enumeration:
  - Annual Holiday Paid/Paid Time Off (PTO)
  - Special Leave
  - Parental Leave
  - Unpaid
- Add **VacationRequestStatus** enumeration:
  - Draft
  - Submitted
  - Approved
  - Rejected
- Add cross module associations to **Account** entity
- Configure access rights
  - Employees should only see their own requests
  - Managers can only edit Status




mx mendix

# Commit your work



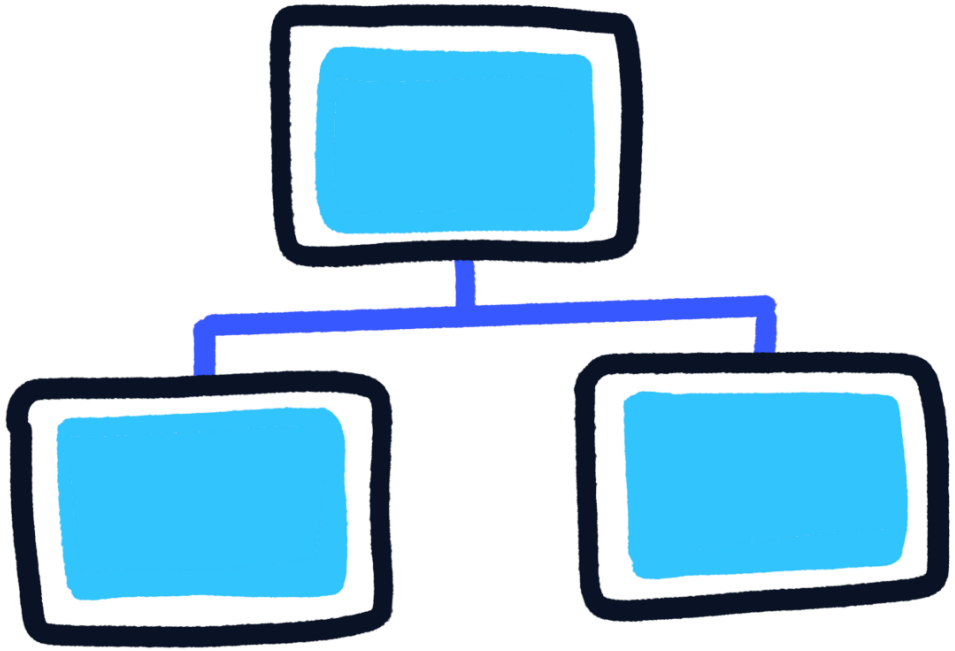
# User story

*As a user I want to see my upcoming vacation and my new requests on my homepage so I have an overview of all relevant requests*

# Using more complex constraints

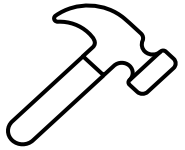
- XPath Keywords & System Variables
  - NULL/empty
  - System variables
  - Time-related variables
- XPath Constraint Functions
- XPath Operators

| | |
|---|---|
| = | Equal to |
| != | Not equal to |
| < | Less then |
| <= | Less then or equal to |
| > | More then |
| >= | More then or equal to |
| or | Or |
| and | And |



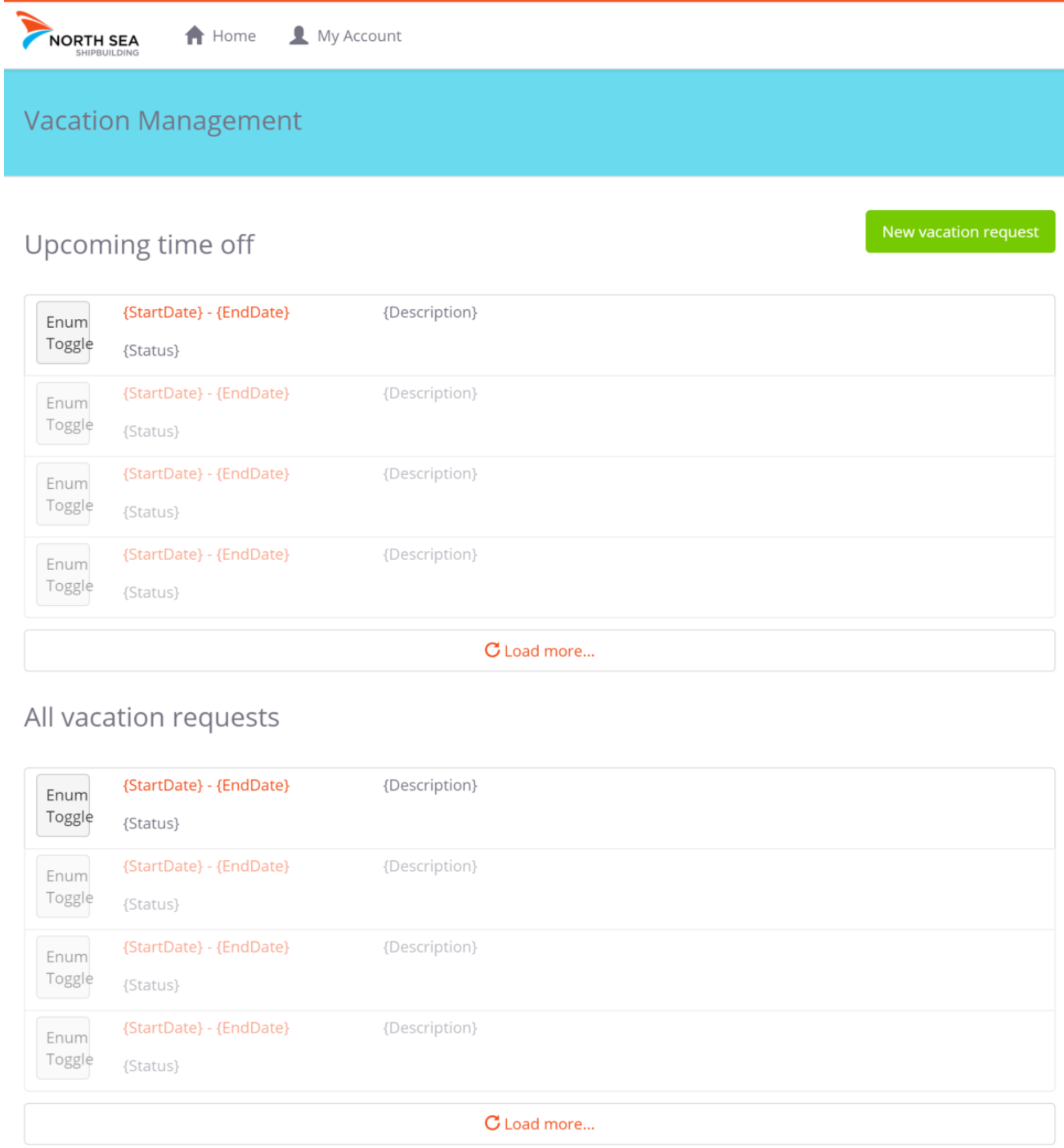
# What is an XPath operator?

| | | |
|---|---|---|
| = | Equal to | price = 9.80 |
| != | Not equal to | price != 9.80 |
| < | Less than | price < 9.80 |
| <= | Less than or equal to | price <= 9.80 |
| > | Greater than | price > 9.80 |
| >= | Greater than or equal to | price >= 9.80 |
| or | Or | price = 9.80 or price = 9.70 |
| and | And | price = 9.80 and amount = 1 |



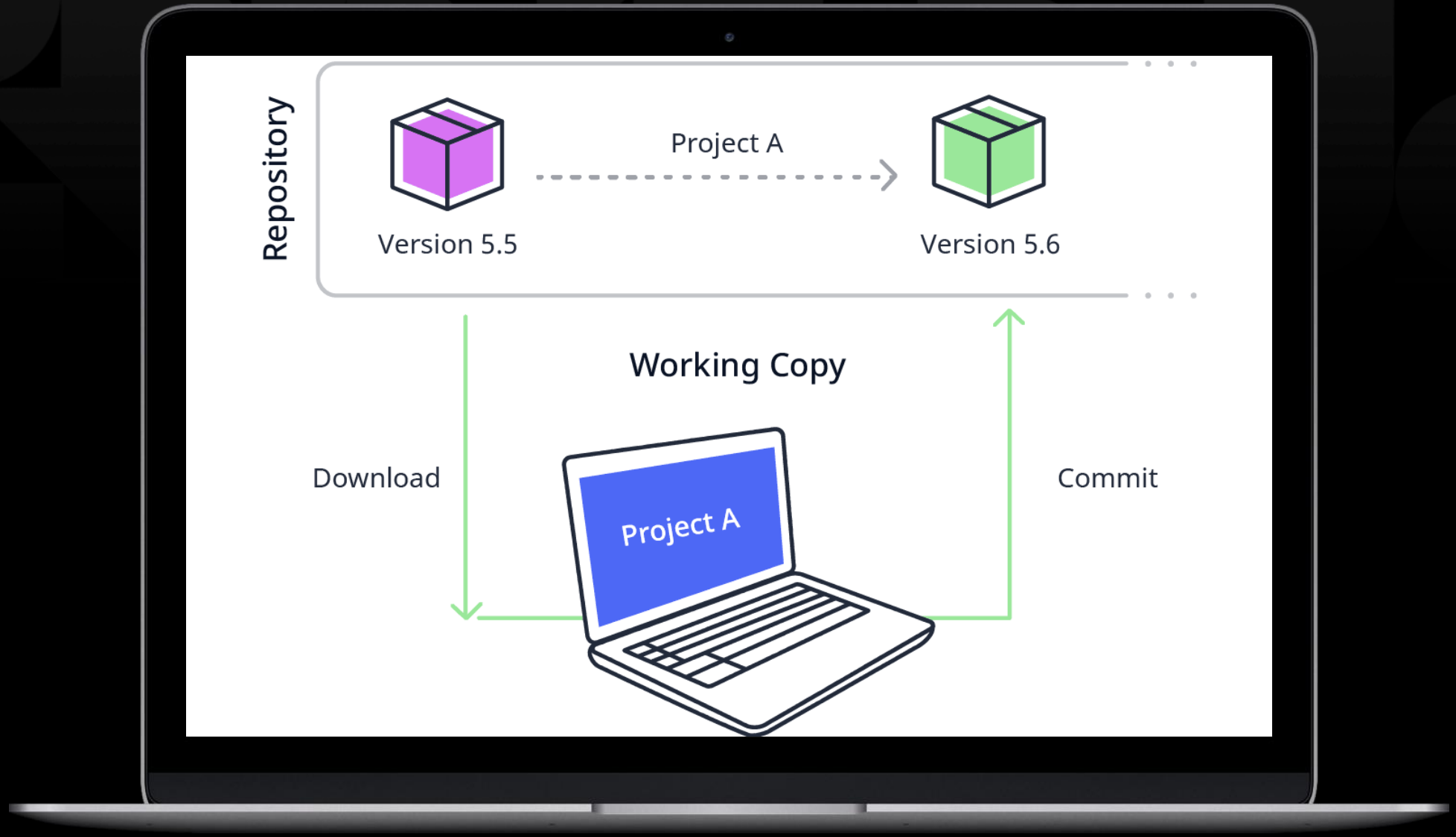
# Adding requests: Home page

- Add a second **row** to the **layout grid**
- Add two **list views** (one in each row) and connect them to the **VacationRequest** entity
- In the first list view, only show the first 4 upcoming requests
- Add a **New vacation request** button, set the on-click action to **Do nothing** for now
- Download the **Enum Toggle** widget from the appstore
- Add an image collection **VacationRequestTypeImages** and add the icons to the Enum widget
- Match the design




mx mendix

# Commit your work

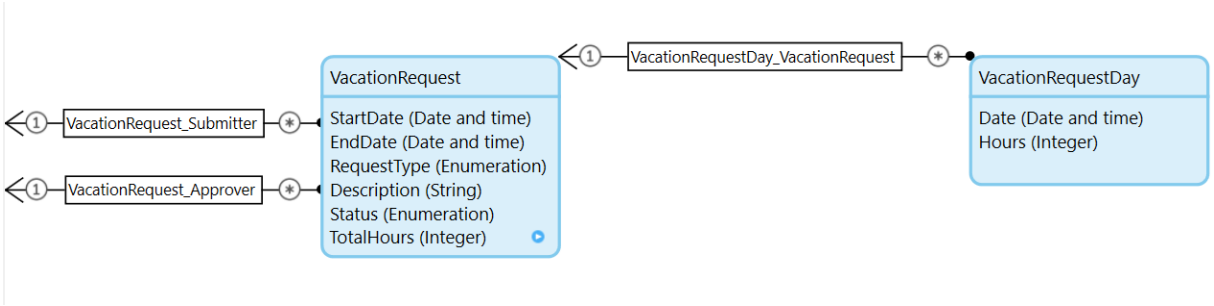


# User story

*As a user I want to be able to bundle several time-off requests so I can ask time off for several timeframes*

# The Wizard page template

- Create three wizard pages
  - Page 1 allows requestor to fill in start and end date, type, and description
  - Page 2 allows requestor to select number of hours per day
  - Page 3 will review time requested – both dates and the total number of hours
- Configure progress bar at the top to show which step of the wizard it is
- Think about the transaction between pages



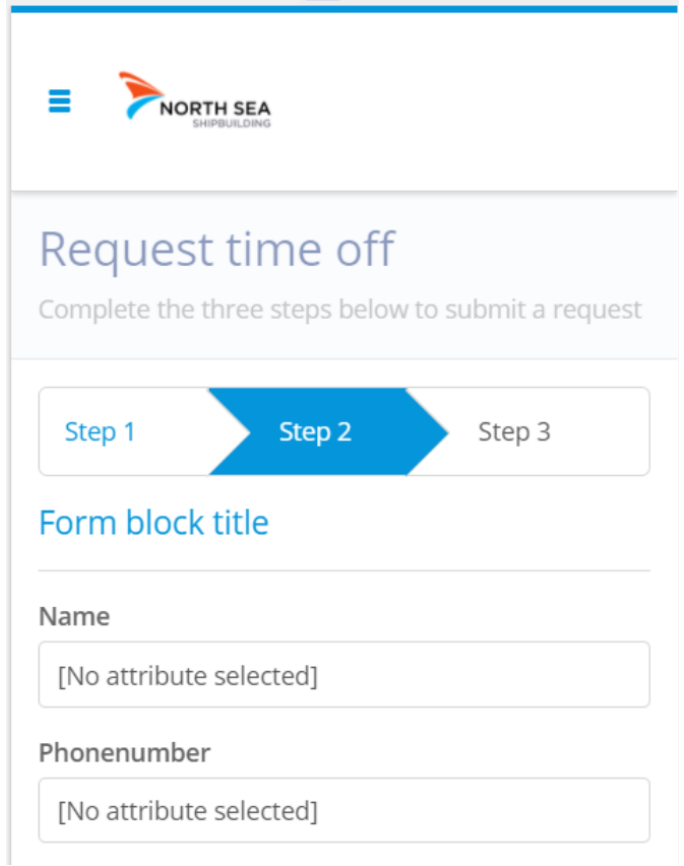
# Building out the request wizard

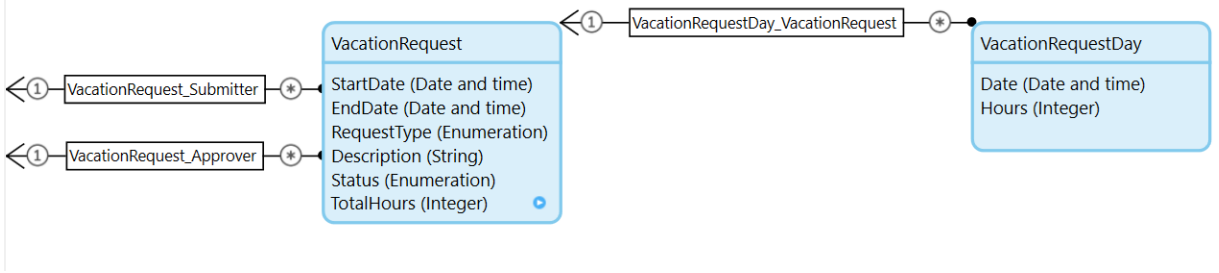
Let’s build out our request wizard:

- Ability to add a request
- Request custom hours per day
- Automatically set 0 hours for weekend dates
- Ability to edit a single request
- Three pages
  1. Create request
  2. Choose hours
  3. Review and submit
- Use CSS classes to configure wizard pages

# Adding requests

- Add new Entity **VacationRequestDay**
  - Give Employee full CRUD
- Create wizard pages in new folder:
  **VacationRequest_Wizard_Step1**
  **VacationRequest_Wizard_Step2**
  **VacationRequest_Wizard_Step3**
  - Atlas_TopBar
  - Wizard Progress
- Set classes accordingly for current step
  - wizard-step
  - wizard-step wizard-step-visited
  - wizard-step wizard-step-active
- Create a snippet **WizardHeader** from one of the headers, and add some nice text to it

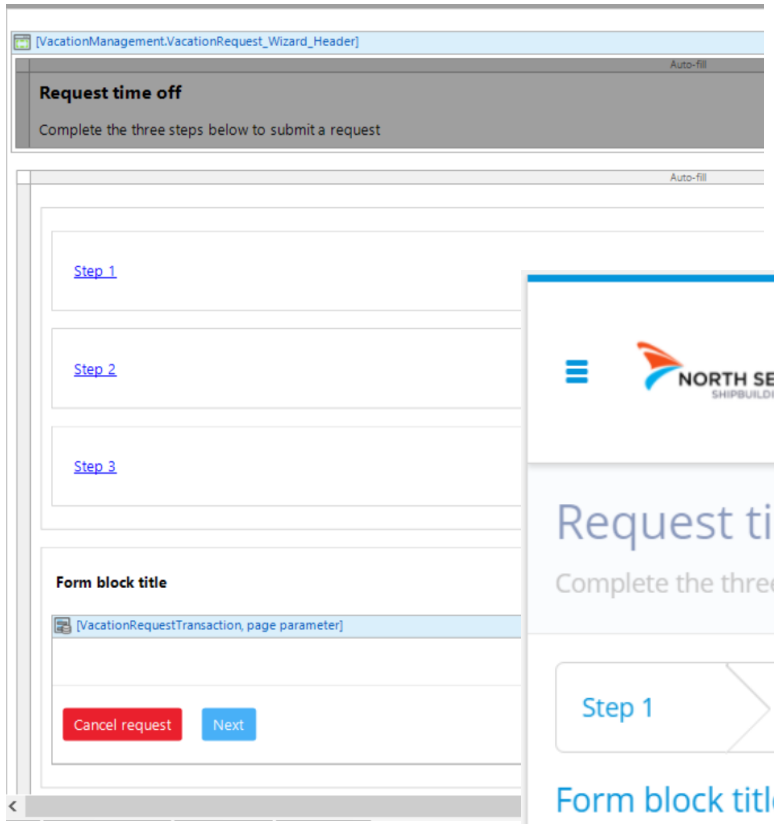


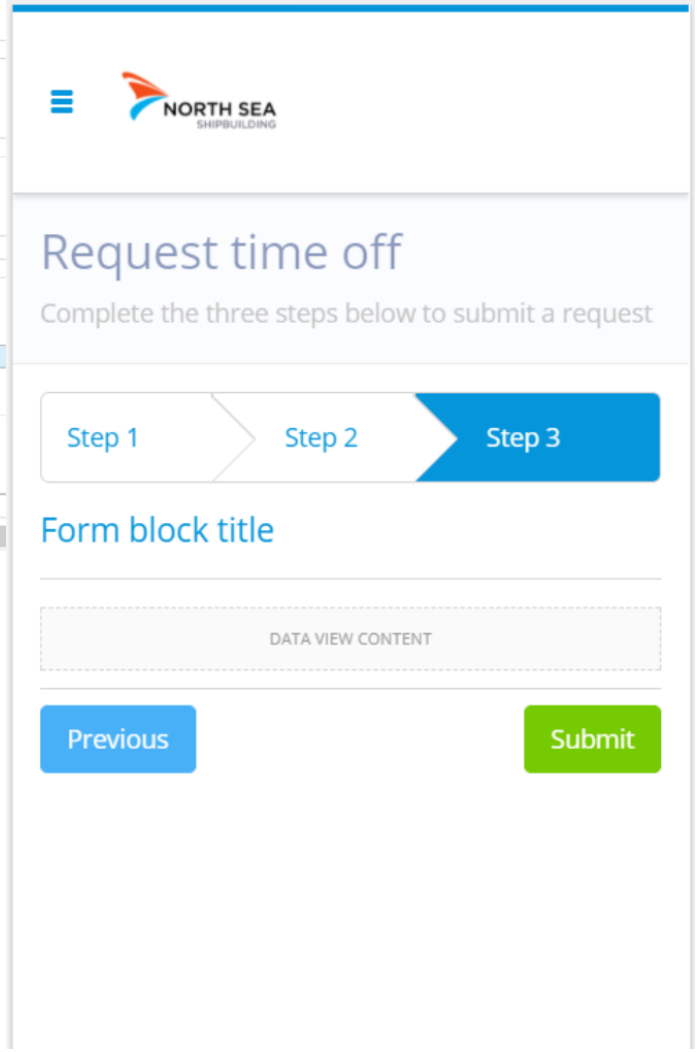



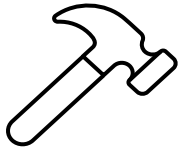

mx mendix

# Adding requests

- On each page:
  - Give **Employee** access
  - Replace the header with the snippet **WizardHeader**
  - Delete step 4
  - Replace list view with data view
  - Connect data view to **VacationRequest**
  - Empty data view contents
  - Create navigation buttons at bottom
    (don’t worry about making them functional yet)

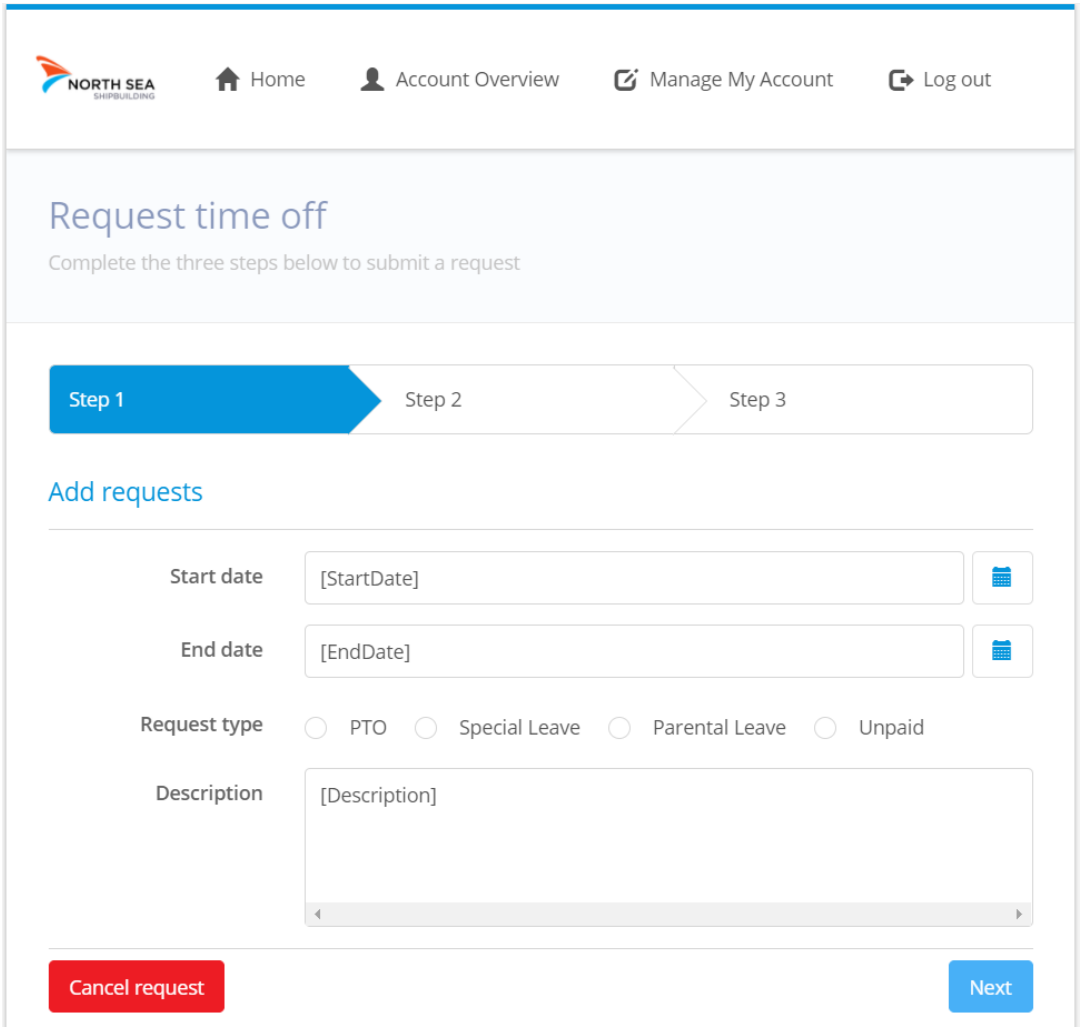




# Adding requests: Wizard step 1

- Match the design in the image
- Set the Cancel button to **Cancel changes**
- Configure the **New vacation request** button on the homepage
- Set the Next button to Call a microflow
  - Create a new microflow **ACT_GenerateVacationDays**



# Working with lists

## Loops

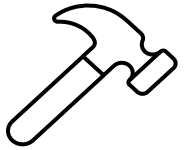
- Iterate over a list of objects
- Can contain all regular microflow elements
  - Except for start & end event
  - Additional: break event & continue event
- Object is called iterator

## Batches

- Processing large amounts of data
- Work with subsets of the data (one batch at a time)
- Safeguards performance

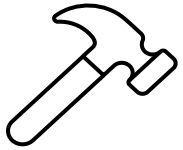
# Generating vacation days: Plan of attack

1. Check if both date fields are filled in
2. Generate vacation days from start date to end date
3. Set hours requested to 8 for weekdays, 0 for weekends
4. Handle changes in start date and end date by deleting all days and generating them again
5. Adjust microflow to preserve changes (don’t delete days that you’ve already created)
6. Show total hours requested using calculated attribute
7. Create microflows for navigating between three wizard pages

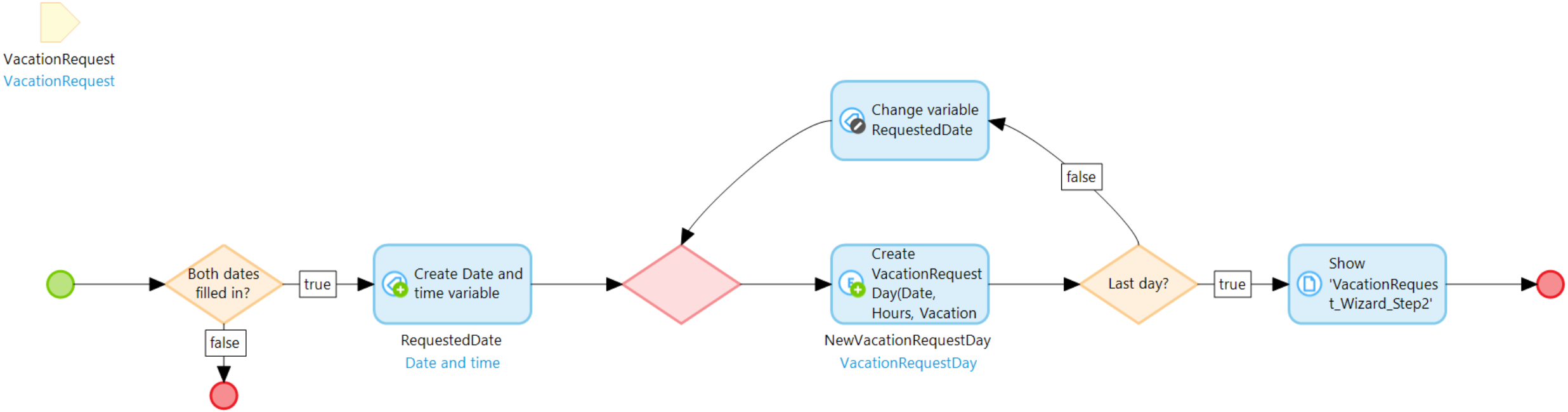


# Generating vacation days

Generate vacation days from start date to finish date when both dates are filled in

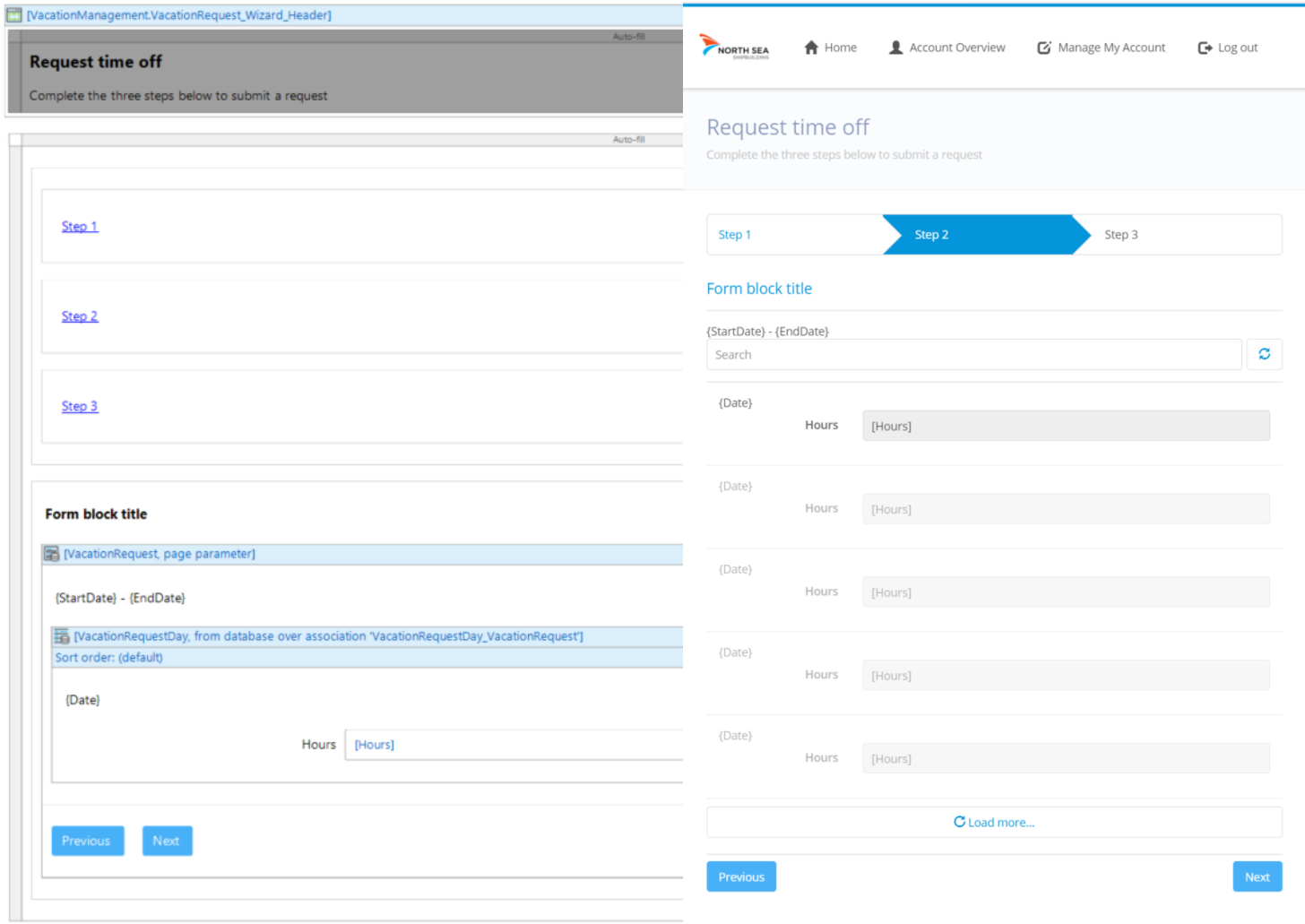


# Generating vacation days



# Adding requests: Wizard step 2

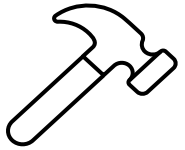
- Display requested period start and end dates
- Add a list view connected to **VacationRequestDay**
- Add a text box which allows the user to set the number of hours requested per day
- Set the Previous button to **Close page** for now




mx mendix

# Try it out!

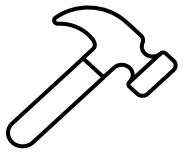
Run locally and test what you've built.



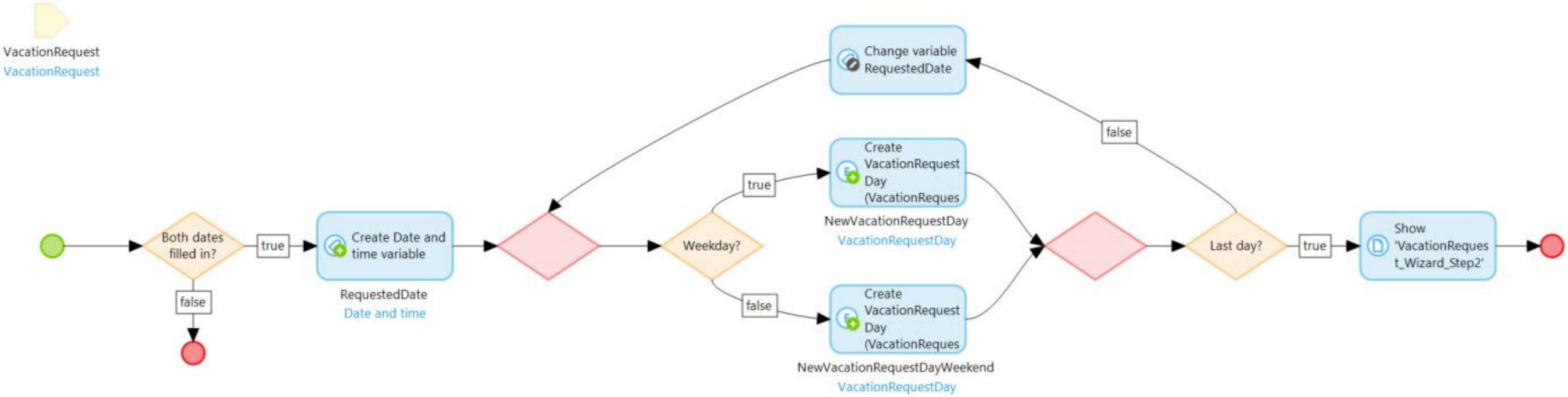
# Generating just weekday hours

Set hours requested to 0 for weekends

Hints:
Use *parseInteger()*
and *formatDateTimeUTC()*
with the *u* modifier to get a number
for day of the week



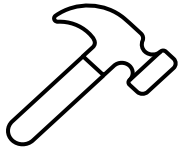
# Generating just weekday hours




mx mendix

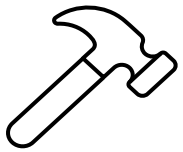
# Try it out!

Run locally and test what you've built.

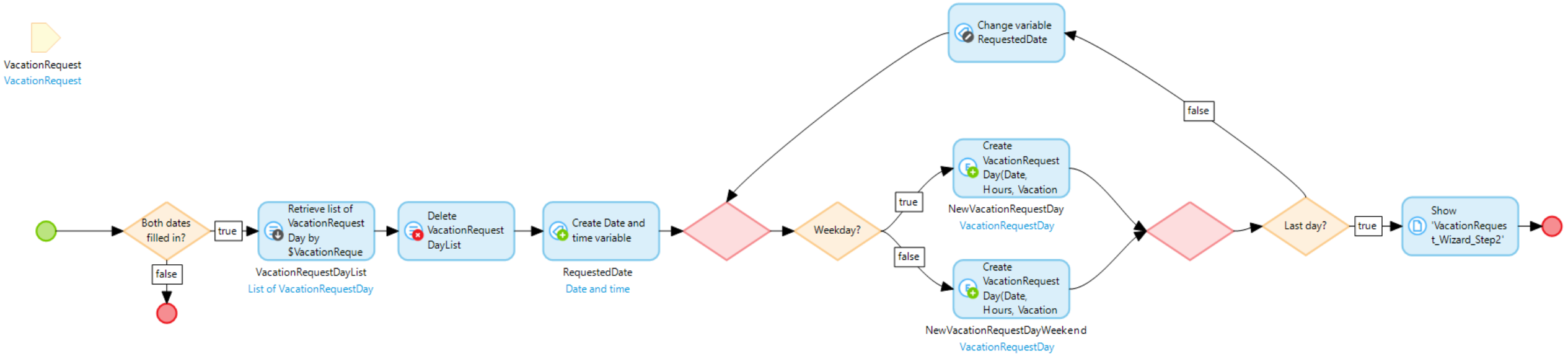


# Not generating duplicate days

Handle changes in start date and end date by deleting all existing days and generating them again



# Not generating duplicate days



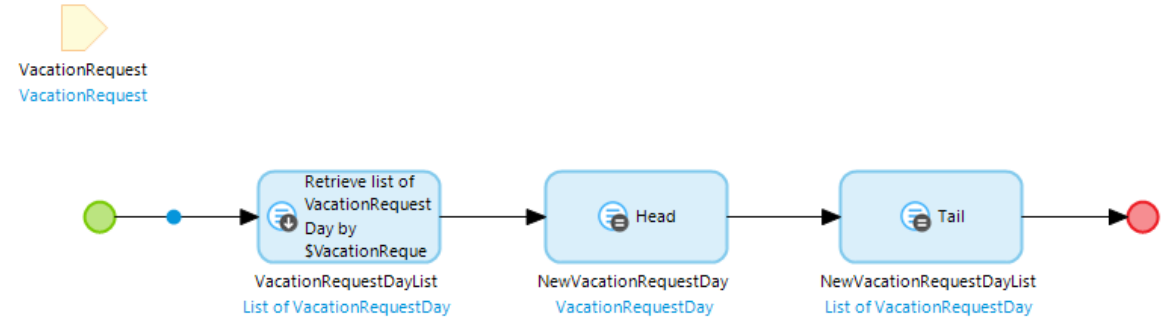

mx mendix

# Try it out!

Run locally and test what you've built.

# List operation: Unary

| Operation | Description | Return Type |
|---|---|---|
| Head | The result is the first element of the list, or empty if the parameter contains zero elements or was initialized as empty | Object |
| Tail | The result is a list containing all elements of the parameter except the first, or an empty list if the parameter contains zero elements or was initialized as empty | List |

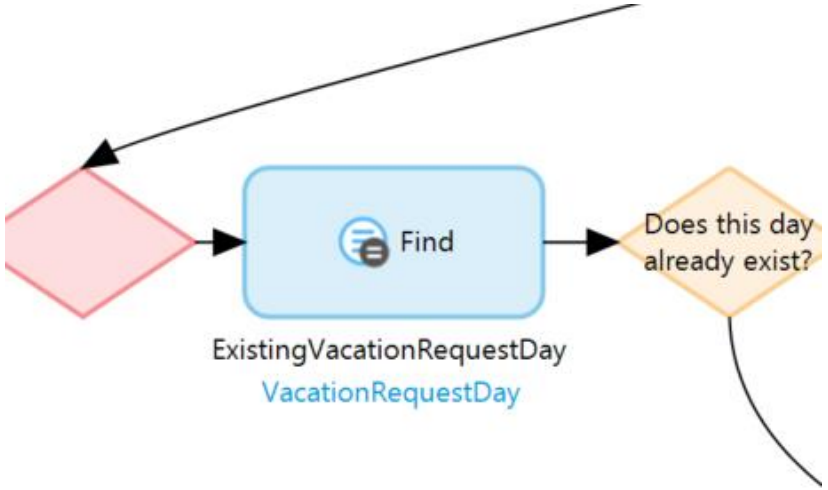


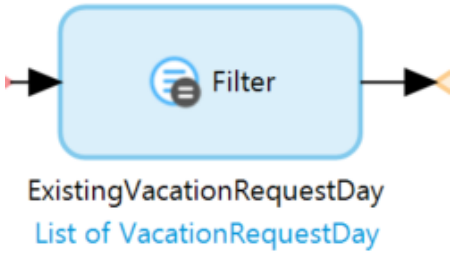
# List operation: Binary

| Operation | Description | Return Type |
| --- | --- | --- |
| Union | The result is a combination of the elements of both parameters avoiding duplicates | List |
| Intersect | The result is a list containing elements that appear in both parameters | List |
| Subtract | The result is the first parameter with the element(s) of the second parameter removed | List |
| Contains | Checks whether all elements of the second parameter are present in the first parameter. | Boolean |
| Equals | Checks whether the lists contain the same elements | Boolean |

# List operation: Member Inspections

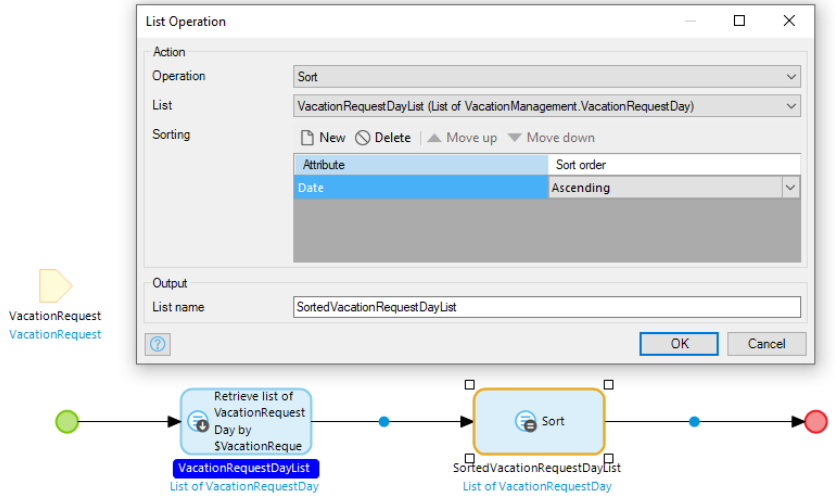
| Operation | Description | Return Type |
|---|---|---|
| Find | Find the first object of which the member has the given value | Object |
| Filter | Find all objects of which the member has the given value | List |

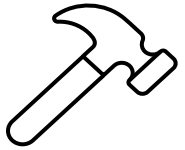


# List operation: Sort

| Operation | Description | Return Type |
| --- | --- | --- |
| Sort | Allows you to sort a list based on a number of attributes. The attributes are ordered to determine their priority while sorting. The input list remains in its original order while the sorted list is stored with the output name. | List |

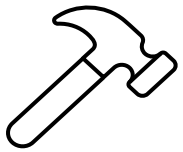




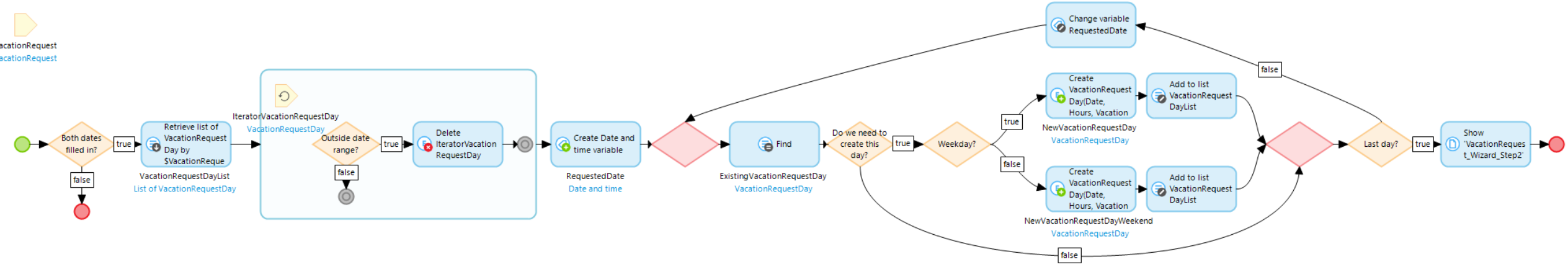
# Sort vacation days

- Generate vacation days without recreating existing ones, and sort by date when finished

- Remove days outside the date range
- Generate days in range that don’t already exist
- You can use:
  - Loops
  - Batches
  - Unary List operations
  - Binary List operations
  - Member inspections
  - Sorting



# Sort vacation days



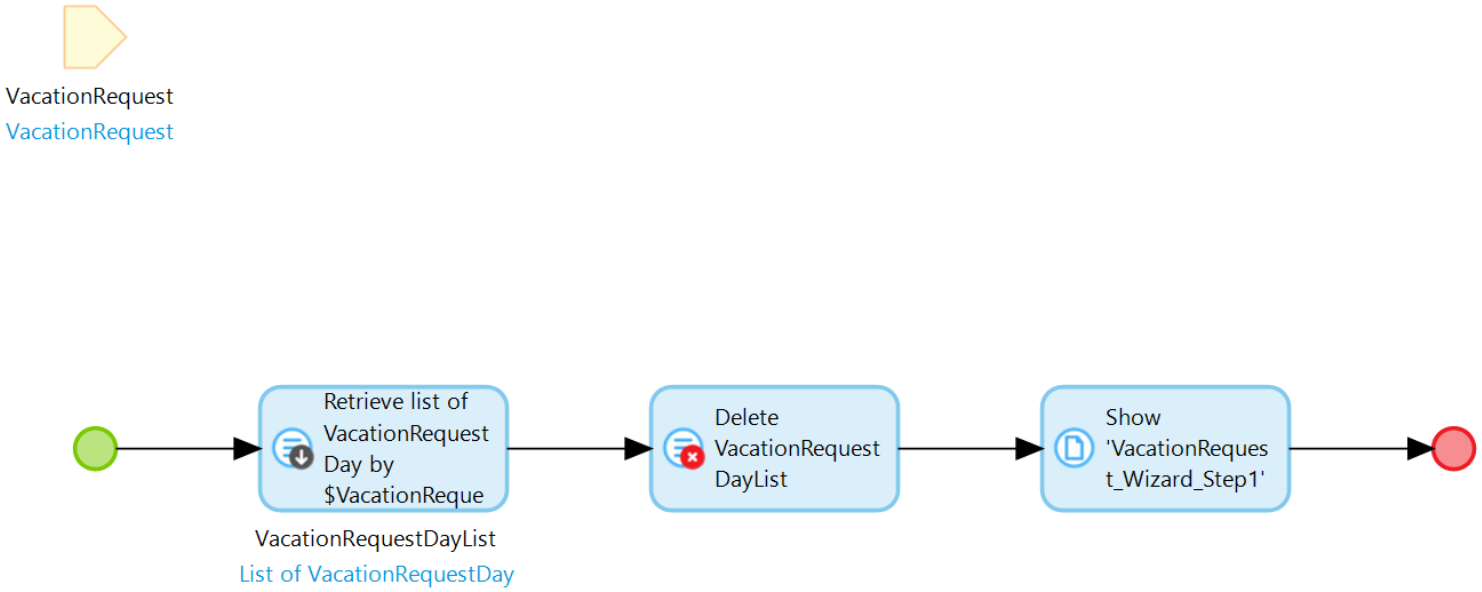

mx mendix

# Try it out!

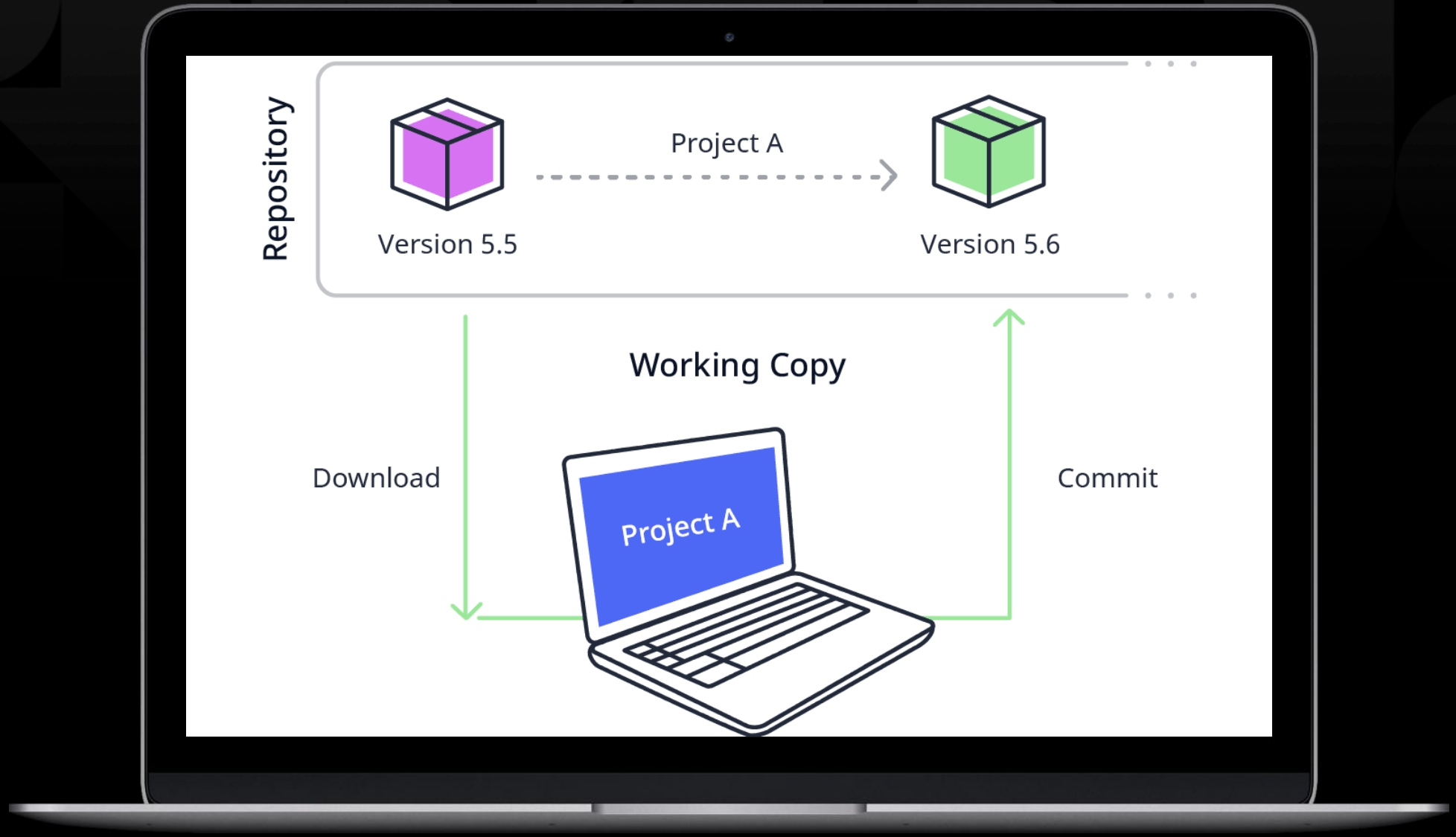
Run locally and test what you've built.

# Navigation microflows

- Previous button on Step 2
  - Retrieve list of vacation days
  - Delete list
  - Show Step 1
- Submit on Step 3
  - Similar, but:
  - Set **Submitter** association to current user's account
  - Commit DayList and VacationRequest



# Commit your work

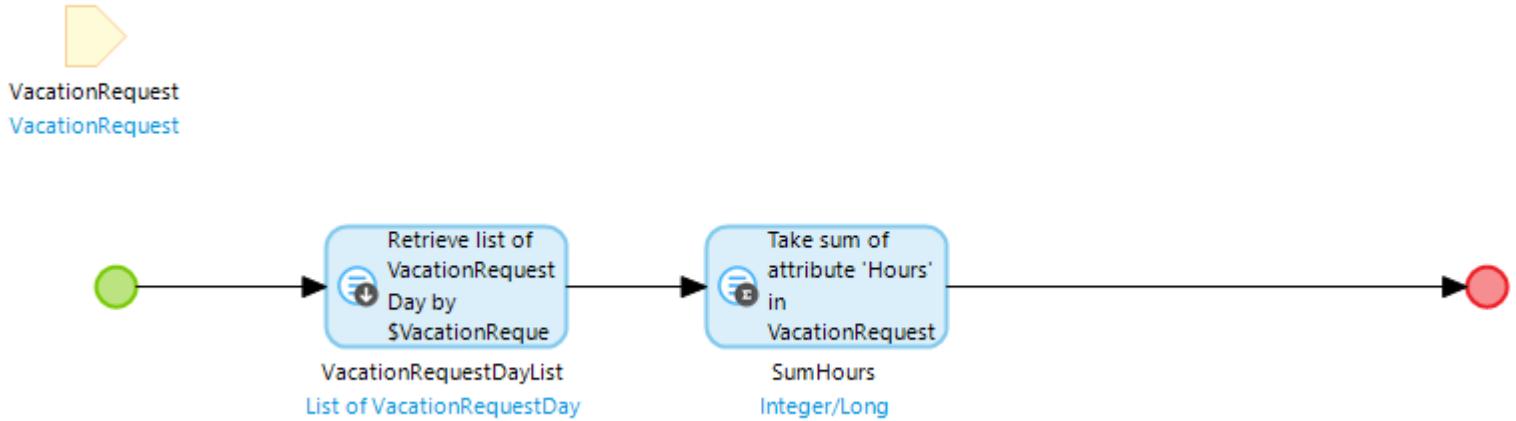


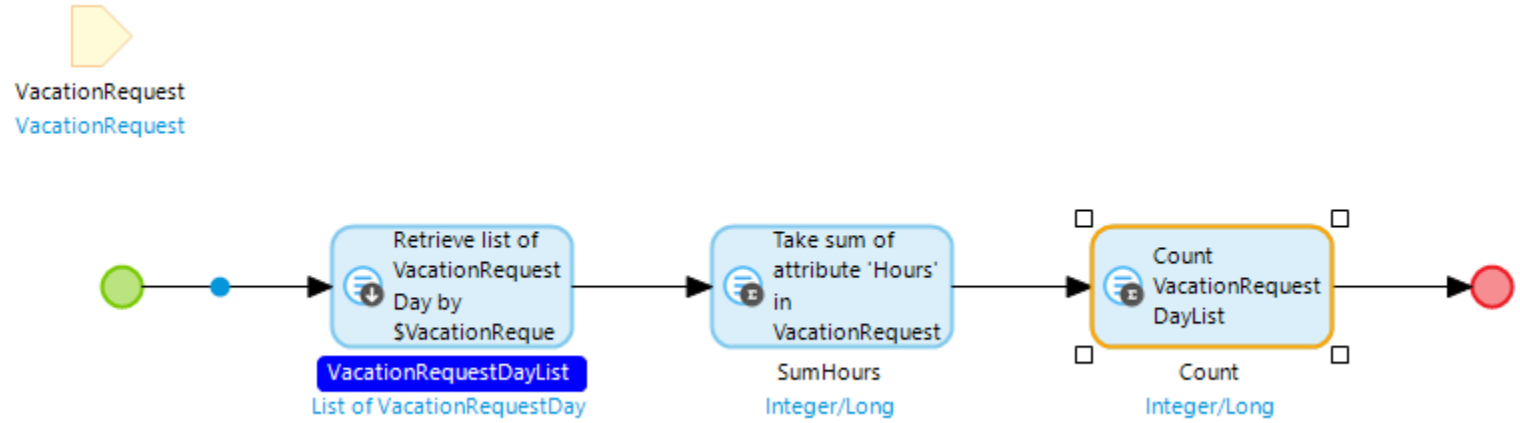
# User story

*As a user I want to see the total time-off for one request so I can determine if I have enough hours left for the year*

# Aggregate List activity

- Calculate aggregated values
  - Sum
  - Average
  - Count
  - Minimum
  - Maximum
- In combination with a retrieve → optimized query to the database

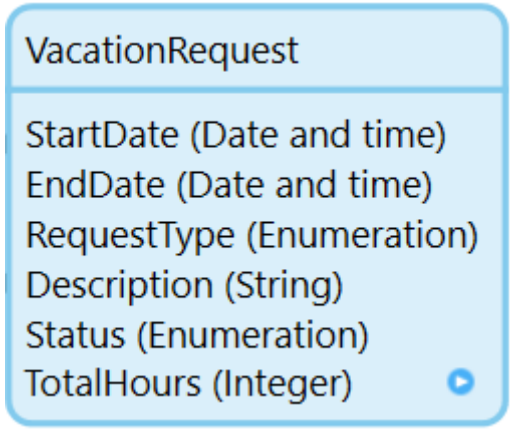



mx mendix

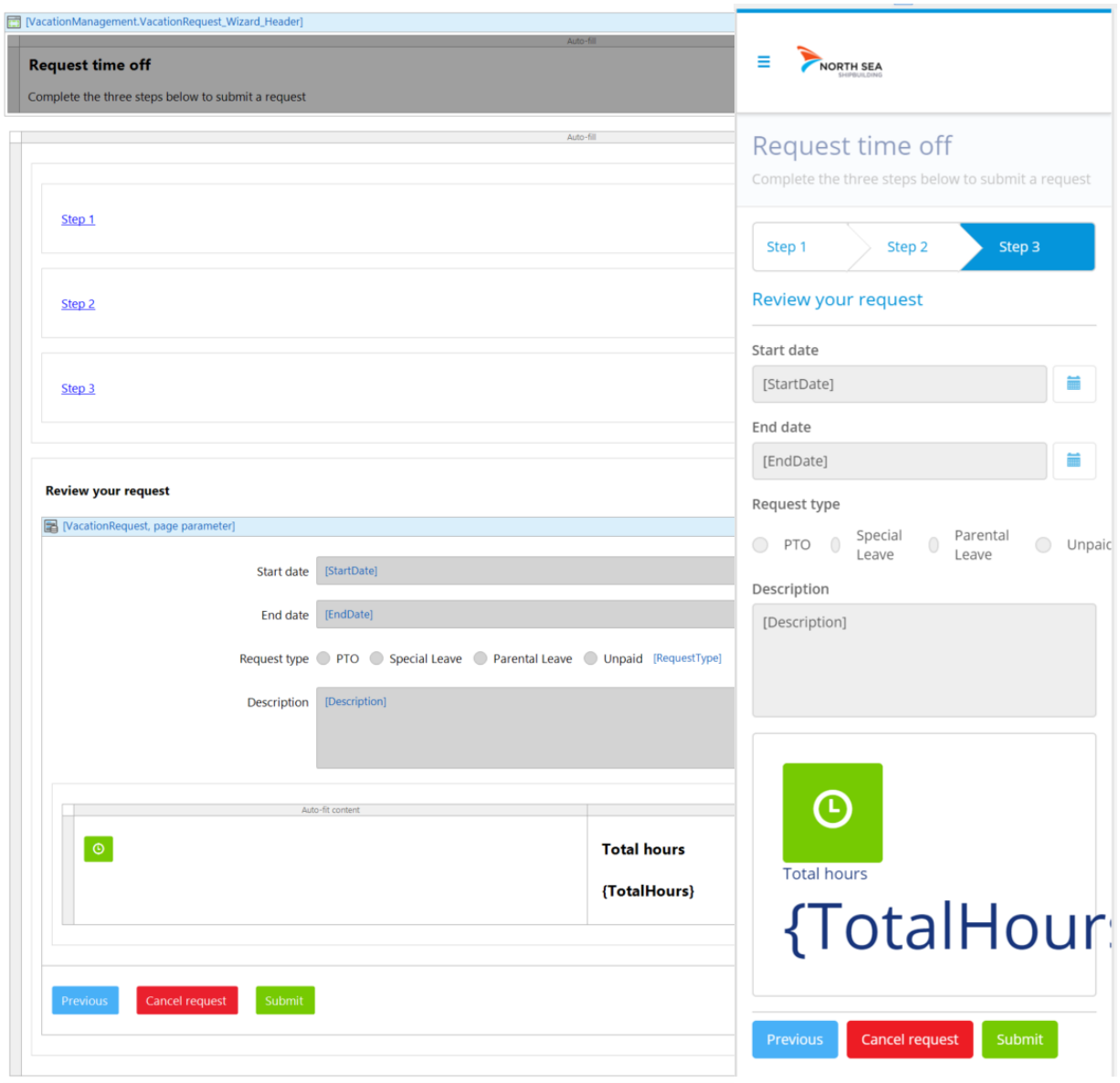
# Calculated attributes

- Calculated when accessed
- Calculation done in a MF
- Always up to date
- Performance?
- Rule of thumb



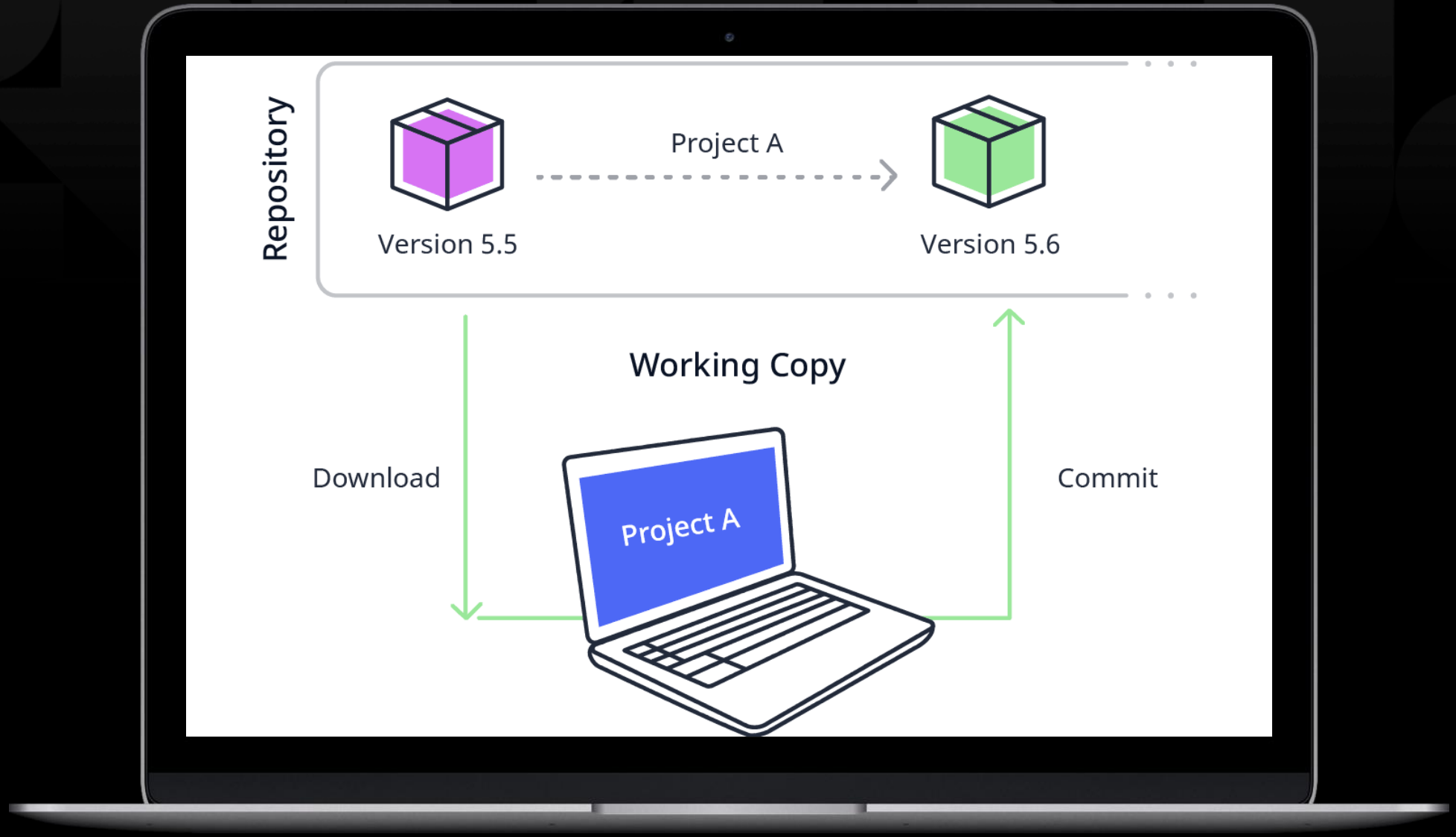
# Adding requests: Wizard step 3

- Show request for final review
- Add a **calculated attribute** to the VacationRequest entity: **TotalHoursRequested**
- Create the microflow that will calculate the value (prefix **CAL**)
- Match the page design
- Make navigation buttons functional; take validation into consideration




mx mendix

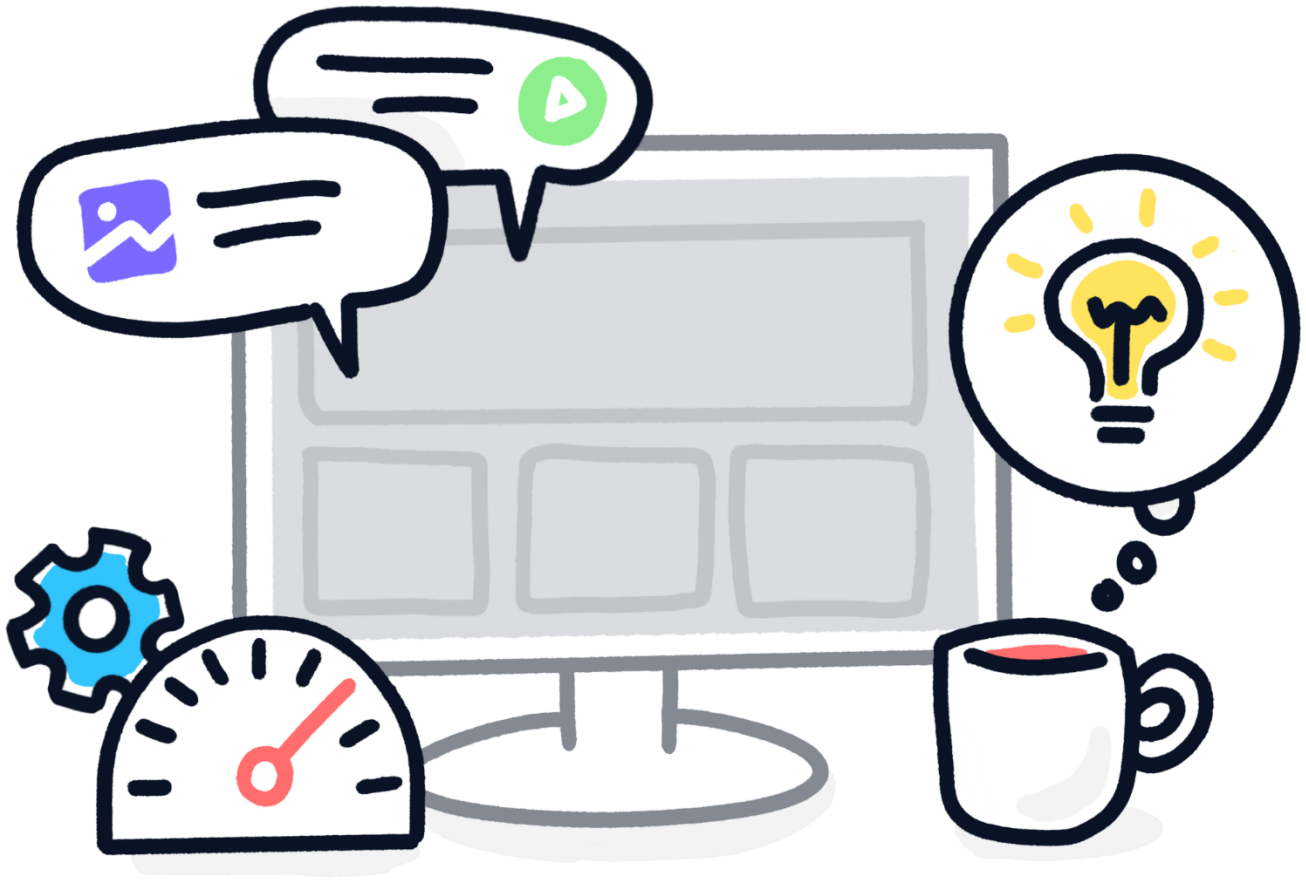
# Commit your work



# Approving requests

# Learning goals

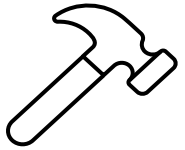
- How to create tabbed pages in Mendix
- How to create popup pages in Mendix



mx mendix

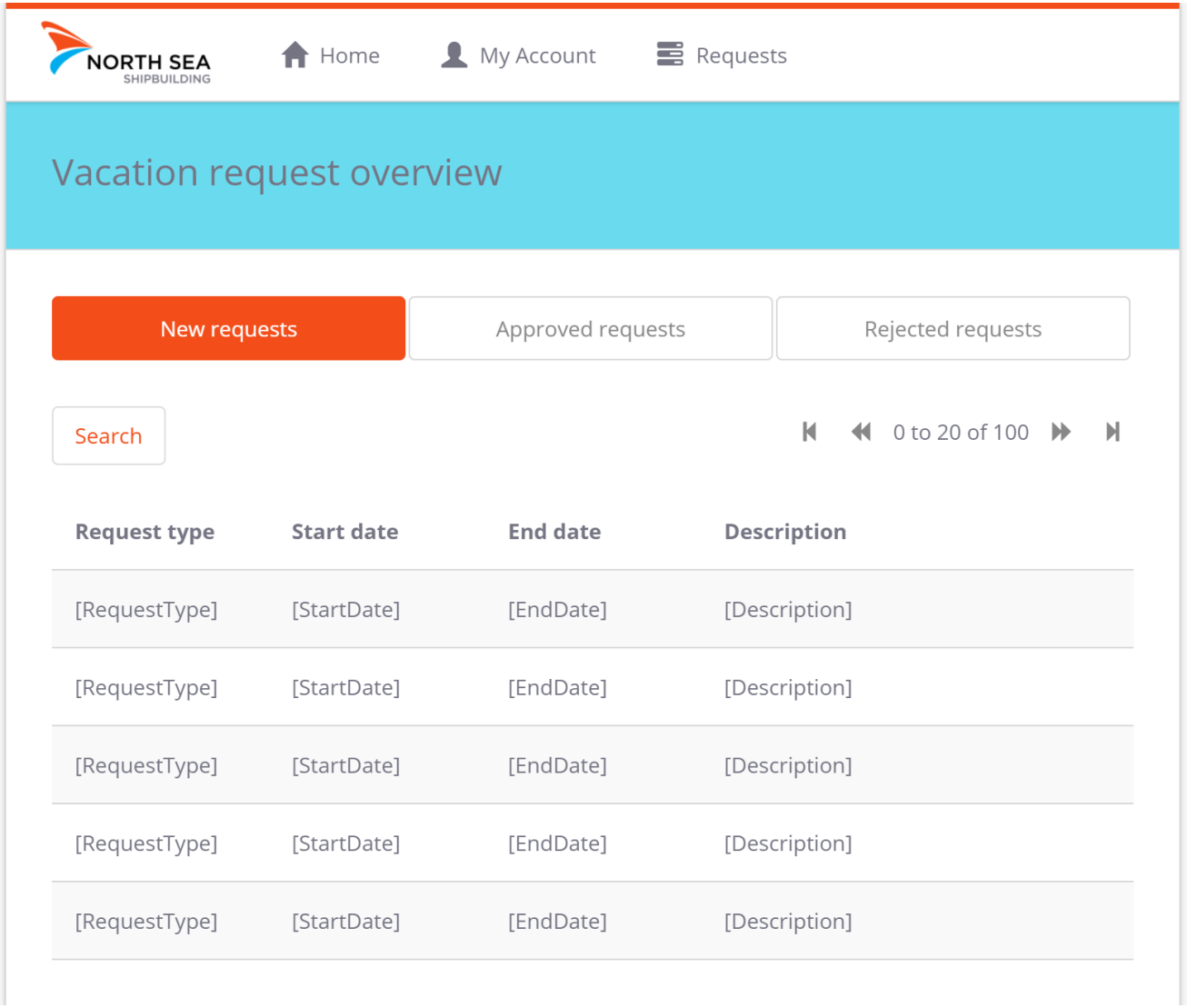
# User story

*As a manager I want to see all requests of my employees in relevant tabs so I have a good overview of my team's time-off requests*



# Tabbed overview page

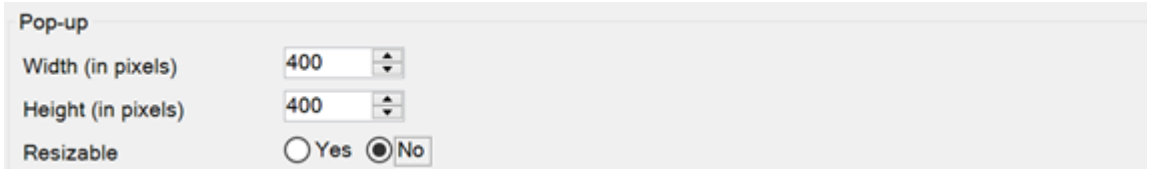
- Create a new page **VacationRequest_Overview** using the **Tabs full width** page template
- Match the design, showing the requests with the correct status on each tab
- Set the **New requests** tab to the Default tab
- Give the **Manager** access to the page and add it to **Navigation**



# Working with popups

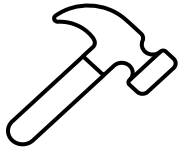
## Resizable popups

- By default: popup windows in Mendix are resizable
- It’s possible to give a popup window a fixed size



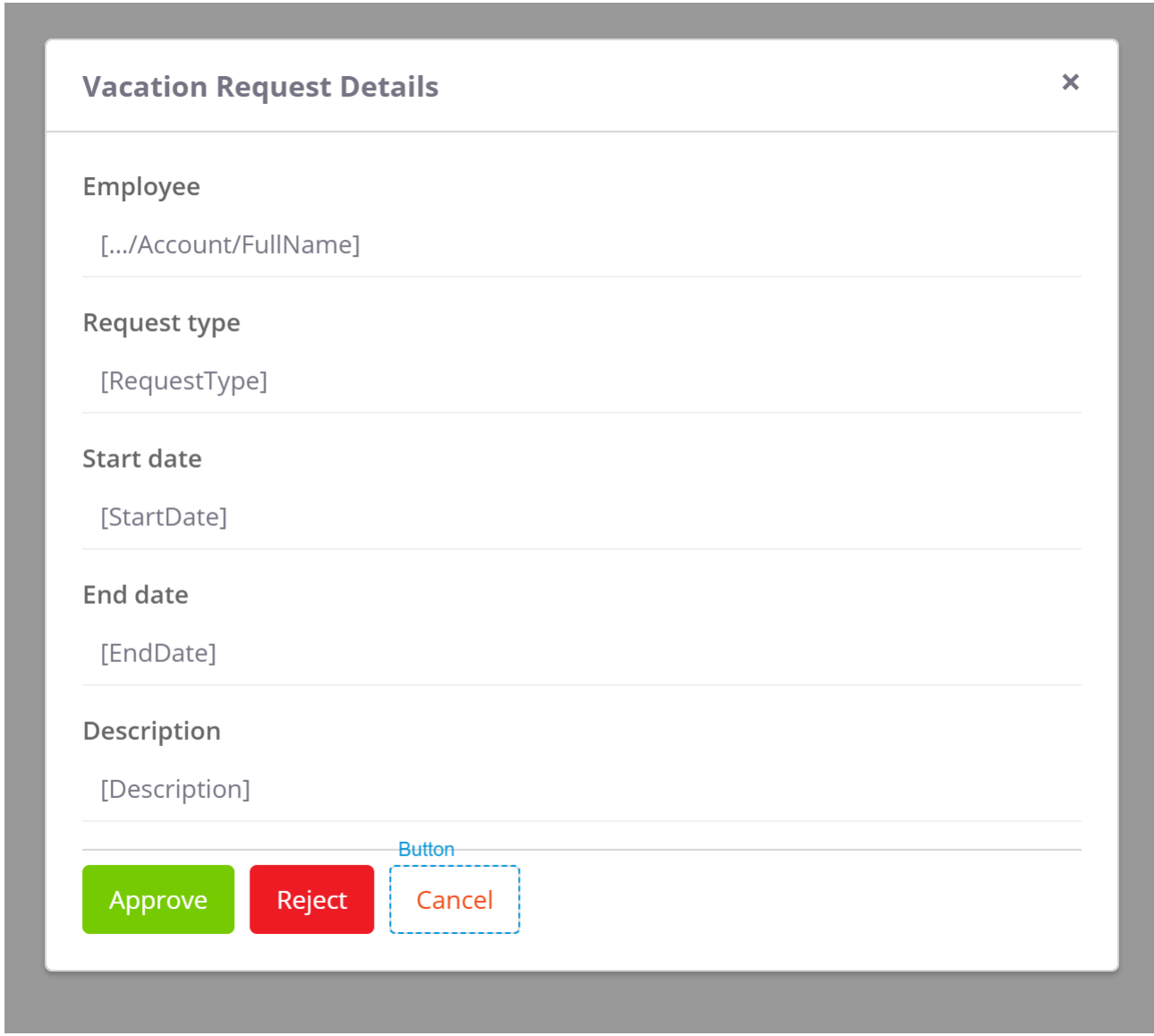

## Popup footer

- Sticky footer vs scrollable area

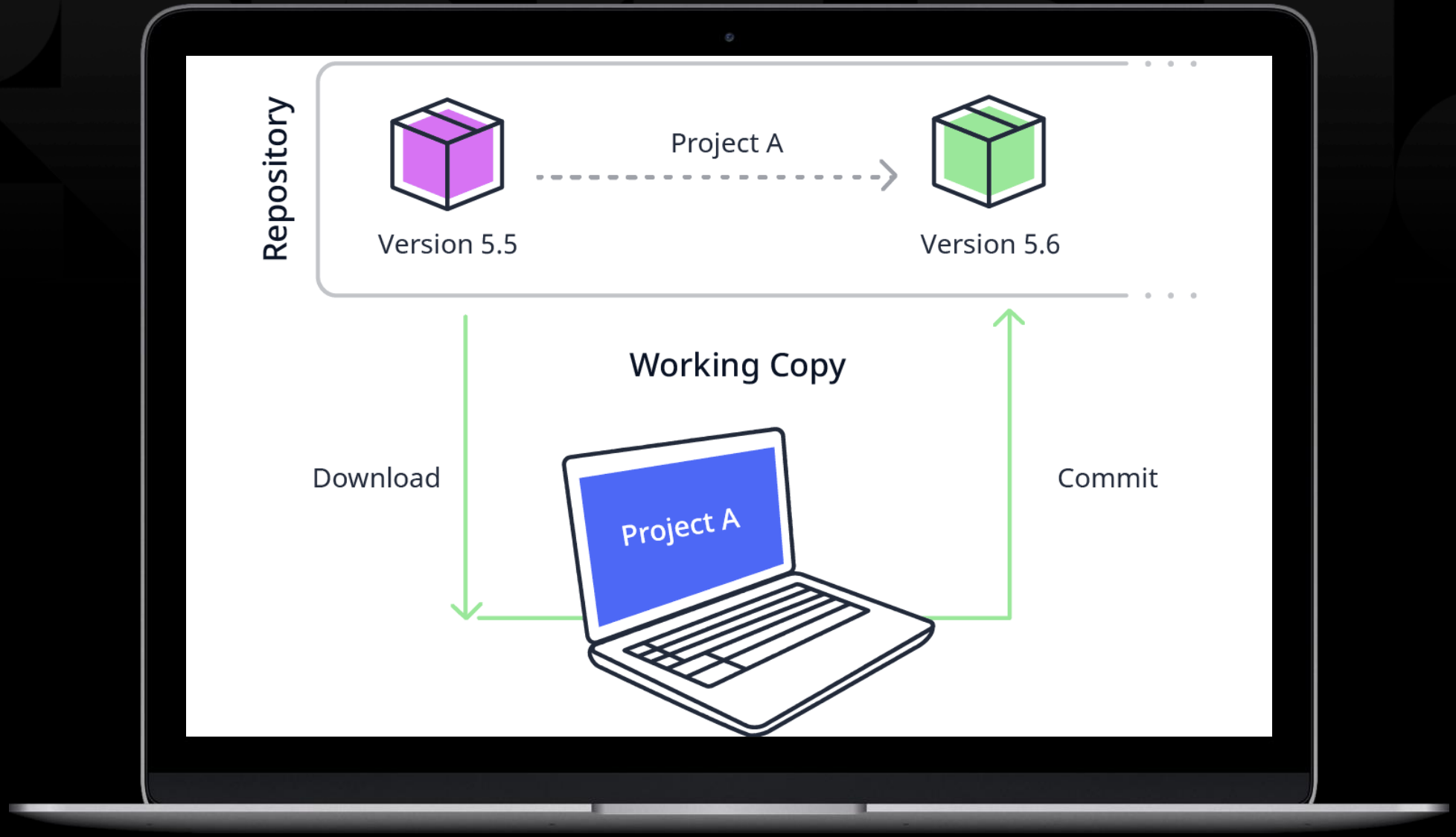


# Approve a request

- Add a **Details** button to the **New requests** data grid
  - Set it to default button
  - Primary button style
- Make the button open a new **popup page**: VacationRequest_Details_Manager
- Give the popup a fixed size;
  - Height 800px
  - Width 500px
- Match the design
- Add buttons and microflows that **approve** or **reject** the request



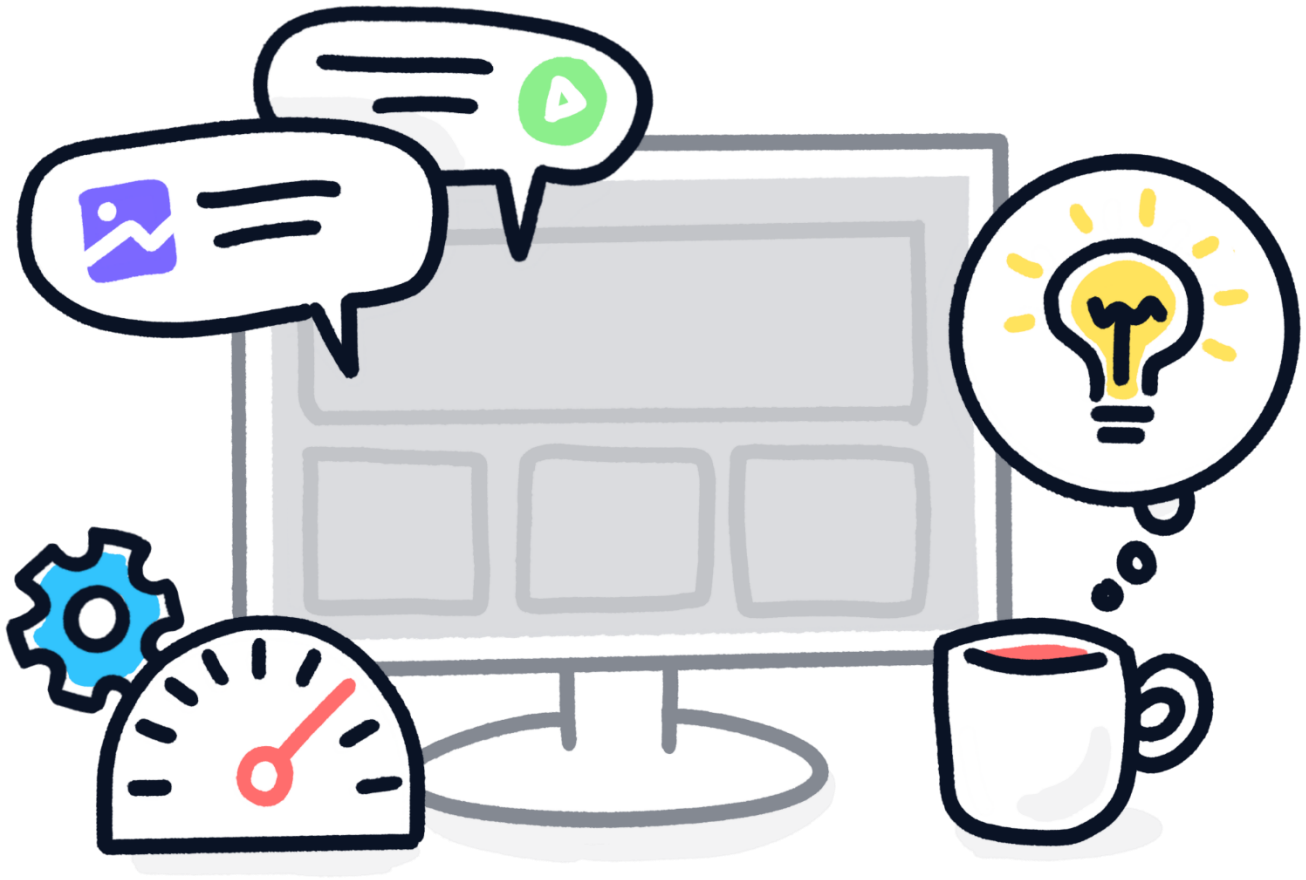
# Commit your work



# Notifications

# Learning goals

In this module we will use the knowledge we gained during this course to build a notification system for our Vacation Request app



# User story

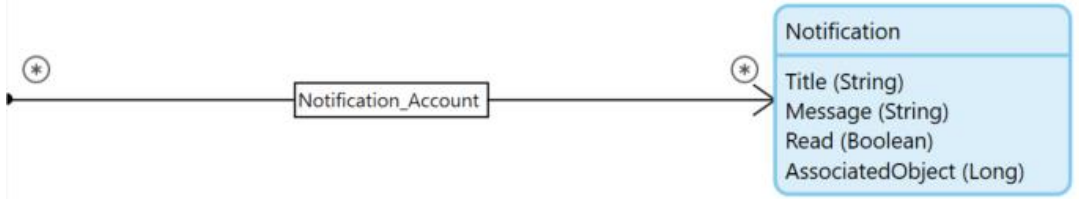
*As a user I want to be notified when the status of my request changes so I can take the appropriate action.*

Directionality is not correct, the text is leading, the screenshot is wrong.

No link to the actual request, needs to be added

# Building the basics

- Add a new module, called **Notifications**
- Add the **Notification** entity to the Domain Model
- Add a *-* association to the **Account** entity
- Extend the **Request** entity with a new attribute:
  - **RequestID**, type **AutoNumber**



# Setup security

In the **Notifications** module

- Add a new module role: **User**

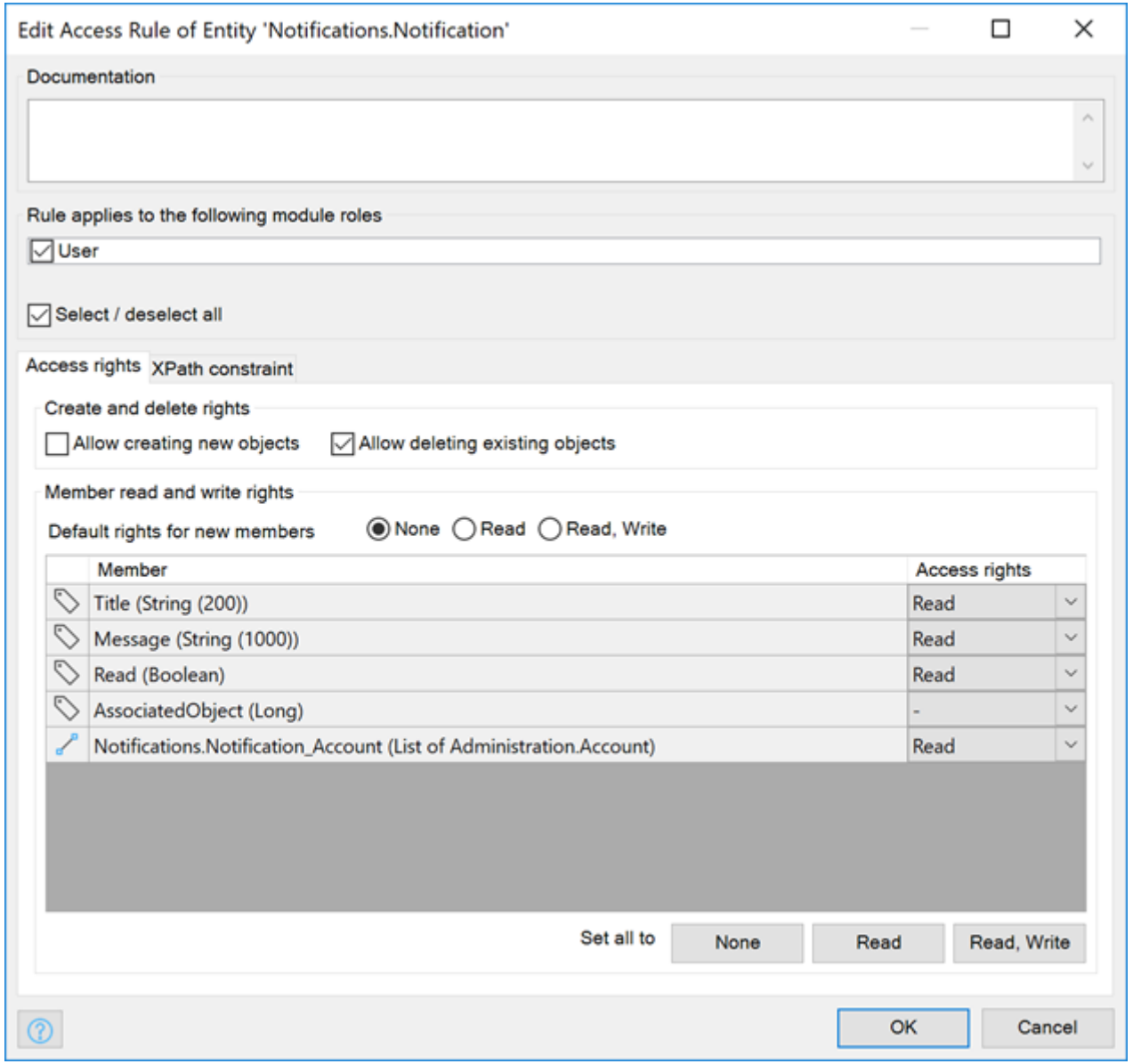
- Add entity access to the **Notification** entity
  - In the **XPath constraint** tab use the **Path to User** button

In the VacationManagement module

- Give the **Employee Read** access to the **RequestID** attribute

In Project security

- Connect the **Employee** and **Manager** user roles to the **User** module role of the **Notifications** module



# Create notifications

- When a request is created by an **Emloyee**, all **Managers** should receive a notification.
- When a request is approved by a **Manager**, the **Employee** that created the request should receive a notification.
- When a request is rejected by a **Manager**, the **Employee** that created the request should receive a notification. In this case, the **Manager** should also be able to add a message, as to why the request has been rejected.

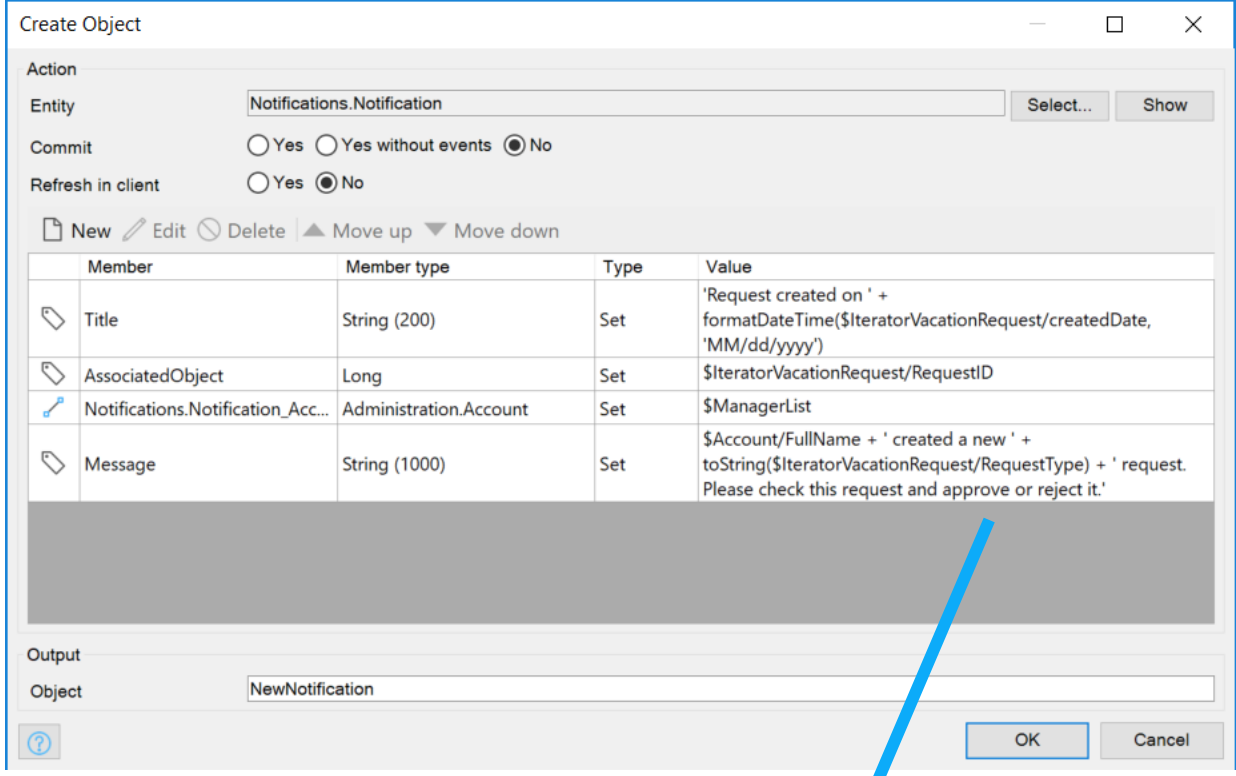
# Creation notification

- Open the microflow you connected to the **Submit request** button on the last page of the wizard

- Extend the microflow:
  - Retrieve all **Managers** from the database

  *[System.UserRoles = '[%UserRole_Manager%]']*

  - Create a new **Notification** object for each **VacationRequest**
  - Use the image to configure the **Create object** activity
  - Commit the **Notification(s)** and **Refresh the client**




Value

'Request created on ' + formatDateTime($IteratorVacationRequest/createdDate, 'MM/dd/yyyy')

$IteratorVacationRequest/RequestID

$ManagerList

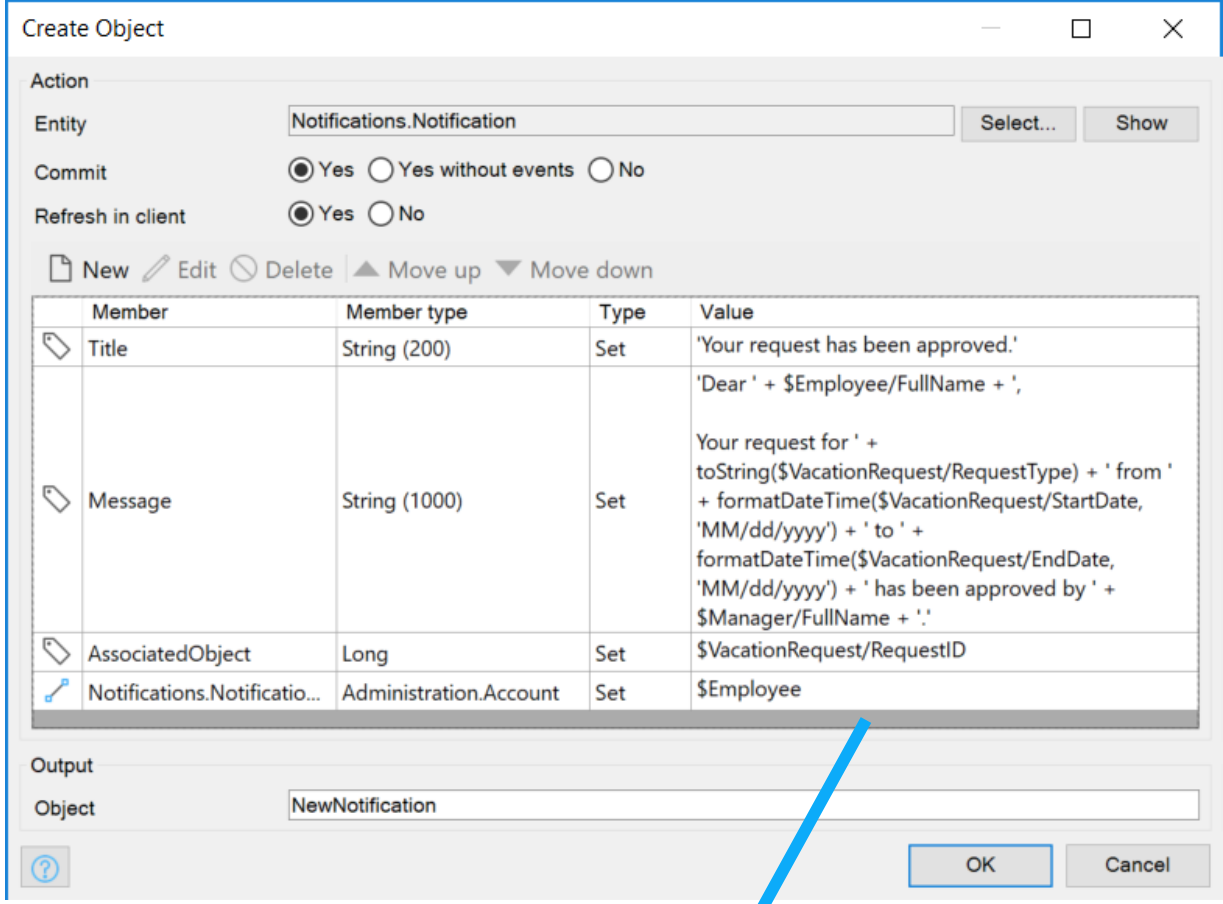
$Account/FullName + ' created a new ' + toString($IteratorVacationRequest/RequestType) + ' request. Please check this request and approve or reject it.'

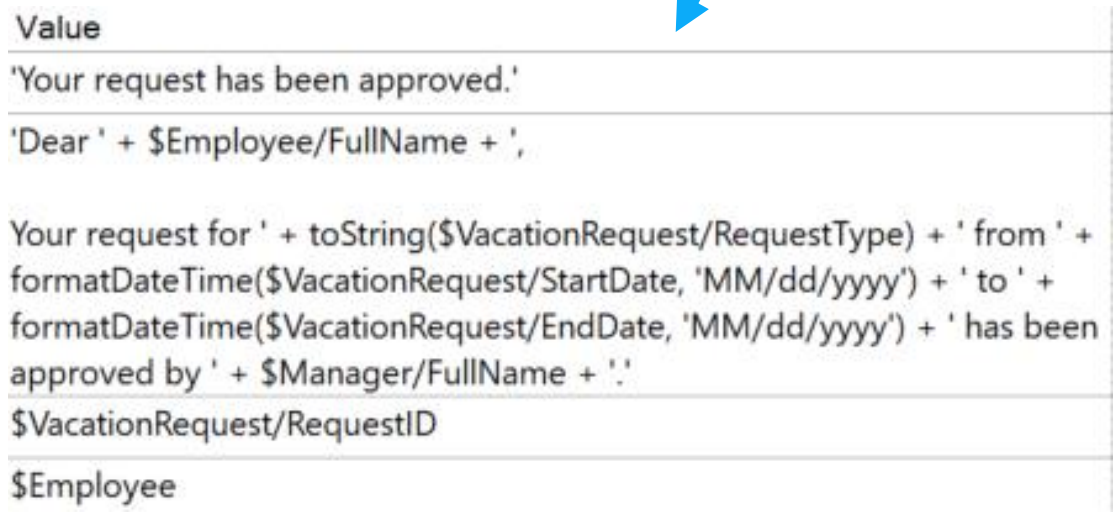
mendix

# Approval notification

- Open the microflow to approve requests
- Extend the microflow:
  - Retrieve the **Account** of the **Employee** that submitted the request.
  - Retrieve the **Account** of the **Manager** who is approving the request

  - Create a new **Notification** object
  - Use the image to configure the **Create object** activity

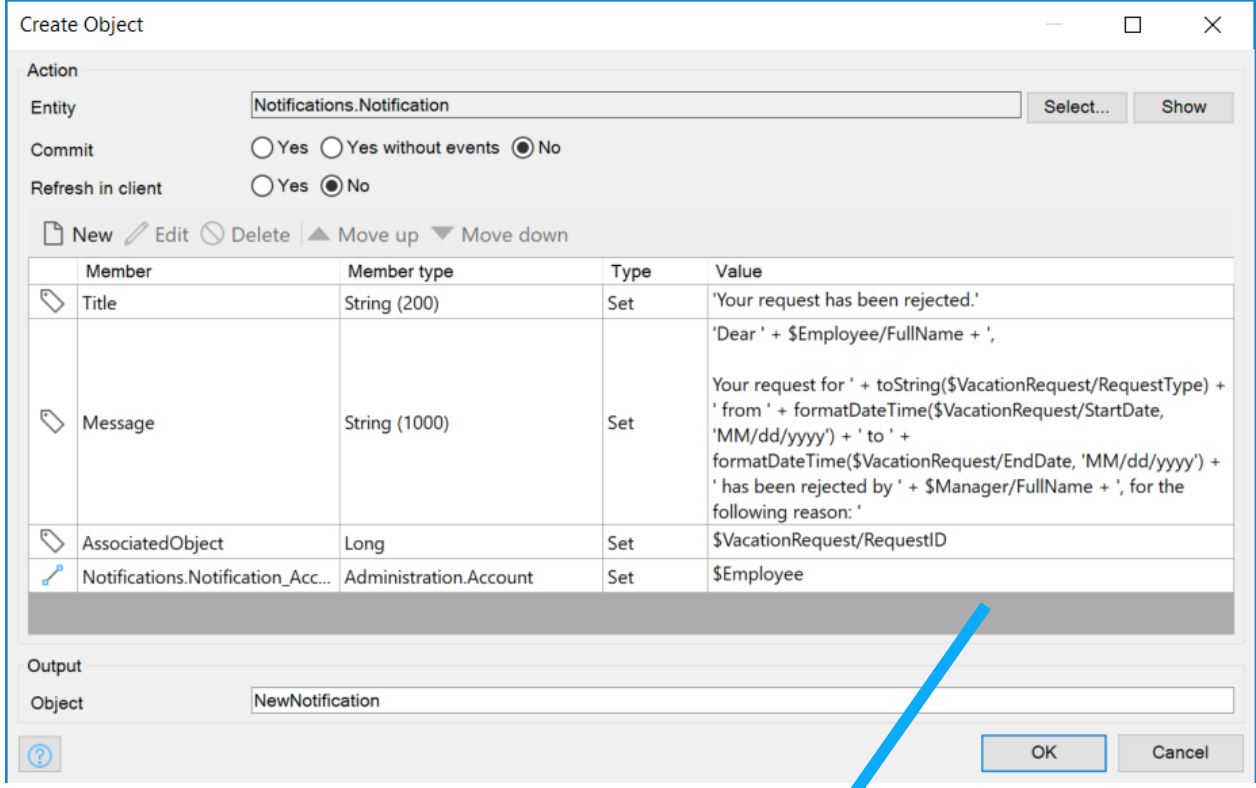


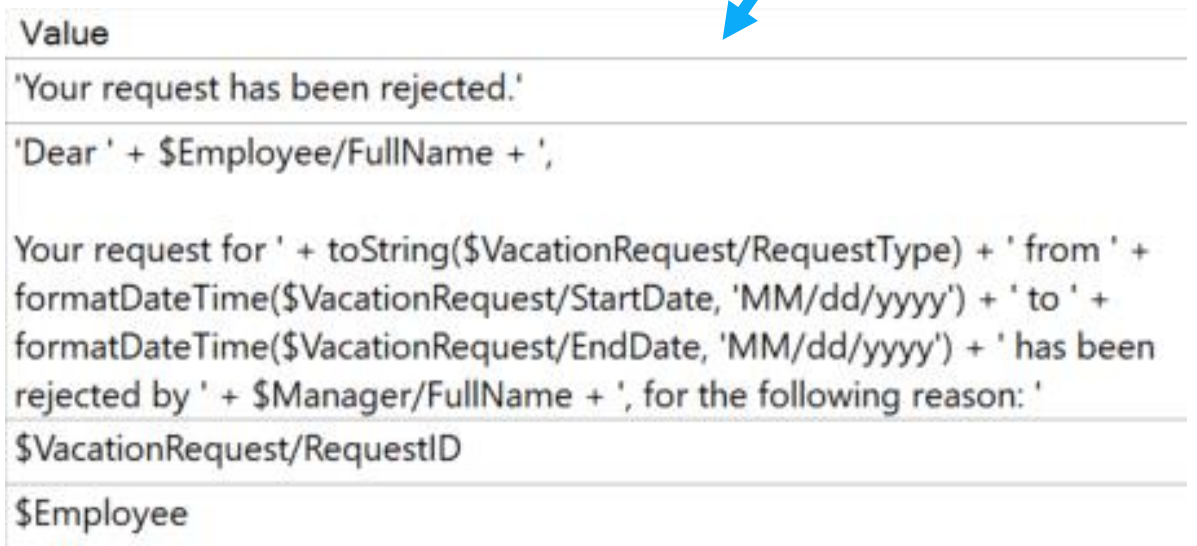




mx mendix

# Reject notification

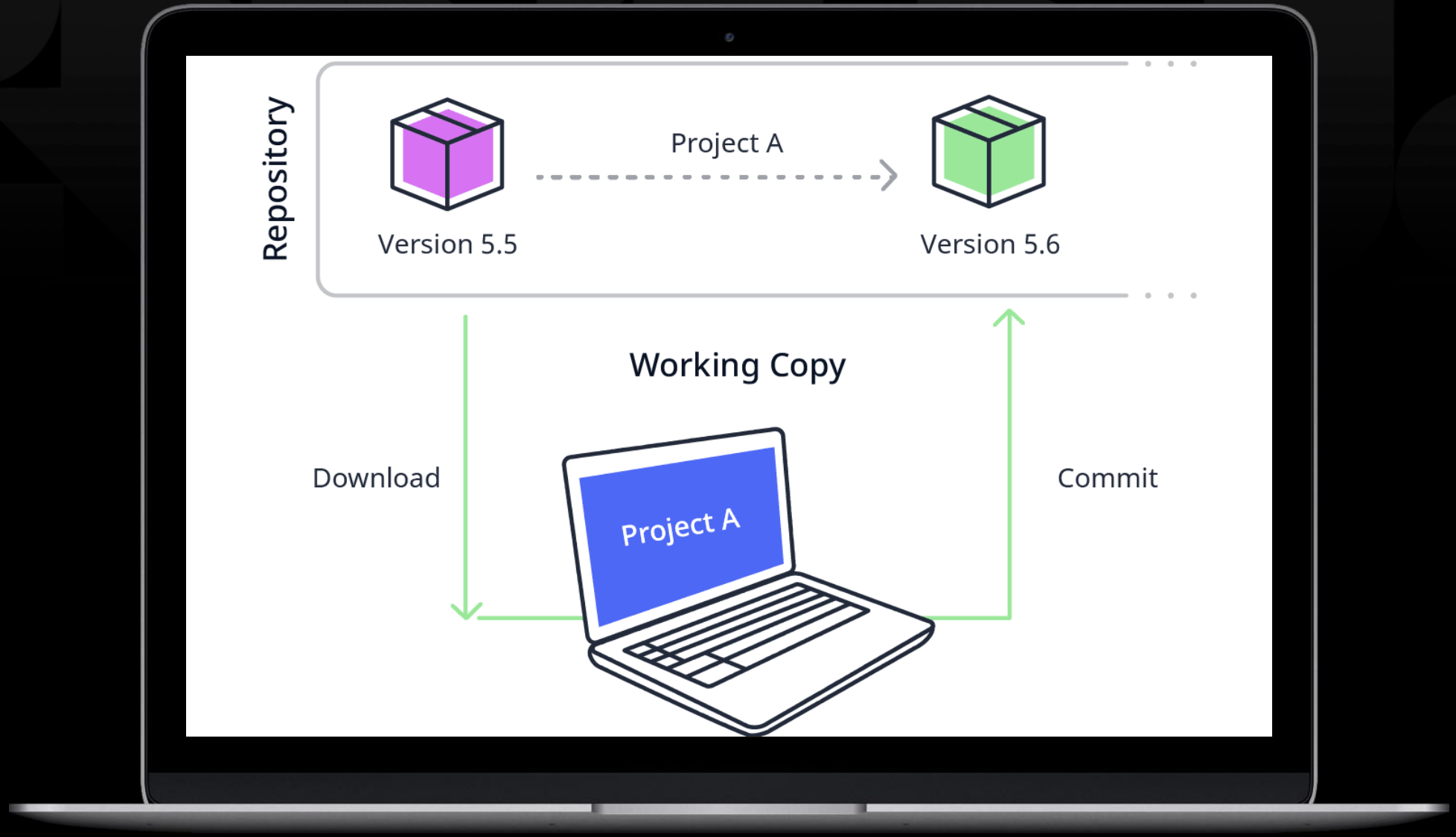
- Open the microflow to reject requests
- Extend the microflow:
  - Retrieve the **Account** of the **Employee** that submitted the request.
  - Retrieve the **Account** of the **Manager** who is approving the request
  - Create a new **Notification** object
  - Use the image to configure the **Create object** activity
- At the end of the microflow add a **Show page** activity, where the **Manager** can add a reason for rejection:




mx mendix

# Commit your work

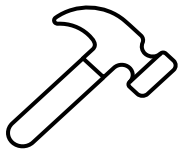


# User story

*As a user I want an easy way to access my notifications so I don't have to search for them*

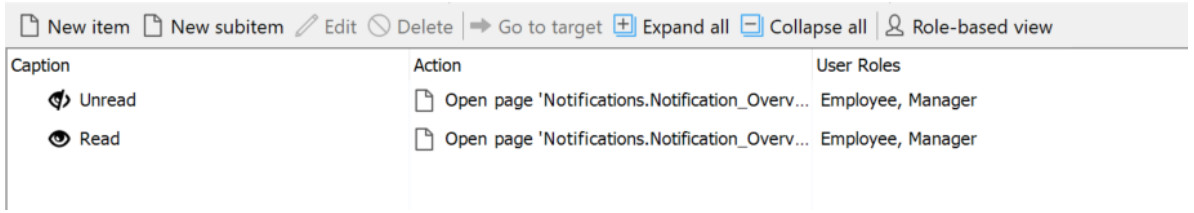
# Custom navigation menu

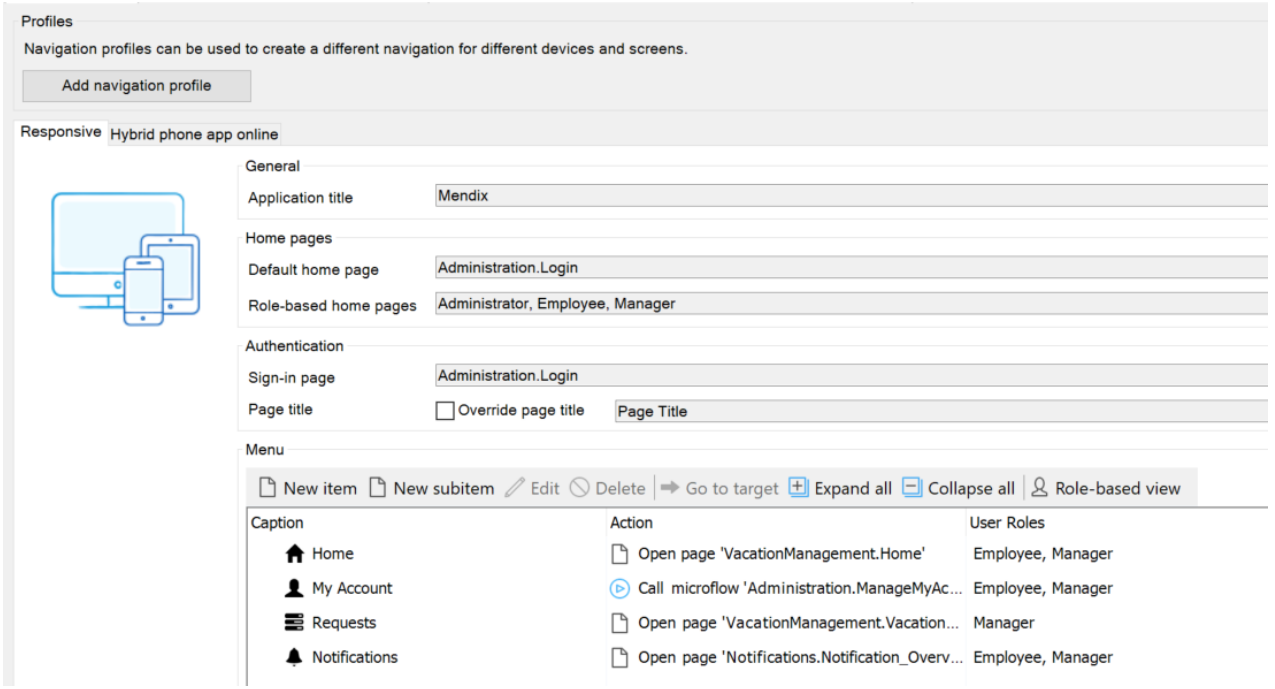
- Use **Menu** document to display a different menu then the main navigation tree.
- Can be used by a **menu widget**
- Mostly used for auxiliary menus, e.g. a side bar
- Can have multiple levels
- Can be displayed anywhere in the app



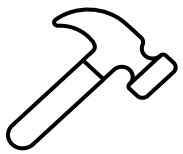
# Build the notifications navigation

- Create two new pages (**Blank** template)
  - One for unread notifications
  - One for read notifications
- Create a **Notifications** menu, with links to the two pages
- Add the unread notifications page to the main navigation menu as well
- Update security



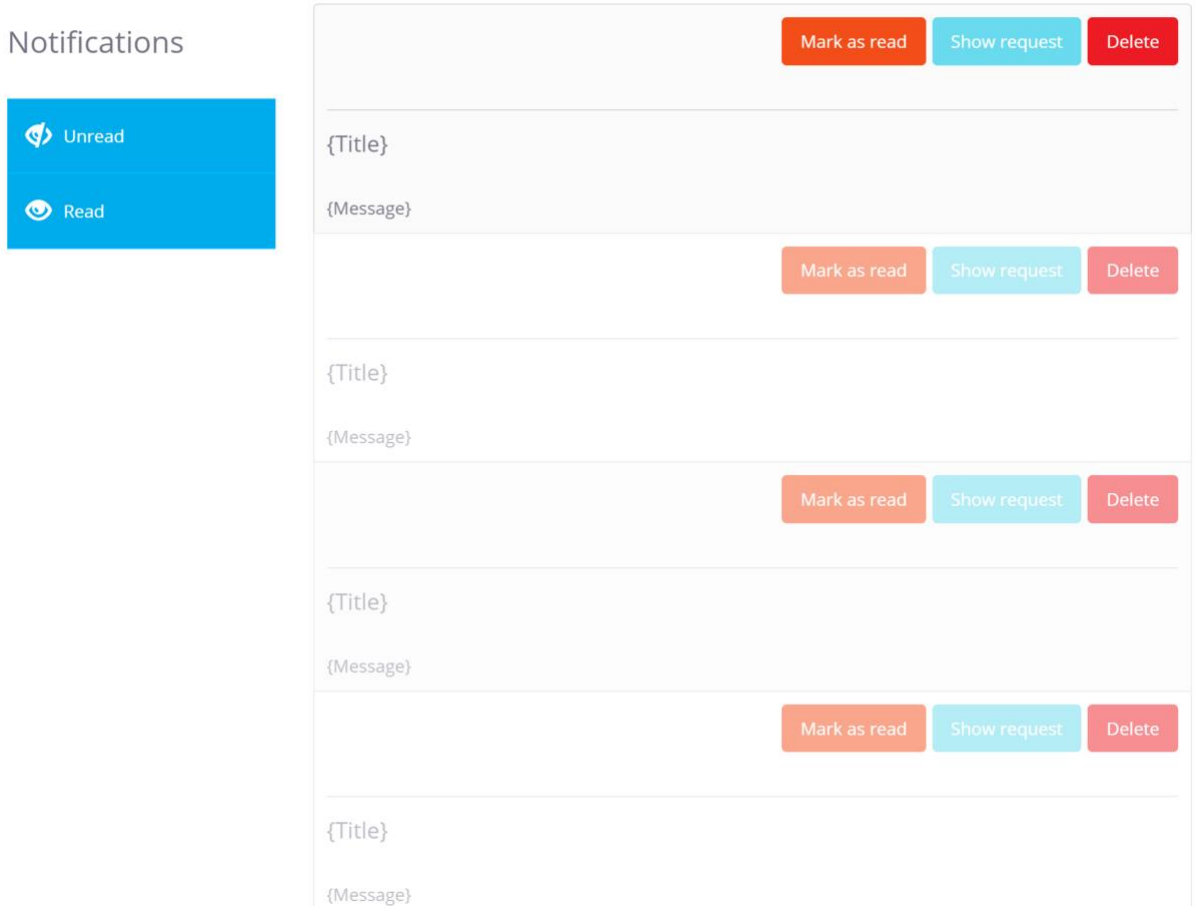




mx mendix

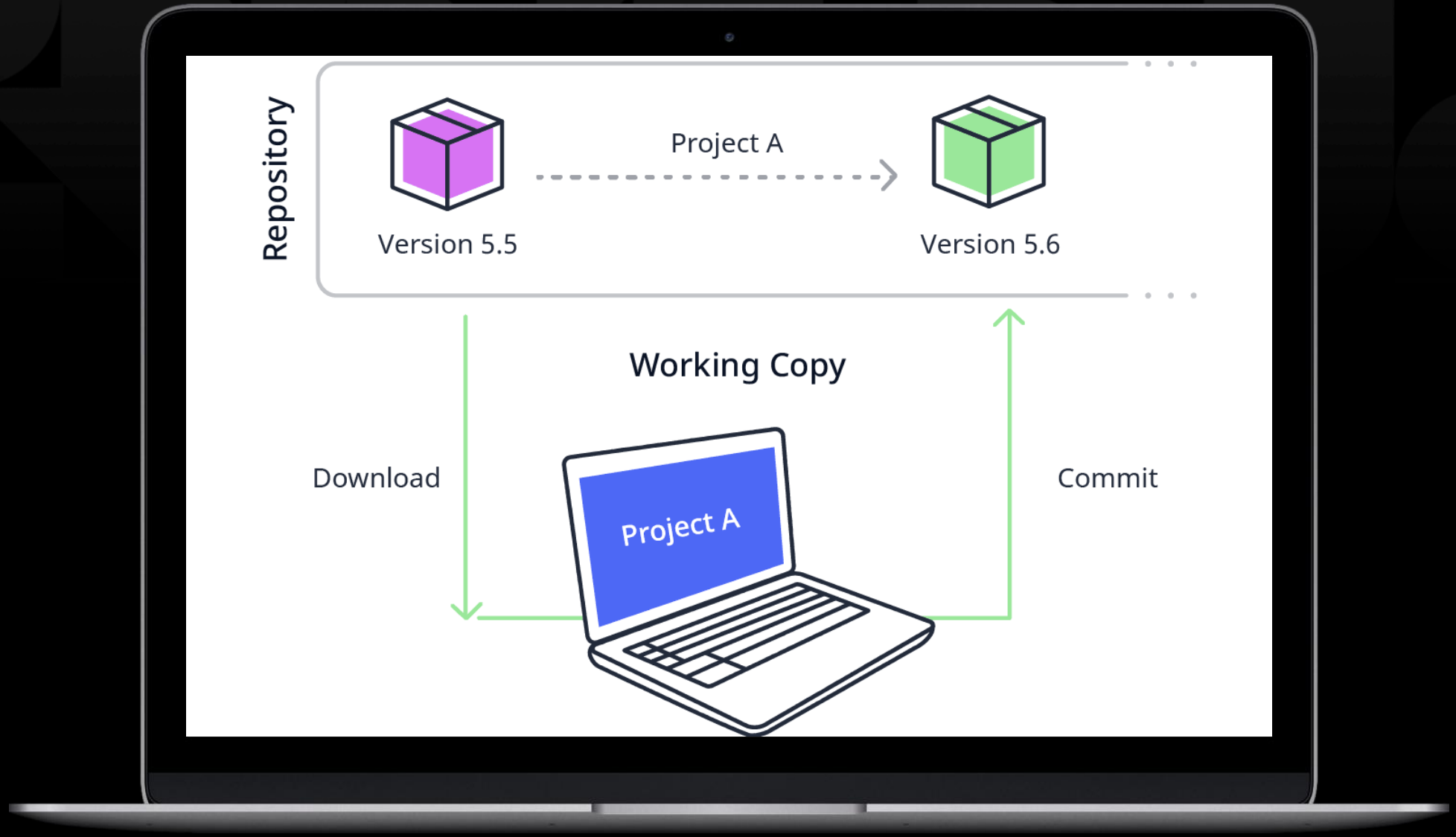


# Build the notification pages

- Using the **layout grid**, build out the two pages
  - Read should show a list view of read notifications for this user
  - Unread should show a list view of unread notifications for this user
- On the unread notifications page, use the **Pageheader controls** building block to add 3 buttons
- Where necessary, build microflows for these buttons
- Match the design



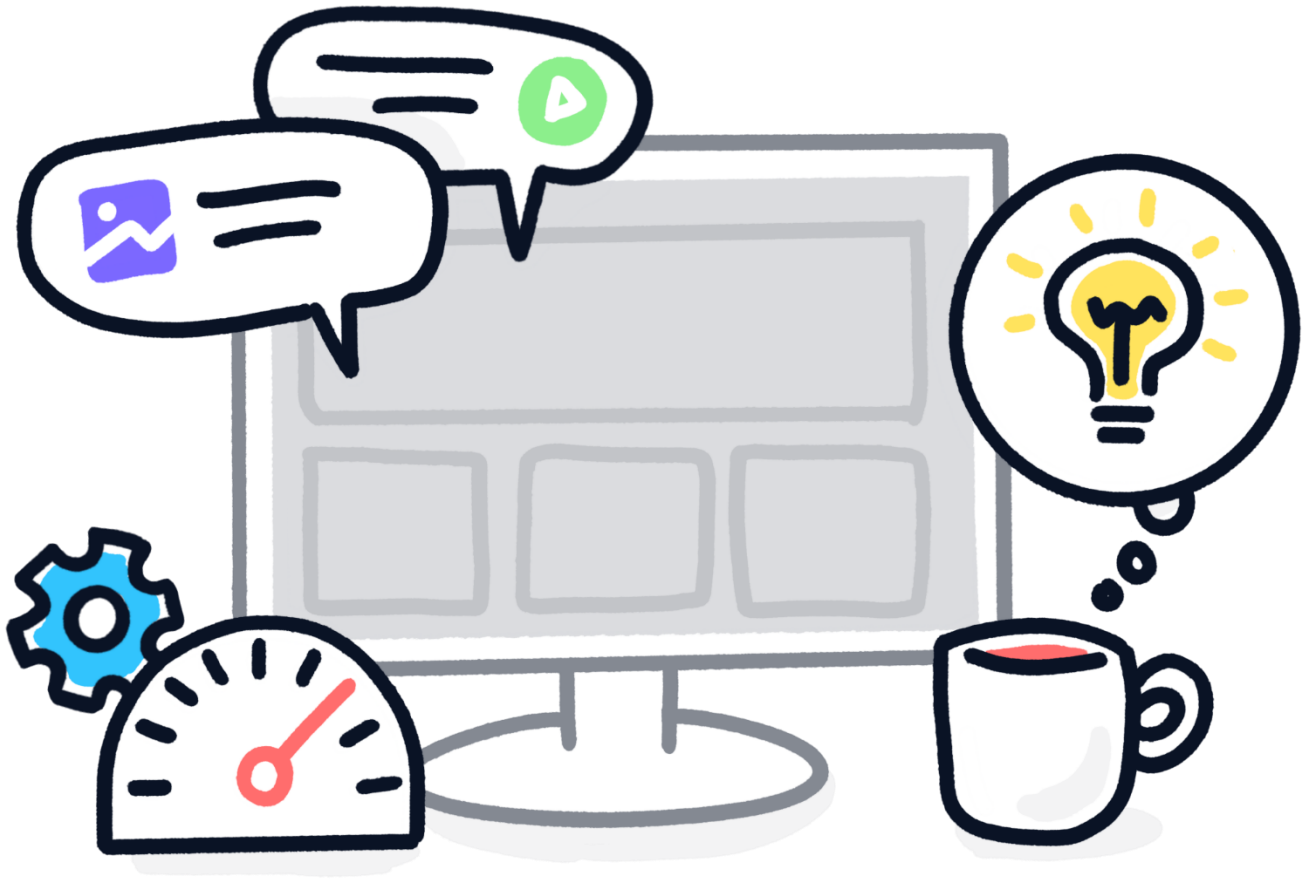
# Commit your work



# Translating the app

# Learning goals

- How to translate your app



# User story

*As a user I want to use the app in my own language, so I don’t misunderstand the labels in the app.*

# Language settings

- Project language
- Development language
- Translatable texts

  <Vacation requests>

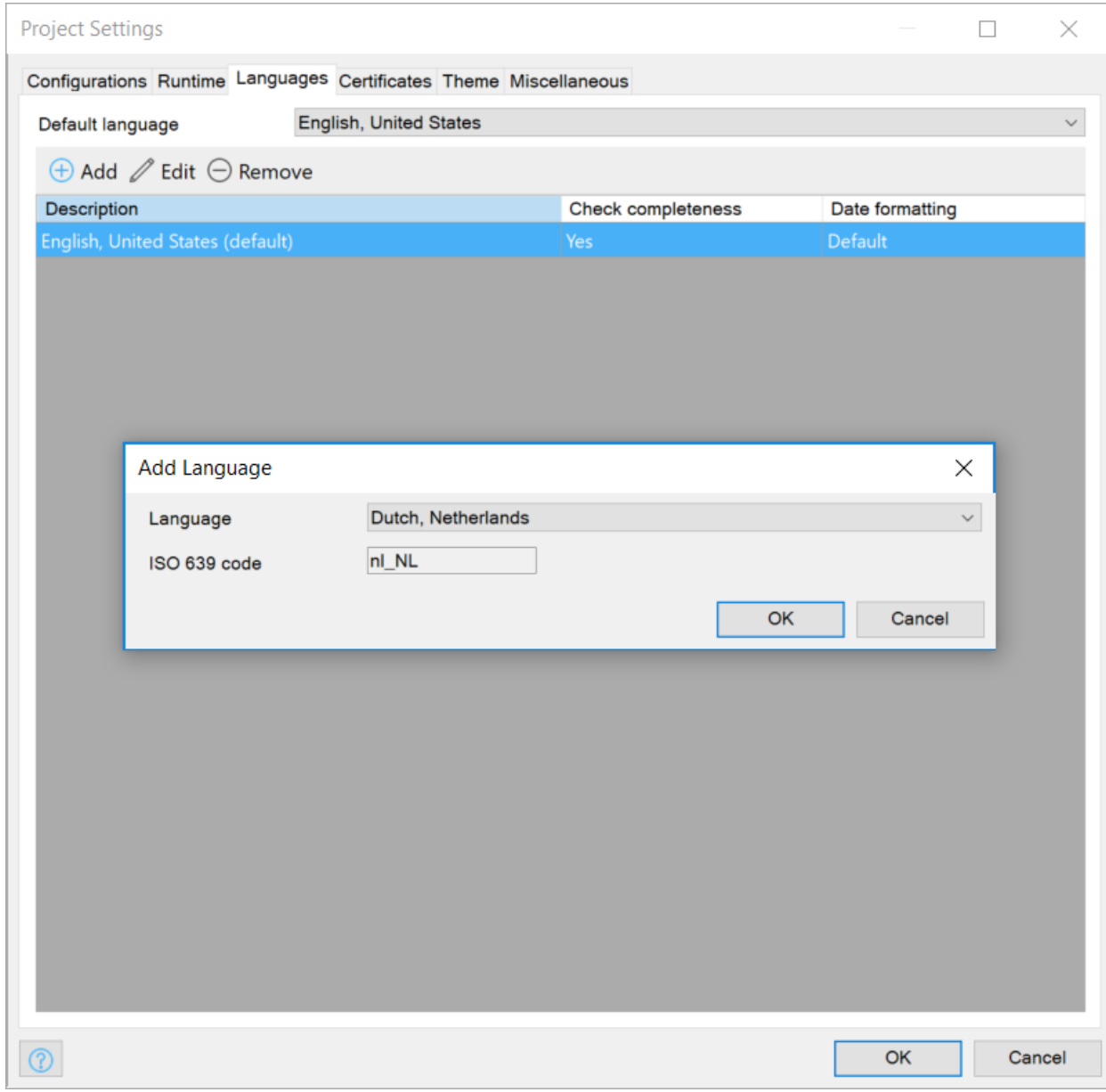
# Translation tools

- Batch Translation
- Text Occurence
- Export/Import Tranlations

# Add a language

- From the **Project Explorer** open the **Settings** menu
- Go to the **Languages** tab

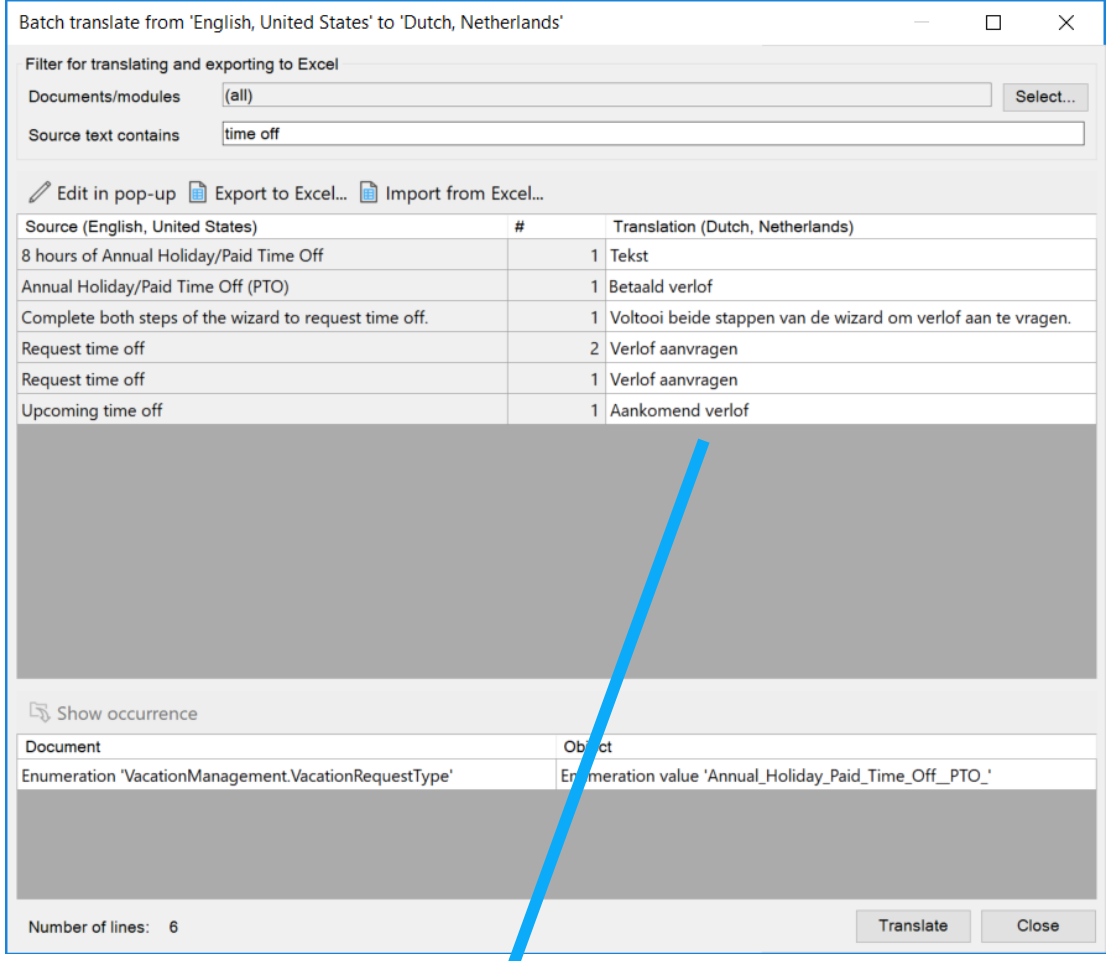
- Add **Dutch, Netherlands** as a new language




mx mendix

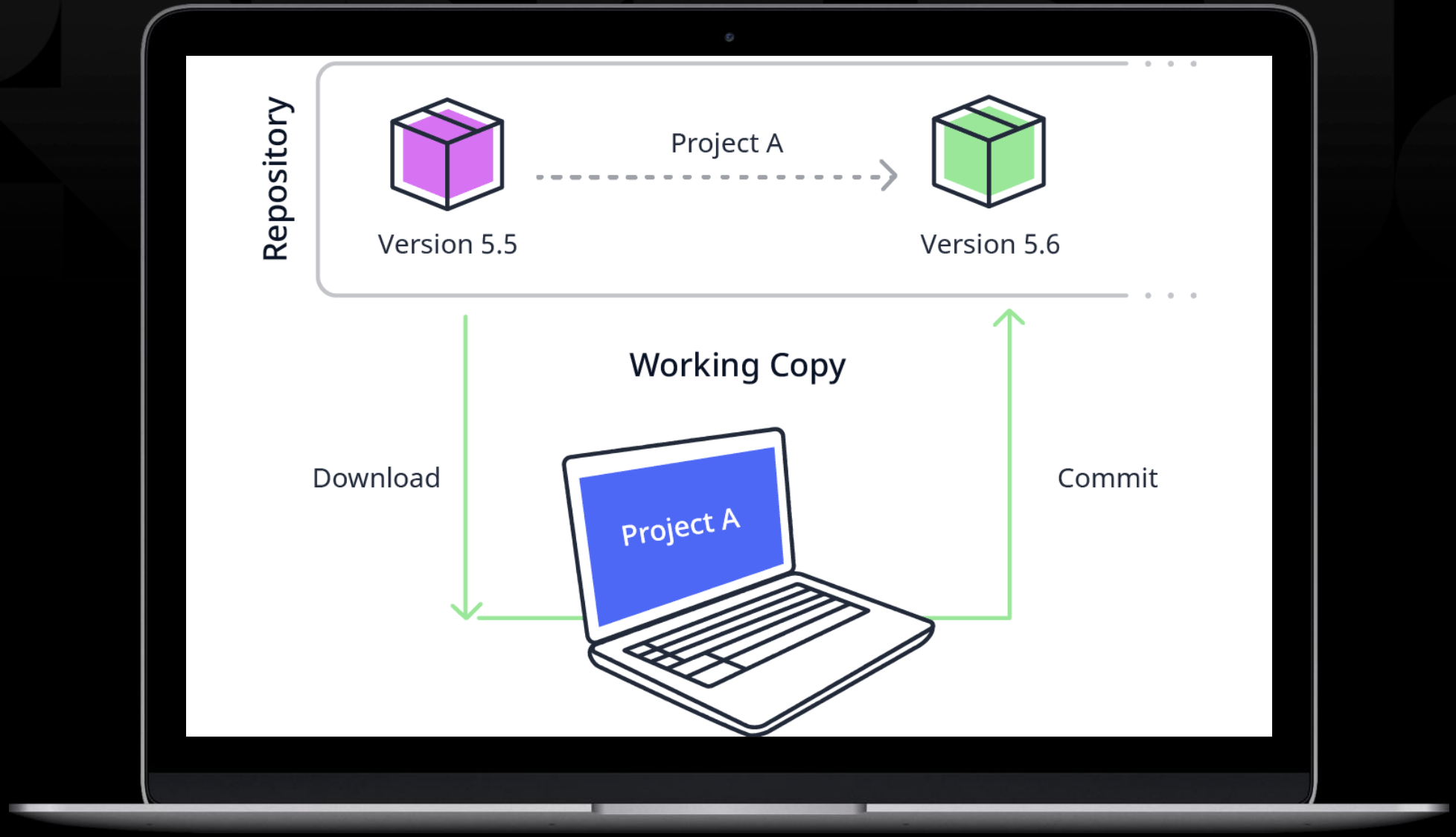
# Translate the app

- Set **Current language** to **Dutch**
- Open the **Home** page
  - Some text has <> around it
  - Some is translated to Dutch system text
- Use **Batch Translate** to translate from English (**Source**) to Dutch (**Destination**)




| | | |
|---|---|---|
| Annual Holiday/Paid Time Off (PTO) | 1 | Betaald verlof |
| Complete both steps of the wizard to request time off. | 1 | Voltooi beide stappen van de wizard om verlof aan te vragen. |
| Request time off | 2 | Verlof aanvragen |
| Request time off | 1 | Verlof aanvragen |
| Upcoming time off | 1 | Aankomend verlof |

# Commit your work



# Conclusion

# Feedback

mx mendix