

The training
will start at
9:00

Intermediate Developer

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

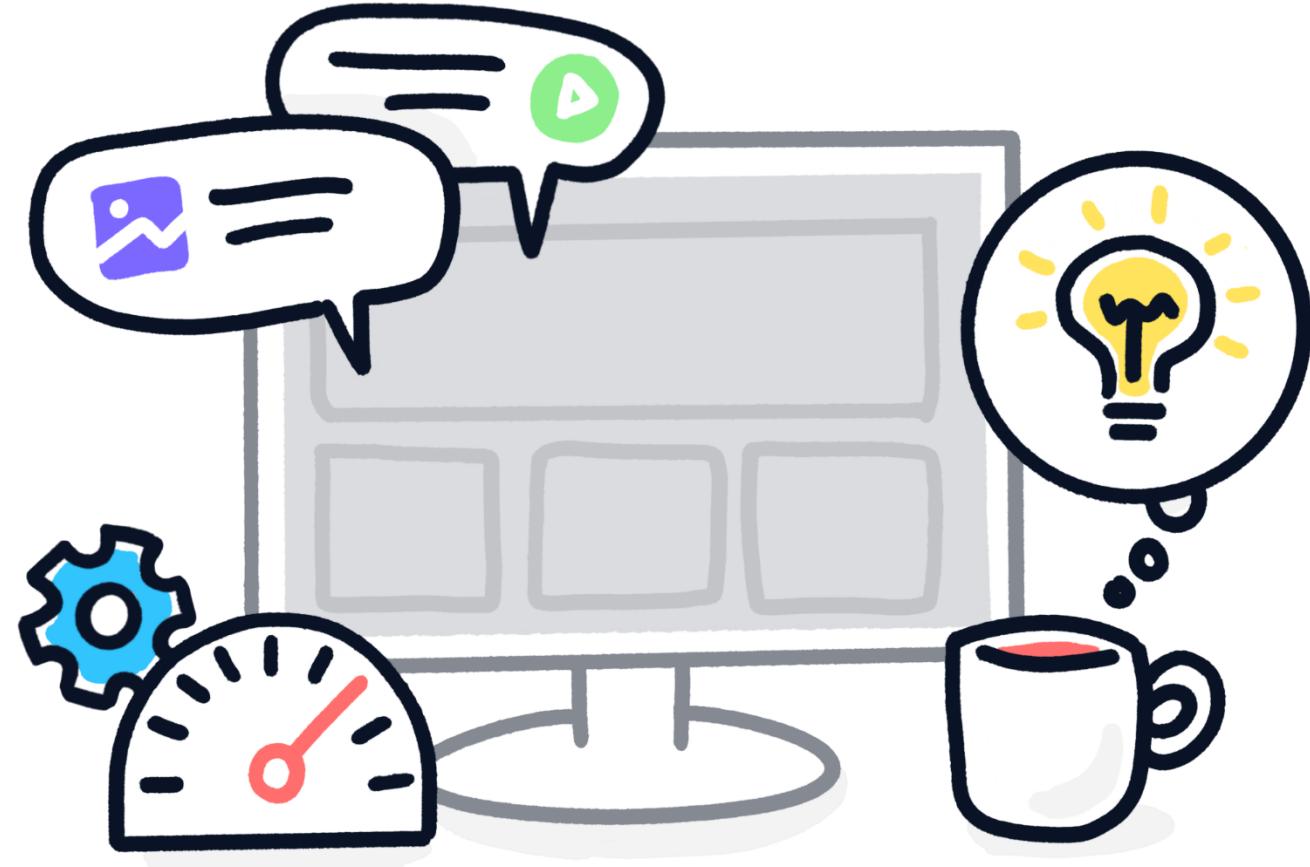
7 Translation

8 Conclusion

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

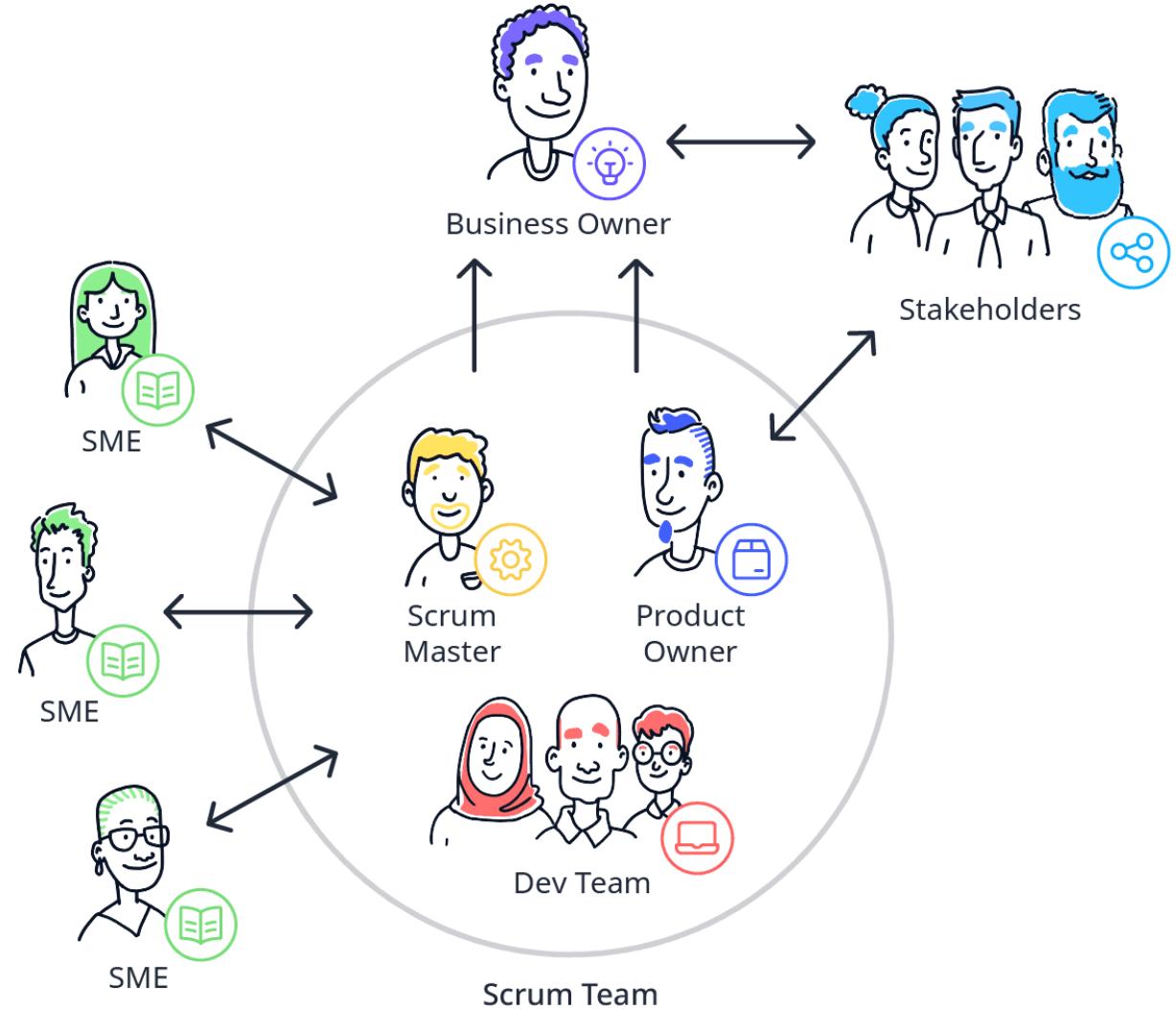
Openness

Commitment

Respect

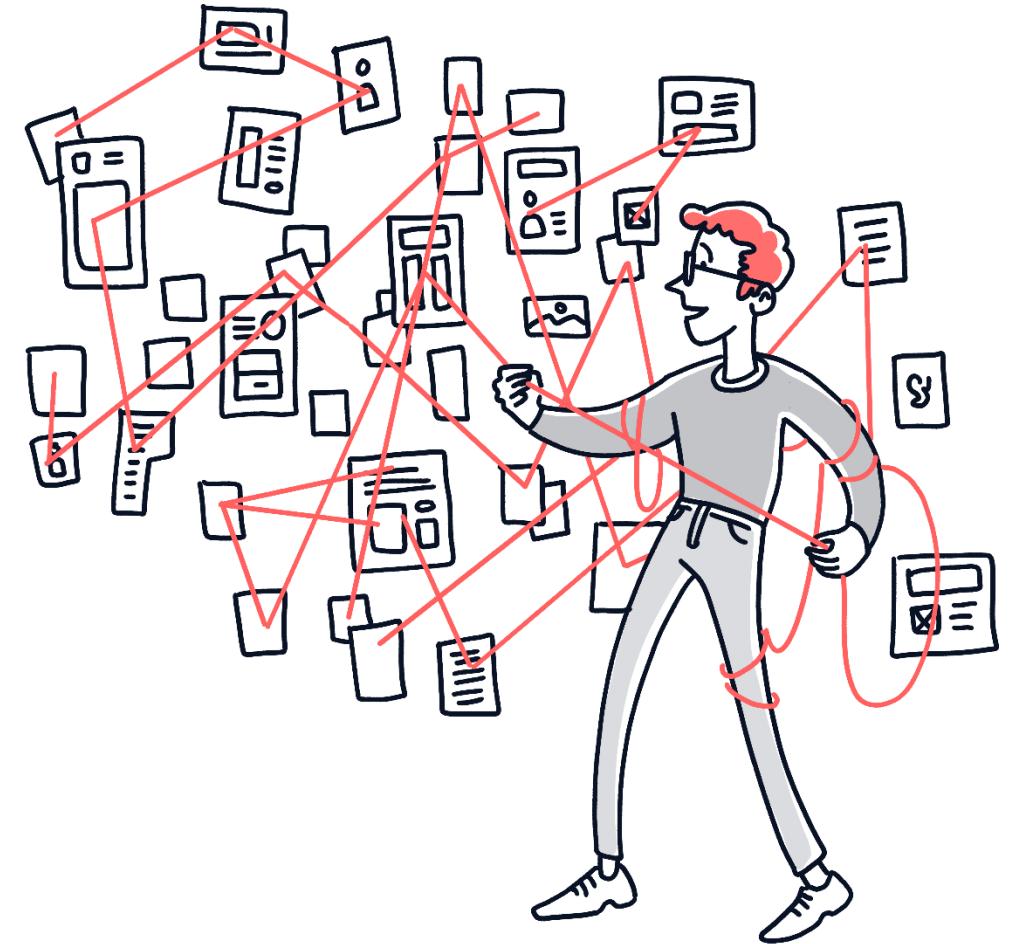
Team Size & Location

- 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

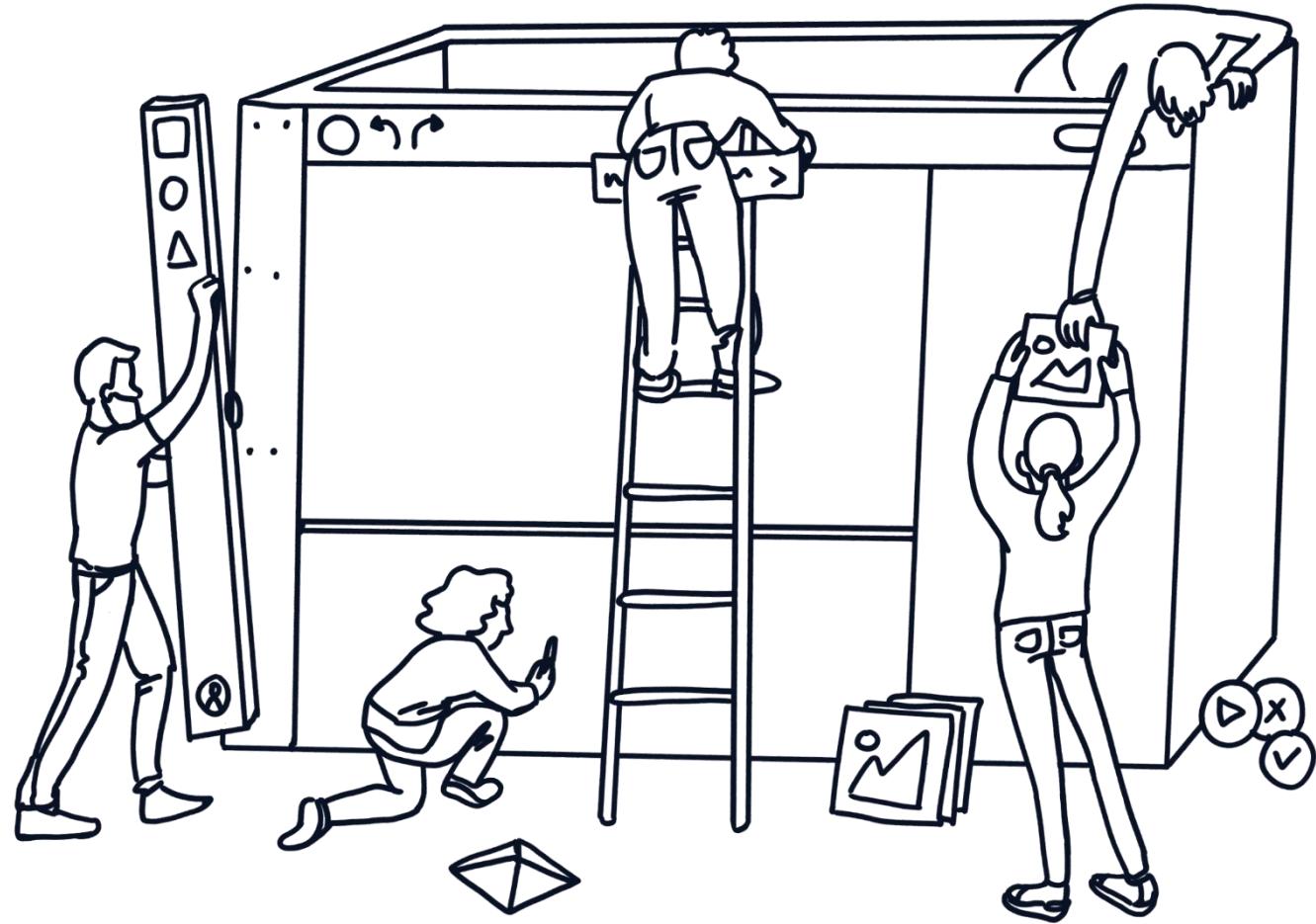
- Preparation period of one or two weeks
- Get familiar with:
 - overall picture
 - organization
 - stakeholders & SMEs
 - availability of team and stakeholders/SMEs
- Have official kick-off meeting
- Product Backlog Refinement meeting



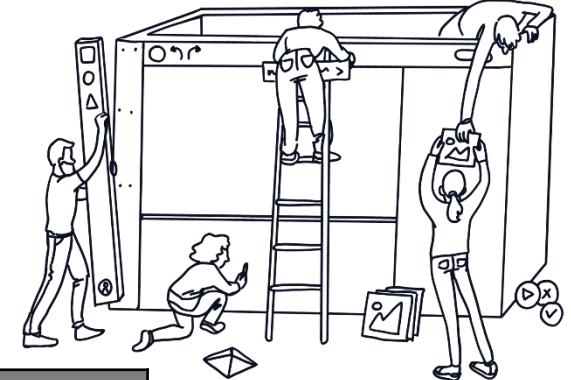
SCRUM events

Sprints contain and consist of:

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



SCRUM events



Event	Inspection	Adaptation
Sprint Planning	<ul style="list-style-type: none">• Product Backlog• Increment• Actions of the last Sprint• Definition of Done	<ul style="list-style-type: none">• Sprint Goal• Forecast• Sprint Backlog
Daily Scrum	<ul style="list-style-type: none">• Progress towards Sprint Goal	<ul style="list-style-type: none">• Sprint Planning• Sprint Backlog
Sprint Review	<ul style="list-style-type: none">• Increment• Sprint• Product Backlog	<ul style="list-style-type: none">• Product Backlog• Change Planning• Release
Sprint Retrospective	<ul style="list-style-type: none">• Work Method/ Process• Engineers practitioners• Definition of Done	<ul style="list-style-type: none">• Actionable committed improvements

SCRUM events

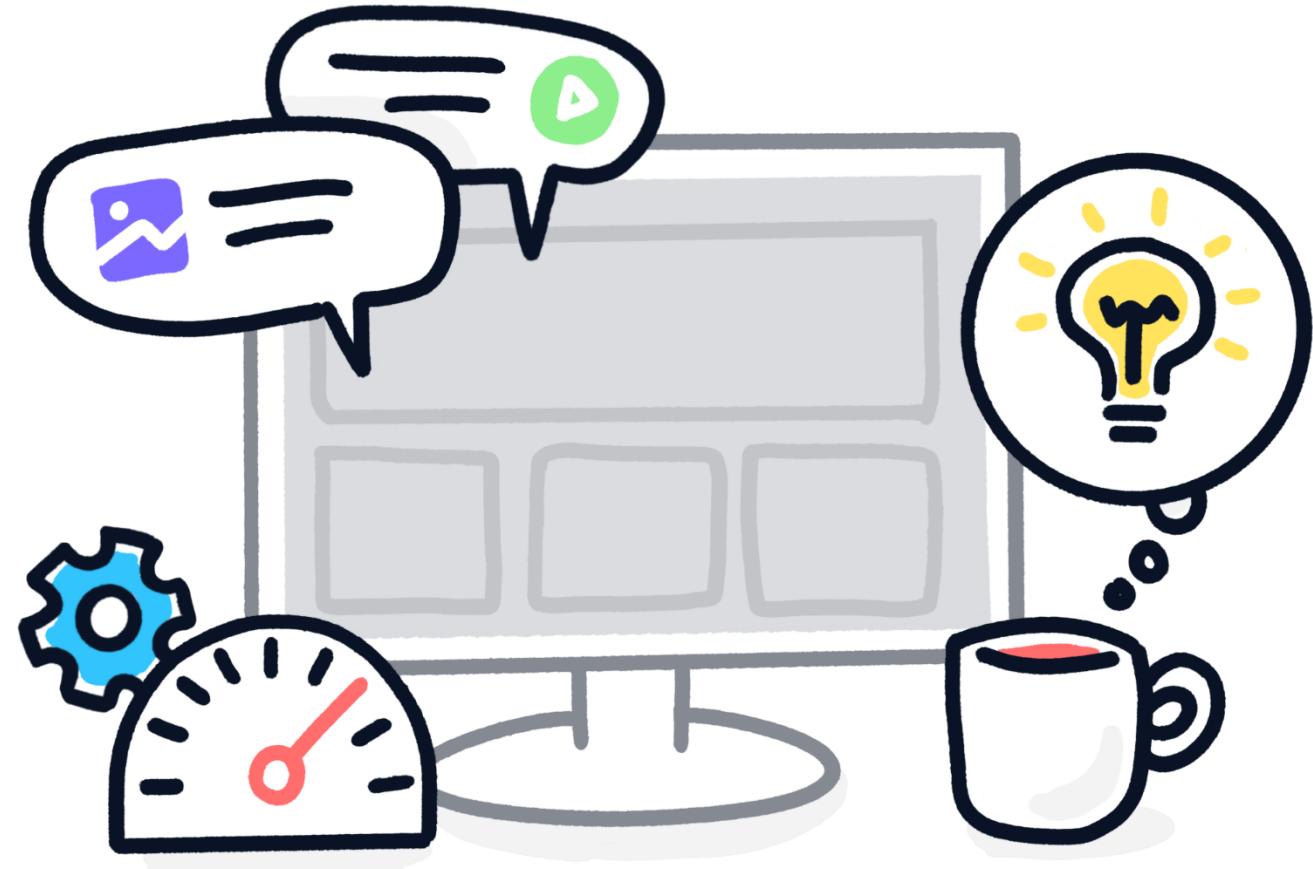


Event	4 Weeks	3 Weeks	2 Weeks	1 Week
Sprint Planning	8 hrs	6 hrs	4 hrs	2 hrs
Daily Scrum	15 min daily	15 min daily	15 min daily	15 min daily
Sprint Review	4 hrs	3 hrs	2 hrs	1 hr
Sprint Retrospective	3 hrs	2.25 hrs	1.5 hrs	0.75 hr

Laying out the basics

Learning goals

- Import modules from the AppStore
- Extending AppStore modules
- Layout Grid
- XPath basics



Our Use Case: North Sea Shipbuilding

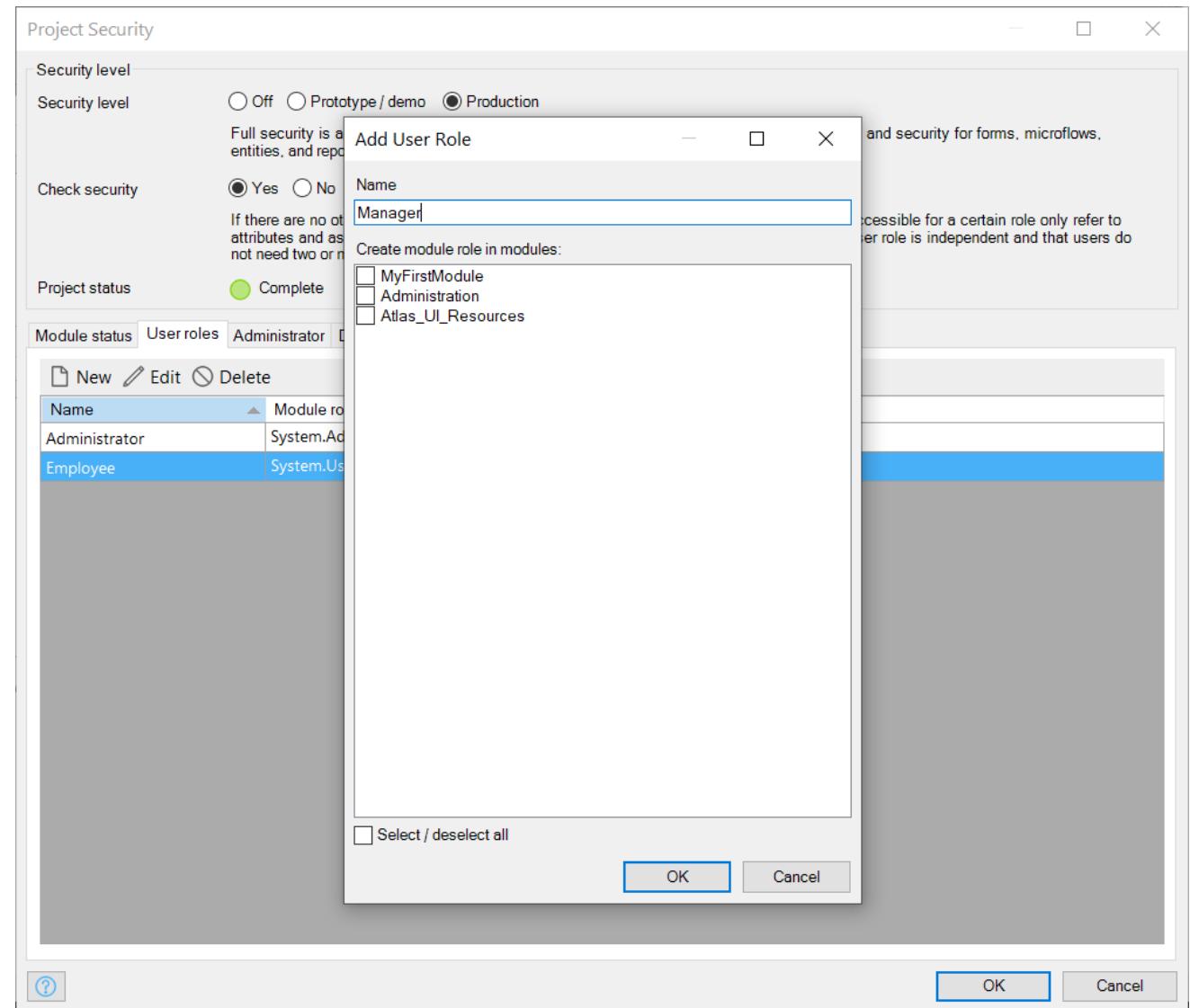


Andrea is the CEO of North Sea Shipbuilding. She wants her employees to have an easy way of requesting time off. It's also important that managers can approve or reject those requests, and that they can see who has time off when.



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





Import your user stories

- Download the stories from the shared project
- Open your own app in the Developer portal
- Go to the **Stories** tab
- Import the user stories

The screenshot shows the Mendix Stories tab interface. At the top right, there are buttons for "New Sprint", "New Story", and "More ▾". The "More ▾" menu is open, showing options: "Import / Export" (which is highlighted in blue), "Manage Labels", "Completed Sprints", and "History". Below the menu, there is a search bar with the placeholder "Select or type your search text (press Enter to search)" and a magnifying glass icon. Two sections are visible: "Get started" and "Backlog". Under "Get started", it says "Active sprint" and "No stories yet! Add a new story to get started.". Under "Backlog", it says "No stories yet! Add a new story to get started." and has an ellipsis (...).



User story

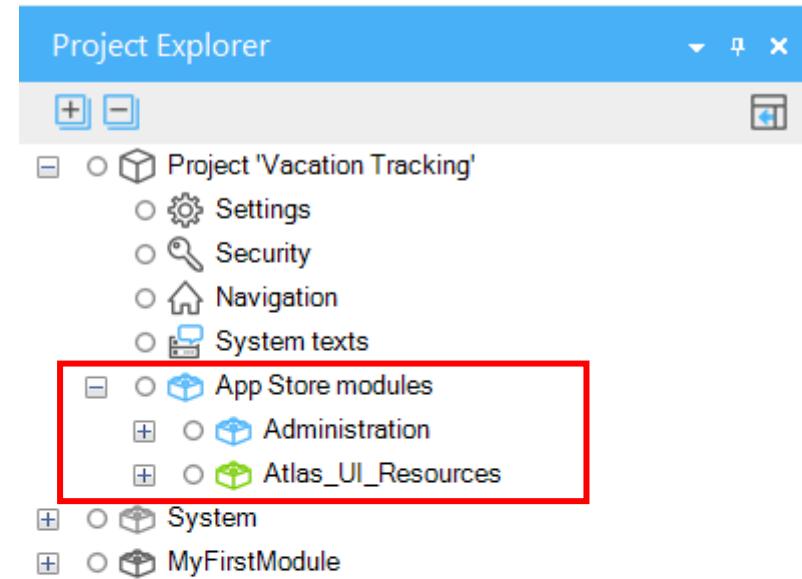
As an authorized user, I want to be able to add a profile picture, so people can easily identify me

Appstore modules

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

Use extension because:

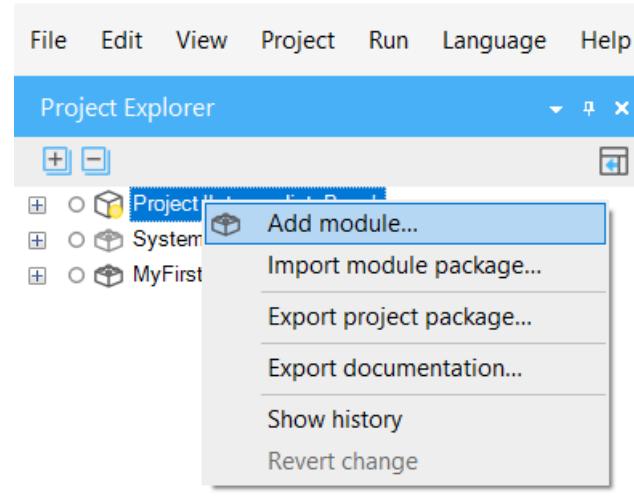
- Any changes are overwritten when you update
- Preferred way to add functionality is extending





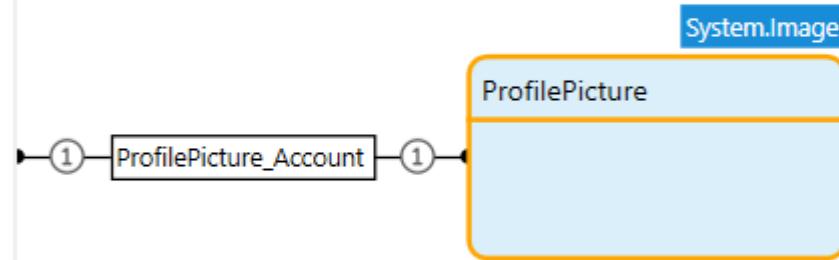
Create new GeneralExtentions module

- 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



Cross module associations

- System & Administration module
- **GeneralExtentions**
- Big app = multiple modules?





Managing Accounts

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

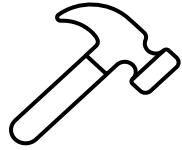
The diagram shows a Mendix model. At the top, there's a toolbar with icons for Entity, Annotation, View, Update security, and Import web service/XML file... Below the toolbar, there's a navigation bar with tabs like Home, Model, Data, Pages, etc. In the center, there's a class diagram. A blue-bordered box labeled 'System.Image' contains a white rectangle labeled 'ProfilePicture'. An association line connects 'ProfilePicture' to another rectangle labeled 'ProfilePicture_Account'. The 'ProfilePicture_Account' rectangle has a circled '1' at both ends of the line, indicating a 1-1 relationship. Below the diagram is a properties grid for the 'ProfilePicture' entity.

Properties of Entity 'GeneralExtentions.ProfilePicture'

General		System members	Documentation
Name	ProfilePicture	<input checked="" type="checkbox"/> Store 'createdDate' <input checked="" type="checkbox"/> Store 'changedDate' <input checked="" type="checkbox"/> Store 'owner' <input checked="" type="checkbox"/> Store 'changedBy'	
Generalization	System.Image	Select...	
Image	(none)	Select...	
Persistable	<input checked="" type="radio"/> Yes <input type="radio"/> No	Objects of this entity can only be stored in the database if it is persistable.	

Attributes Associations Validation rules Event handlers Indexes Access rules

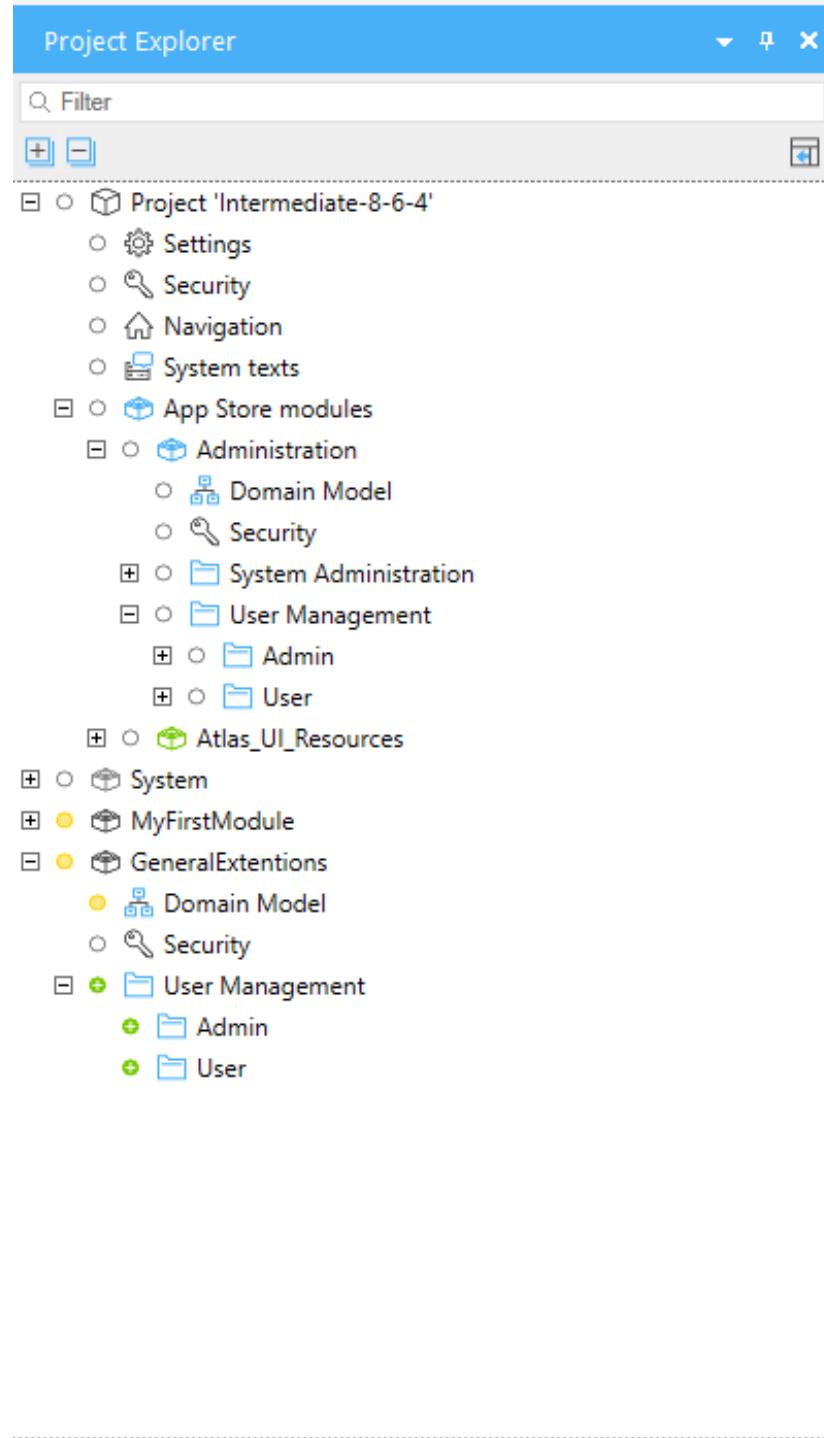
Name	Type	Owner	Parent	Child
ProfilePicture_Account	Reference	Both	GeneralExtentions.ProfilePicture	Administration.Account



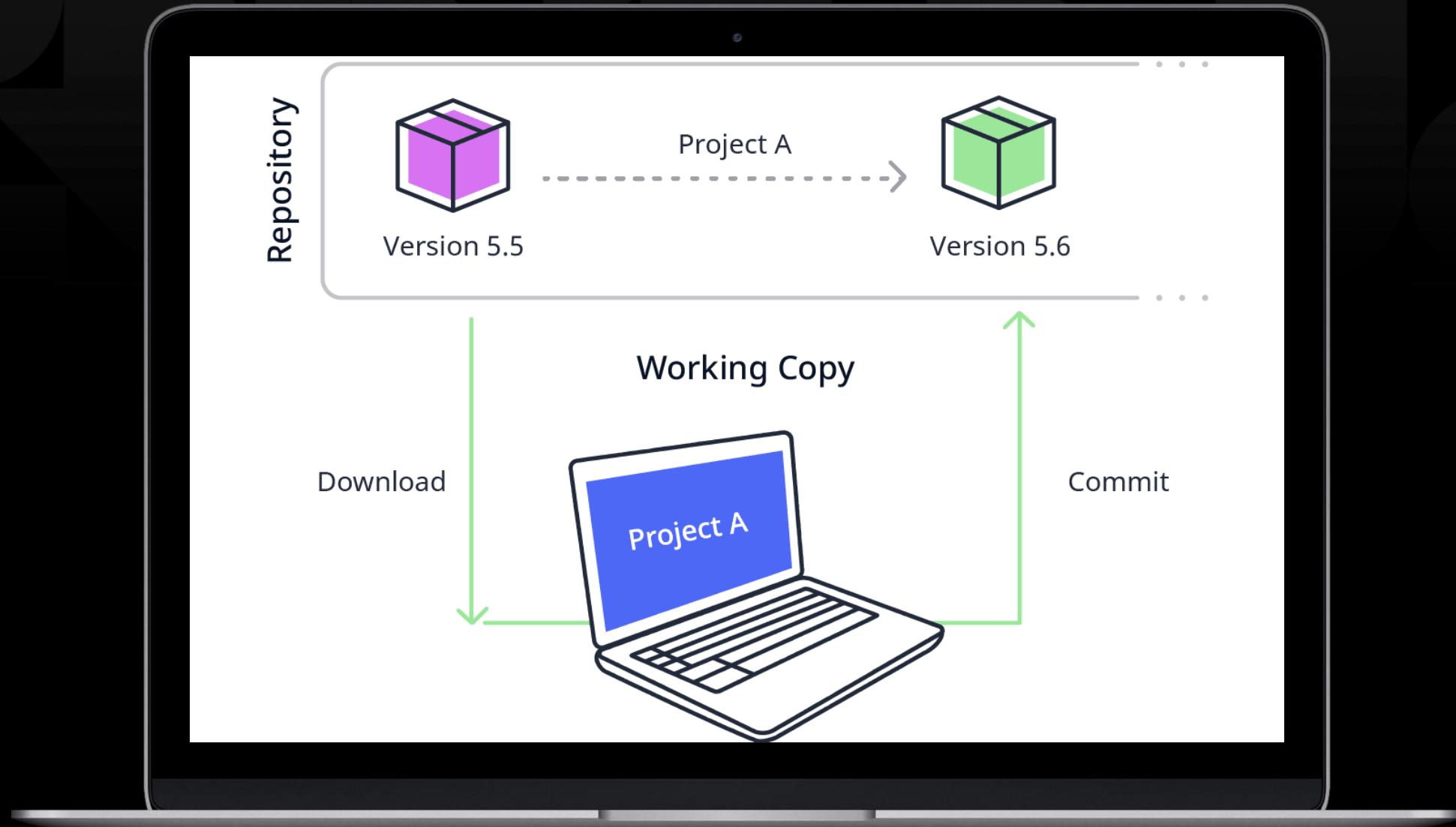
Organize your project

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



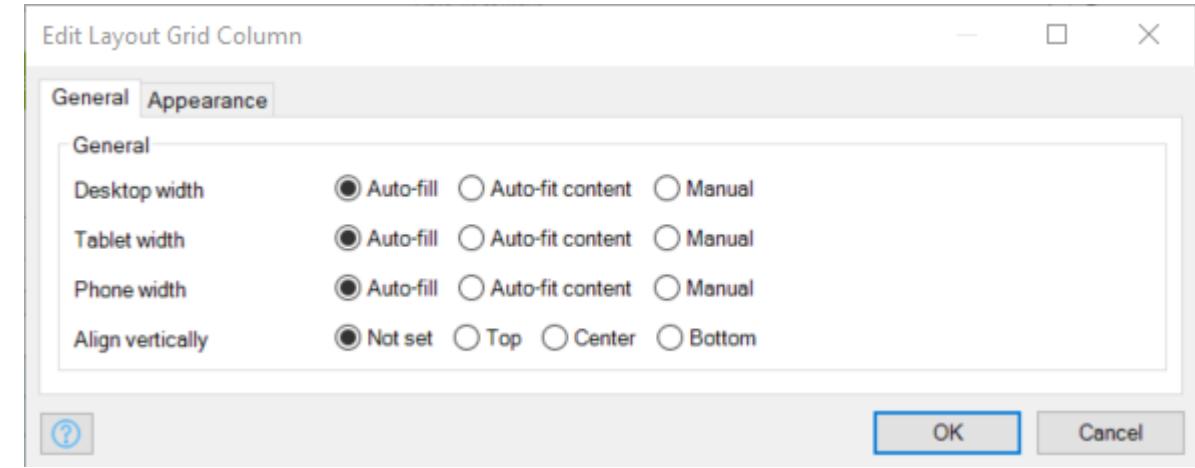


User story

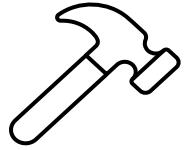
As an administrator, I want an account management interface that displays the user's profile picture, so I can quickly recognize them when managing users.

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



The screenshot shows a Mendix application interface. At the top, there is a header with 'Auto-fill' and 'Auto-fit content' sections. The 'Auto-fit content' section contains a 'Team member management' card with a subtitle placeholder and an 'Add team member' button. Below this is a list of accounts from a database. The main content area features a layout grid with three columns. The first column contains a profile picture with an 'Edit' button below it. The second column contains a 'FullName' field and an 'Email' field. The third column is empty and labeled 'Auto-fit'. At the bottom of the grid, there are 'Edit' and 'Delete' buttons, and a 'Load more...' link.



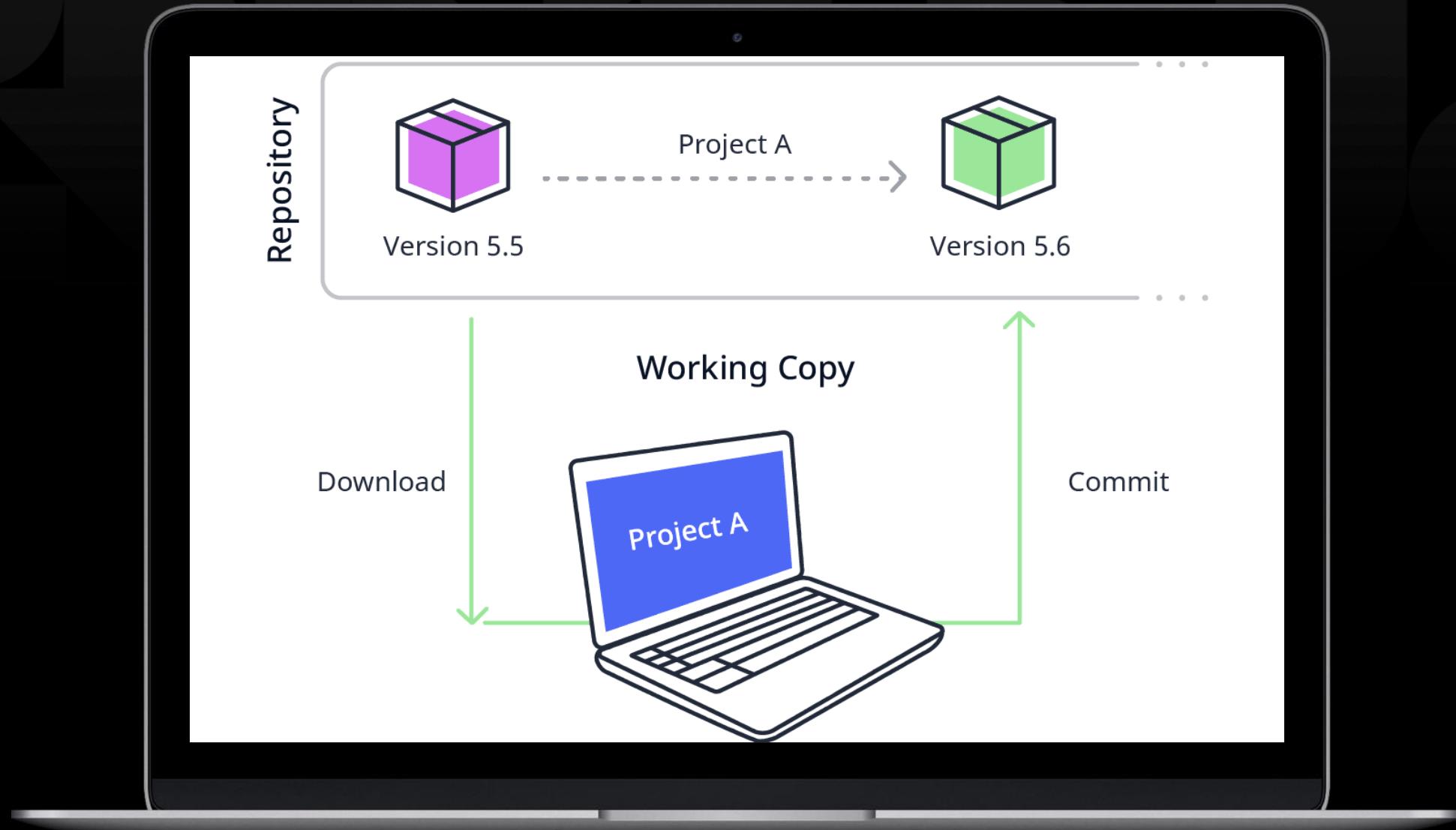
Managing Accounts

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

The screenshot shows a Mendix application page titled "Team member management". At the top, there is a navigation bar with the Mendix logo, a "Home" link, and an "Accounts" link. On the right side of the header, there is a green button labeled "Add team member". Below the header, there is a subtitle "Here you can put a paragraph as subtitle". The main content area is titled "List View" and displays a grid of six items, each representing a team member. Each item has a small profile picture, the member's full name, their email address, and two buttons: a blue "Edit" button and a red "Delete" button. The entire list is enclosed in a dashed border.

Commit your work





User story

As an employee, I want to have a page where I can manage my account information, so I can update things like my profile picture.

Constraining data with XPath

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

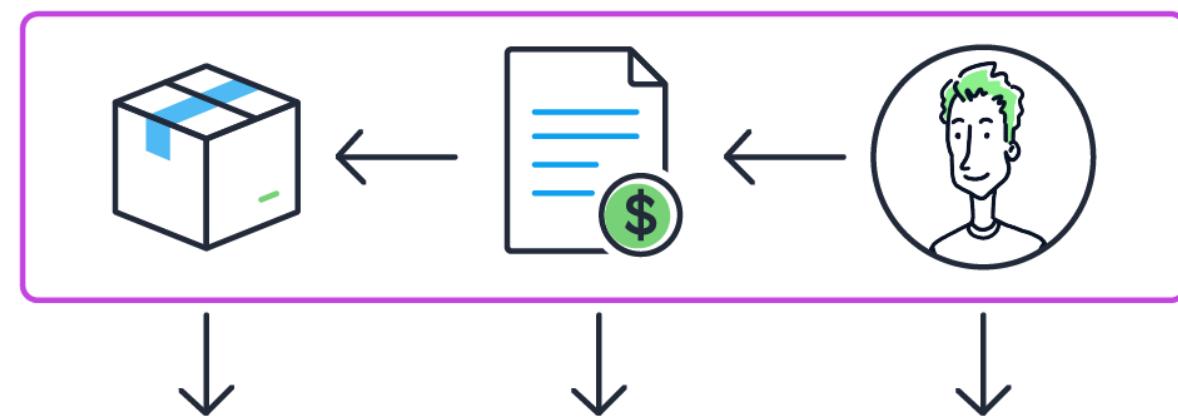
Product



Order



Me



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

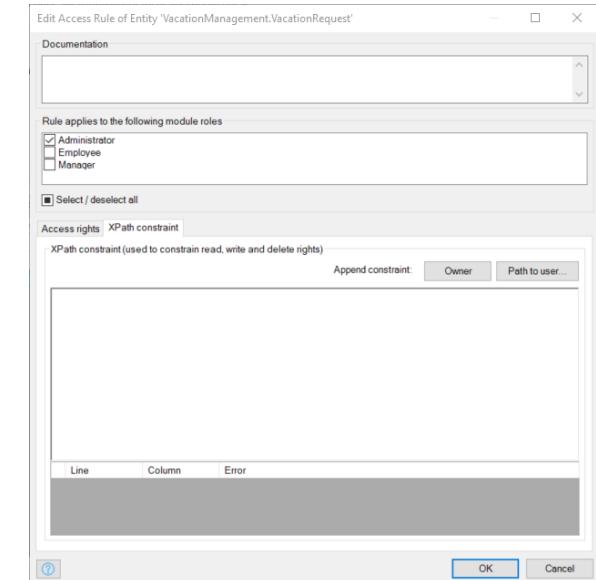
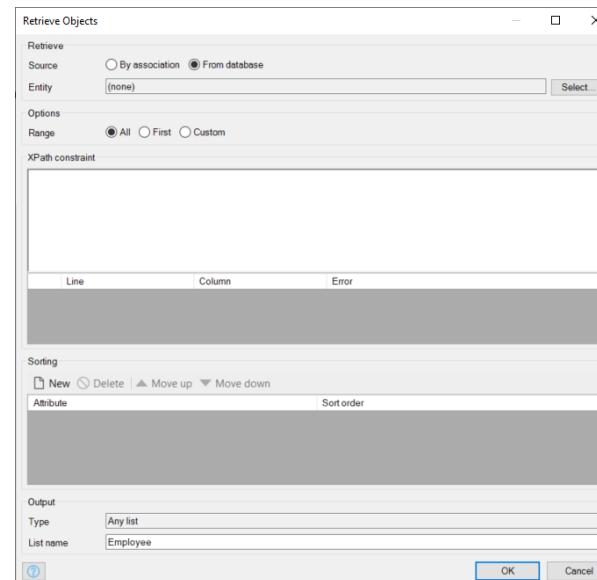
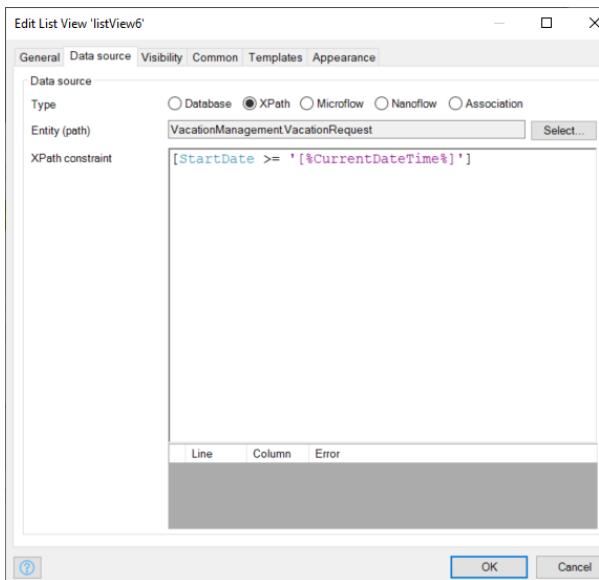
Where do we use XPath?

- 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

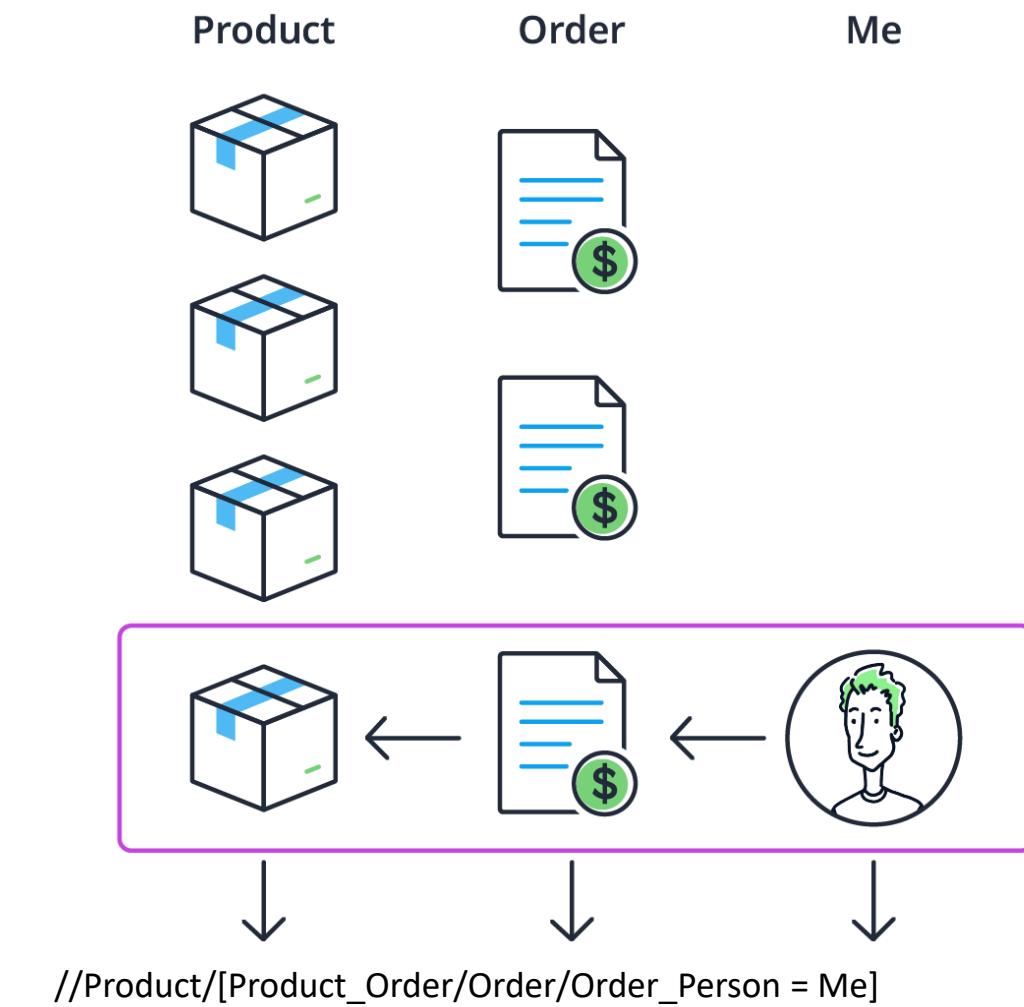


How to structure an XPath

- 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)

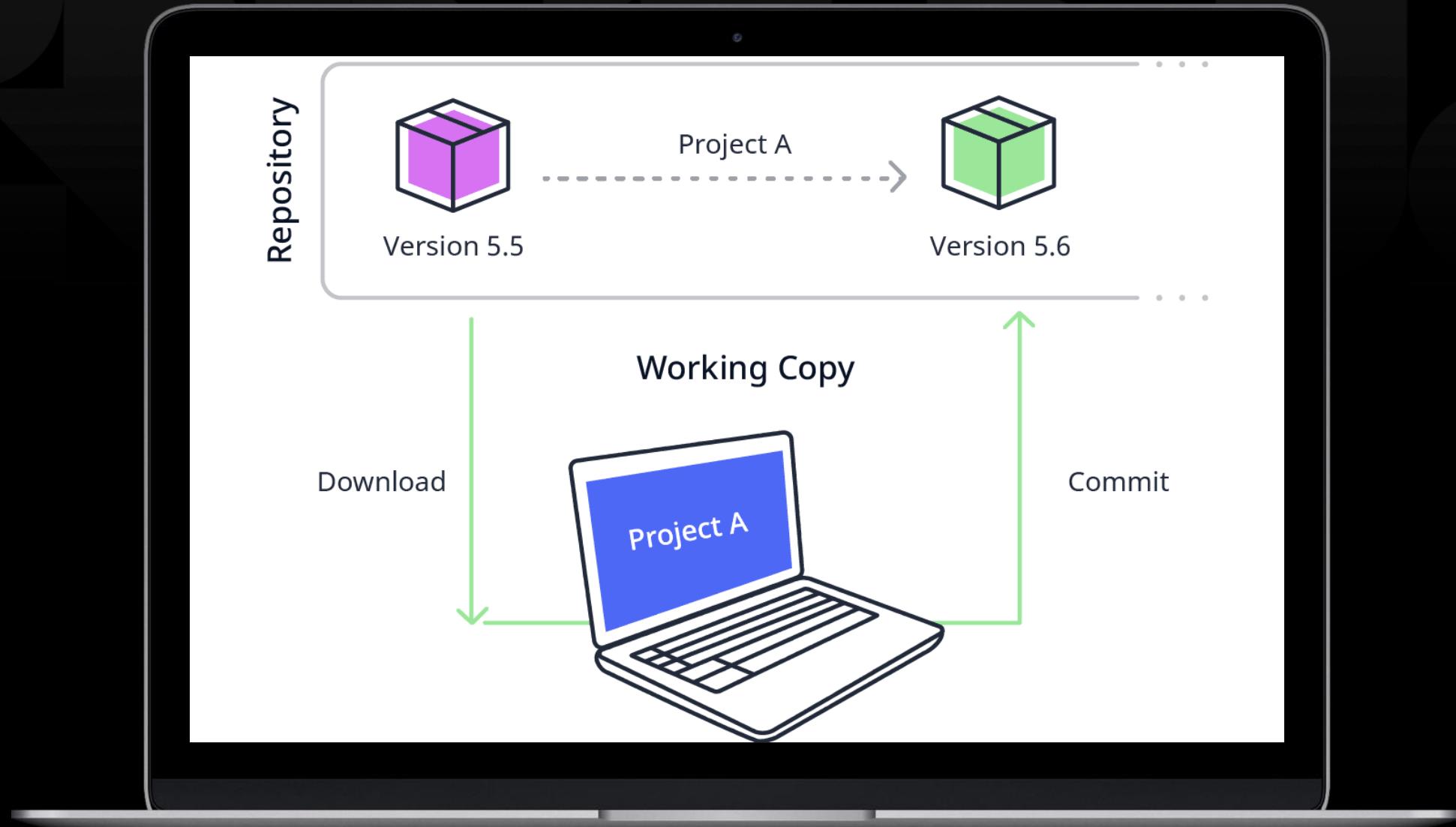


Managing My Account

- Build Manage My Account functionality
 - Use or copy the **ManageMyAccount** microflow
 - 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
- Create a user
- View with new user

The screenshot shows a Mendix application interface titled 'Manage My Account'. At the top, there is a navigation bar with icons for Home, Accounts, and My Account. The main content area has a title 'Manage My Account'. Below this, there are four input fields: 'Full name' (placeholder '[FullName]'), 'Email' (placeholder '[Email]'), 'User name' (placeholder '[Name]'), and 'Language' (placeholder '[.../Language/Description]') with a dropdown arrow. To the right of the language field is an orange button labeled 'Change password'. Below these fields is a section for 'Profile image' featuring a circular placeholder image of a woman with red hair. To the right of the image are buttons for 'Upload image', '...', and 'Browse...'. At the bottom of the form are two buttons: a green 'Save' button and a white 'Cancel' button.

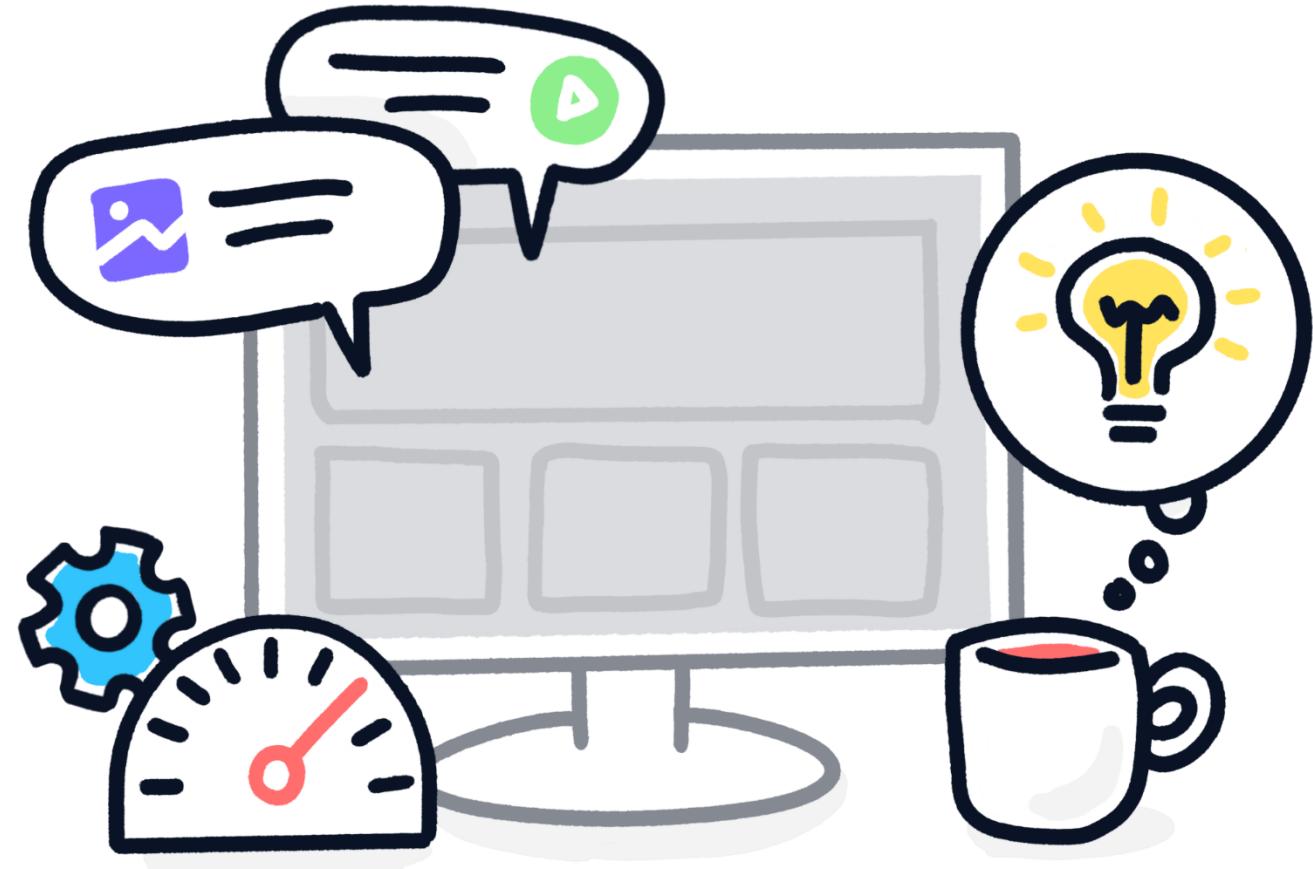
Commit your work



Branding

Learning goals

- Stylesheet (Custom styling) – adding a custom theme to the app
- Content of your project directory



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

The screenshot shows the Mendix Theme Customizer interface. At the top, there's a header with a small icon and a close button (X). Below the header, the title "Theme Customizer" is displayed, followed by the subtitle "Customize the look and feel of your Mendix application".

The interface is divided into several sections:

- Settings:** A section titled "Customize your app with the following settings:"
- UPLOAD LOGO:** A step 1 labeled "UPLOAD LOGO" with instructions: "After you have uploaded your logo, we will generate a color palette you can use for your brand colors. The logo colors and brand colors are available for use in the color picker below." It includes a "Upload logo" input field and a "Select File" button.
- BRAND COLORS:** A step 2 labeled "BRAND COLORS" with instructions: "These are the main colors for your app. Set them here and reuse them throughout your theme. This will make your theme look nice and consistent!" It shows a color palette with seven color swatches: Default (light grey), Primary (blue), Inverse (black), Info (light blue), Success (green), Warning (orange), and Danger (red).
- Preview:** A section titled "Get an impression of how your app will look:" which contains a large, empty placeholder box.
- Buttons preview:** A section showing a grid of colored buttons labeled "Default", "Primary", "Inverse", "Info", "Success", "Warning", "Danger", and "Danger".
- Toolbar preview:** A section showing a toolbar with icons for "Default", "Primary", "Inverse", "Info", "Success", "Warning", and "Danger".

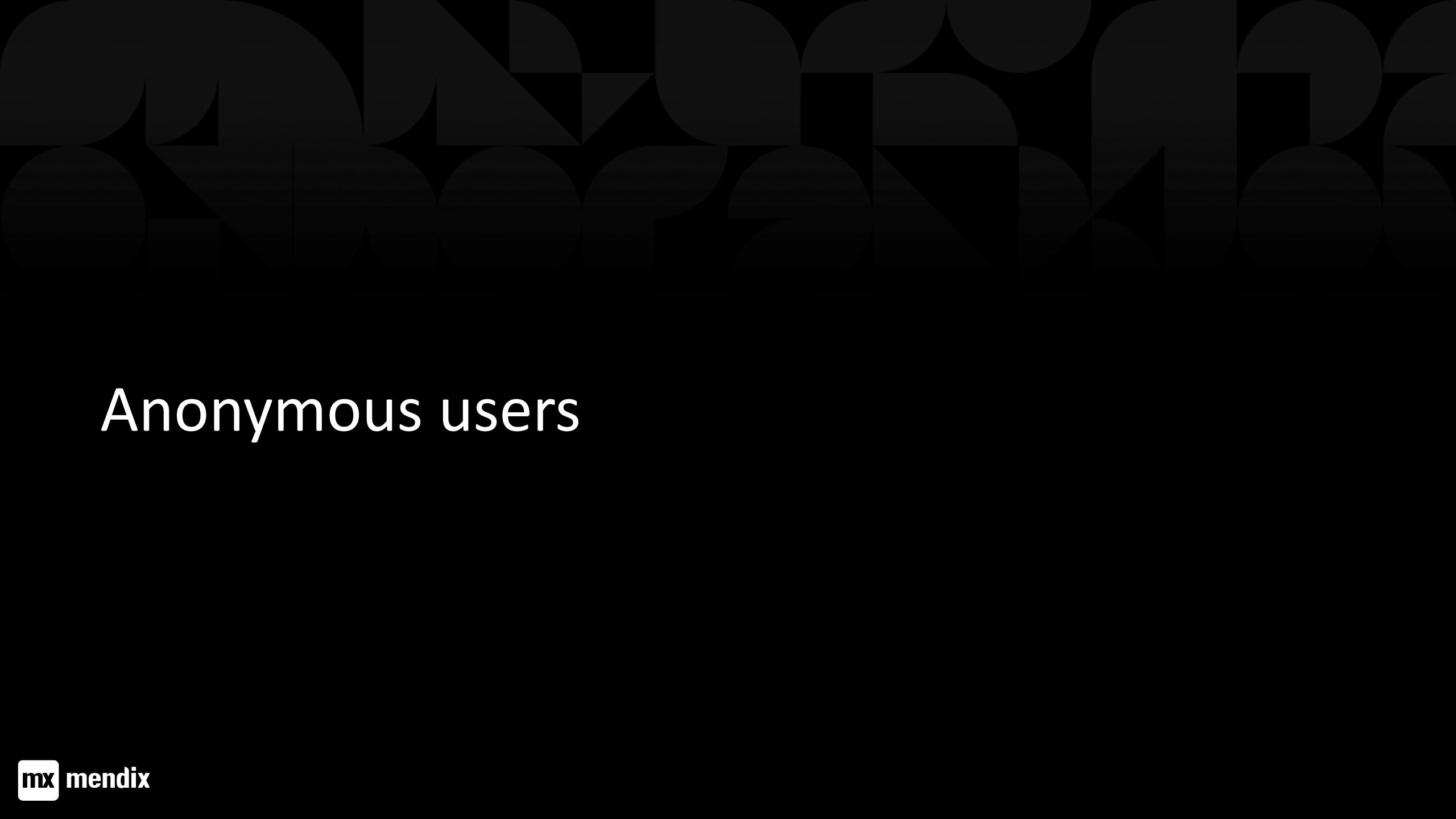
The Project Directory



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)

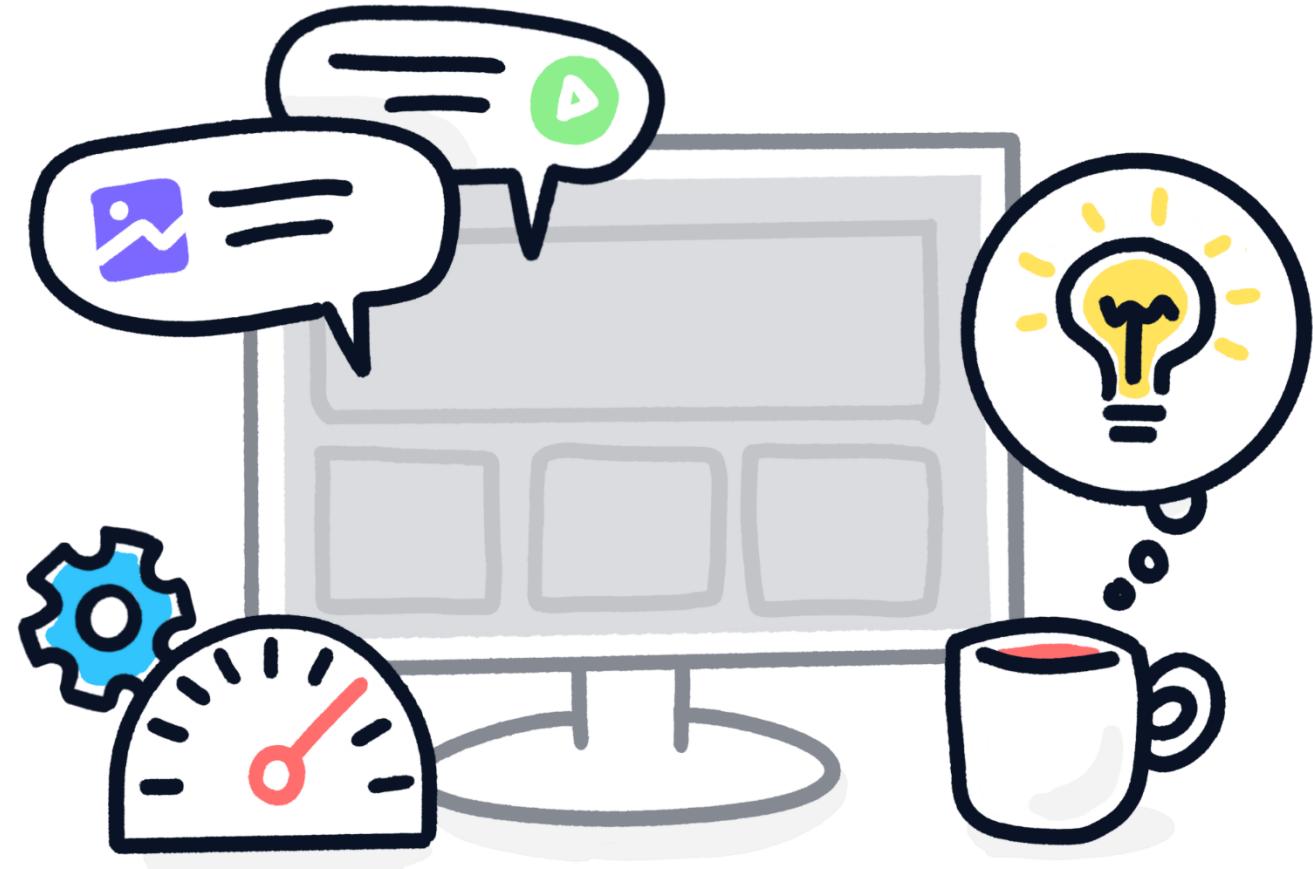
📁 .history
📁 .mendix-cache
📁 .svn
📁 deployment
📁 javascriptsource
📁 javasource
📁 resources
📁 theme
📁 userlib
📁 widgets
📄 .classpath
📄 .project
ｍｐｒ Intermediate 9-10 April 2020.mpr
ｍｐｒ Intermediate 9-10 April 2020.mpr.bak
ｍｐｒ Intermediate 9-10 April 2020.mpr.lock
📄 Vacation_Tracking_main_2.launch



Anonymous users

Learning goals

- What is an anonymous user in Mendix?
- What are the benefits and drawbacks?
- How to use an anonymous user in 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

Introduction

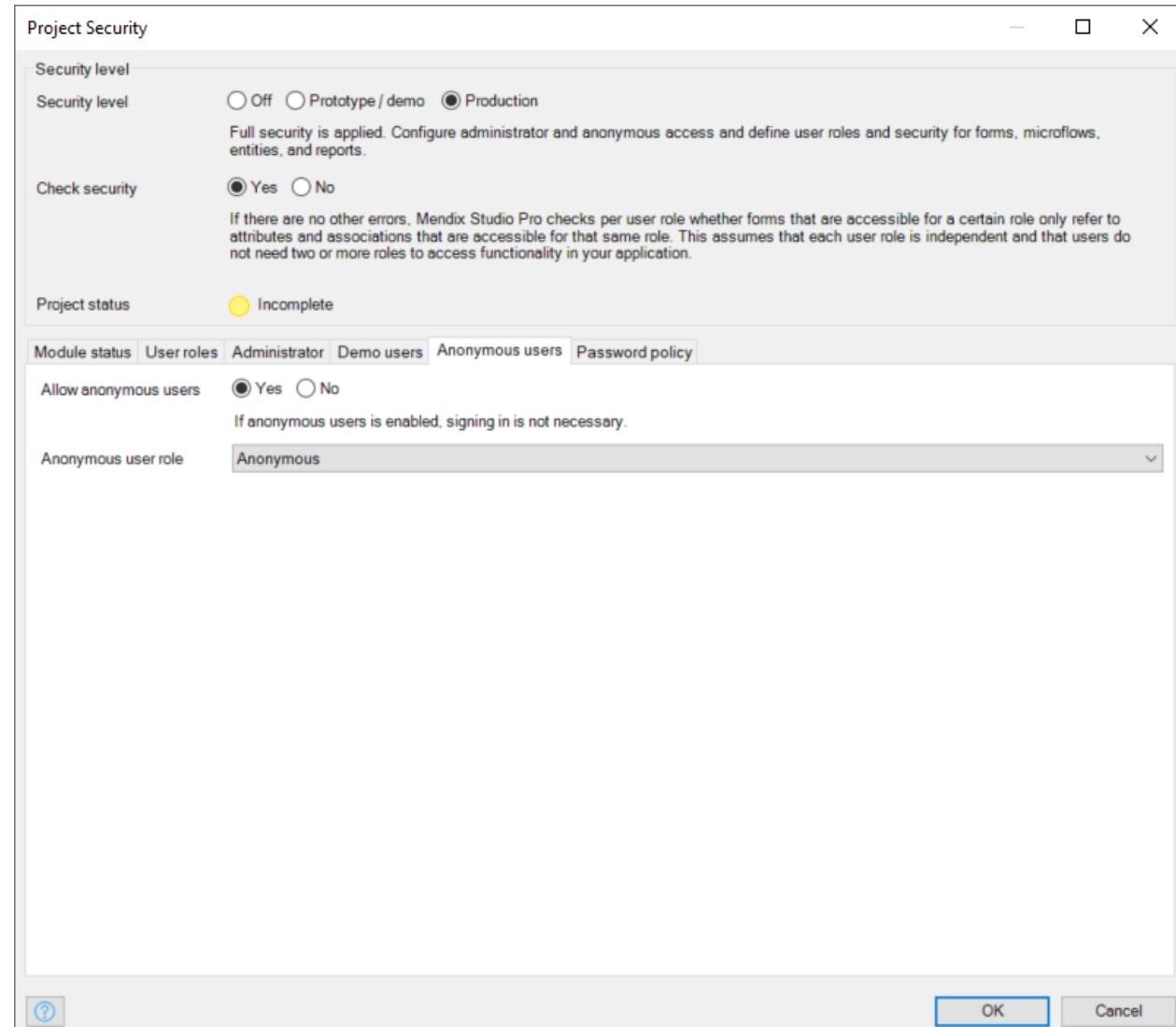
Anonymous users allow people without accounts to have access to a Mendix app

Benefits

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

Drawbacks

- Potential security risk



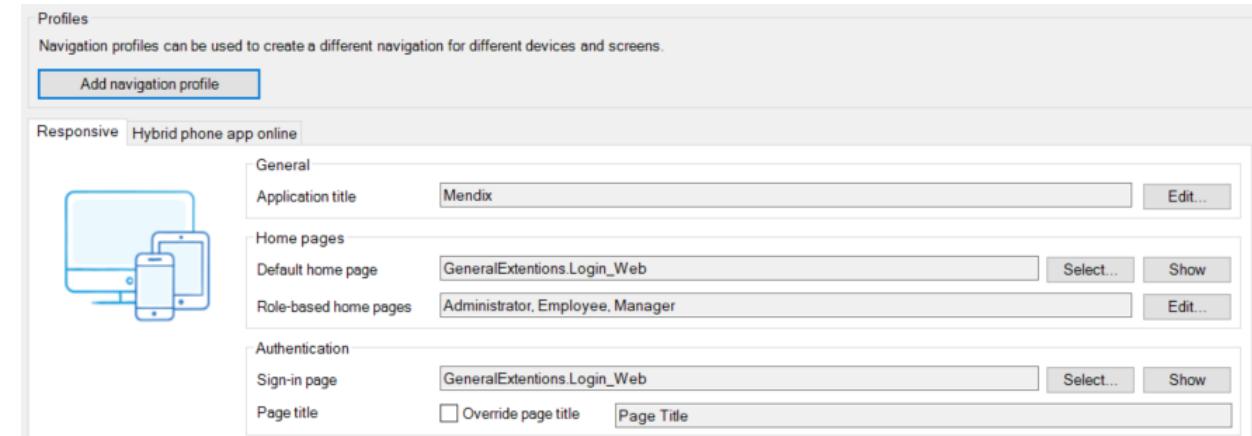
Custom login page

- 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)

	Mendix page	HTML page
Easy to use		
HTML knowledge required		
Allow Anonymous Users		
Works on Hybrid Mobile		
Mx loaded after login		

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



The screenshot shows the 'Profiles' section in the Mendix application. It displays a navigation profile named 'Hybrid phone app online'. The profile includes settings for 'General', 'Home pages', 'Role-based home pages', and 'Authentication'. Under 'General', the application title is set to 'Mendix'. In 'Home pages', the 'Default home page' is set to 'GeneralExtentions.Login_Web'. Under 'Role-based home pages', the roles 'Administrator, Employee, Manager' are listed. In 'Authentication', the 'Sign-in page' is also set to 'GeneralExtentions.Login_Web'. There is an option to 'Override page title' which is checked, and the 'Page Title' field contains 'Page Title'.



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 role based homepages for Administrator, Manager & Employee
- Build out custom login page to match this design

Welcome to North Sea
Shipbuilders Vacation
requests

Use this app to get your vacation requests
approved!

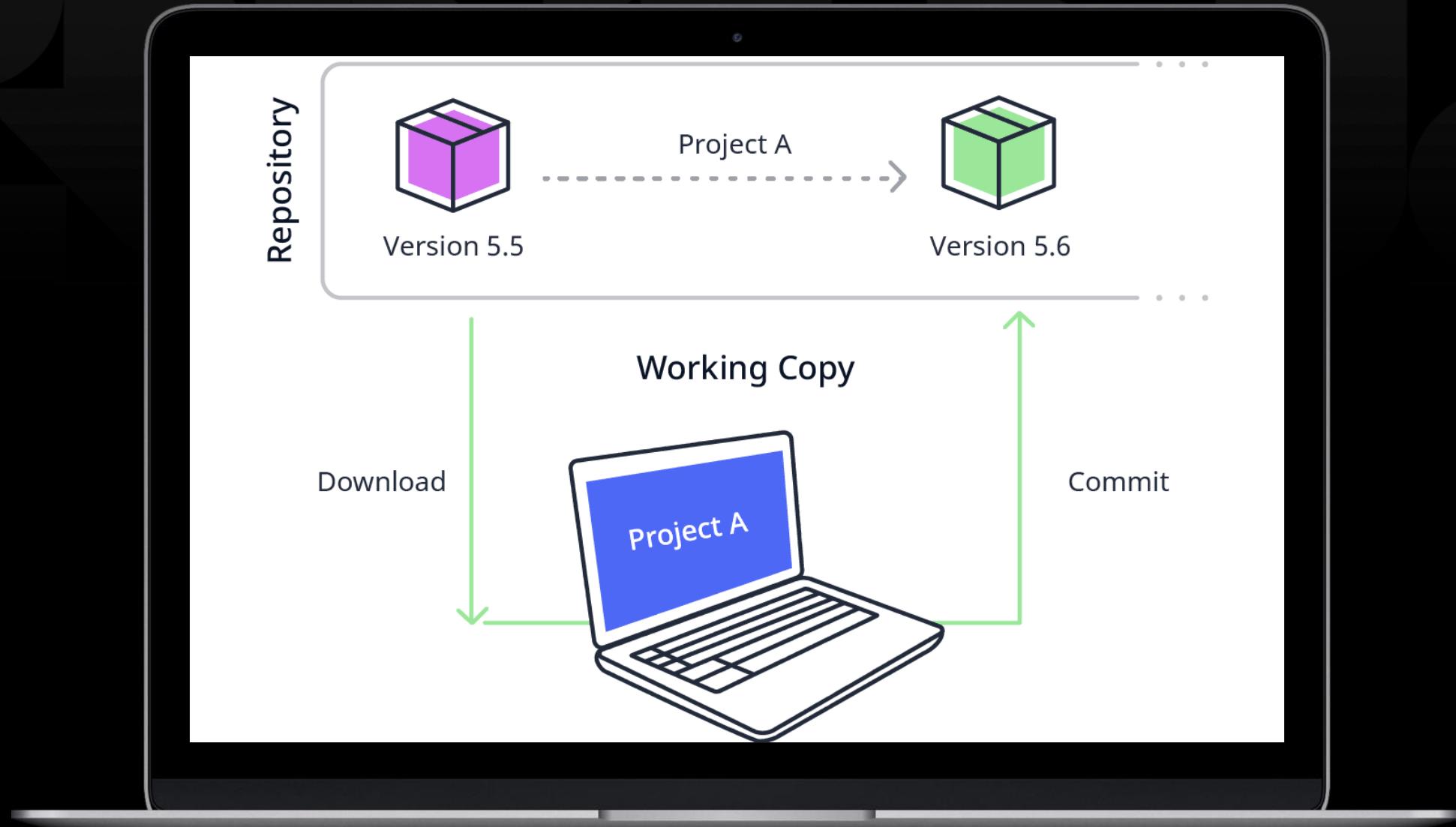
Sign in

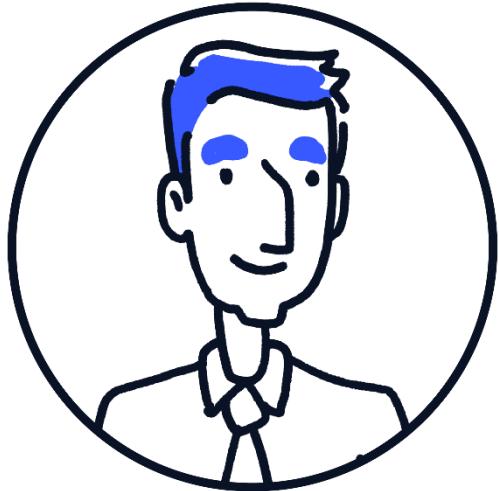
Username

Password

Sign in

Commit your work



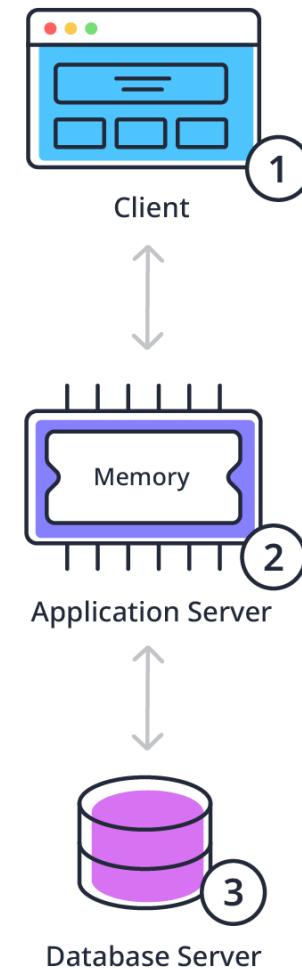


User story

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

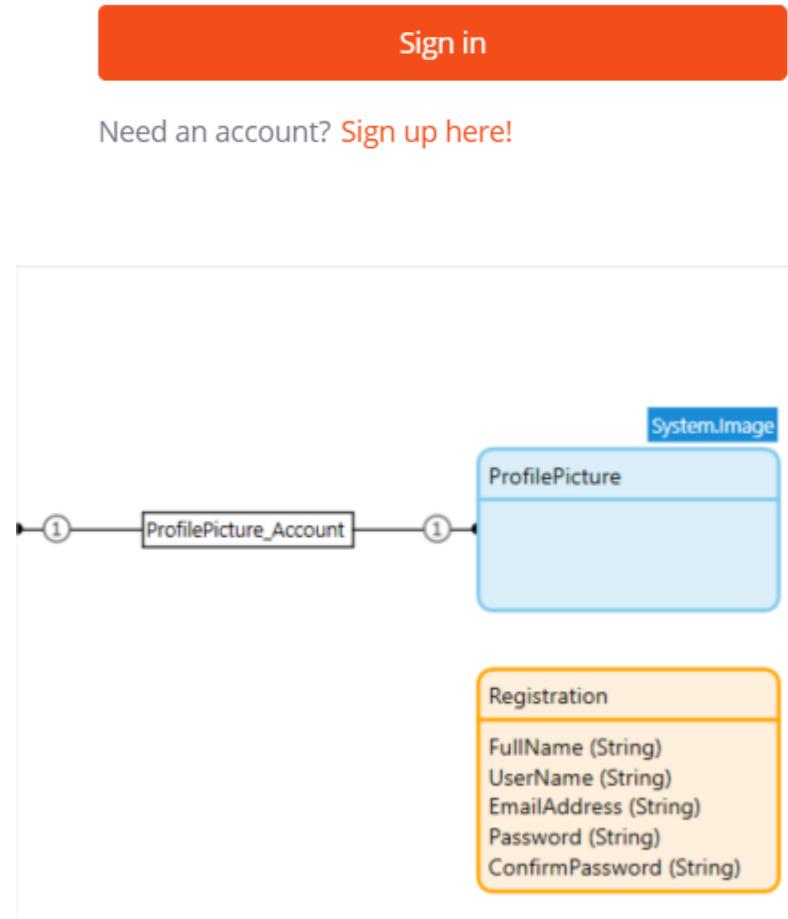
- Persistable and non-persistable
- Transient objects
- Non-persistable associations (1-1 not possible)
- Other limitations
 - No AutoNumber
 - No validation rules on Domain Model
 - No indexes
- Why use them?





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)



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

Retrieve Objects

Source By association From database

Entity System.UserRole

Options

Range All First Custom

XPath constraint

```
[id = '%UserRole_Employee%']
```

Line	Column	Error

Sorting

New Delete | ▲ Move up ▼ Move down

Attribute	Sort order

Output

Type System.UserRole

Object EmployeeUserRole

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

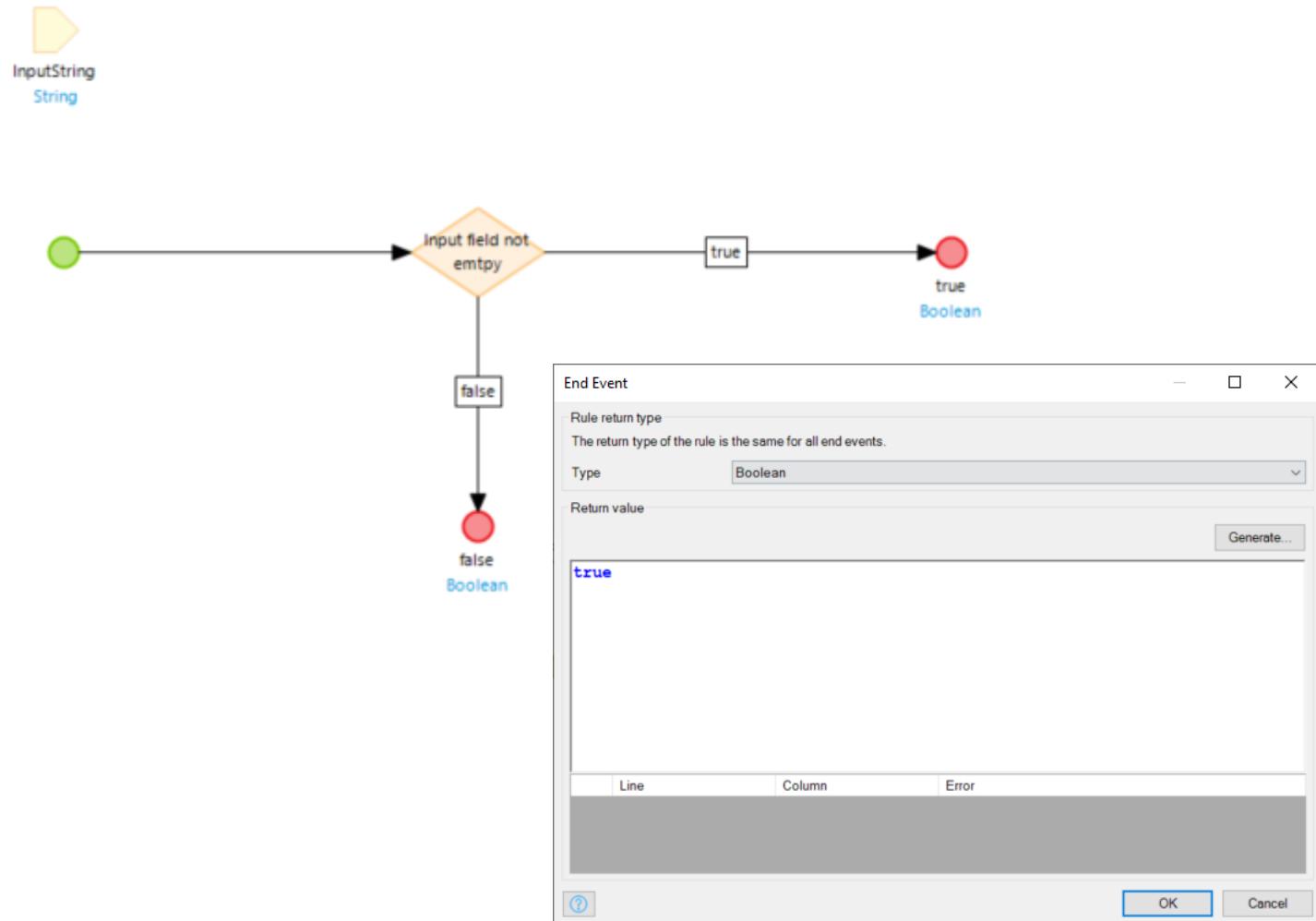
Validating string inputs

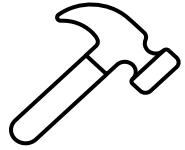
- `$InputString != empty` doesn't cover it!
- `empty` is not the same value as "
- Correct way of checking empty string:

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

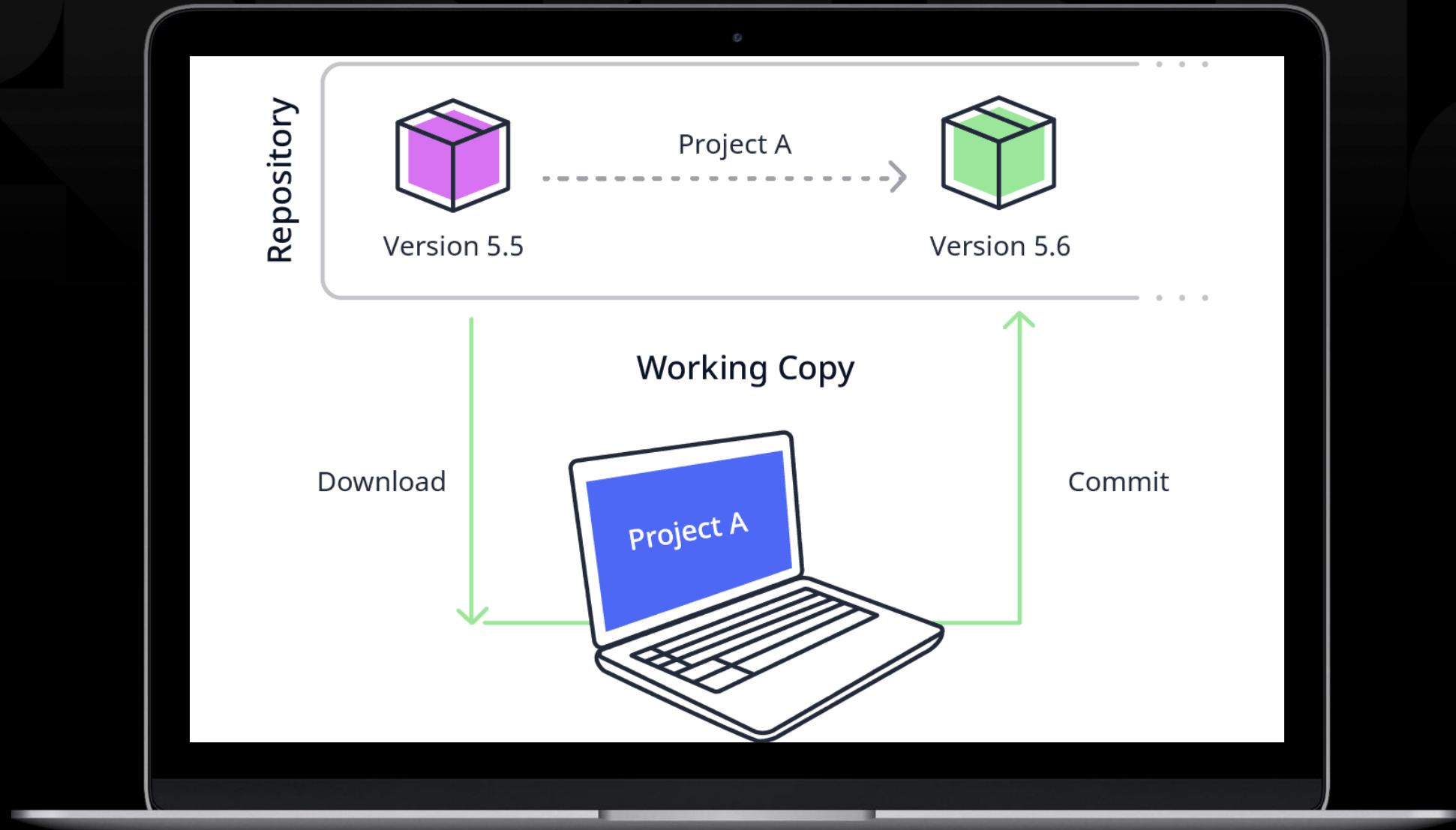




Extract submicroflow

- 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

Commit your work



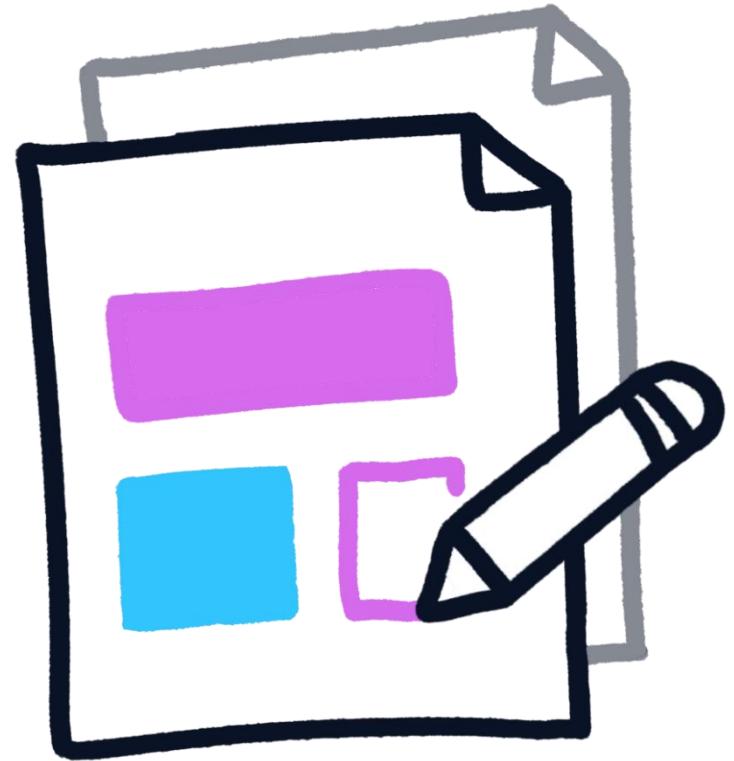


User story

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

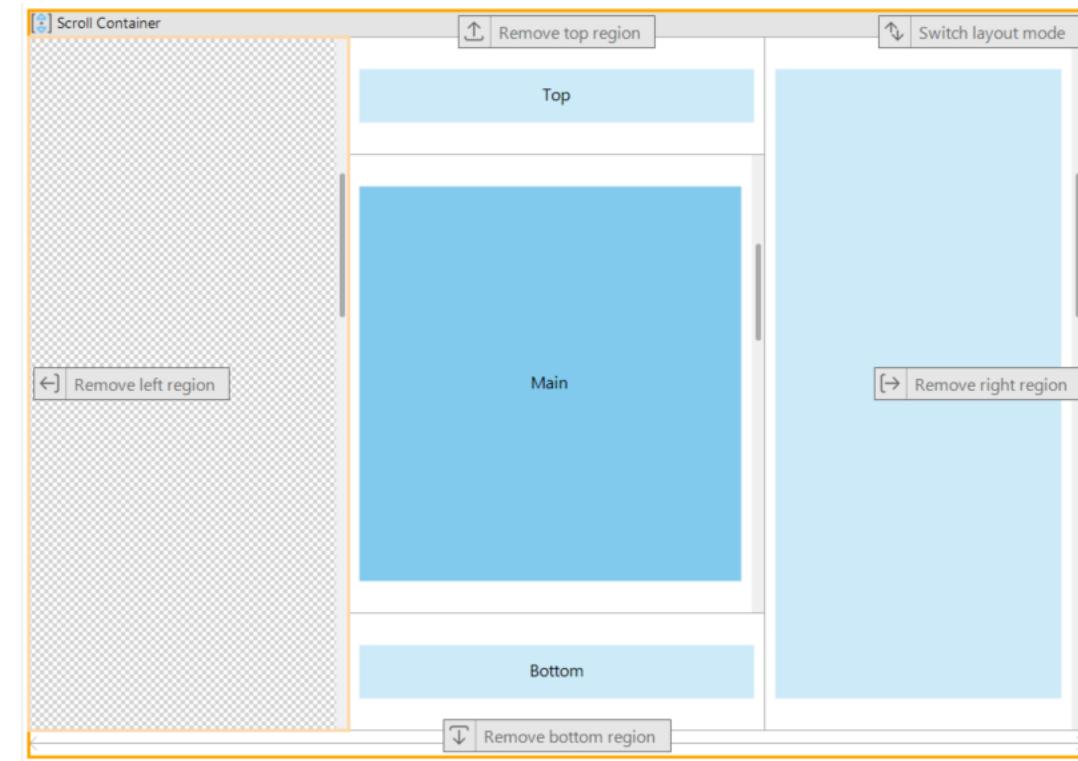
Layouts

- 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



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



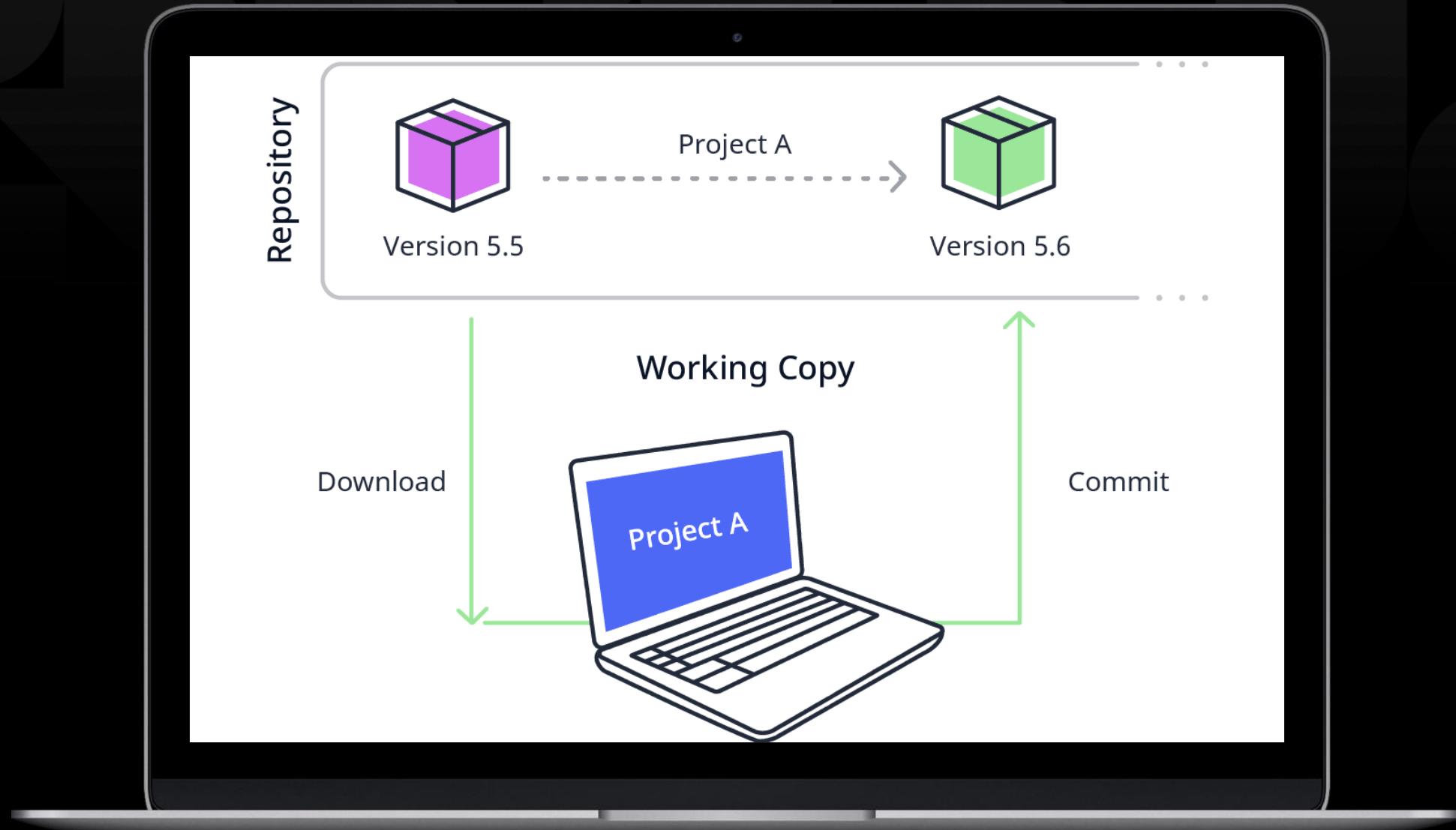


Using and creating navigation layouts

- 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

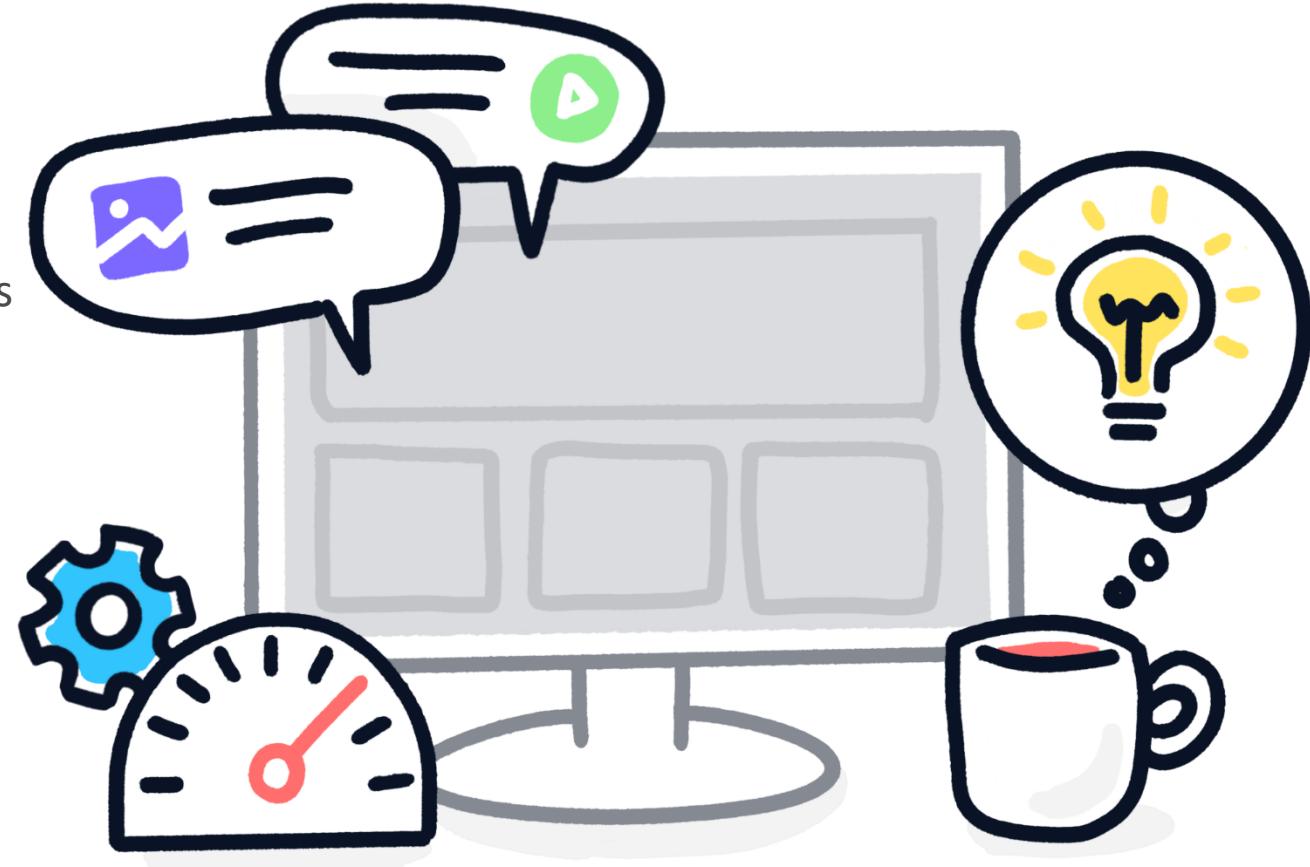
Commit your work

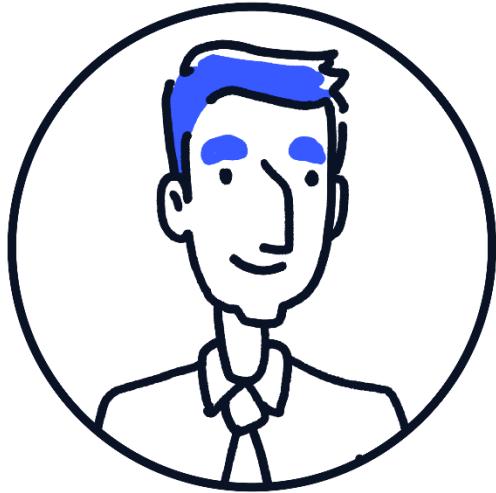


Adding requests

Learning goals

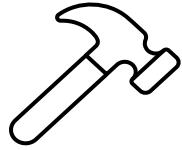
- How to use enumerations in Mendix
- Why you would want to create multiple associations between two entities
- How to create a wizard in Mendix
- How to use functions in your expressions to your advantage
- How to update derived values





User story

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



Configure MyFirstModule

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

Project Security

Security level

Security level Off Prototype / demo Production

Full security is applied. Configure administrator and anonymous access and define user roles and security for forms, microflows, entities, and reports.

Check security Yes No

If there are no other errors, Mendix Studio Pro checks per user role whether forms that are accessible for a certain role only refer to attributes and associations that are accessible for that same role. This assumes that each user role is independent and that users do not need two or more roles to access functionality in your application.

Project status Complete

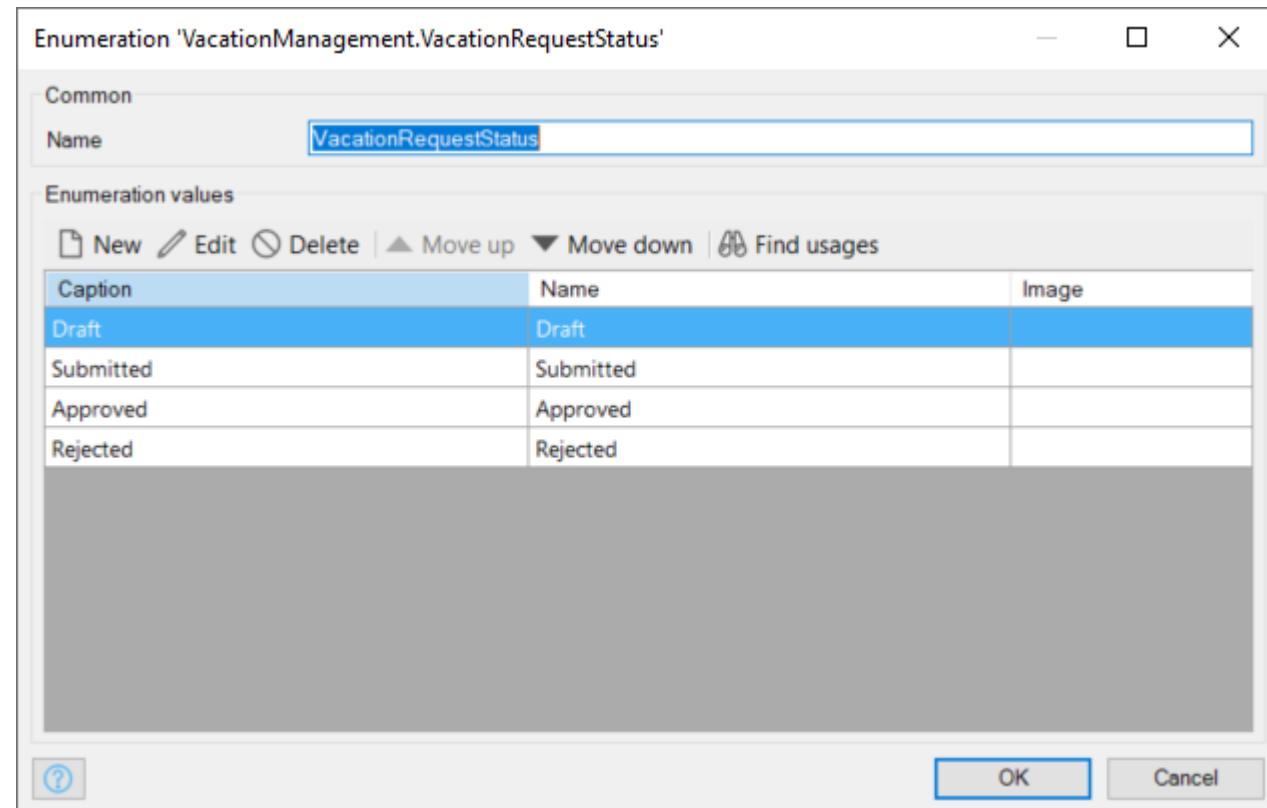
Module status User roles Administrator Demo users Anonymous users Password policy

New Edit Delete

Name	Module roles
Administrator	Administration.Administrator, System.Administrator, VacationManagement.Administrator, GeneralExtentions.Administrator
Anonymous	System.User, GeneralExtentions.Anonymous
Employee	Administration.User, System.User, VacationManagement.Employee, GeneralExtentions.User
Manager	Administration.User, System.User, VacationManagement.Manager, GeneralExtentions.User

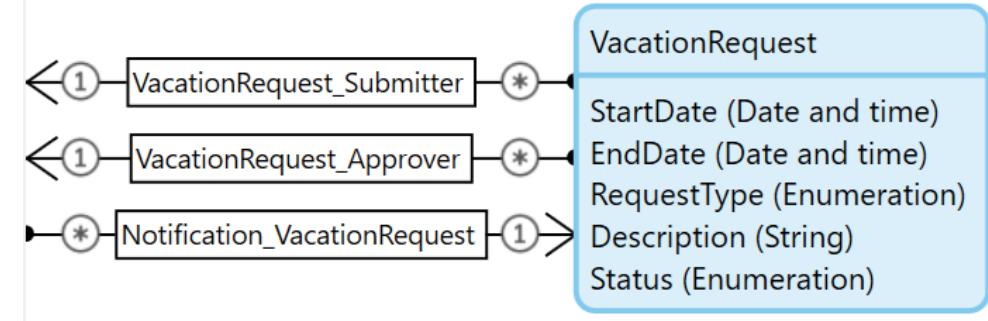
What is an Enumeration Value?

- Pre-defined list of string values
- Can be selected in a **Drop-down** or from **Radio button**
- Can be used to constrain data



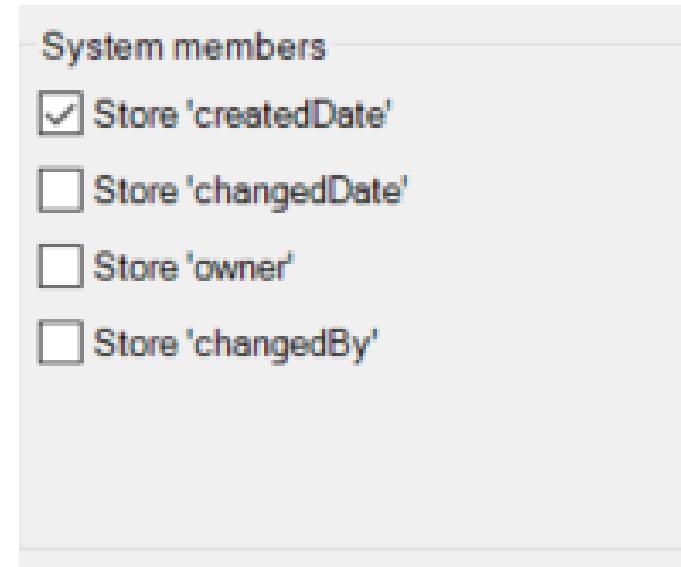
Multiple associations between entities

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



System Members

- createdDate
- changedDate
- owner
- changedBy



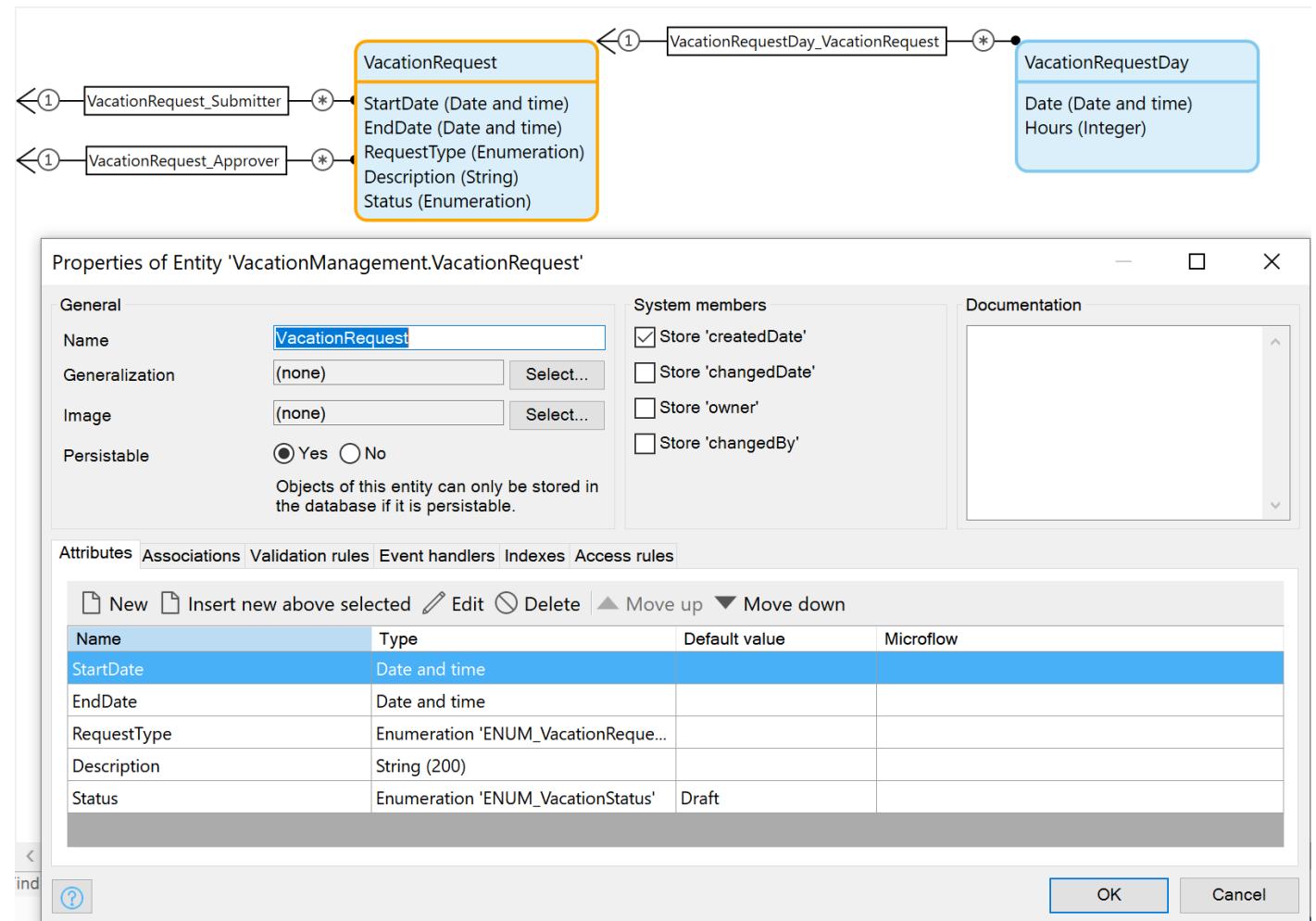
How do we store a vacation request?

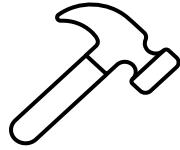
- Start-date and end-date
- Type of request
- Status of the request
- Description
- Number of hours
 - Number of hours per day
 - We need a **VacationRequestDay** entity!



Adding requests: extend Domain Model

- Add **VacationRequest** and **VacationRequestDay** entities
 - Add attributes
- 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





Adding requests: Set Access Rules

- Configure access rules for **VacationRequest** entity
 - Employees should only see their own requests
 - Employees cannot edit the Status
 - Managers can only edit Status
 - Employee and Manager should not have write access to the associations

Properties of Entity 'VacationManagement.VacationRequest'

Module roles	Create	Delete	Member access	XPath constraint
Employee	Yes	No	Full Read, Limited Write	[VacationManagement.VacationRequest_S...]
Manager	No	No	Full Read, Limited Write	
Administrator	Yes	Yes	Full Read, Full Write	

General System members Documentation

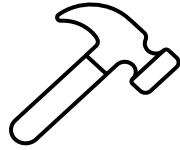
Name: VacationRequest Generalization: (none) Image: (none) Persistable: Yes

System members:
 Store 'createdDate'
 Store 'changedDate'
 Store 'owner'
 Store 'changedBy'

Attributes Associations Validation rules Event handlers Indexes Access rules

New Edit Delete Duplicate

OK Cancel



Adding requests: Set Access Rules

- Configure access rules for **VacationRequestDay** entity
 - Employees should only be able to change hours
 - Managers should only be able to read
 - Manager and Employee should only be able to read the association

Properties of Entity 'VacationManagement.VacationRequestDay'

Module roles	Create	Delete	Member access	XPath constraint
Employee	No	No	Full Read, Limited Write	
Manager	No	No	Full Read, No Write	
Administrator	Yes	Yes	Full Read, Full Write	

General

Name: VacationRequestDay

Generalization: (none)

Image: (none)

Persistable: Yes

System members:

- Store 'createdDate'
- Store 'changedDate'
- Store 'owner'
- Store 'changedBy'

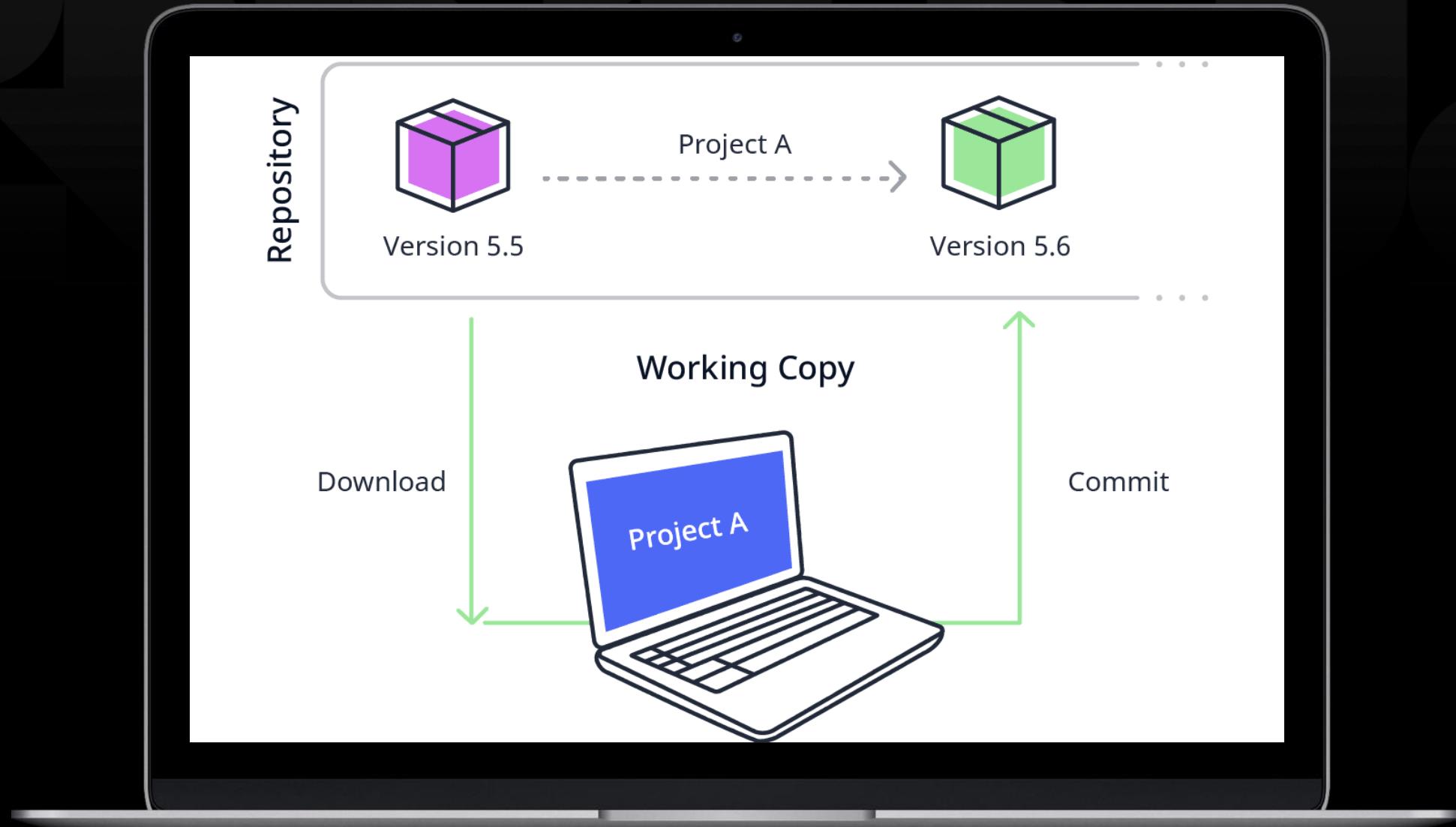
Documentation:

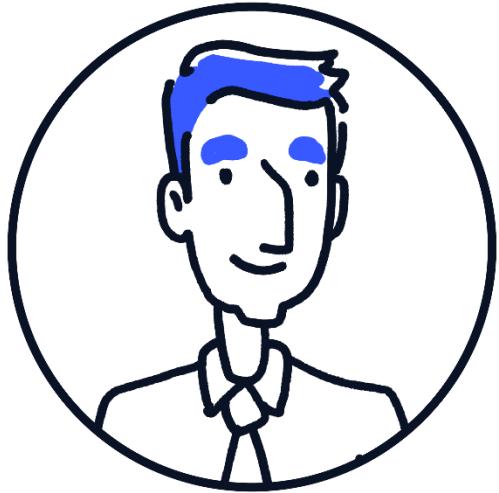
Attributes Associations Validation rules Event handlers Indexes Access rules

New Edit Delete Duplicate

OK Cancel

Commit your work





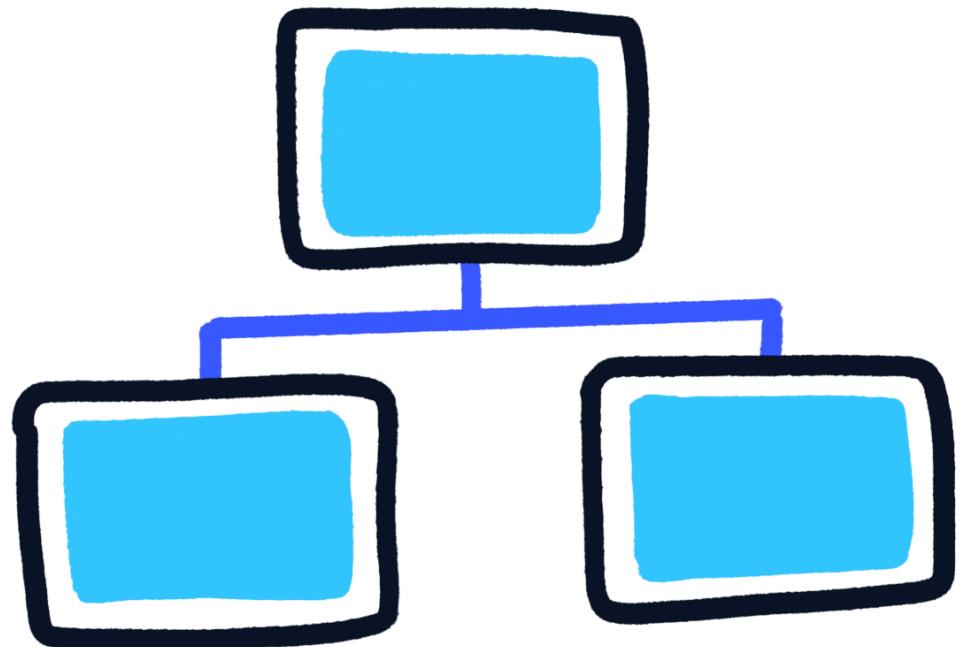
User story

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 than
<=	Less than or equal to
>	More than
>=	More than or equal to
or	Or
and	And



What is an XPath operator?

Operator	Description	Example	
=	Equal to	price = 9.80	true if price is 9.80, false if price is 9.90
!=	Not equal to	price != 9.80	true if price is 9.90, false if price is 9.80
<	Less than	price < 9.80	true if price is 9.70, false if price is 9.80
<=	Less than or equal to	price <= 9.80	true if price is 9.80, false if price is 9.90
>	Greater than	price > 9.80	true if price is 9.90, false if price is 9.80
>=	Greater than or equal to	price >= 9.80	true if price is 9.80, false if price is 9.70
or	Or	price = 9.80 or price = 9.70	true if price is 9.80, false if price is 9.60
and	And	price = 9.80 and amount = 1	true if price is 9.80 and amount is 1, false if price is 9.70 and amount is 1, false if price is 9.80 and amount is 2, false if price is 9.70 and amount is 2



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
- Match the design

The screenshot shows the Mendix Home Page Editor interface. It features a layout grid with two rows. The top row contains a 'Vacation Management' section with a green 'New vacation request' button and an 'Upcoming time off' list view. The 'Upcoming time off' view is connected to the 'VacationRequest' entity via XPath and sorts by Start Date. The bottom row contains an 'All vacation requests' list view, also connected to the 'VacationRequest' entity via database and sorted by Start Date. Both views have search filters for Description.

Vacation Management

New vacation request

Upcoming time off

[VacationRequest, by XPath]
Sort order: StartDate (ascending)
Search on: [Description]

Auto-fit content	Auto-fit content	Auto-fit content
	{StartDate} - {EndDate}	{Description}
	{Status}	

Load more...

All vacation requests

[VacationRequest, from database]
Sort order: StartDate (ascending)
Search on: [Description]

Auto-fit content	Auto-fit content	Auto-fit content
	{StartDate} - {EndDate}	{Description}
	{Status}	



Adding requests: Enum widget

- Add an image collection using the icons provided



- Download the **Enum Toggle** widget from the appstore
- Add the icons to the Enum widget
- Match the design

Vacation Management

Upcoming time off

	{StartDate} - {EndDate}	{Description}

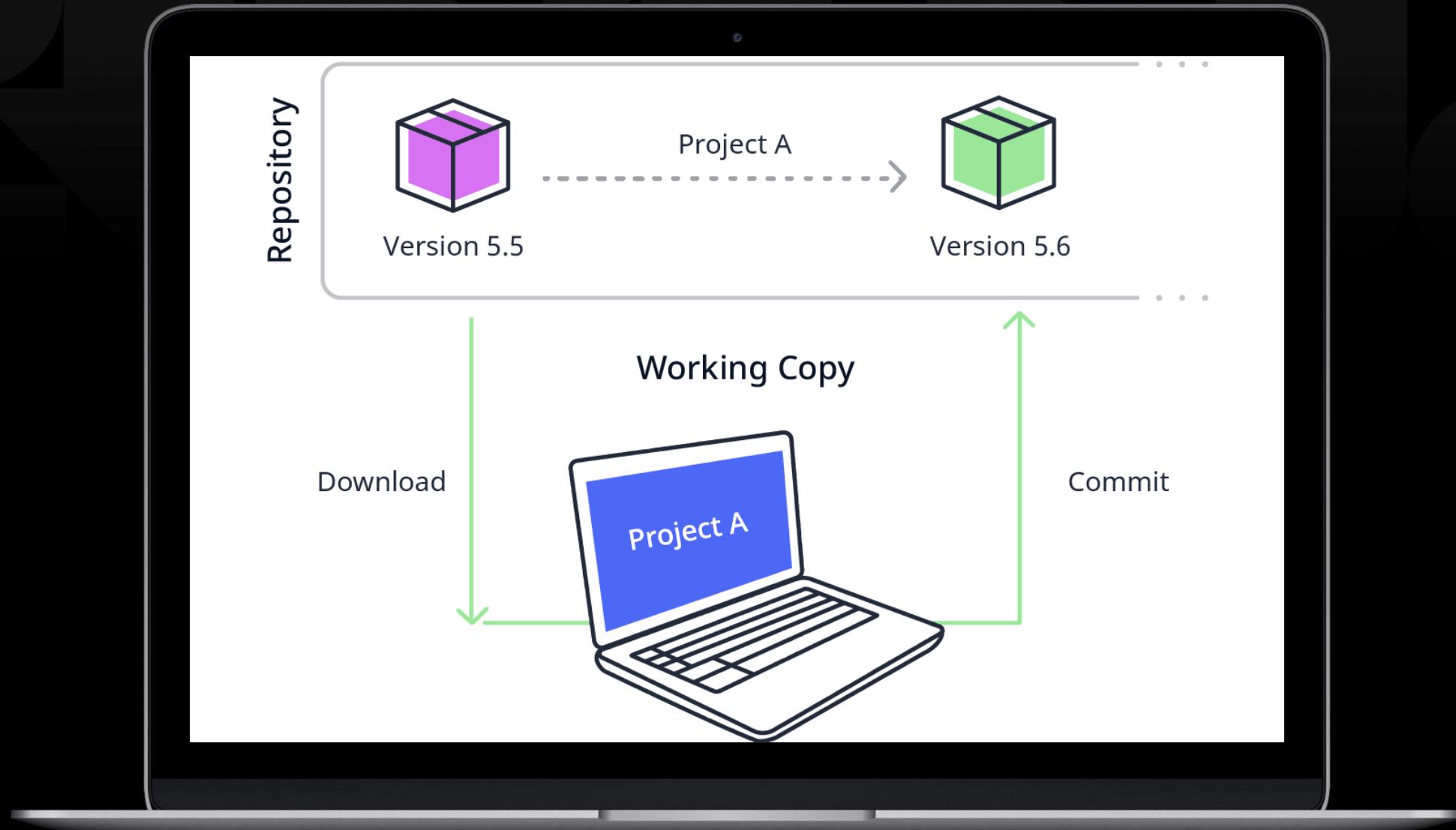
 Load more...

All vacation requests

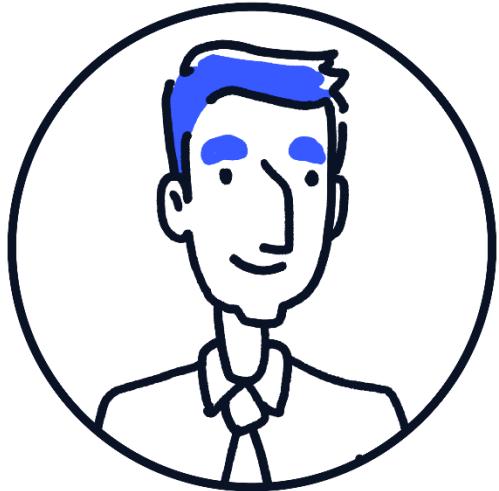
	{StartDate} - {EndDate}	{Description}

 Load more...

Commit your work



User story



As an Employee, I want to indicate for each day in my request how many hours of vacation time I want to take so I don't have to use my vacation hours for weekends and national holidays

What do we need?

Requirements:

- Ability to enter start and end date for time off request
- For each day specify the hours of holiday
- Ability to review request before submitting

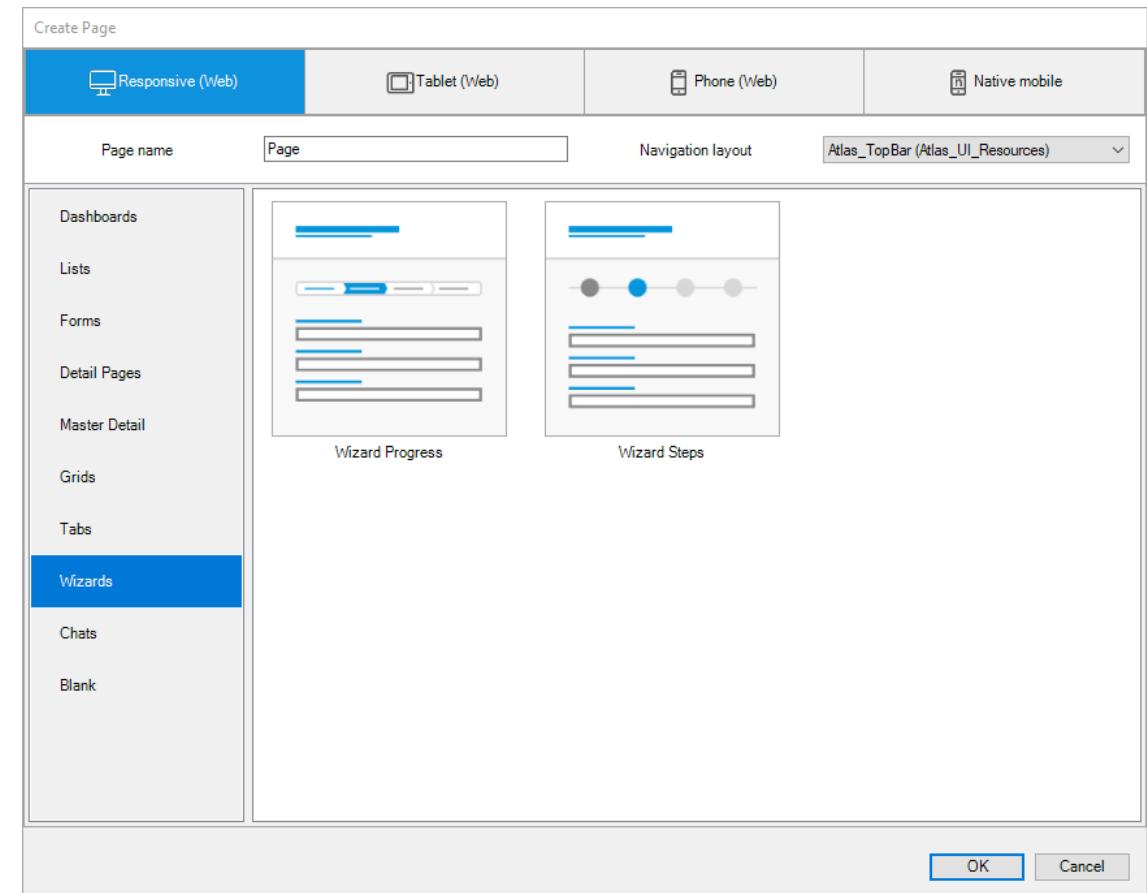
How do we do this in Mendix?

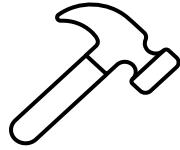
- Use a wizard with 3 pages
 1. Create request
 2. Create days
 3. Review and submit



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





Adding requests: Wizard pages

- Create wizard pages in new folder:
VacationRequest_Wizard_Step1
VacationRequest_Wizard_Step2
VacationRequest_Wizard_Step3
 - Atlas_TopBar
 - Wizard Progress
- Delete step 4
- Set design property classes accordingly for each step:
 - wizard-step wizard-step-visited
 - wizard-step wizard-step-active
 - wizard-step
- Extract **Header** as a Snippet

The screenshot shows the Mendix 'Edit Container' dialog for 'container10'. The 'Appearance' tab is selected. In the container preview, there are three steps labeled 'Step 1', 'Step 2', and 'Step 3'. The 'Step 1' button is highlighted with an orange border. Below the steps is a 'Form block title' section with fields for 'Name' and 'Phonenumber', both currently showing '[No attribute selected]'. On the right, the 'Design properties' section includes fields for 'Spacing top', 'Spacing bottom', 'Spacing left', and 'Spacing right', each with a corresponding gray slider bar. The 'Resulting classes' field contains 'wizard-step wizard-step-visited'. Under the 'Common' section, the 'Class' field is set to 'wizard-step wizard-step-visited' and the 'Style' field is empty. At the bottom are 'OK' and 'Cancel' buttons.



Adding requests: Configure pages

- On each page:
 - Give **Employee** access
 - Create and use snippet **WizardHeader** as a header
 - Connect data view to **VacationRequest**
 - Empty data view contents
 - Create appropriate navigation buttons at bottom
(don't worry about making them functional yet)

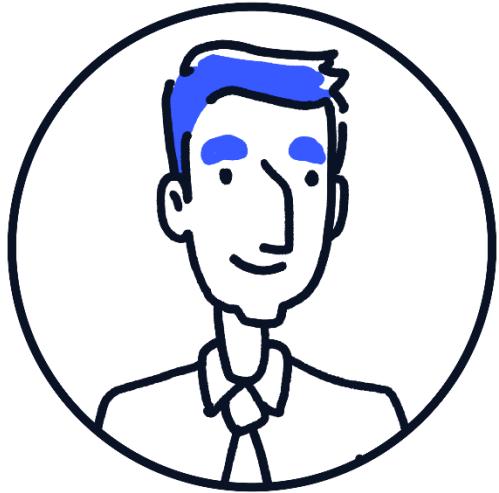
The screenshot shows the Mendix studio interface for configuring a wizard page. At the top, there's a header snippet named [VacationManagement.Wizard_Header] with an 'Auto-fill' button. The main content area has a title 'Request time off' and a subtitle 'Complete the three steps below to submit a request'. On the left, there are three steps labeled 'Step 1', 'Step 2', and 'Step 3'. Step 1 contains a 'Form block title' and a data view with a page parameter [VacationRequest]. It includes 'Cancel Request' and 'Next' buttons. Step 2 features the 'NORTH SEA SHIPBUILDING' logo. Step 3 has a large blue arrow pointing right with the text 'Step 1', 'Step 2', and 'Step 3' above it. Below the steps, there's a 'Form block title' and a 'DATA VIEW CONTENT' section with 'Previous' and 'Submit' buttons.



Adding requests: Wizard step 1

- Match the design in the image
- Set the Cancel button to **Cancel**
- Set **Delete behaviour** to cascading delete for **VacationRequestDays** when a **VacationRequest** is deleted
- Set the Next button to Call a microflow
 - Create a new microflow **ACT_VacationRequest_GenerateVacationDays** – we will configure this later
- Configure the **New vacation request** button on the homepage to create an object and go to Step1

The screenshot shows the 'Request time off' wizard Step 1. At the top, there's a navigation bar with the North Sea Shipbuilding logo, Home, Account Overview, Manage My Account, and Log out links. Below the navigation, the title 'Request time off' is displayed with the sub-instruction 'Complete the three steps below to submit a request'. A progress bar at the top right indicates 'Step 1' is active, followed by 'Step 2' and 'Step 3'. The main section is titled 'Add requests' and contains fields for 'Start date' (with a calendar icon), 'End date' (with a calendar icon), 'Request type' (radio buttons for PTO, Special Leave, Parental Leave, and Unpaid), and a 'Description' text area. At the bottom are 'Cancel request' and 'Next' buttons.



User story

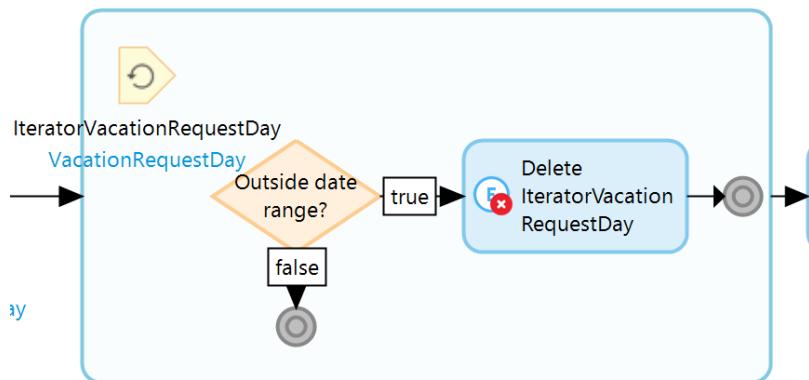
As an Employee, I want to indicate for each day in my request how many hours of vacation time I want to take so I don't have to use my vacation hours for weekends and national holidays

Next task: Create a microflow to generate vacation days

Working with lists

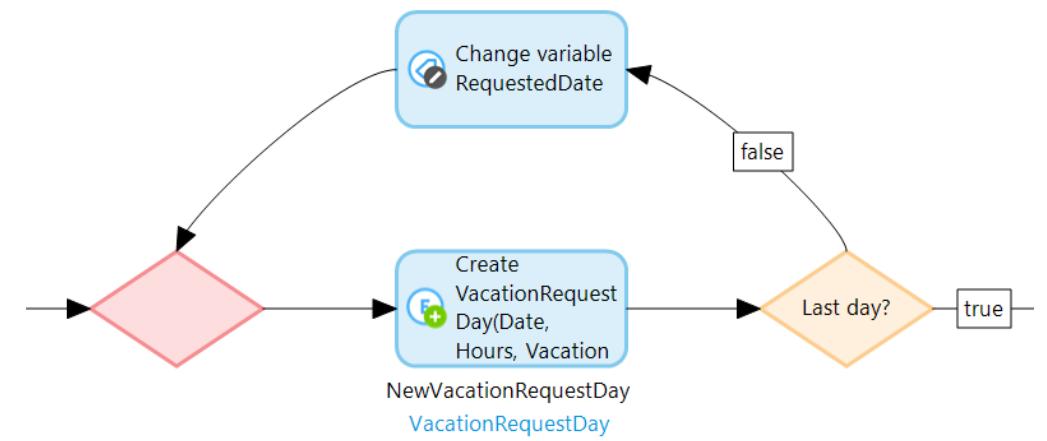
Loop activity

- 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



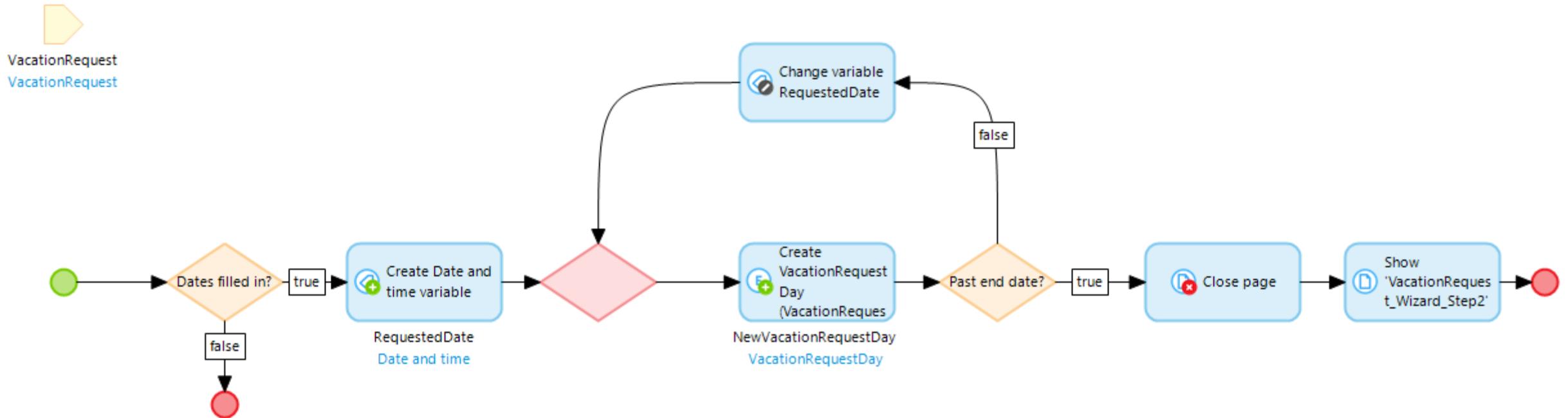
While loop

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





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 **Show page** and select **VacationRequest_Wizard_Step1** as the target page

The image shows two views of a Mendix application. On the left, the 'VacationManagement.Wizard_Header' page is displayed, featuring a header with the North Sea logo and navigation links for Home, Account Overview, Manage My Account, and Log out. Below the header, a title 'Request time off' and a subtitle 'Complete the three steps below to submit a request' are shown. Three large rectangular input fields are labeled 'Step 1', 'Step 2', and 'Step 3'. Below these fields, a section titled 'Form block title' contains a parameter 'StartDate - EndDate' and a search bar. To the right of the search bar, there are four rows of date and hour inputs, each consisting of a date field, a 'Hours' button, and a 'Hours' input field. At the bottom of this section are 'Go back' and 'Next' buttons. On the right, the 'Request time off' page is shown. It has a similar header and subtitle. A horizontal progress bar at the top indicates 'Step 1' is active, 'Step 2' is the current step, and 'Step 3' is the next step. Below the progress bar, the 'Form block title' section is identical. The main content area displays a list view of 'VacationRequestDay' items, with a specific item highlighted. This item has a date field and a 'Hours' button. At the bottom of this area are 'Previous' and 'Next' buttons.



Try it out!

Run locally and test what you've built.

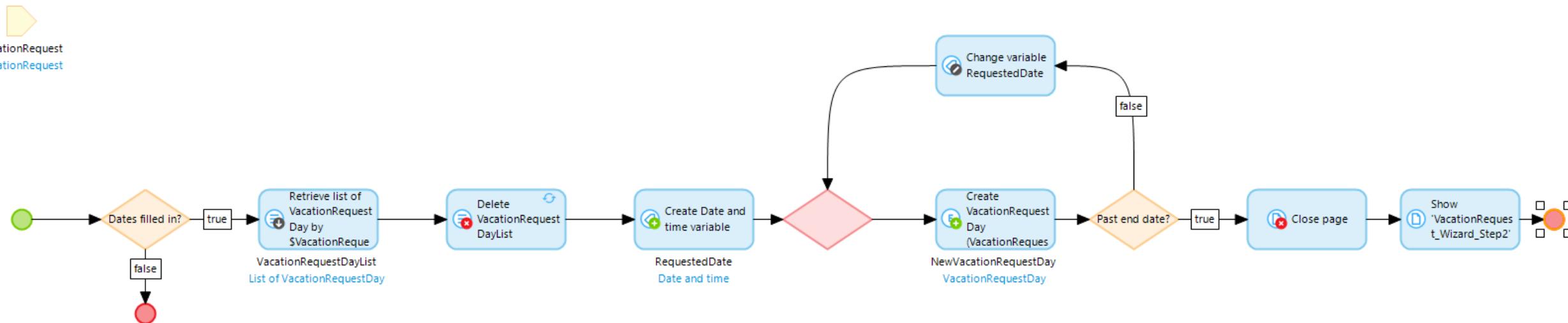


Not generating duplicate days

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



Not generating duplicate days





Generating just weekday hours

Set hours requested to 0 for weekends

Hints:

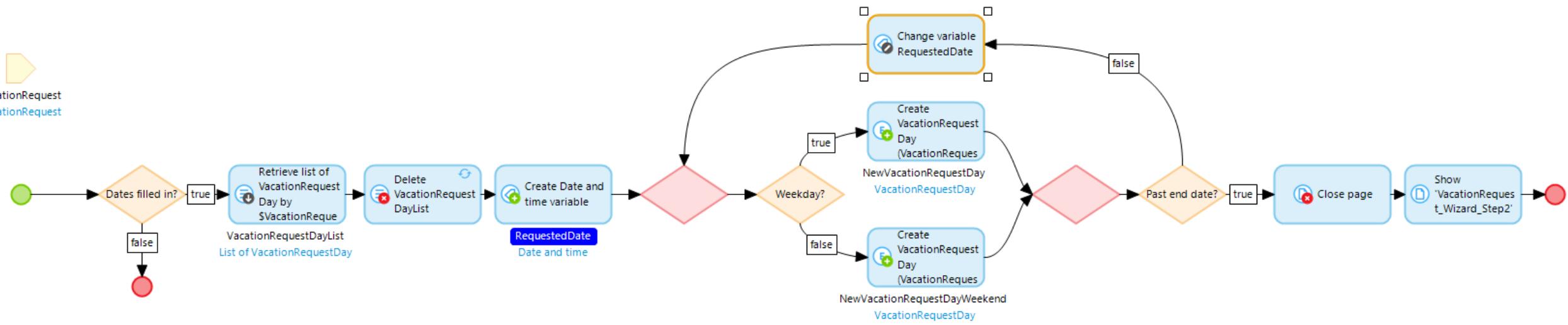
Use `parseInteger()`

and `formatDateTimeUTC()`

with the `u` modifier to get a number for day of the week



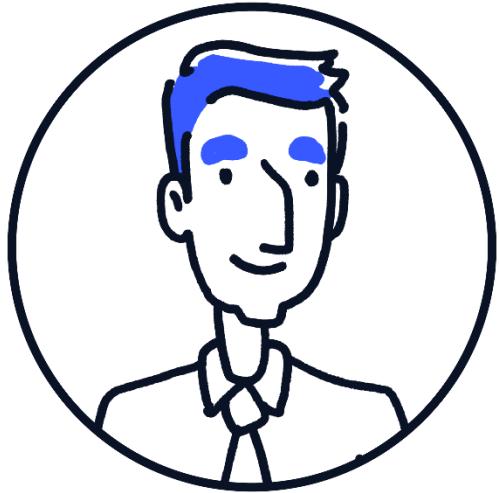
Generating just weekday hours





Try it out!

Run locally and test what you've built.



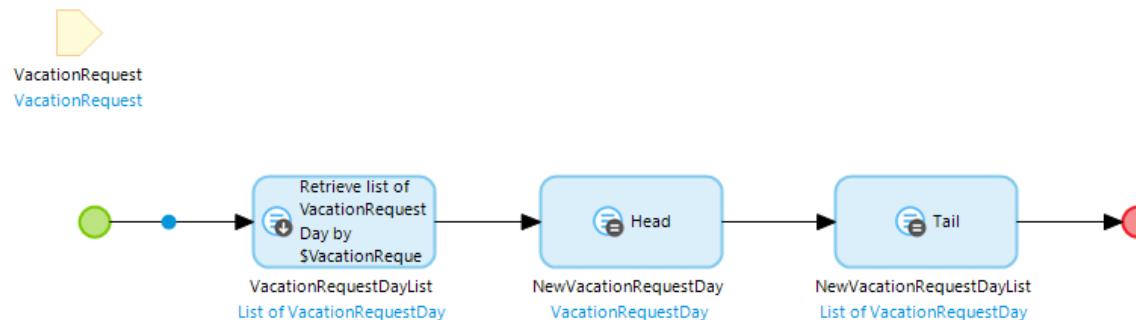
User story

As an Employee, I want to indicate for each day in my request how many hours of vacation time I want to take so I don't have to use my vacation hours for weekends and national holidays

Next task: Keep existing days

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

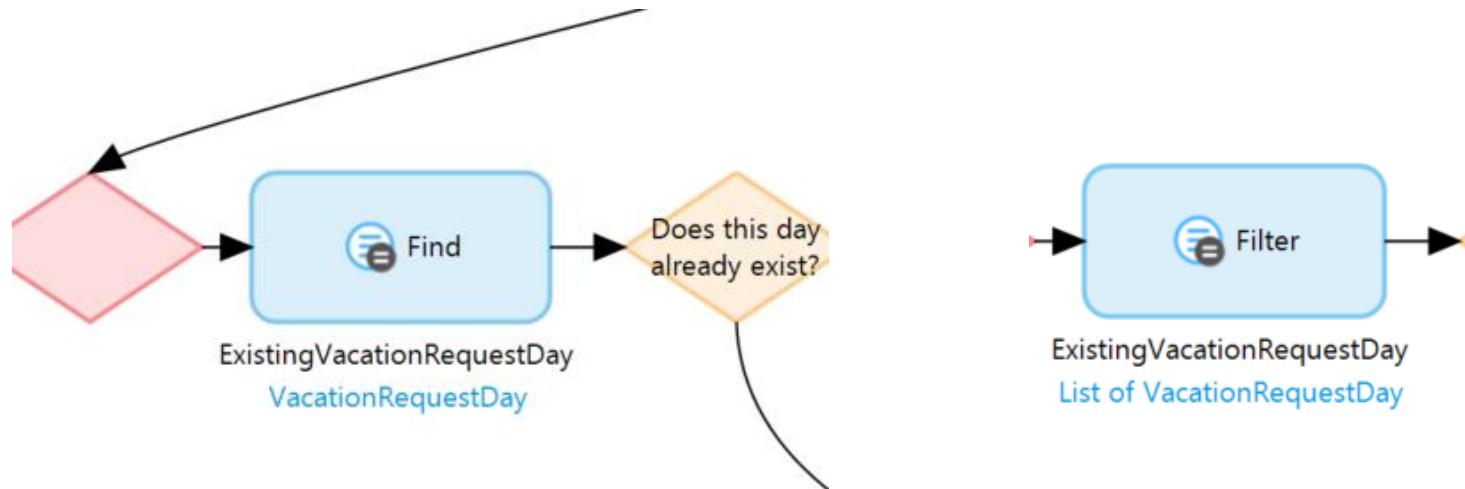


List operation: Binary

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



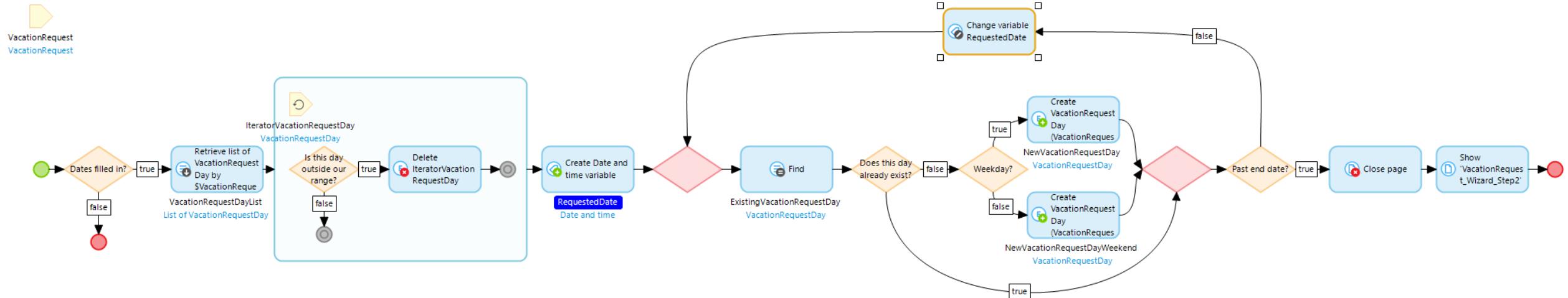


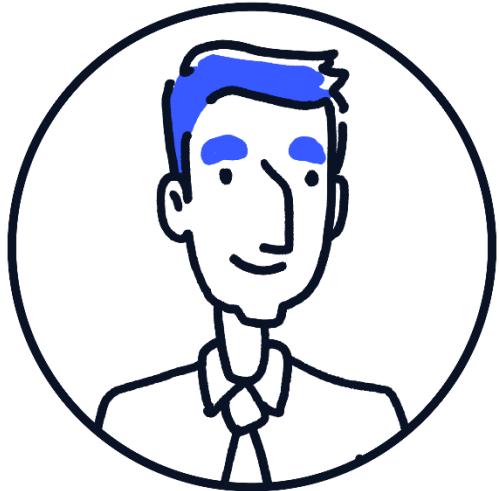
Keep existing days

- Generate vacation days without recreating existing ones
- 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



Keep existing days





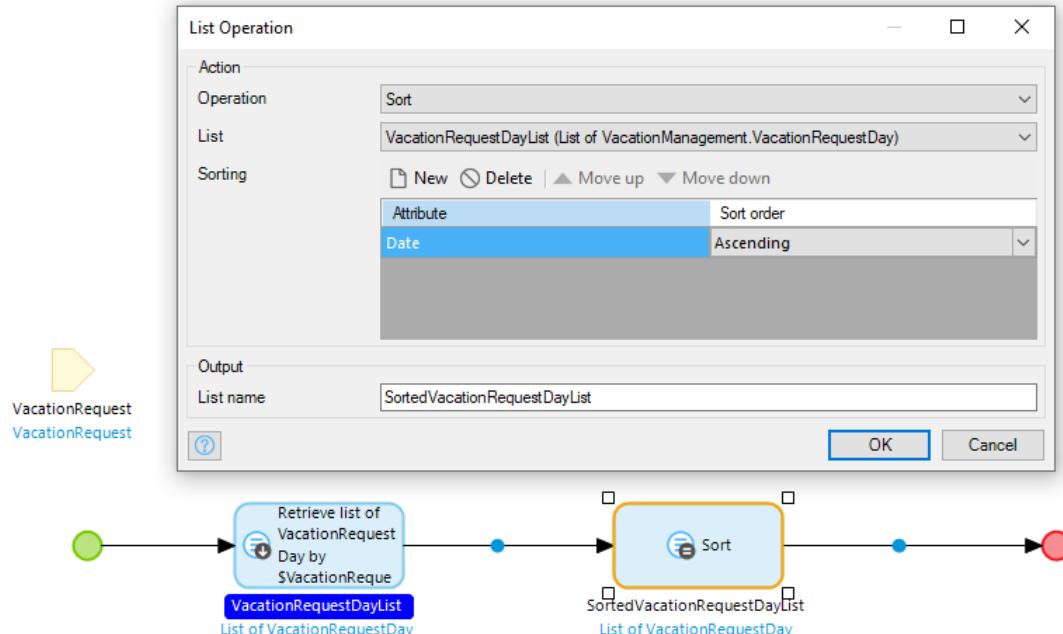
User story

As an Employee, I want to indicate for each day in my request how many hours of vacation time I want to take so I don't have to use my vacation hours for weekends and national holidays

Next task: Sort vacation days

List operation: Sort

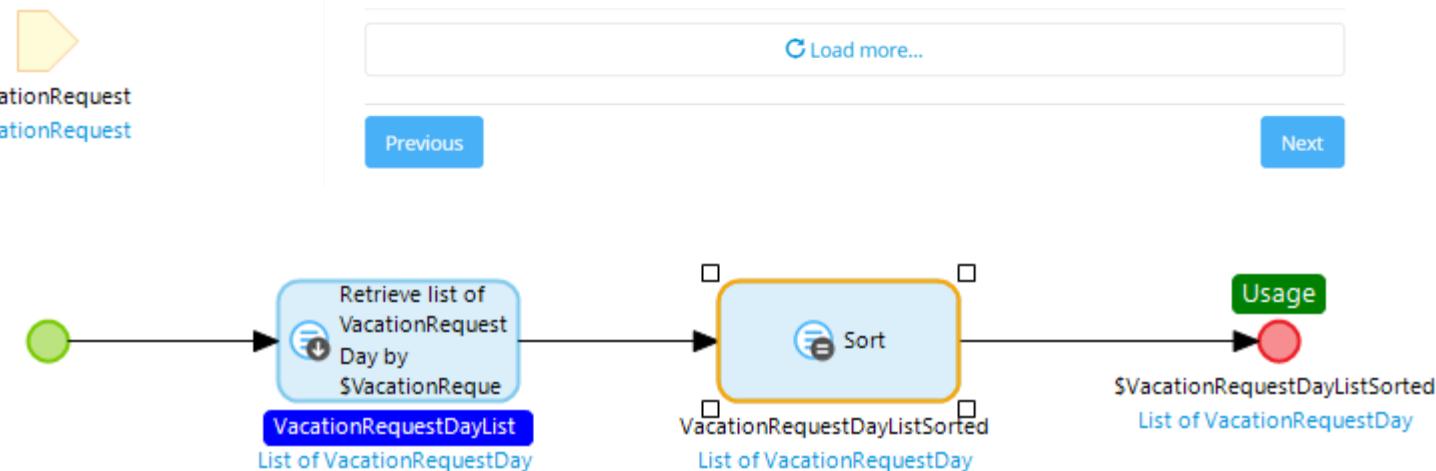
Operation	Description	Return Type
Sort	<p>Allows you to sort a list based on a number of attributes. The attributes are ordered to determine their priority while sorting. The input list remains in its original order while the sorted list is stored with the output name.</p>	List





Adding requests: Wizard step 2

- Edit the VacationDay list view
- Set the **Data Source** to **Microflow**
- Create a microflow called **DS_VacationRequestDay_Sorted**
- Build a microflow that sorts the list of vacation requests and passes it to the list view

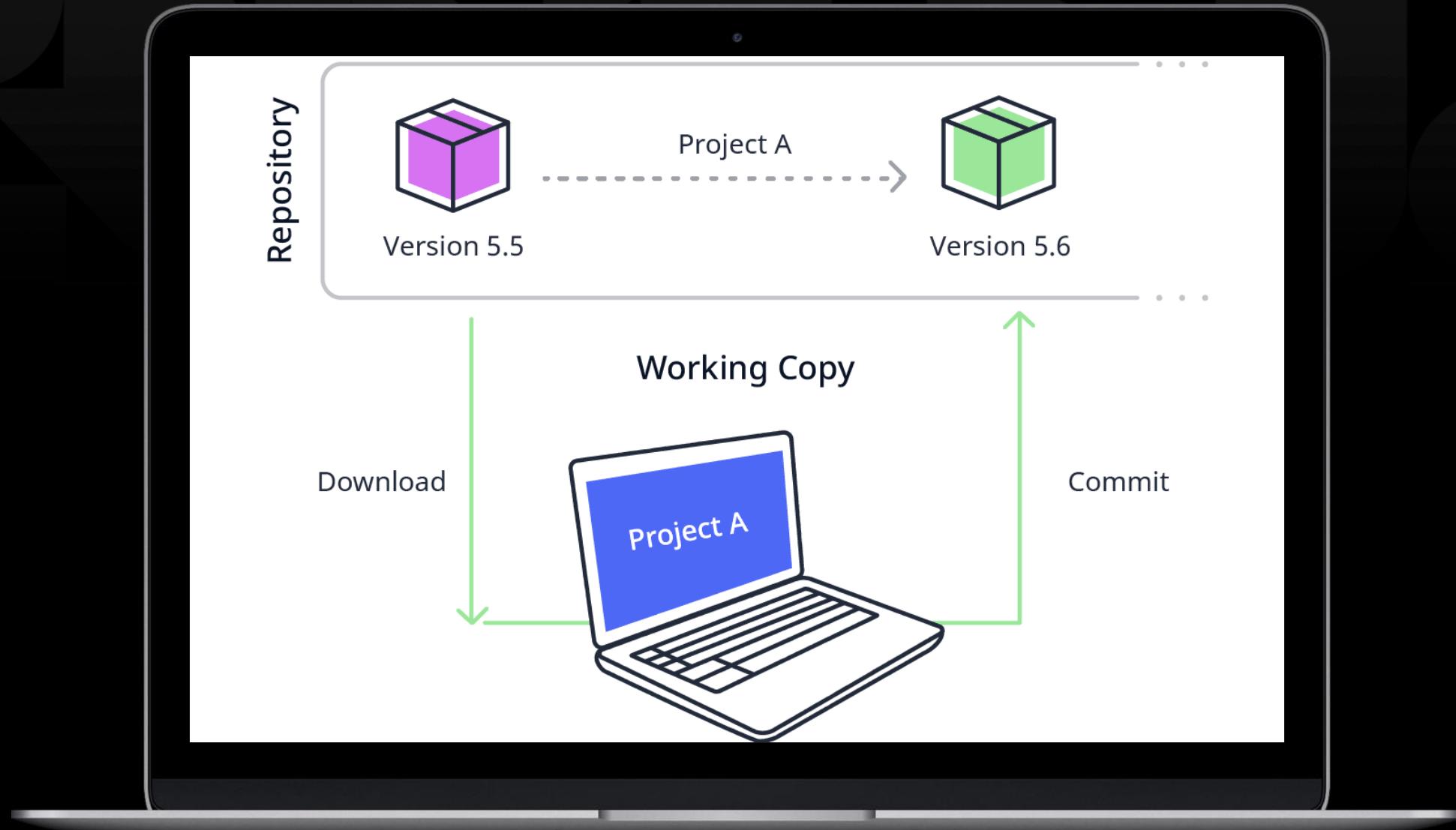


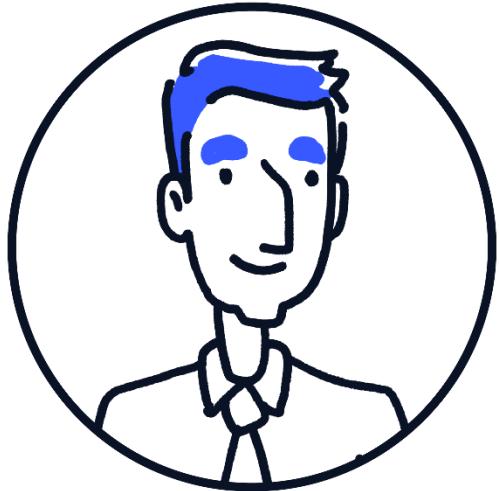


Try it out!

Run locally and test what you've built.

Commit your work





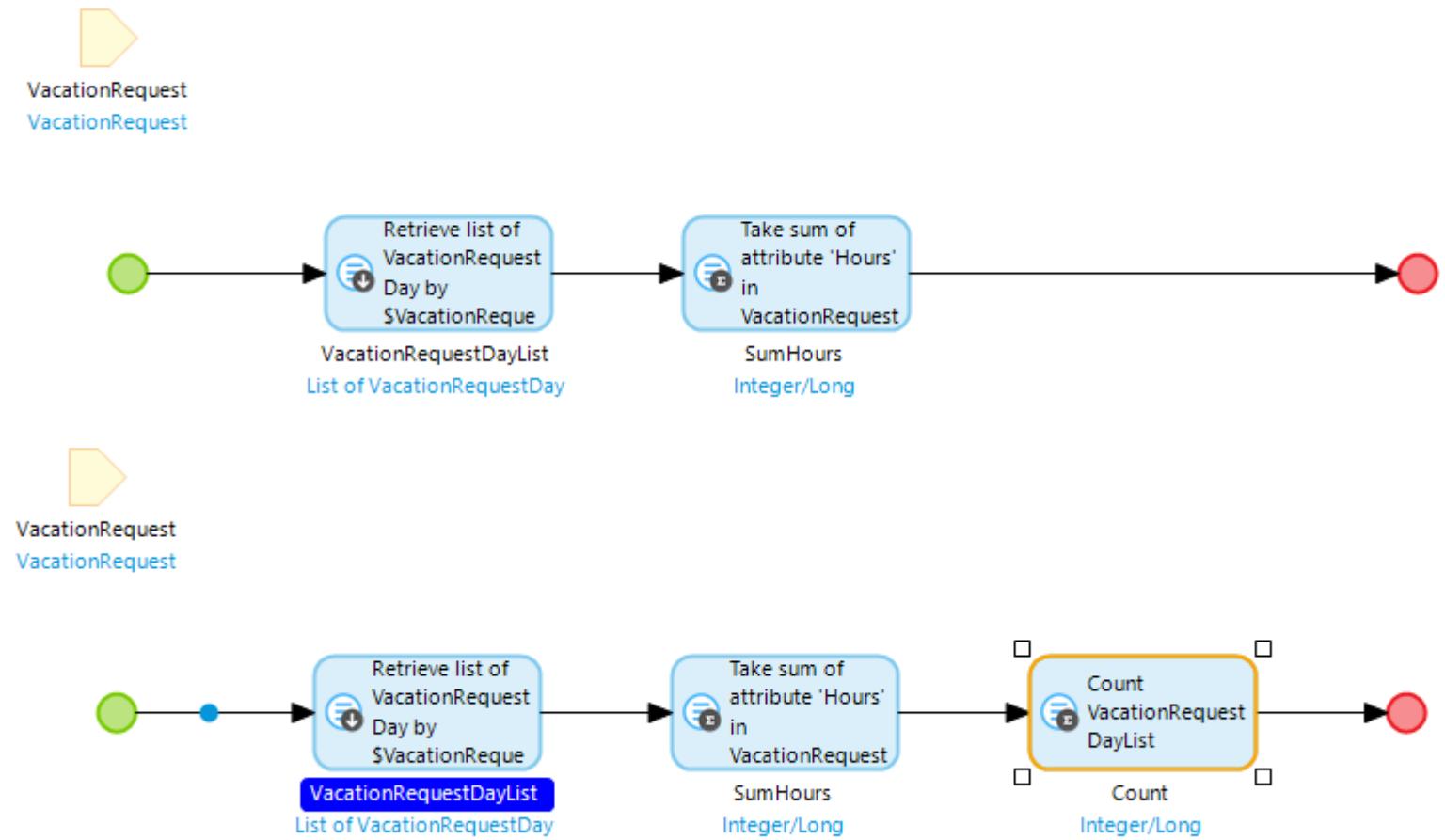
User story

As an Employee, I want to indicate for each day in my request how many hours of vacation time I want to take so I don't have to use my vacation hours for weekends and national holidays

Next task: Configure review page

Aggregate List activity

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



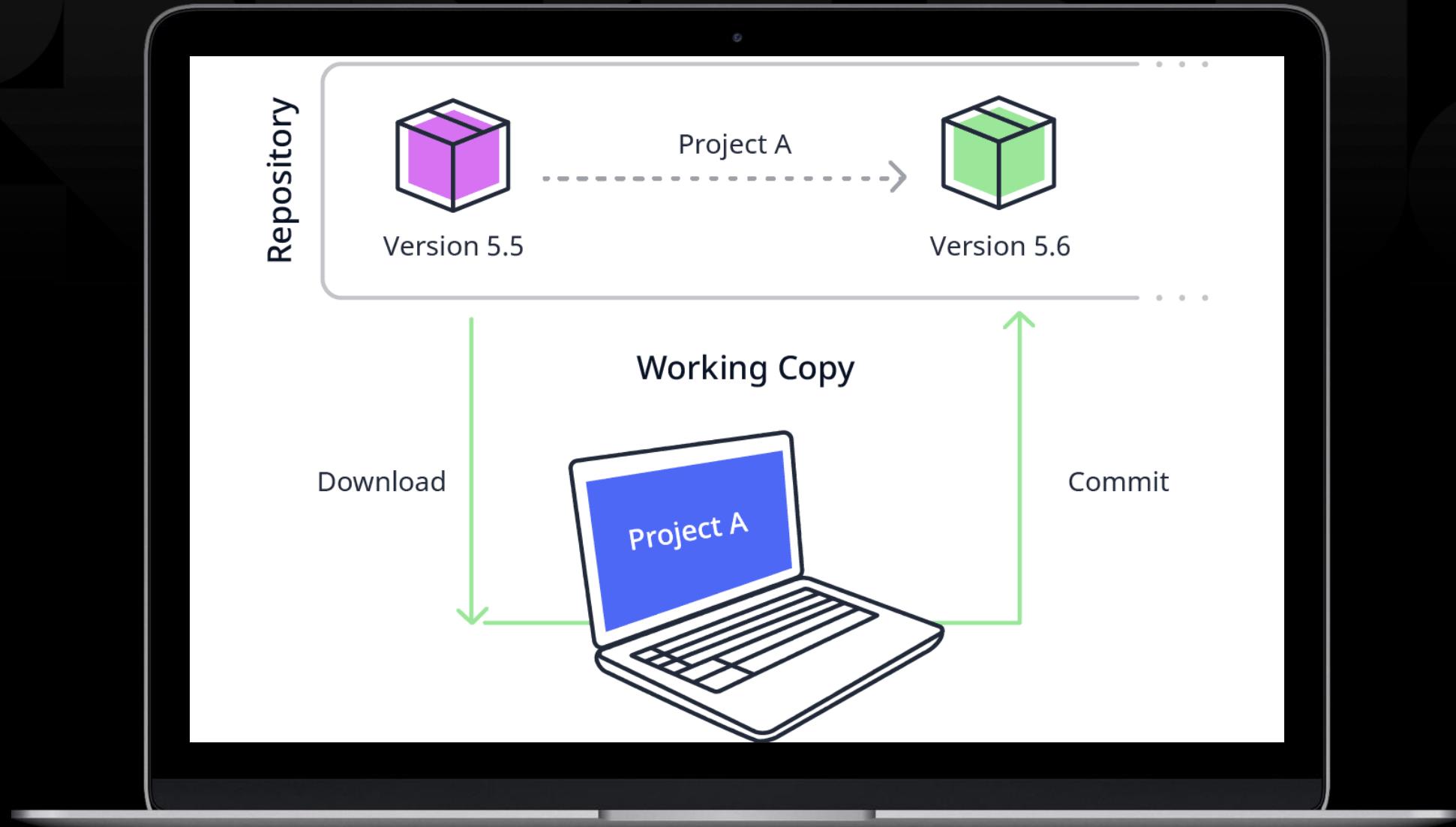


Adding requests: Wizard step 3

- Show request for final review
- Add an **attribute** to the **VacationRequest** entity: **TotalHoursRequested**
- Create a microflow for the next button on page two that calculates the total number of hours requested and sets it on the **VacationRequest** object
- Set the **Previous** button to **Show a page** and select step 2
- Match the page design
- Make navigation buttons functional; take validation into consideration
- In the **ACT_VacationRequest_Submit** microflow you set the **Submitter**, set the **Status** and **Close** the page

The screenshot displays the Mendix interface for creating a vacation request. On the left, the 'VacationManagement.VacationRequest_Wizard_Header' page is shown, featuring three steps: Step 1, Step 2, and Step 3. Step 3 is currently active, titled 'Review your request'. It contains fields for Start date ([StartDate]), End date ([EndDate]), Request type (radio buttons for PTO, Special Leave, Parental Leave, Unpaid), and Description ([Description]). Below this is a table with a single row showing a green circular icon and the label 'Total hours (TotalHours)'. At the bottom are buttons for 'Previous', 'Cancel request', and 'Submit'. On the right, the 'NORTH SEA SHIPBUILDING' logo is visible above the 'Request time off' page. This page also has three steps: Step 1, Step 2, and Step 3. Step 3 is active, showing the same input fields as the header page. A large green box on the right displays a clock icon and the text '{TotalHours}'.

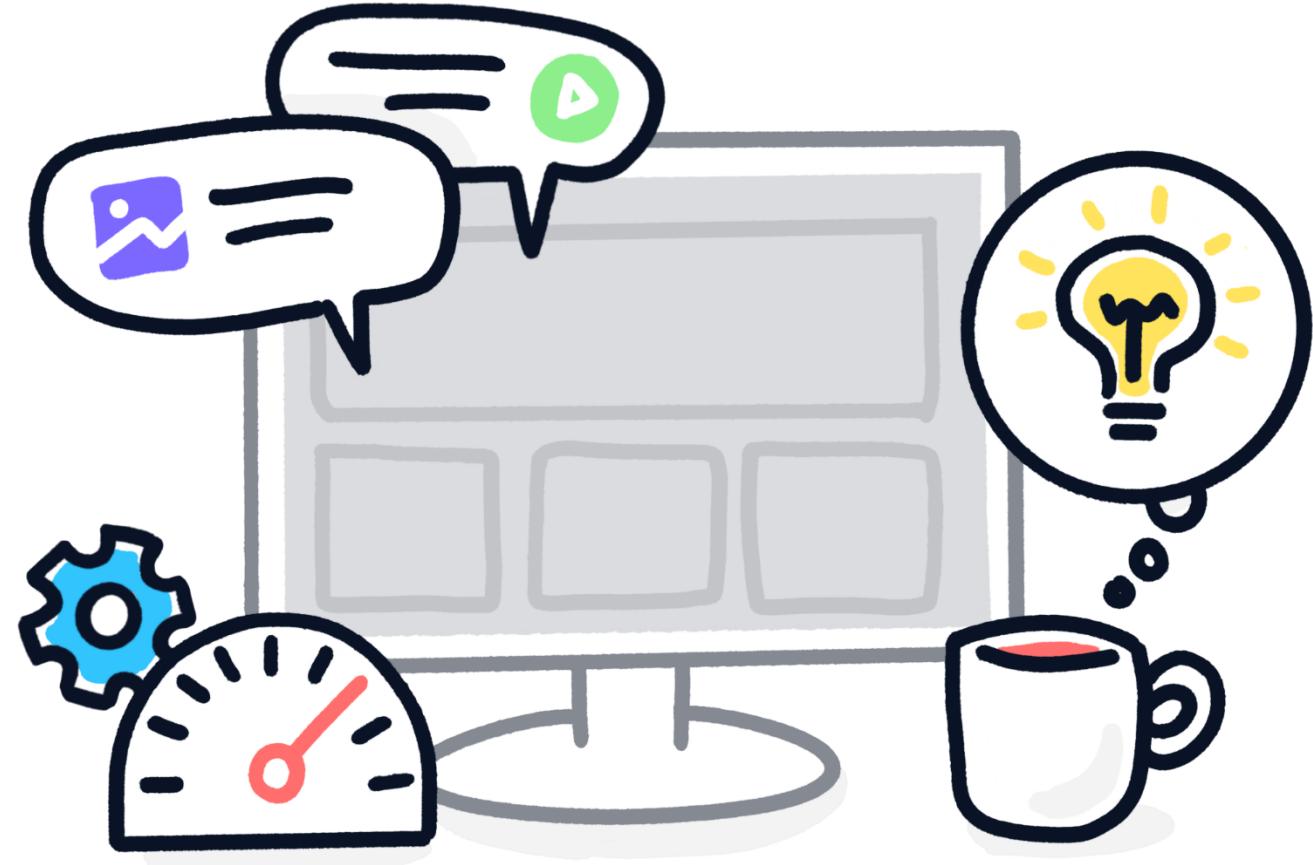
Commit your work

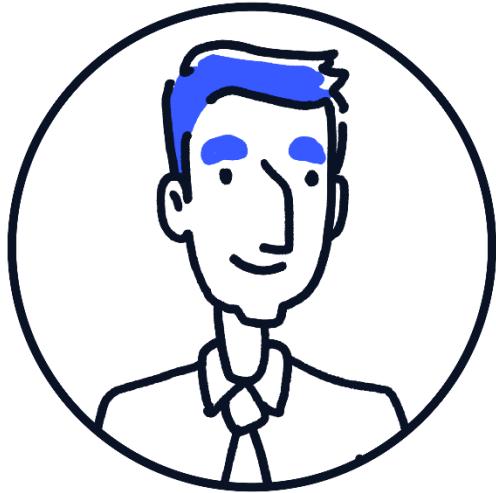


Approving requests

Learning goals

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

The screenshot shows a navigation bar at the top with links: Home, Manage Accounts, Approve requests, and Manage my account. Below the navigation is a 'Log out' button. The main content area is titled 'Vacation request overview'. It features three tabs: 'New requests' (selected), 'Approved requests', and 'Rejected requests'. Below the tabs is a search bar with 'Search' and 'Details' buttons. To the right of the search bar are navigation arrows for pagination. A table below lists vacation requests with columns: Request type, Start date, End date, and Description. Each row contains placeholder text for these fields.

Request type	Start date	End date	Description
[RequestTy...]	[StartDate]	[EndDate]	[Description]
[RequestTy...]	[StartDate]	[EndDate]	[Description]
[RequestTy...]	[StartDate]	[EndDate]	[Description]
[RequestTy...]	[StartDate]	[EndDate]	[Description]
[RequestTy...]	[StartDate]	[EndDate]	[Description]

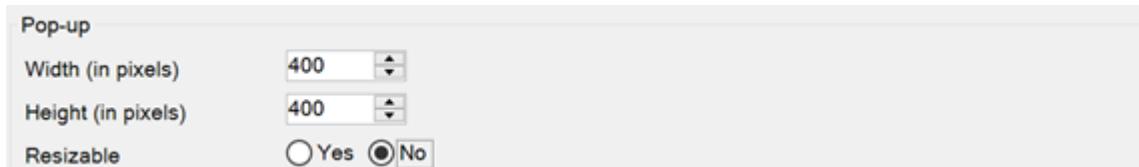
Working with popups

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. Make sure only **Managers** have access to the microflows

Vacation Request Details ×

Employee
[.../Account/FullName]

Request type
[RequestType]

Start date
[StartDate]

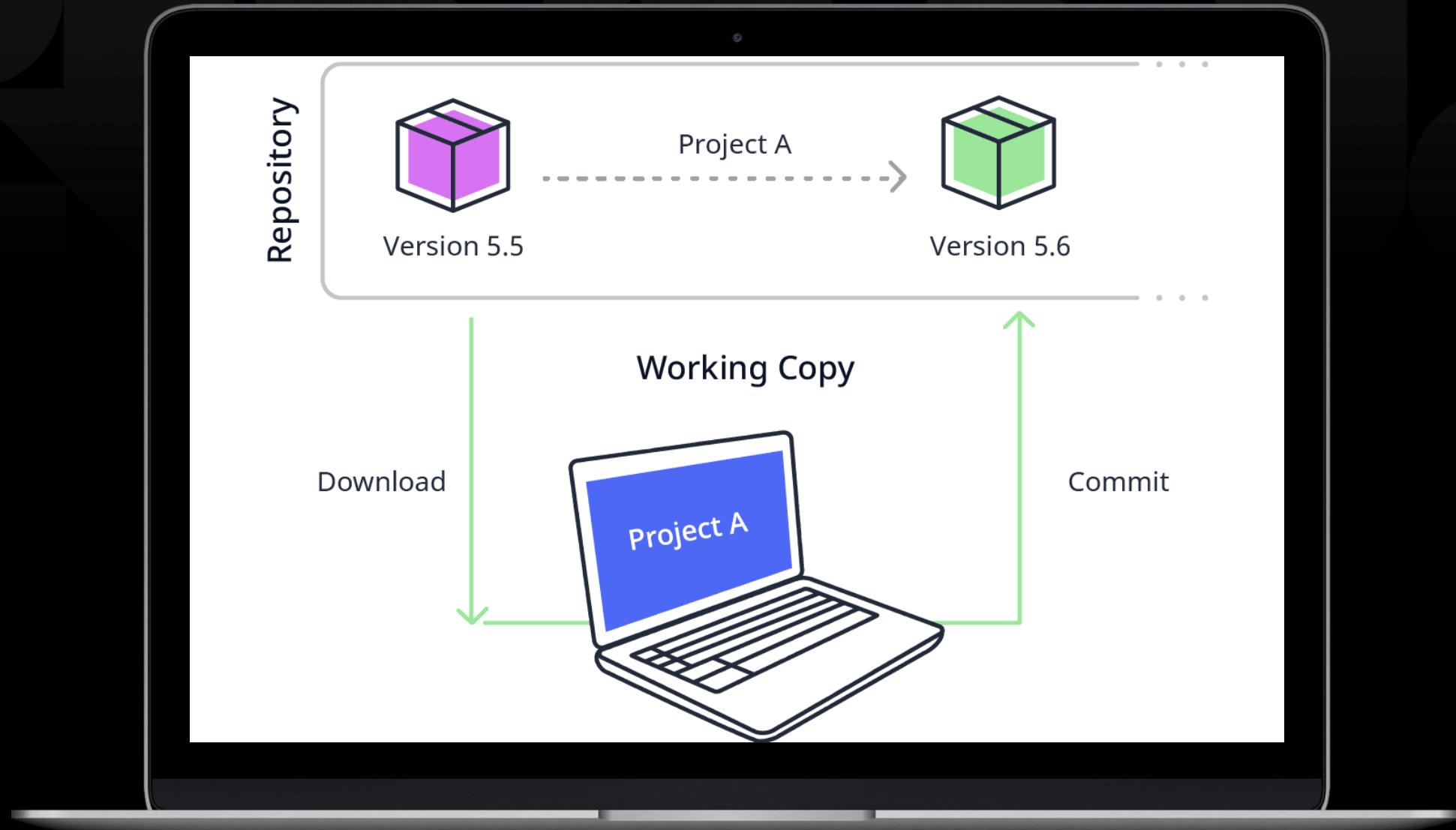
End date
[EndDate]

Description
[Description]

Button

Approve Reject Cancel

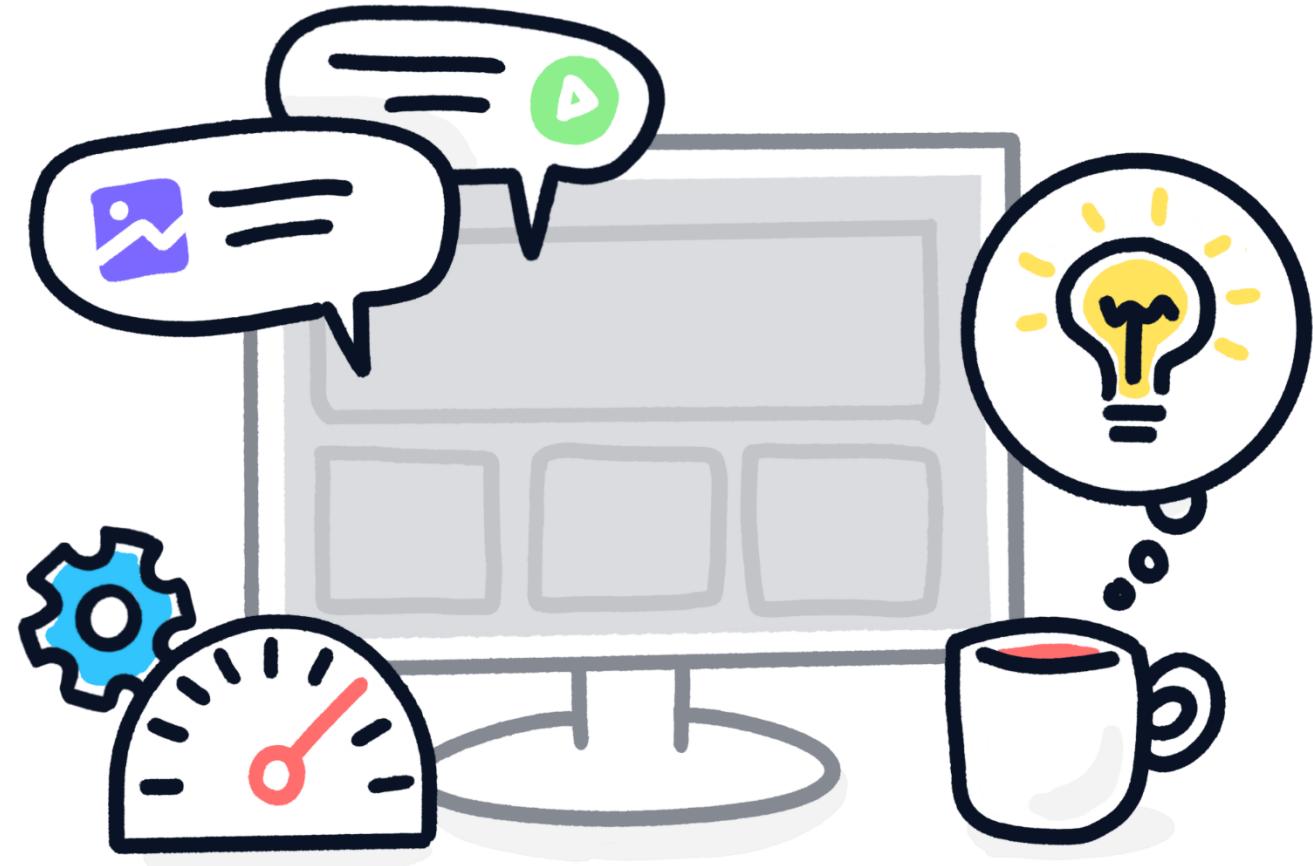
Commit your work

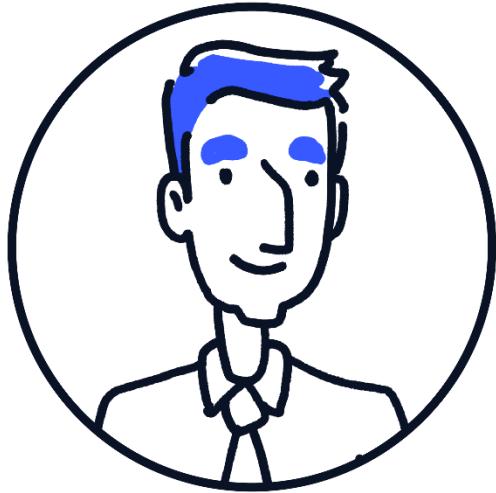


Notifications

Learning goals

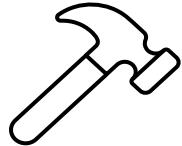
In this module we will use the knowledge we gained during this course to build a notification system for our Vacation Request app





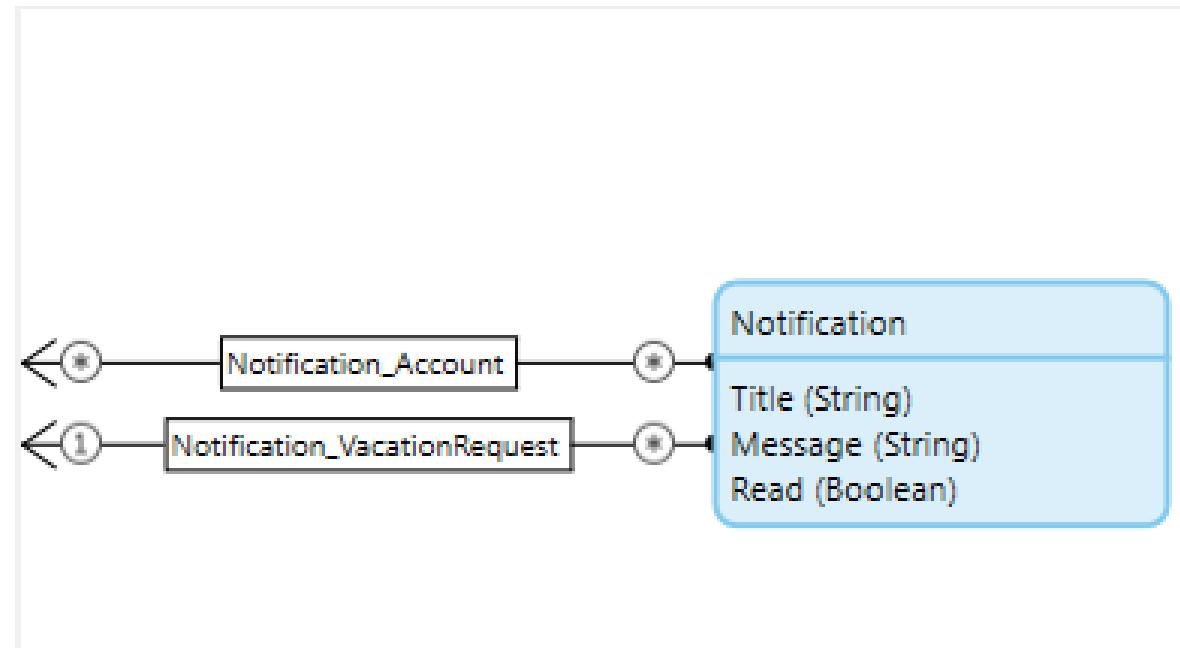
User story

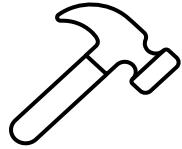
As a user, I want to be notified when the status of my request changes, so I can take the appropriate action.



Building the basics

- Add a new module, called **Notifications**
- Add the **Notification** entity to the Domain Model
- Add a *-* association to the **Account** entity
- Add a 1-* association from **Notification** to **VacationRequest**





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 Project security

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

New Access Rule of Entity 'Notifications.Notification'

Documentation

Rule applies to the following module roles

User

Select / deselect all

Access rights XPath constraint

Create and delete rights

Allow creating new objects Allow deleting existing objects

Member read and write rights

Default rights for new members None Read Read, Write

Member	Access rights
Title (String (200))	Read
Message (String (200))	Read
Read (Boolean)	Read
Notifications.Notification_Account (List of Administration.Account)	Read
Notifications.Notification_VacationRequest (VacationManagement.VacationRequest)	Read

Set all to

OK Cancel

Create notifications

- When a request is created by an **Employee**, 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%']
 - Use the image to configure the **Create object** activity
 - Commit the **Notification** and Refresh the client

Create Object

Action

Entity: Notifications.Notification

Commit: Yes

Refresh in client: Yes

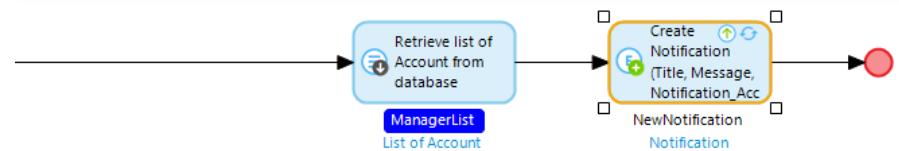
New Edit Delete | ▲ Move up ▼ Move down

Member	Member type	Type	Value
Title	String (200)	Set	'Request created on ' + formatDateTime(\$VacationRequest/createdDate, MM/dd/yyyy)
Message	String (200)	Set	\$VacationRequest/VacationManagement.VacationRequest_ + ' created a new ' + toString(\$VacationRequest/RequestType) + ' request.'
Notifications.Notification_...	Administration.Account	Set	\$ManagerList
Notifications.Notification_...	VacationManagement.Vac...	Set	\$VacationRequest

Output

Object name: NewNotification

OK Cancel



Value

```
$VacationRequest/VacationManagement.VacationRequest_Submitter  
/Administration.Account/FullName + ' created a new ' +  
toString($VacationRequest/RequestType) +  
' request. Please check this request and approve or reject it'
```

Generate



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

Create Object

Action

Entity Notifications.Notification

Commit Yes Yes without events No

Refresh in client Yes No

New Edit Delete ▲ Move up ▼ Move down

Member	Member type	Type	Value
Title	String (200)	Set	'Your request has been approved.'
Message	String (200)	Set	'Dear ' + \$VacationRequest/VacationManagement.VacationRequest_Submitter /Administration.Account/Name + ', Your request for ' + toString(\$VacationRequest/RequestType) + ' from ' + formatDate(\$VacationRequest/StartDate, 'MM/dd/yyyy') + ' to ' + formatDate(\$VacationRequest/EndDate, 'MM/dd/yyyy')
Notifications.Notification_...	Administration.Account	Set	\$VacationRequest/VacationManagement.VacationRequest_Submitter
Notifications.Notification_...	VacationManagement.Vac...	Set	\$VacationRequest

Output

Object name NewNotification

OK Cancel

```
'Dear ' + $VacationRequest/VacationManagement.VacationRequest_Submitter  
/Administration.Account/Name + ',  
  
Your request for ' +  
toString($VacationRequest/RequestType) + ' from ' +  
formatDate($VacationRequest/StartDate, 'MM/dd/yyyy') + ' to ' +  
formatDate($VacationRequest/EndDate, 'MM/dd/yyyy')  
+ ' has been approved by ' +  
$Manager/FullName + '.'
```



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:

Create Object

Action

Entity Notifications.Notification

Commit Yes Yes without events No

Refresh in client Yes No

New Edit Delete | ▲ Move up ▼ Move down

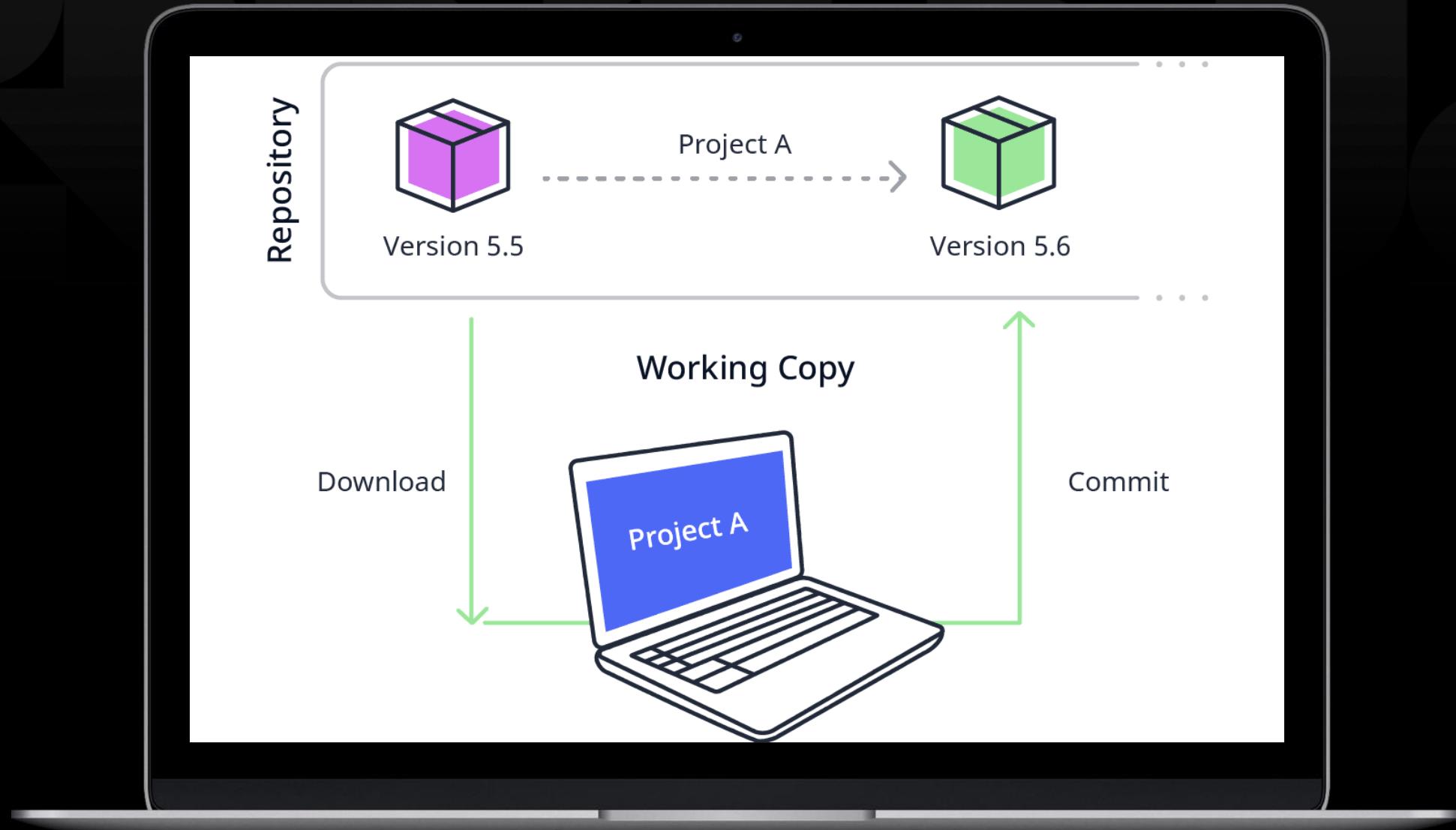
Member	Member type	Type	Value
Title	String (200)	Set	'Your request has been rejected.'
Message	String (200)	Set	Your request for ' + toString(\$VacationRequest/RequestType) + ' from ' + formatDateTime(\$VacationRequest/StartDate, 'MM/dd/yyyy') + ' to ' + formatDateTime(\$VacationRequest/EndDate, 'MM/dd/yyyy')
Notifications.Notification_...	Administration.Account	Set	\$VacationRequest/VacationManagement.VacationRequest_Submitter
Notifications.Notification_...	VacationManagement.Vac...	Set	\$VacationRequest

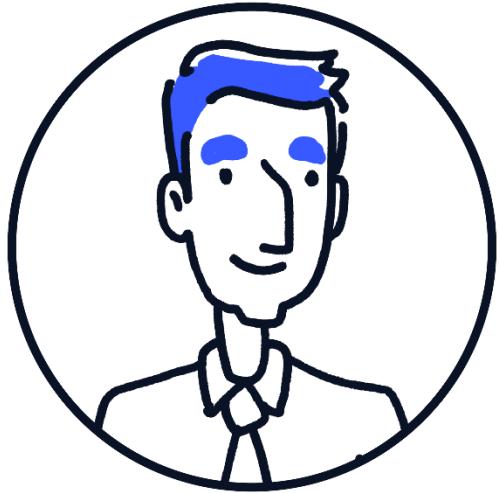
Output

Object name NewNotification

```
'Dear ' + $VacationRequest/VacationManagement.VacationRequest_Submitter  
/Administration.Account/Name + ',  
  
Your request for ' +  
toString($VacationRequest/RequestType) + ' from ' +  
formatDateTime($VacationRequest/StartDate, 'MM/dd/yyyy') + ' to ' +  
formatDateTime($VacationRequest/EndDate, 'MM/dd/yyyy')  
+ ' has been rejected by ' +  
$Manager/FullName + '.'
```

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 than 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

Caption			Action	User Roles
	Unread		Open page 'Notifications.Notification_Overv...'	Employee, Manager
	Read		Open page 'Notifications.Notification_Overv...'	Employee, Manager

Profiles

Navigation profiles can be used to create a different navigation for different devices and screens.

Add navigation profile

Responsive Hybrid phone app online

General

Application title	Mendix
-------------------	--------

Home pages

Default home page	Administration.Login
-------------------	----------------------

Role-based home pages

Administrator, Employee, Manager

Authentication

Sign-in page	Administration.Login
--------------	----------------------

Page title

<input type="checkbox"/> Override page title	<input checked="" type="checkbox"/> Page Title
--	--

Menu

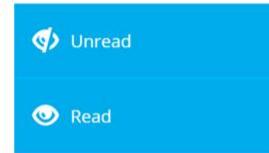
Caption			Action	User Roles
	Home		Open page 'VacationManagement.Home'	Employee, Manager
	My Account		Call microflow 'Administration.ManageMyAc...'	Employee, Manager
	Requests		Open page 'VacationManagement.Vacation...'	Manager
	Notifications		Open page 'Notifications.Notification_Overv...'	Employee, Manager



Build the notification pages

- Using the **layout grid**, build out the two pages
 - Use Navigation tree widget to show Notifications menu
 - 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

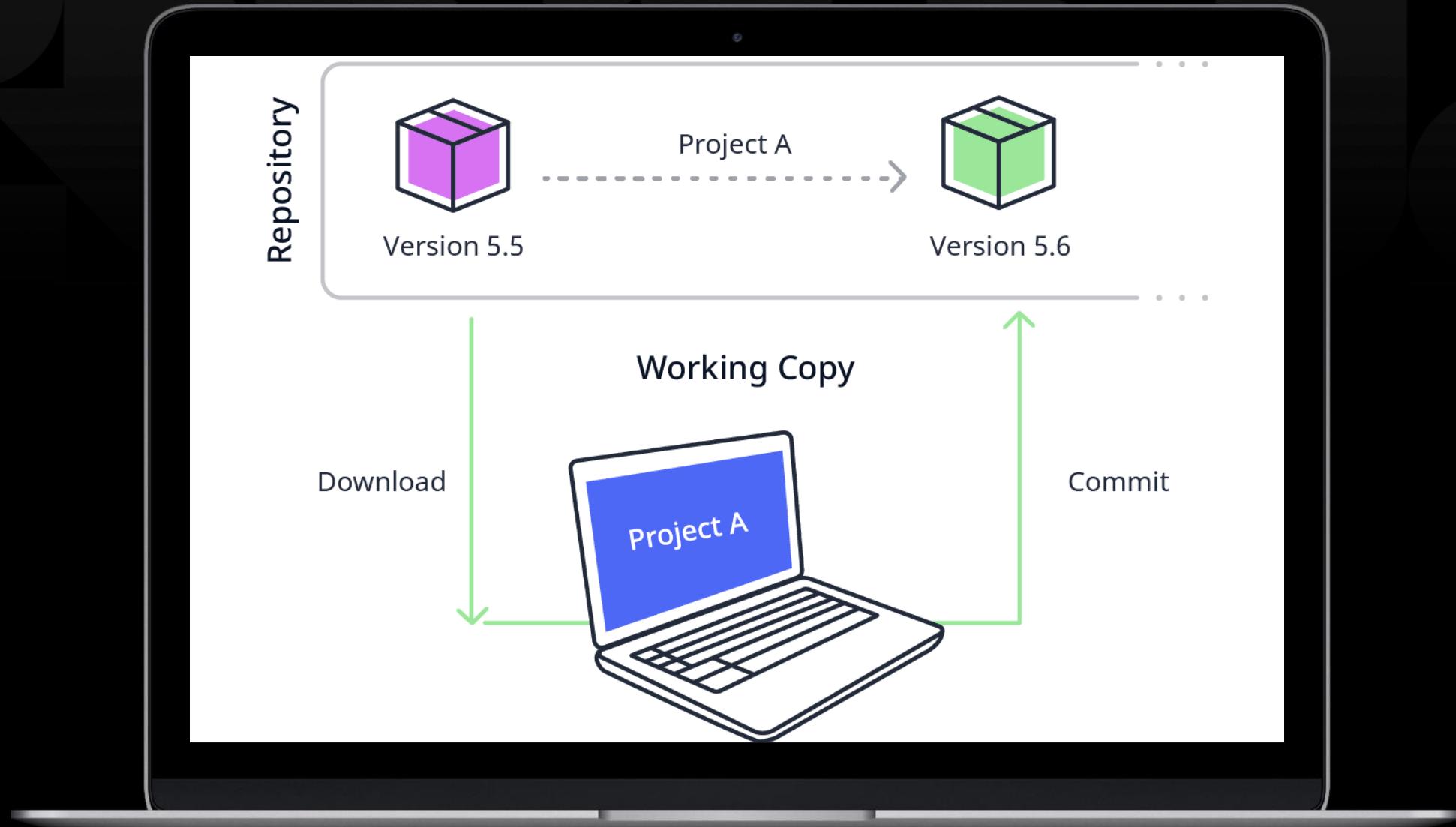
Notifications

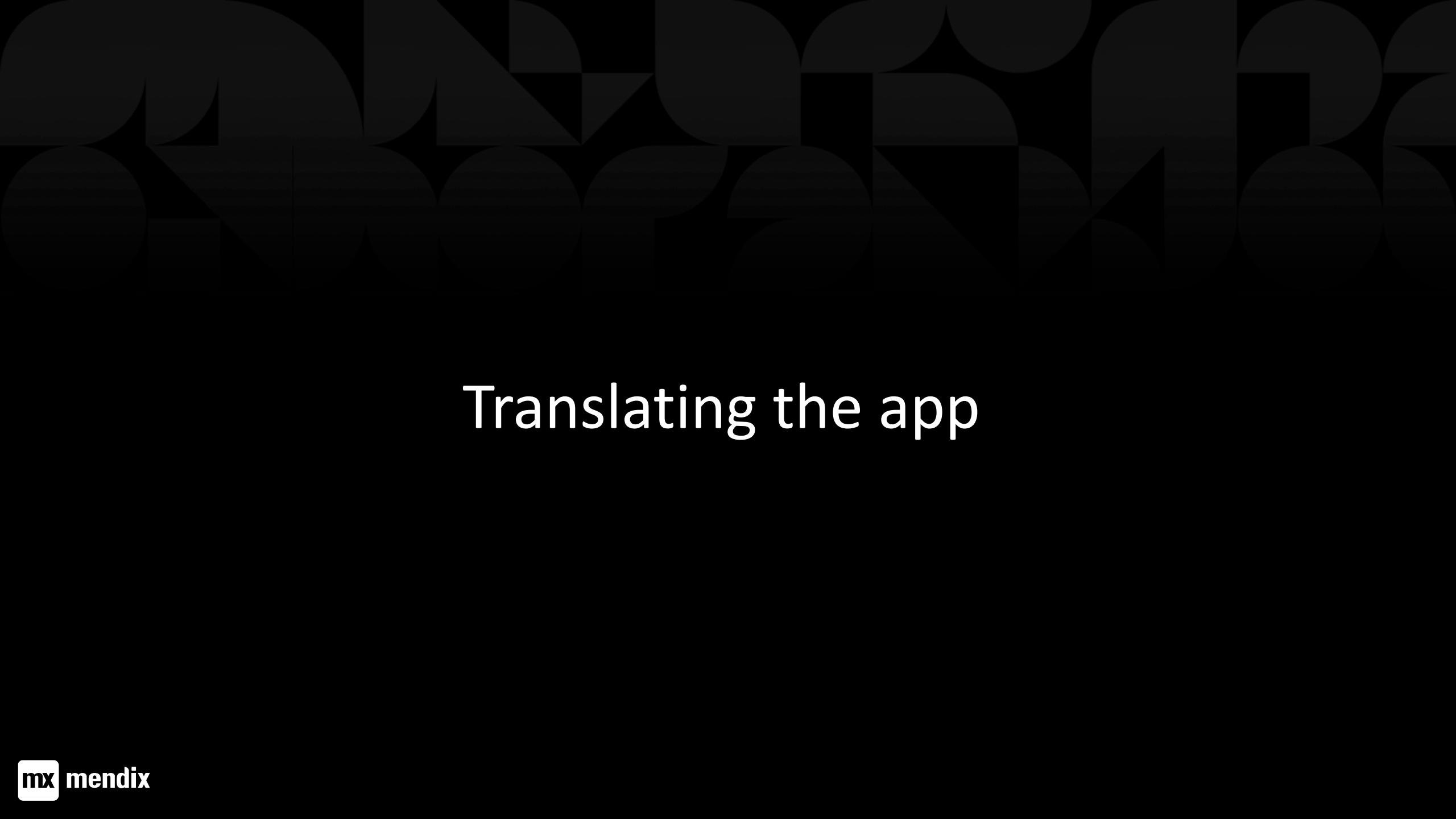


The screenshot shows a list of three notifications. Each notification has a title and a message. Below each message are three buttons: 'Mark as read' (orange), 'Show request' (light blue), and 'Delete' (red).

Notification Index	Title	Message	Action Buttons
1	{Title}	{Message}	Mark as read Show request Delete
2	{Title}	{Message}	Mark as read Show request Delete
3	{Title}	{Message}	Mark as read Show request Delete

Commit your work

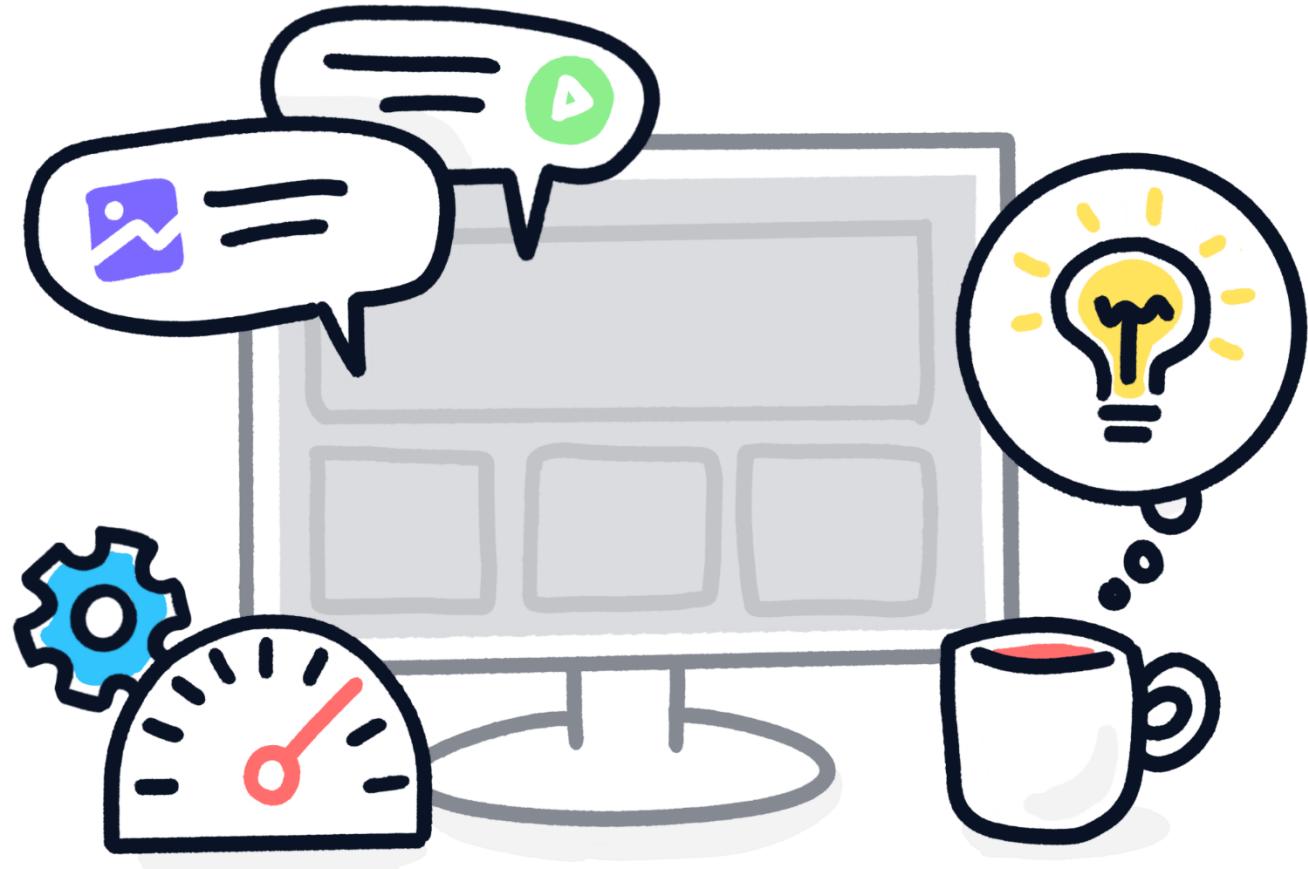


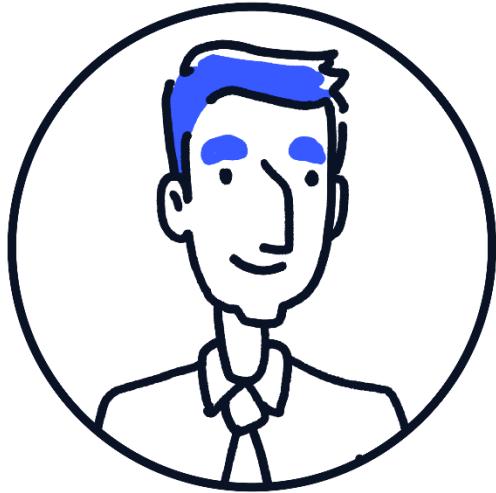


Translating the app

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 Occurrence
- Export/Import Translations



Add a language

- From the **Project Explorer** open the **Settings** menu
- Go to the **Languages** tab
- Add **Dutch, Netherlands** as a new language

The screenshot shows the Mendix Project Settings interface. The 'Languages' tab is selected. A table lists languages, with 'English, United States (default)' as the current default. An 'Add' button is visible. A modal dialog titled 'Add Language' is open, showing 'Dutch, Netherlands' selected in the 'Language' dropdown and 'nl_NL' in the 'ISO 639 code' input field. The 'OK' button is highlighted.

Description	Check completeness	Date formatting
English, United States (default)	Yes	Default



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	Voltoooi 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

Batch translate from 'English, United States' to 'Dutch, Netherlands'

Filter for translating and exporting to Excel

Documents/modules (all)

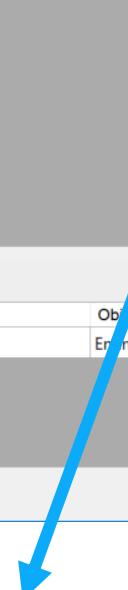
Source text contains time off

Source (English, United States)	#	Translation (Dutch, Netherlands)
8 hours of Annual Holiday/Paid Time Off	1	Tekst
Annual Holiday/Paid Time Off (PTO)	1	Betaald verlof
Complete both steps of the wizard to request time off.	1	Voltoooi 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

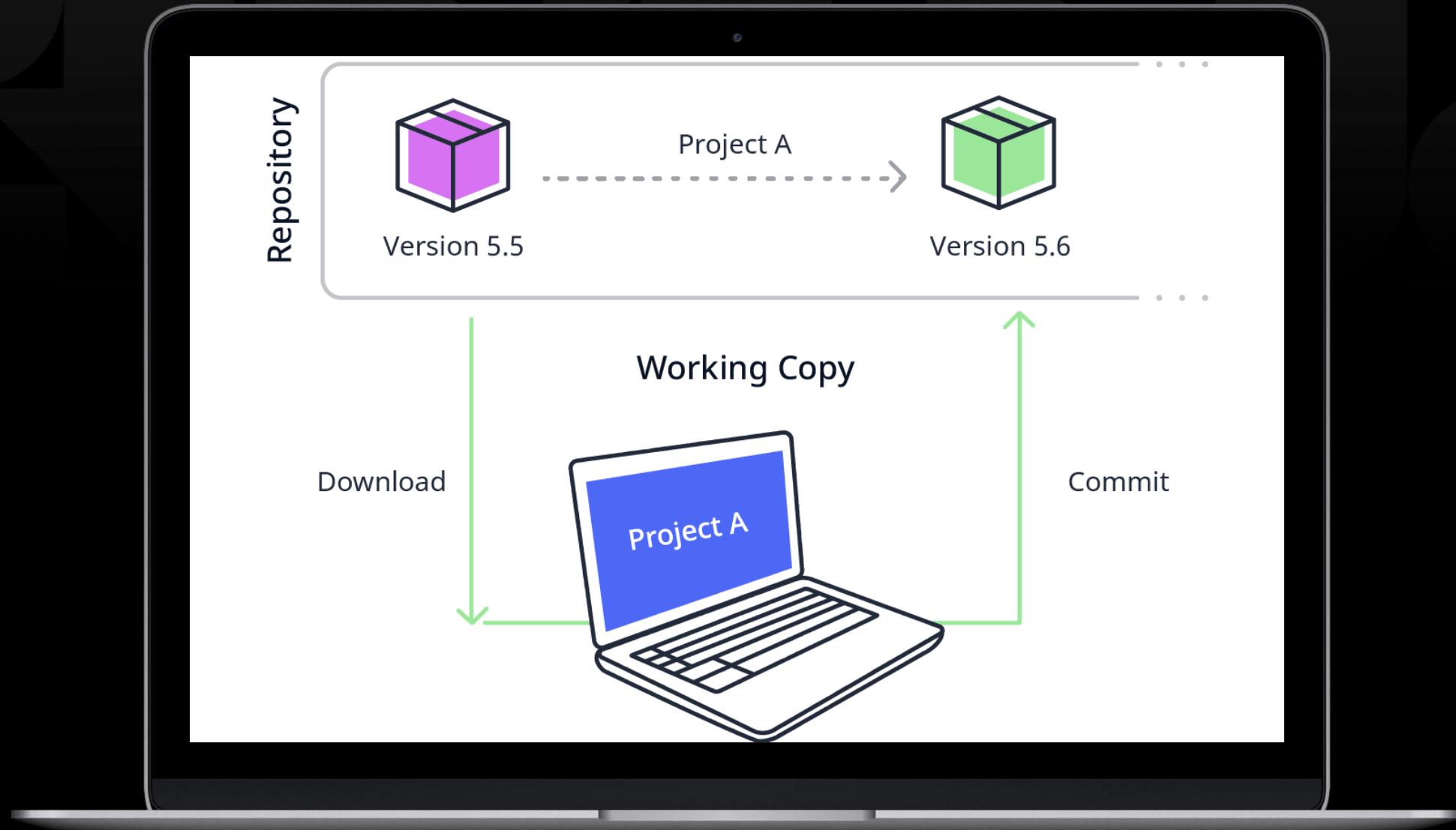
Show occurrence

Document	Object
Enumeration 'VacationManagement.VacationRequestType'	Enumeration value 'Annual_Holiday_Paid_Time_Off_PTO'

Number of lines: 6



Commit your work



Conclusion

Feedback

