BUILDING AN APP WITH OUTSYSTEMS

Learning materials

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus, Hämeenlinnan korkeakoulukeskus
syksy, 2022

Elina-Maria Zamora Lopez

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittely		Tiivistelmä
Tekijä	Elina-Maria Zamora Lopez	Vuosi 2022
Työn nimi	Building an app with OutSystems – Learning Materials	
Ohjaaja	Esa Huiskonen	

Opinnäytetyön tavoitteena on kehittää yksinkertainen tehtävälista-sovellus OutSystemsin low-code-alustalla ja luoda tästä oppimateriaalia HAMK:n englanninkieliselle OutSystems-aiheiselle kurssille, joka pohjautuu OutSystemsin Reactive Web Developer -sertifiointipolkuun.

Teoriaosuudessa käsitellään low-coden lisäksi no-code ja high-code, sivutaan hieman low-coden historiaa, ja käsitellään citizen developmentin käsitettä ja ilmiötä. Näiden lisäksi käsitellään OutSystems, OutSystems Service Studio 11 ominaisuuksineen, ja Service Studiolla kehittämisen eri vaiheet.

Käytännön osassa kuvataan sovelluksen suunnitteluprosessia, kehitystyön osalta käsitellään pääkohdat yksityiskohtaisesti myös kuvankaappauksin. Sovellus on pidetty tarkoituksellisesti yksinkertaisena, jotta opiskelija saa käsityksen siitä miten helppoa low-code voi olla. Kehitystyö käsitellään päävaiheittain, edeten työn aloittamisesta luomalla entiteetit/tietokannat, jatkaen ensimmäisten ruutujen luomiseen sovellukseen aluksi tuodun tiedon avulla, ja päättyen viimeistelyyn ja julkaisuun.

Työn tuloksena on kuvaus yksinkertaisen sovelluksen luomisesta OutSystemsin Service Studio 11-sovelluksella, sekä erilaista oppimateriaalia kurssia varten. Lukijan tulisi saada peruskäsitys OutSystemsin toiminnasta, ja ymmärrys yksinkertaisen sovelluksen kehittämisestä. Opettajat saavat materiaalia oppitunnin pitämiseen ja oppilaille jaettavaksi.

Sovelluksen kehittäminen oli yksinkertaista ja suoraviivaista, isoin haaste oli pitää valmis sovellus järkevän laajuisena. Tarkoitus on kuitenkin myös osoittaa, että low-code-kehittäminen on yksinkertaista.

Valmis sovellus toimii, kuten on tarkoitus. Sovelluksessa ei ole tällä hetkellä mahdollista poistaa jo tehdyiksi merkittyjä tehtäviä, mutta tämä voisi olla haaste opiskelijoille kehittää itse sovellusta edelleen, ja halutessaan lisätä sovellukseen poistamisen lisäksi muitakin ominaisuuksia.

Avainsanat	Low-code, No-code, OutSystems
Sivut	34 sivua ja liitteitä 10 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Name of Degree Programme		Abstract
Author	Elina-Maria Zamora Lopez	Year 2022
Subject	Building an app with OutSystems – Learning Materials	
Supervisors	Esa Huiskonen	

The thesis aims to develop a simple to-do list application on OutSystems' low-code platform and create learning material from this for HAMK's English-language OutSystems-related course, which is based on OutSystems' Reactive Web Developer certification path.

In addition to low-code, no-code and high-code are discussed in the theory part, the history of low-code is looked at a little, and the concept and phenomenon of citizen development are discussed. In addition to these, OutSystems, OutSystems Service Studio 11 with its features, and the different stages of development with Service Studio are discussed.

In the practical part, the design process of the application is described in terms of the development work, and the main points are discussed in detail, with screenshots. The application has been deliberately kept simple so that the student gets an idea of how easy low-code can be. The development work is handled by main stages: creating the entities/databases, making the first screens using the information initially brought into the application, and finishing and publishing.

The result of the thesis is a description of creating a simple application using the OutSystems Service Studio 11 application and different learning materials for the course. The reader should get a basic understanding of how OutSystems works and an understanding of developing a simple application. Teachers receive material to teach the lessons and distribute it to the students.

The application development was simple and straightforward, and the biggest challenge was keeping the finished application reasonable in scope. However, the purpose is also to show that low-code development is simple.

The finished application works as intended. In the application, it is not currently possible to delete already marked tasks. Still, it could be considered a challenge for students to develop the application itself further and if they want to add other features besides deletion.

Keywords Low-code, No-code, OutSystems, development
Pages 34 pages and appendices 10 pages

Contents

Figures

Annexes

Annex 1	Material management plan

Attachments

Attachment 1	MyTasks.xlsx
Attachment 2	PowerPoint presentations

1 Introduction

Market research company Forrester coined the term "Low-code development" in 2016. Forrester defined low-code as follows “Low-code platforms enable rapid delivery of business applications with a minimum of hand-coding and minimal upfront investment in setup, training, and deployment” (obviously.ai, 2022). It is estimated that 70% of all development will be made with low-code platforms and technologies by 2025.(Wong & Iijima, 2021)

This thesis aims to make study materials for an upcoming course on low-code development, focusing on developing using OutSystems. In this thesis, I will open the concept of low-code by comparing it to high-code and no-code and tracing some of its histories. OutSystems will also get a deeper view of the platform and development using it.

I will also go through the development process in detail, which will be the basis for the PowerPoint -presentations. The presentations and other possible additional materials will result from this thesis.

The questions this thesis aims to answer are: How to develop a simple application in OutSystems Service Studio, and what are the main phases of development?

2 Low-Code, No-Code, and High-Code

Low-Code and No-Code development's roots are deeper than one would think. The first programming language for non-programmers was developed in 1955. FLOW-MATIC was devised by Grace Hopper so that businessmen could use computers. It used English-like statements instead of mathematical symbols. (nocode.tech, 2022)

FLOW-MATIC was a programming language. But what about no-code tools? In 1983 VisiCalc was launched. One could use software to run calculations and store, analyze, and organize data in a spreadsheet. Before that, one needed a programmer to run the calculations. VisiCalc was followed by Microsoft Excel, probably one of the more familiar no-code/low-code tools. (nocode.tech, 2022)

Low-code/no-code platforms enable users to create applications and automation workflows using graphical tools with drag-and-drop functions. The platforms are easily integrated into existing products, such as cloud services. (Hay, 2021)

2.1 Citizen development and Citizen developer

Citizen development is a business process that enables employees to develop the business applications they need using low-code or no-code platforms, thus supporting them to become software developers even without experience or training in coding. This also includes customizing existing business applications to fit their needs better. These employees are then citizen developers. (Liptak & Horwitz, 2021)

Citizen developers can help relieve the IT department by developing applications for smaller user groups or niche needs. However, the platforms used for development should be approved by the IT department, or an in-company program for citizen development should be created. This is a way to reduce and avoid breaching regulations or safety issues. Setting up the program could be time-consuming, but it will release resources for the IT department in the future. (Rubens, 2014)

2.2 Low-Code and No-Code

Low-code development platforms are made for users with a limited understanding of coding. A good example would be someone who knows and understands processes and workflows but does not know how to code. They are the users to whom these platforms are targeted. (Hay, 2021)

There are several providers of low-code/no-code platforms; for example, Forrester included these 14 platforms in their Q2/2021 assessment: AgilePoint, Appian, GeneXus, HCL Software, Mendix, Microsoft, Oracle, OutSystems, Pegasystems, Salesforce, ServiceNow, Thinkwise, Unqork, and WaveMaker. (Bratincevic & Koplowitz, 2021)

2.3 High-Code

Whereas citizen developers use no-code and low-code, high-code use needs experience in developing and understanding programming languages. High-code is perceived as "traditional programming," where developers are more experienced and write the code from the beginning. (Kissflow, 2022)

High-code is best to be used when a unique application is needed, one that is customized to your business needs. Developers control the experience and interface, while data can be used and controlled by remote systems. It is important to note that high-code is very reliant on experienced developer resources. (Northcutt, 2021)

3 OutSystems

OutSystems was founded in 2001 by Paulo Rosado in Lisbon, Portugal. OutSystems is a low-code platform that one can use to develop an application, either starting from scratch or using an earlier application as a base to build on. The available options for developing are reactive web apps, tablet apps, phone apps, external web portals, traditional web development, and service development. (OutSystems, 2022g)

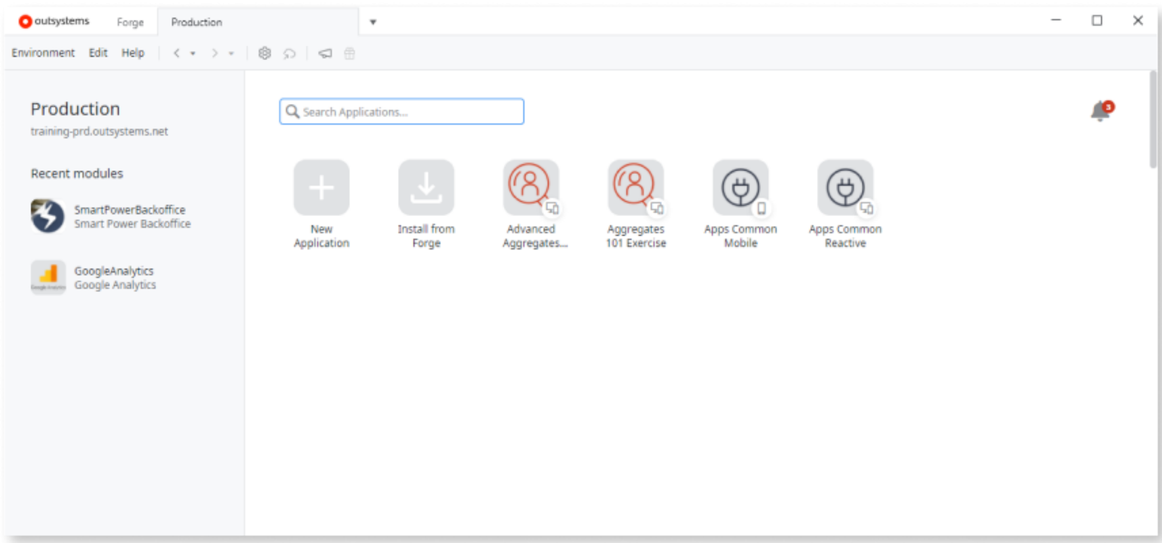
3.1 OutSystems Service Studio 11

Applications are developed in Service Studio, the main IDE (Integrated development environment) in OutSystems. Service Studio is used to build, edit, and debug applications. (OutSystems, 2022k)

To use Service Studio, the user needs access to an OutSystems environment. The free option of OutSystems provides one runtime environment that can be used for training and development. Two subscription options provide more runtime environments that are fully isolated. (OutSystems, 2022b)

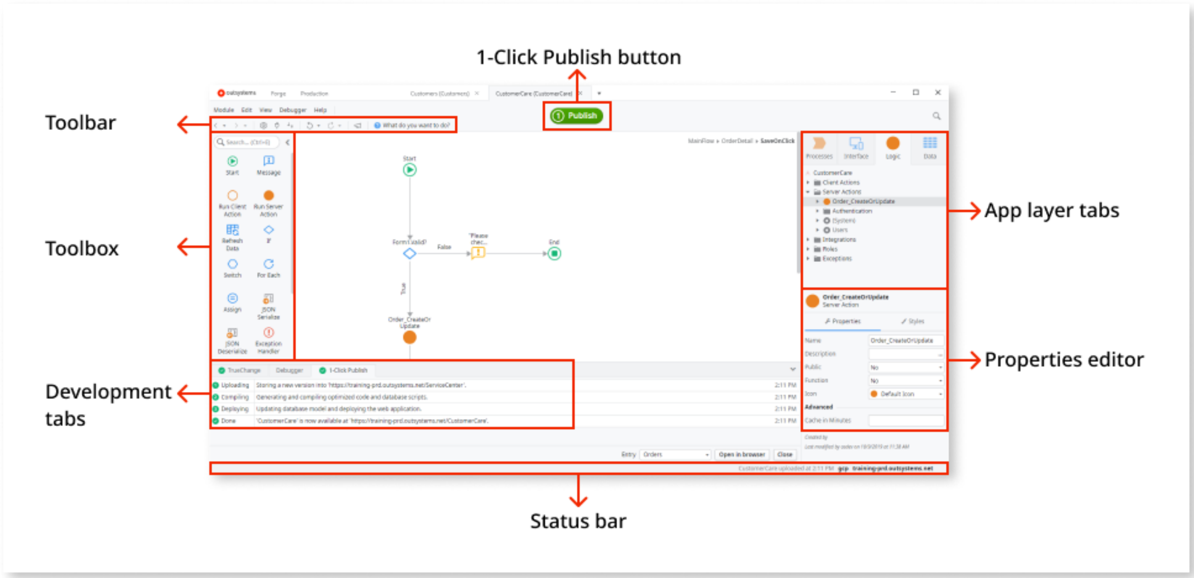
Service Studio opens to the Environment tab (see picture 1), showing users the applications currently on the server. It allows the user to view and access them, create new applications, and install application templates made by the community from the Forge. (OutSystems, 2022k)

Figure 1 Overview of the Environment tab (OutSystems, 2022k)



The other main feature of Service Studio is the workspace (see Figure 2).

Figure 2 Overview of the workspace (OutSystems, 2022k)



The Main editor is the area in the middle of the screen where the interface and logic are designed. Shortcuts to the most common operations can be found in the *Toolbar,* and the *Toolbox* contains the tools and widgets. (OutSystems, 2022k)

The four *development tabs* consist of the *TrueChange tab* that displays errors and warnings, the *debugger tab* that is used for debugging as the name implies, the *1-Click Publish tab* that shows the progress and result of the deployment, and the *Search Results tab* that shows search results for searches performed in the module. (OutSystems, 2022k)

The other elements of the workspace are the *Application layer tabs* that contain the elements for the different layers (processes, user interface, logic, and data model), the *1-click publish button* that starts the deployment process if the application has no errors, the *Status Bar,* and the *Properties editor*. (OutSystems, 2022k)

3.2 Developing in OutSystems

Developing or building can be started from scratch or using application templates. Applications are constructed with modules that Service Studio bootstraps from the selected templates. (OutSystems, 2022f) No previous experience in OutSystems is needed. (OutSystems, 2022l)

There are three kinds of application templates, built-in, custom, and Forge application templates. Built-in templates are maintained by OutSystems and include Reactive Web App, Phone App, and Tablet app. Custom templates are created by the user, and Forge templates are created and shared by the OutSystems community. (OutSystems, 2022f)

3.2.1 Use data

Service Studio lets the user specify and use complicated data structures. Entities are used to keep information persistent, and structures are used for data management. Entities are also used in data modeling and can be used like database tables or views. Entities are defined using Entity Attributes. The data model is made to define the relationships between entities. (OutSystems, 2022a)

The data in the application can be viewed and displayed using database queries or customized SQL queries. The data in the application has been fetched from the database as an aggregate or an SQL element. (OutSystems, 2022a)

3.2.2 Design UI

UI Design in OutSystems is based on The OutSystems UI Framework. On top of the Framework are the Built-in Screen Templates, The Custom Application Templates, and a Style Guide that can be created to define the style of your application. The Built-in Screen Templates are made and supported by OutSystems, and the users make Custom templates. There are also possibilities for making your application multilingual and accessible. (OutSystems, 2022h)

3.2.3 Implement application logic

In OutSystems, the user uses Actions to implement logic in the applications. Like in earlier features, there are Built-in actions and Custom actions. Built-in actions cannot be modified and are defined by the platform, and the user creates Custom actions. With Custom actions, the user can, for example, run integrations and create business rules. There are also actions for handling system events that run at a particular moment of the application duration. The user can implement their own business rules in the design of these System Event handling actions. OutSystems also provides AI-supported development that suggests developing your logic and automatically adding logic nodes to the workflow. Exception and error handling are also available for the actions. (OutSystems, 2022i)

There are differences in where the actions run depending on if the actions are used in Reactive Web Applications and Mobile applications or traditional web and service applications. The actions in Reactive Web applications can run either on the server or on the client, client in this case meaning the device used by the user of the application. In traditional web and service applications, all logic runs on the server. This means that the developer needs to consider that it is done via server requests when the logic on the client side needs to use server-side logic. The communication between the client logic and server logic needs an internet connection. (OutSystems, 2022i)

3.2.4 Use Processes

In OutSystems, a Business Process is called a Process. A process is seen as how a task is carried out in the user's company or organization. Examples include invoice handling and order processing. A process is also known as Business Process Technology or a BPT. (OutSystems, 2022i)

The developer uses the Process Flow Editor to create and edit the processes. The user can use the editor to create a process flow for the activities needed during runtime. The editor has tools to use in the design process. (OutSystems, 2022i)

3.3 AI-assisted development in OutSystems

OutSystems has developed an AI service called AIFusion™ and added AI-assisted development to Service Studio through its OutSystems.ai program. The OutSystems.ai program is intended to research methods to increase efficiency in developing and also ease the addition of AI to the developed applications. (OutSystems, 2022d)

AIFusion™ powers the AI-assisted development in Service Studio; AI-assisted development uses patterns learned from anonymized code to give the developer suggestions on the following steps to take or the tools to use. (OutSystems, 2022c)

There are two different ways for AI-assisted development to give suggestions for the developer, Next-step suggestions for logic flows and OutSystems Smart Guidance. Next-step suggestions for logic flows gives suggestions for all kinds of logic flows within the flow itself. It gives 1-6 suggestions depending on the confidence of the assistant. The more confident the assistant is, the fewer suggestions it gives. Next-step suggestions are often specific and provide the information needed for the parameters of the logic node so that the developer does not have to fill them in.

With OutSystems Smart Guidance, the developer can search for relevant information on the current issue they are facing. There are several triggers within the development environment, including a "What do you want to do?"-button. Clicking the trigger activates a search wizard that shows results relevant to the issue at hand. OutSystems uses AI to figure out the context of the issue from the type of the module or the error messages the developer is seeing. (OutSystems, 2022e)

3.4 Publishing, deploying, and scaling the application

In the free version of OutSystems Service Studio, the user gets one runtime environment. Once the app is published using 1-click Publish, it's also deployed. In OutSystems, you use the Lifetime- console to manage the applications lifecycle and the environments. (Note: Console is separate from Service Studio). More runtime environments are available in the paid subscriptions, and Lifetime is needed to manage them. The user can deploy the App from development to QA or Production, depending on the available runtimes. There is also some variation depending on if OutSystems is used in the cloud or installed on-premise. But all management can be handled through the Lifetime console. (OutSystems, 2022j)

The user size range for apps developed with OutSystems ranges from a few to millions of users. OutSystems provides several scalability options, including vertical and horizontal scalability, according to the user's needs. (OutSystems, 2022n)

4 Purpose of the work

The purpose of this thesis was to produce learning materials and a simple to-do-list application on OutSystems. HAMK needed study materials on the following subjects: custom validations and data modeling, user-interface, utilizing the built-in top menu, using existing templates in their application, and writing custom SQL queries.

The materials were made by collecting information from OutSystems documentation and documenting the development process in writing and screenshots. The text and screenshots were made into PowerPoint presentations, and an Excel -file was also saved for use in development.

The application was designed with a citizen developer in mind, so someone who does not know how to code but is interested in low-code development. The design was deliberately kept relatively simple to showcase the simpleness of low-code development.

The development process was completed several times. First, to achieve a working and valuable application, then a few times to document the phases of development. The process was repeated similarly each time, only differences being the names of the developed applications. Only OutSystems Service Studio and Excel were used to develop the application. Which is a nice coincidence, since Excel was mentioned as one of the early instances of low-code development, and one of the more well-known ones.

5 Development of the application and making of the study materials

Developing the application was simple and straightforward, as expected when using a low-code platform. It was agreed that a simple "to-do list" application would be suitable for the study materials. Since the course the study materials are for follows the OutSystems Reactive Developer certification path, the application will naturally be a reactive web app.

A "to-do-list" application is quite simple to make in OutSystems; one of the default examples of fast development is a Tasks-application, a simple to-do-list app.

I chose to make the materials as PowerPoint presentations, as they are easier to update and recycle than videos. For this, three presentations were made, each with its own subject title. One focuses on general development with OutSystems, one for making a screen from scratch, and one for using aggregates to visualize the data.

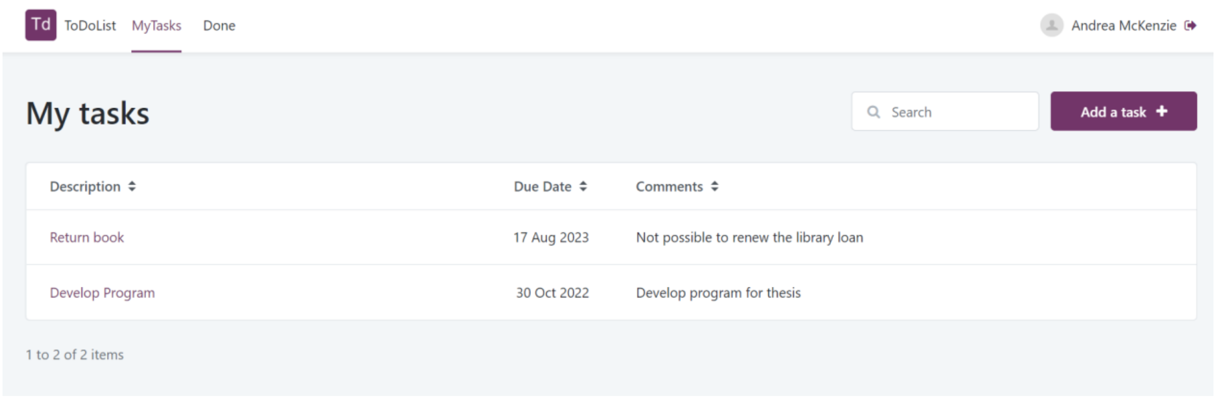
The most challenging part of developing was keeping the scope of the application sensible. It is relatively easy to get too excited and try to come up with new things to add to the application.

5.1 Designing the application

The designing and development of the application had to be done in a way that would be understandable for a beginner. The application’s design had to be simple but also include chances to learn further. At the simplest, the development is importing an excel file to make the database and entities and dragging and dropping the said entities to create two screens.

I designed the application to have three screens. OutSystems creates the two first screens from the database/entities: the main screen (Figure 3), which includes an overall view of the tasks, and a detail screen where the user can edit the task (Figure 5). The detail screen is also used when adding a new task (Figure 4). I wanted to add a third screen to the application so that the study material would include a screen made from scratch. The third screen shows the user all the tasks that have been marked as done (Figure 6).

Figure 3 Main screen

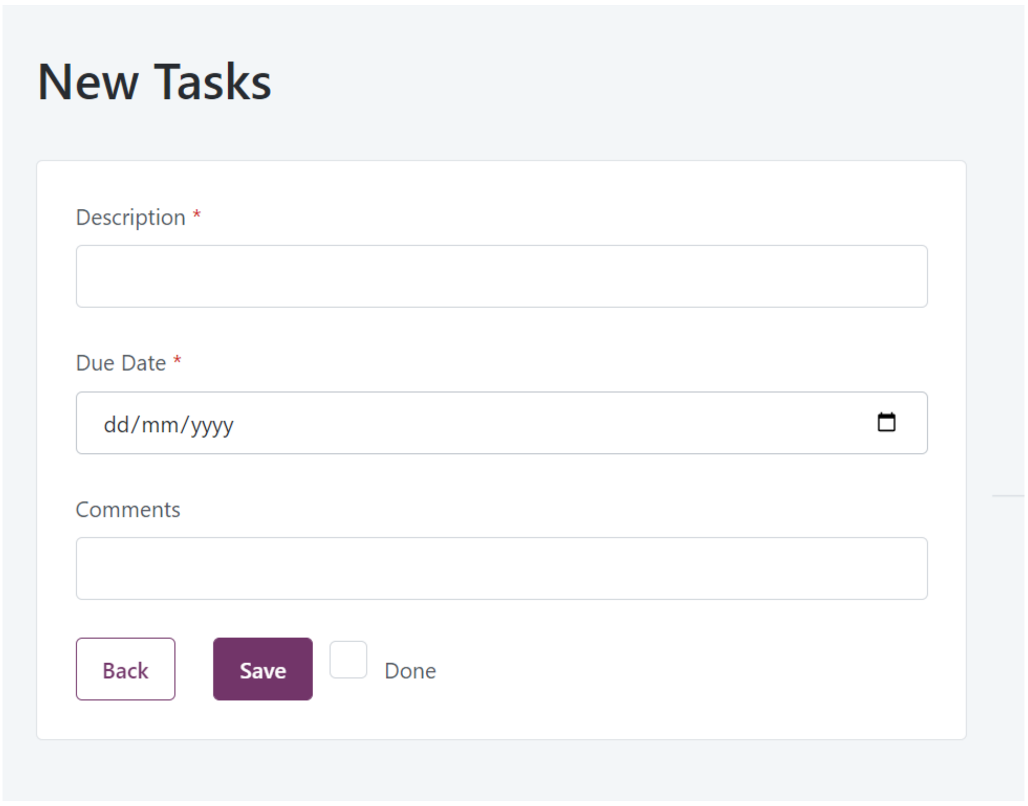


My tasks

Description	Due Date	Comments
Return book	17 Aug 2023	Not possible to renew the library loan
Develop Program	30 Oct 2022	Develop program for thesis

1 to 2 of 2 items

Figure 4 Detail Screen for adding tasks



New Tasks

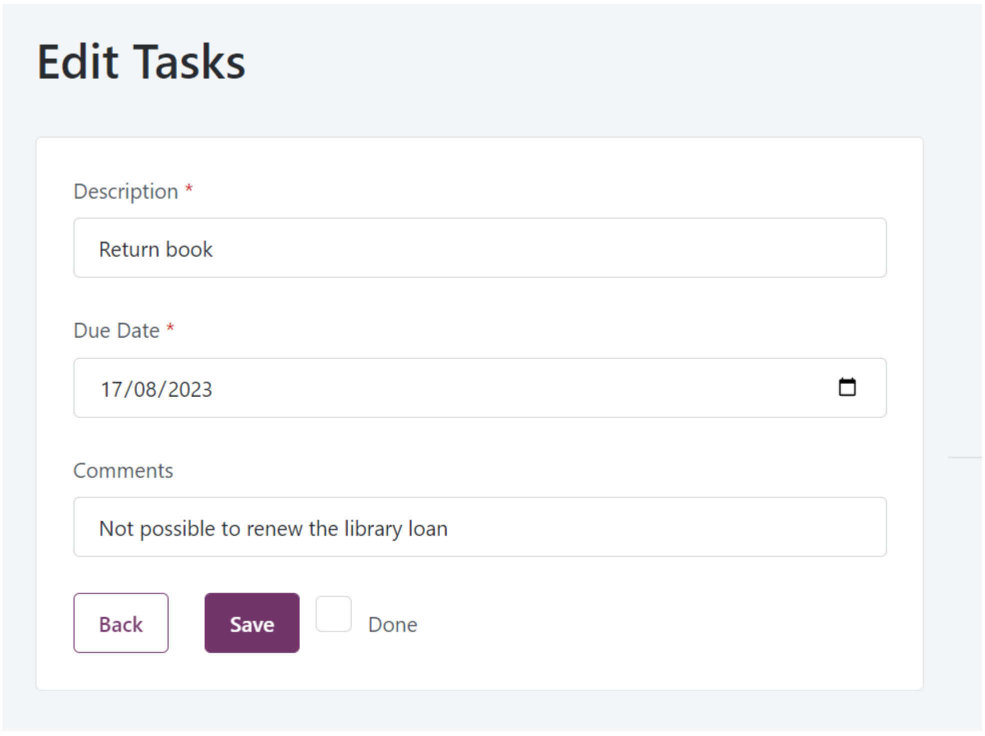
Description *

Due Date *

dd/mm/yyyy

Comments

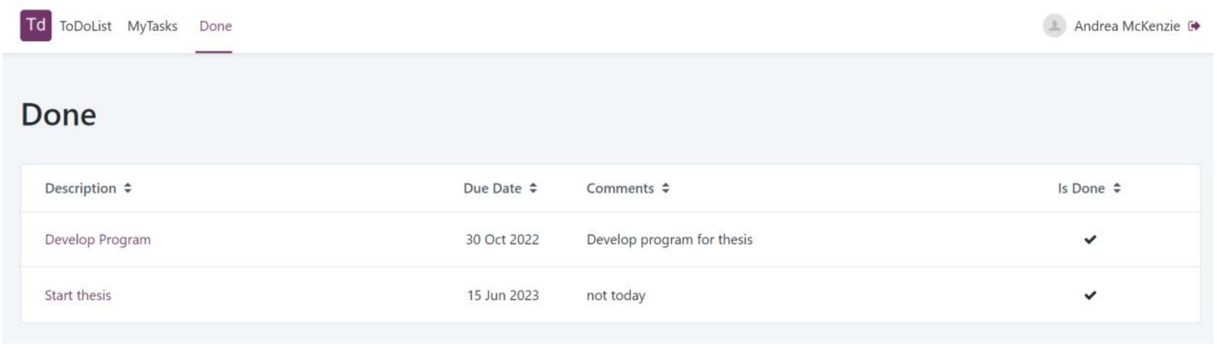
Figure 5 Detail Screen for editing tasks



Edit Tasks

Description *	Return book
Due Date *	17/08/2023
Comments	Not possible to renew the library loan

Figure 6 "Done" screen



Done

Description	Due Date	Comments	Is Done
Develop Program	30 Oct 2022	Develop program for thesis	✓
Start thesis	15 Jun 2023	not today	✓

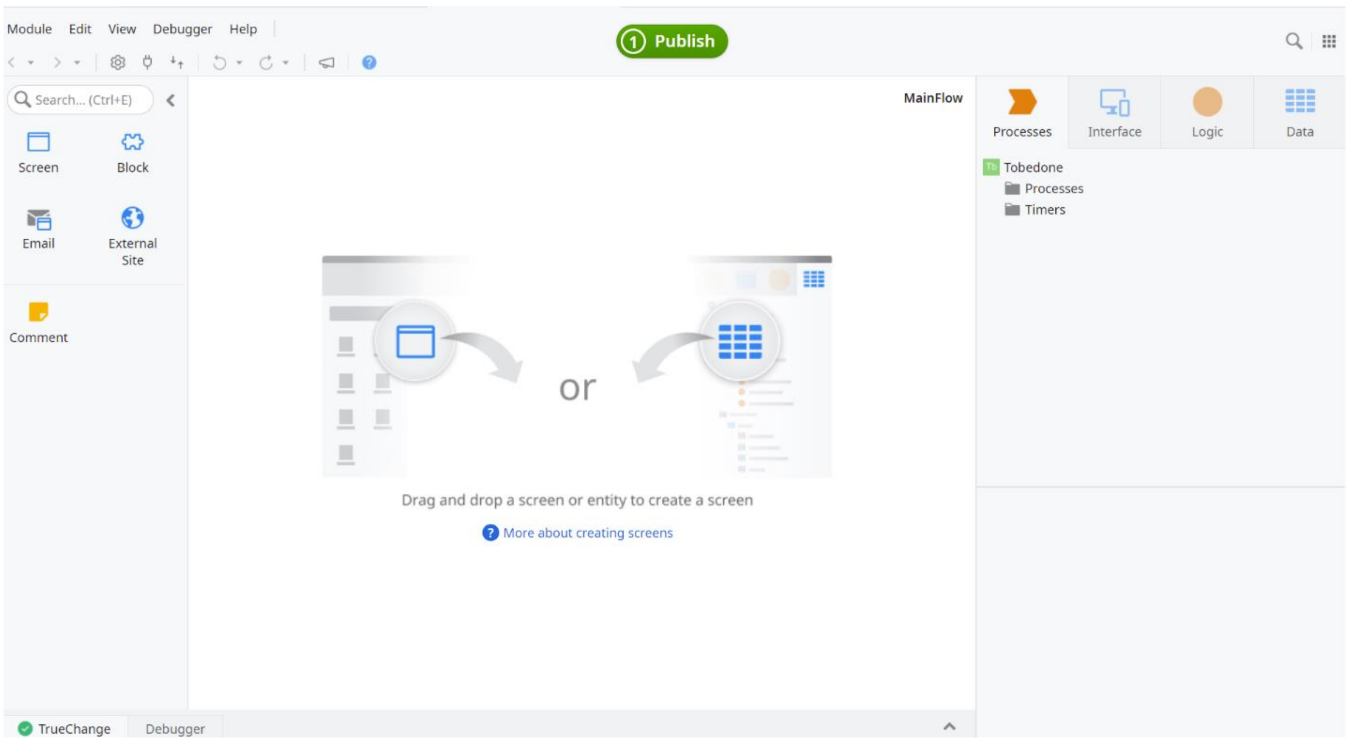
I also wanted to add a checkbox for the tasks. Since it is a "To do" -list, the user would like to be able to check their tasks as done. I added a feature that once the tasks were marked as done, the tasks would be moved to the third screen called “Done” (see fig. 6), where the user will be able to view the tasks marked as done. The app filters the tasks done from the overall view; so that the user only sees tasks still to be done. In case the user marks the task as done by accident, I added a possibility to edit the tasks after they are moved to the “Done” -screen, and the task can be returned to the main screen by removing the check from the checkbox.

5.2 Development

The development could be divided into three stages: setting up the database and entities, setting up the screens, and then using aggregates to control the data seen on the screens.

I chose to create the database and entity attributes by importing them from an Excel file I had previously prepared for this use. This simple Excel file will also be included in the study materials. One entity attribute will be added manually to the database, and it will be used as the filtering value in the aggregates as well.

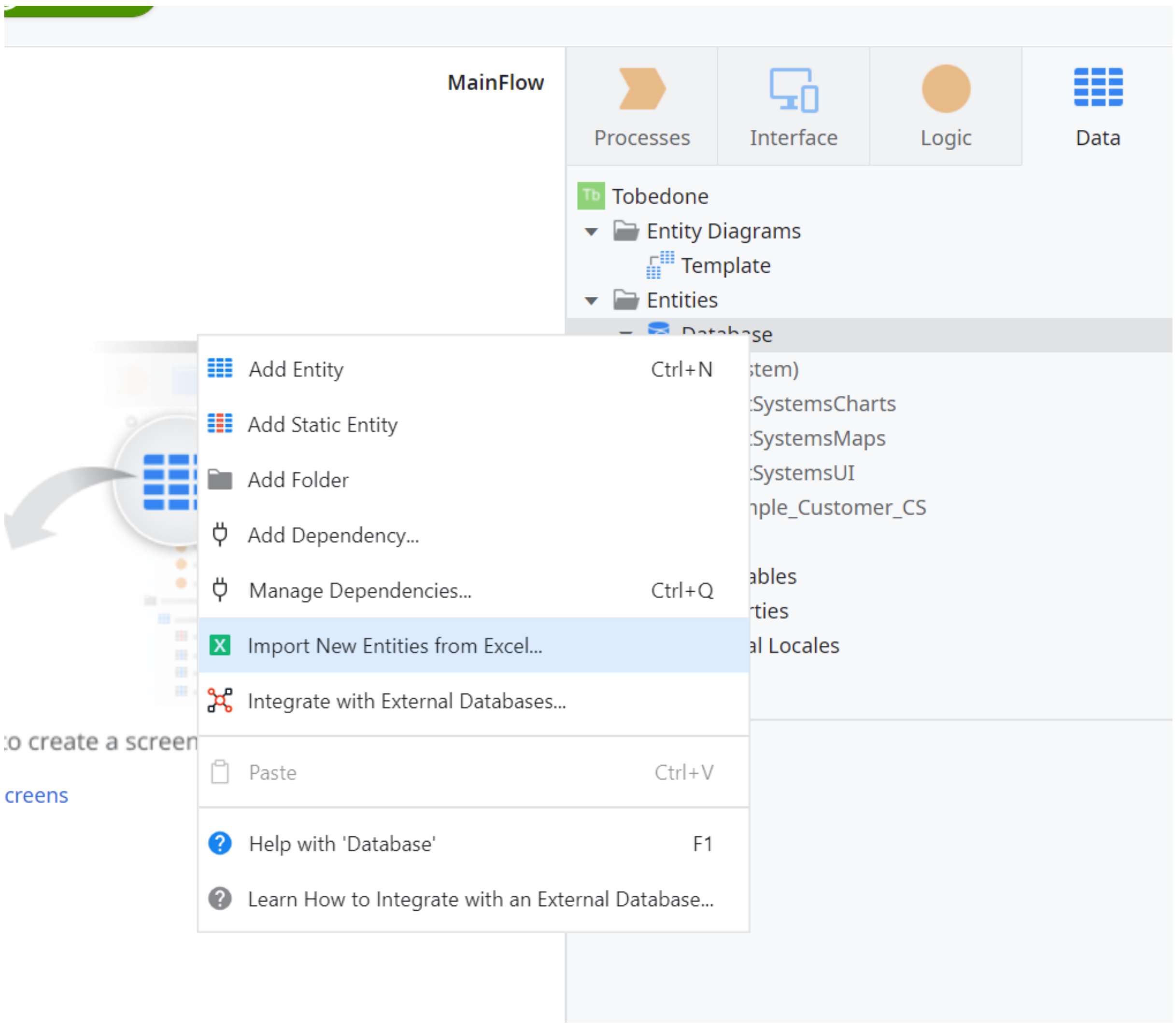
Figure 7 OutSystems Service Studio at the beginning of development



5.2.1 Entities

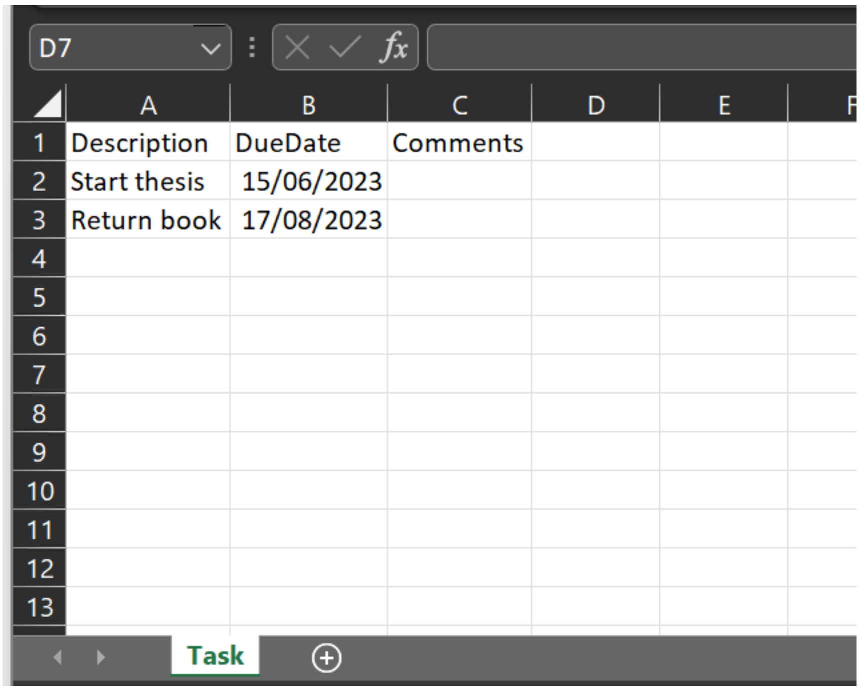
I started the development by importing the Entity and its attributes from Excel. (Figure 8).

Figure 8 Importing entities from Excel.



The first phase of the import process is to choose the file from which the Entity and the entity attributes are from. One thing that should be noted here is that Service Studio will name the Entity according to the name of the sheet, and the entity attributes will be named after the headings. For example, an entity made with the Excel sheet in Figure 9 would be called task, and the entity items are Description, DueDate, and Comments.

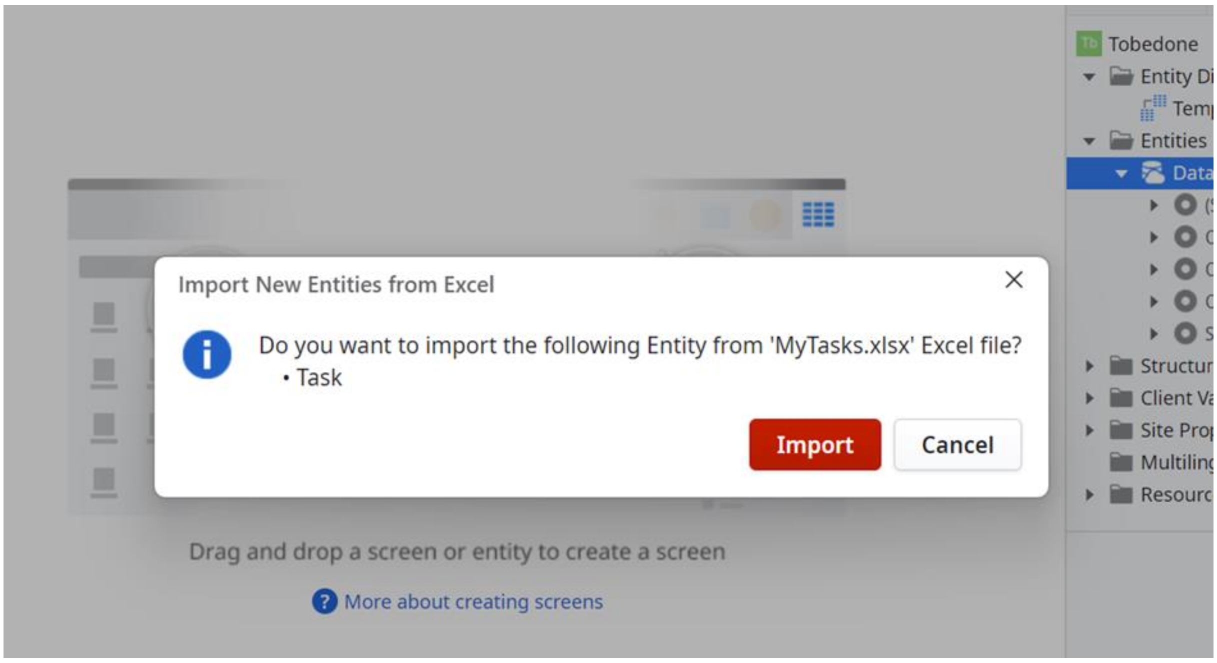
Figure 9 Excel sheet used to create Entities and Entity Attributes



	A	B	C	D	E
1	Description	DueDate	Comments		
2	Start thesis	15/06/2023			
3	Return book	17/08/2023			

Once the file is chosen, Service Studio will pop out a prompt to verify which Entities are to be imported, as shown in Figure 10.

Figure 10 Prompt to confirm import from Excel



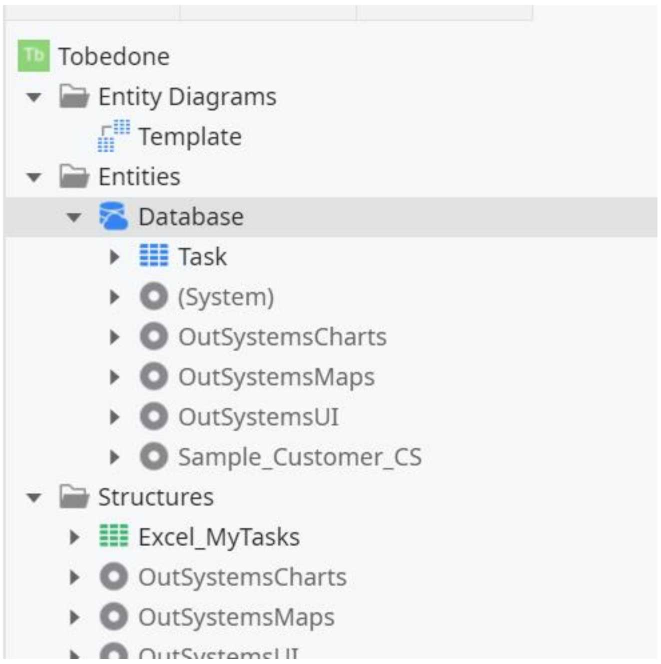
Import New Entities from Excel

Do you want to import the following Entity from 'MyTasks.xlsx' Excel file?
• Task

Drag and drop a screen or entity to create a screen

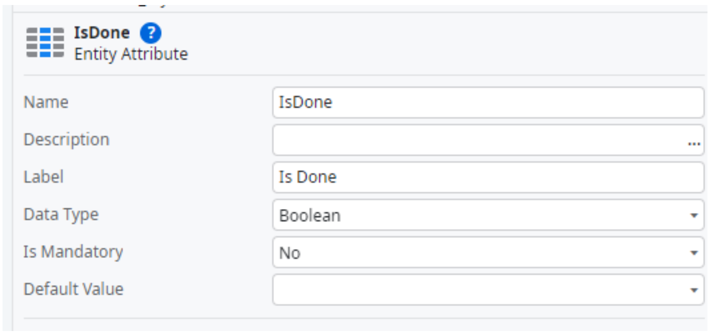
After importing, OutSystems creates the database and logic. The database can be seen in the element tree after it is created, as shown in Figure 11.

Figure 11 Element tree showing the newly created database



After this, the new entity attribute will be added to the database. The attribute is added by right-clicking the Entity on the element tree and choosing “add entity attribute” from the dropdown menu. I chose to call the entity attribute IsDone. I checked the properties and ensured that the Data Type is Boolean since the information needed here is whether the checkbox has been checked or not. The correct properties are shown in Figure 12.

Figure 12 Properties of "IsDone" Entity attribute



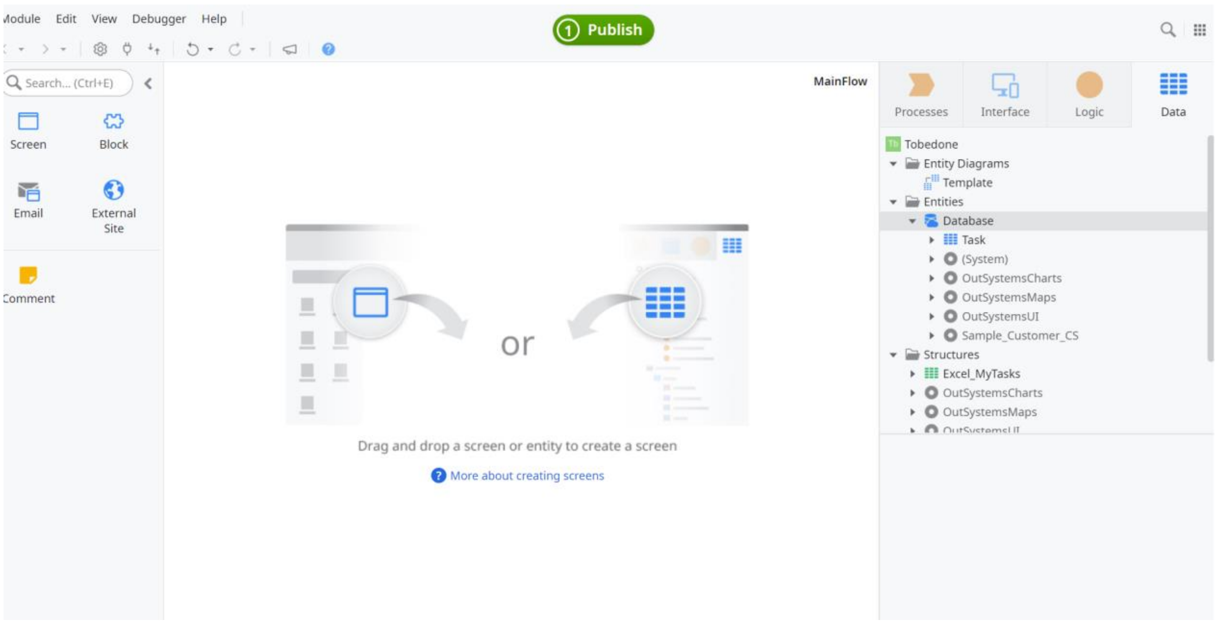
IsDone
Entity Attribute

Name	IsDone
Description	
Label	Is Done
Data Type	Boolean
Is Mandatory	No
Default Value	

5.2.2 Screens and checkbox

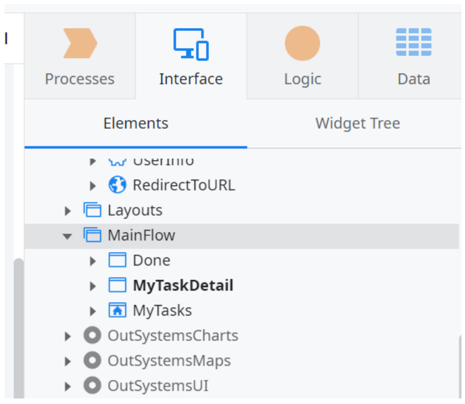
After adding the entity attribute, it was time to add the first two screens by dragging and dropping the TASK-entity to the MainFlow. OutSystems will create screens and logic in the background. Once the screens are ready, the checkbox can be added to the Edit/New task -screen. Service Studio visually guides the developer to drag and drop either a screen or an Entity to the MainFlow to create a screen, as seen in Figure 13.

Figure 13 Service Studio before Screens are added.



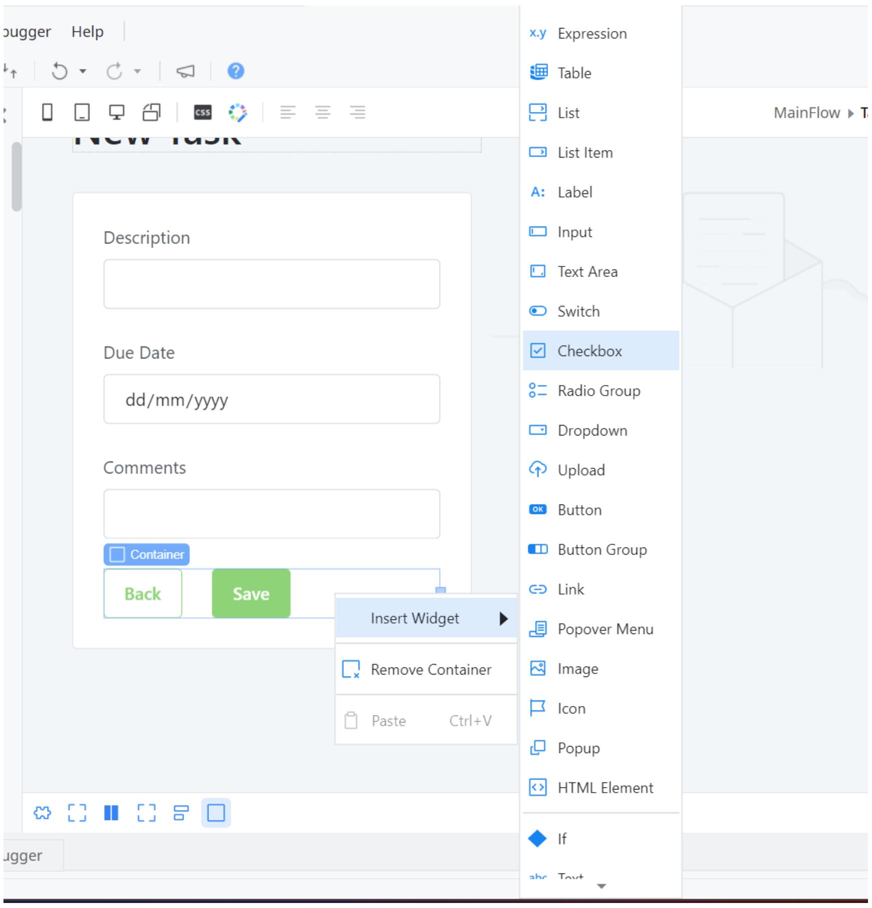
Service Studio will then create two screens, MyTasks, the main screen, and MyTaskDetail, the detail screen used to add or edit the tasks on the list. The screens can be accessed and edited from the Interface-tabs Element tree, as shown in Figure 14.

Figure 14 Elements tree for interface



Next, I added the checkbox by right-clicking on the position I wanted to place it next to the "Back" and "Save"-buttons created by Service Studio. I chose Insert Widget from the dropdown menus and then the checkbox from the widget dropdown, as shown in Figure 15.

Figure 15 Adding the checkbox.

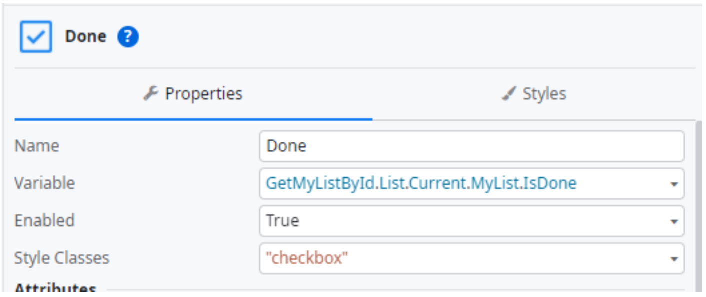


The checkbox could also be added by dragging and dropping the widget from the Toolbox, but I preferred the right-click and dropdown combination for precision.

Then, the checkbox needs an accompanying variable. It is set in the properties box, as shown in Figure 16. Service Studio will automatically suggest different options, but for this case, I chose: “GetMyListById.List.Current.MyList.IsDone”.

By choosing this, the application will update the IsDone Entity Attribute as TRUE or FALSE, depending on whether the checkbox is checked. The update will take place when the Save button is pressed. Service Studio has created the logic for the updating, and I chose to use it here, too, instead of adding new logic on top.

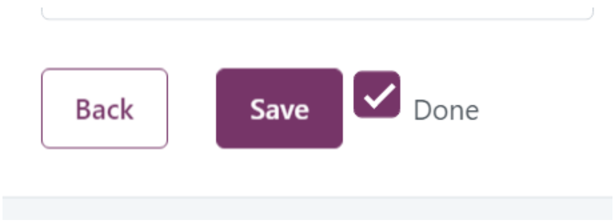
Figure 16 Properties of the Checkbox



Name	Done
Variable	GetMyListById.List.Current.MyList.IsDone
Enabled	True
Style Classes	"checkbox"

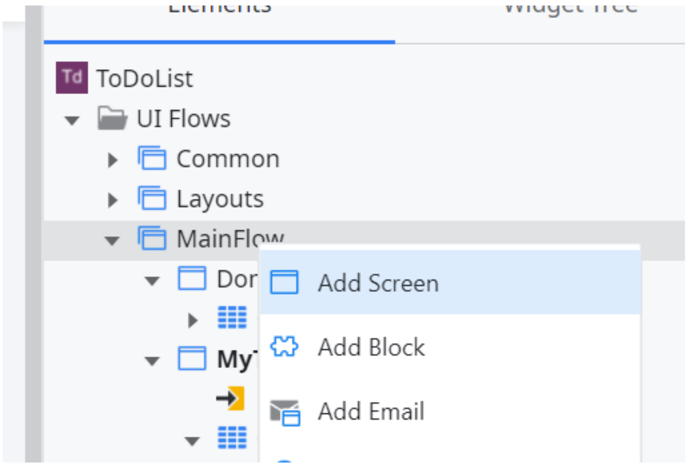
Once the checkbox was complete, I added a label next to the checkbox and typed “Done” on the label. The label can be added by either dragging it from the widget toolbox or, as how I added the checkbox itself, right-clicking and choosing "add widget" from the dropdown. See Figure 17 for the result and placement.

Figure 17 Finished Checkbox with label



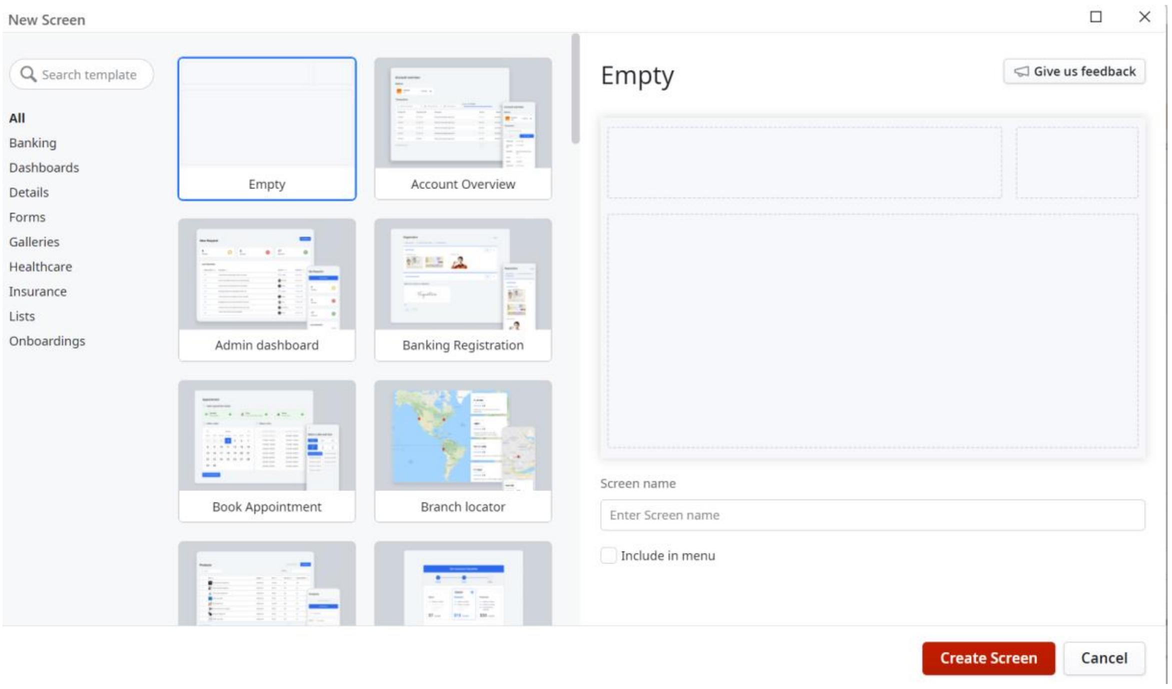
Next, the new screen is added. We start by right-clicking MainFlow in the Elements tree of the Interface tab. We will select Add screen from the dropdown menu, as shown in Figure 18.

Figure 18 Adding a new screen from the Elements tree.



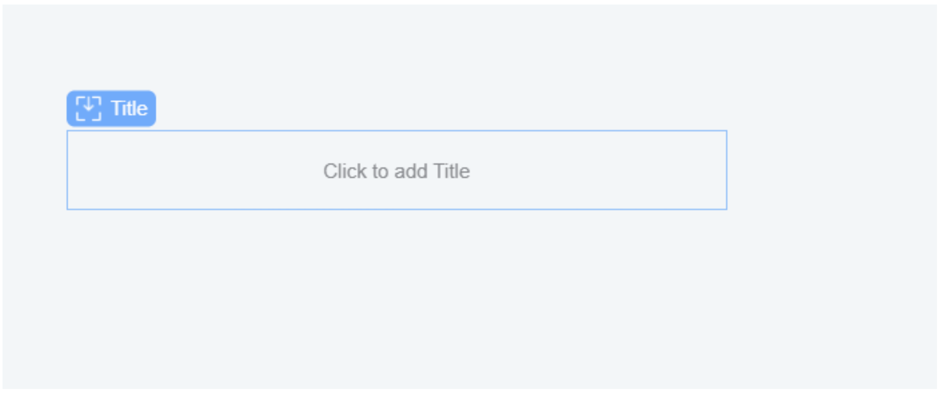
Service Studio will show a pop-up (Figure 19) where the developer can choose a template and name the new screen. Here we choose an empty screen and name it "Done."

Figure 19 Choosing a template for the new screen.



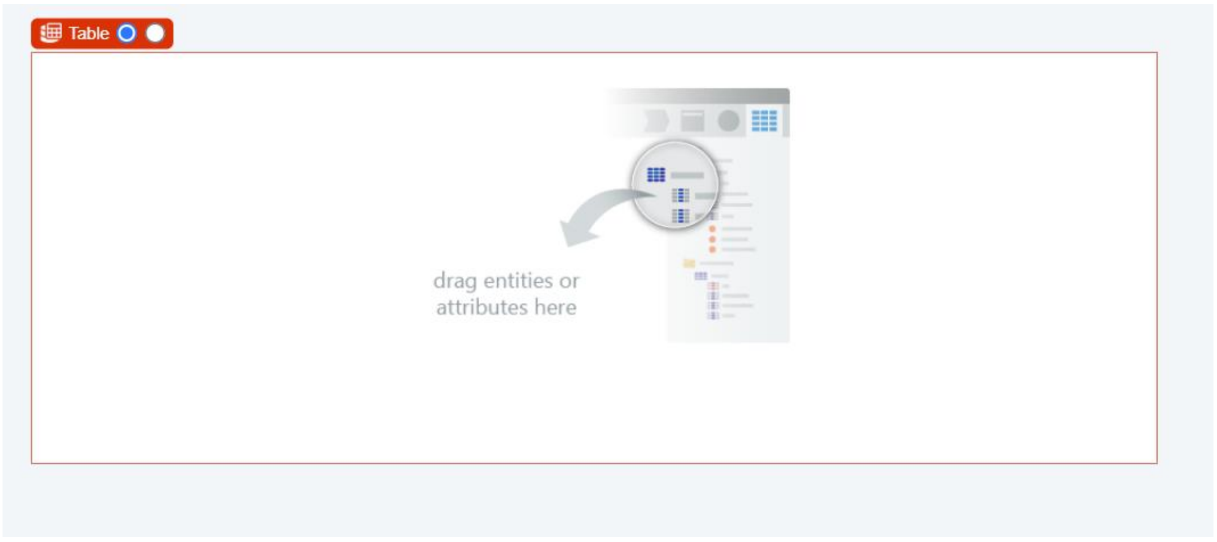
Service Studio does give the developer some help with adding placeholders for the screen title (Figure 20) and Main Content. First, the title needs to be added by clicking the placeholder and typing the title. Since this screen lists out the tasks that are done, I named it "Done."

Figure 20 Placeholder for the screen title



Then I added a Table widget to the screen by dragging and dropping it from the Toolbox to the MainContent-area of the screen. As before, the widget can be added by right-clicking the MainContent -area and choosing the widget from the dropdown menu.

Figure 21 Table widget without data



I added the data to the table by right-clicking the red "Table" button. From the dropdown menu, I chose "Select Source." I checked all Entity attributes from the MyList Entity I created earlier by importing it from Excel and manually adding the "IsDone" entity attribute. For these steps, see Figures 22 and 23.

Figure 22 Selecting a Source for the table.

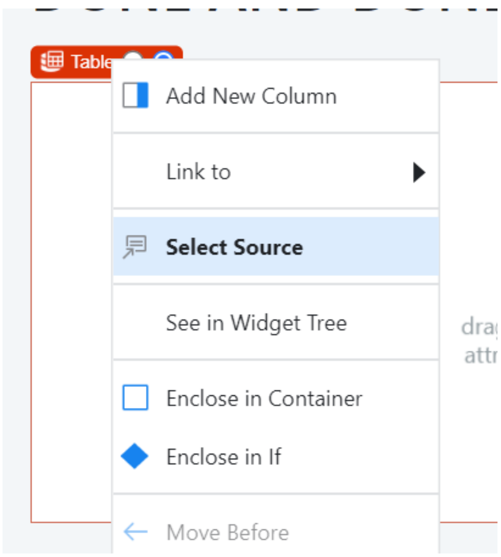
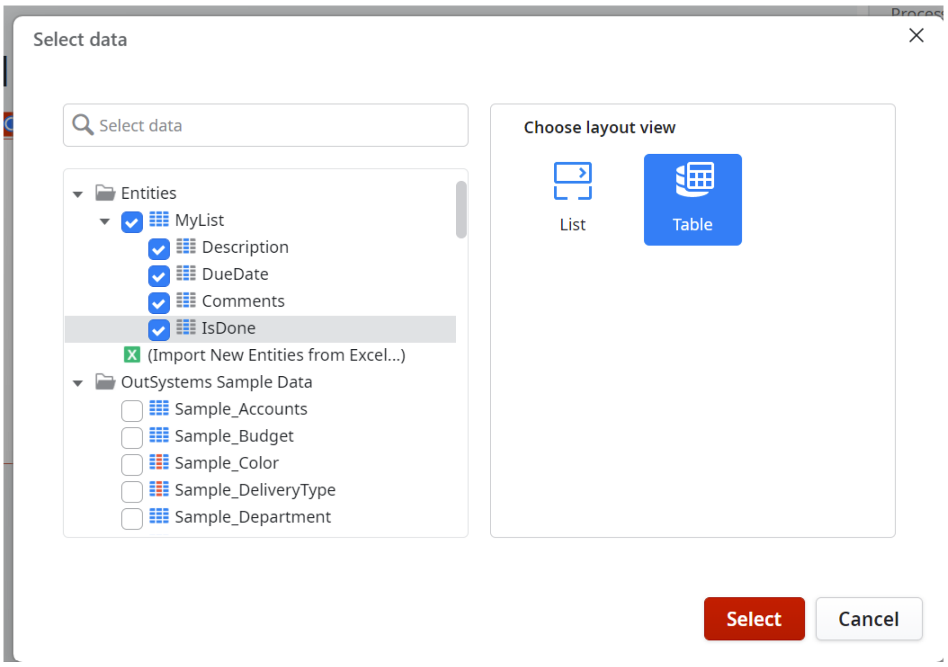


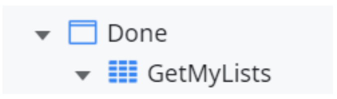
Figure 23 Selecting data for the table from the source.



5.2.3 Aggregate editing

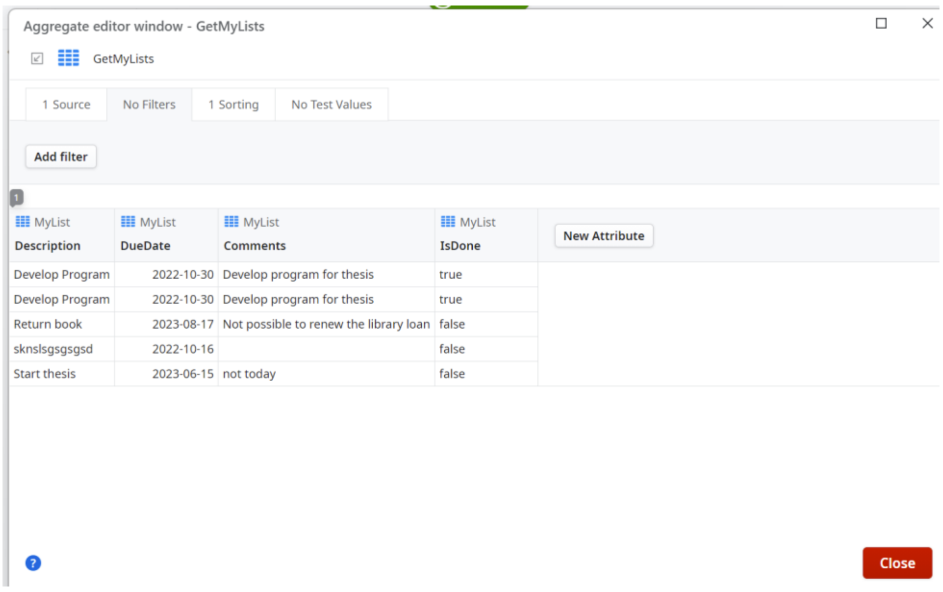
For the Done screen to show only the tasks that have been marked as done with the checkbox, I needed to edit the aggregate already created by Service Studio. The aggregate defines what data is shown in the table. An aggregate called GetMyLists has been created for my new screen by Service Studio and can be accessed from the widget tree, as shown in Figure 24.

Figure 24 GetMyLists aggregate in the widget tree



I opened the Aggregate editor window by double-clicking the GetMyLists aggregate (see Figure 25). The aggregate editor offers options to edit the data shown in the table. It is possible to add sources to combine data from several sources, such as entities and variables. Filters can be used to only show the data according to the desired filtering conditions. Lastly, the data can be sorted with the aggregate.

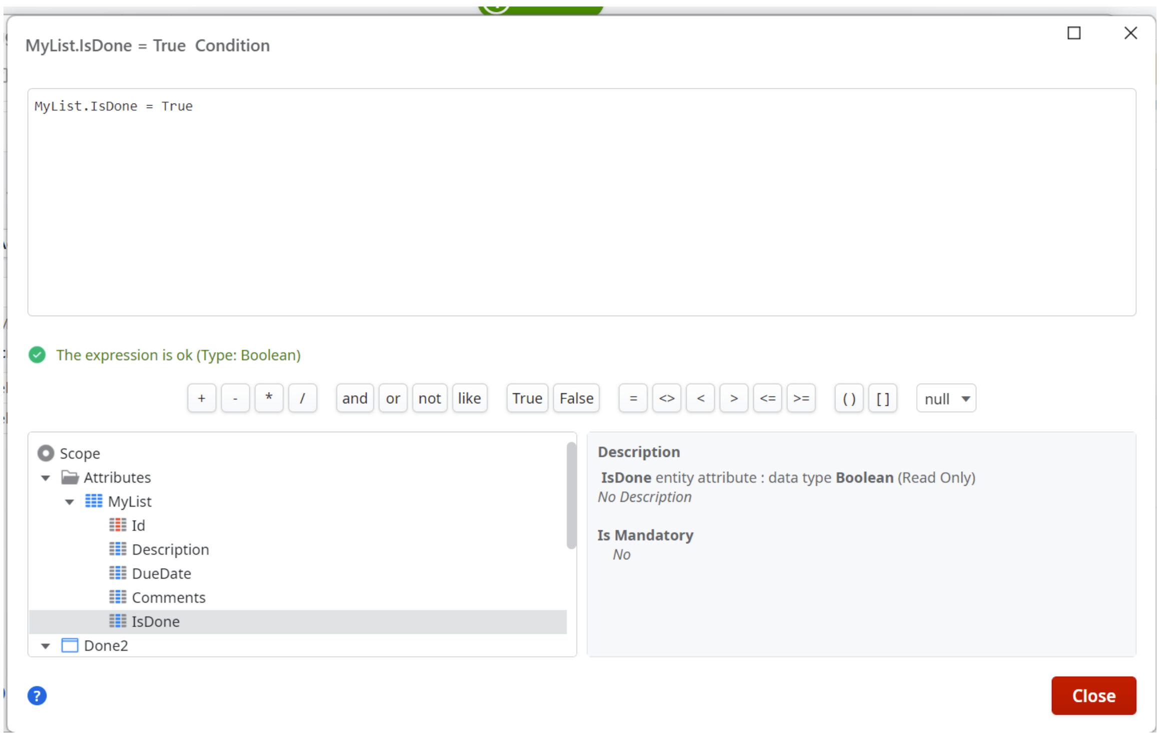
Figure 25 Aggregate editor



MyList Description	MyList DueDate	MyList Comments	MyList IsDone
Develop Program	2022-10-30	Develop program for thesis	true
Develop Program	2022-10-30	Develop program for thesis	true
Return book	2023-08-17	Not possible to renew the library loan	false
sknslsgsgsgsd	2022-10-16		false
Start thesis	2023-06-15	not today	false

In the editor, I opened the Filters tab (or No Filters tab if none have been made). After clicking on "Add filter," a Filter condition editor opens. There I added the following expression: `MyList.IsDone = True`, and now the aggregate only shows items that have been marked as done using the checkbox created earlier. The filter condition editor with the expression used in this case is shown in Figure 26.

Figure 26 Filter condition editor


MyList.IsDone = True Condition

```
MyList.IsDone = True
```

The expression is ok (Type: Boolean)

Description
IsDone entity attribute : data type Boolean (Read Only)
No Description

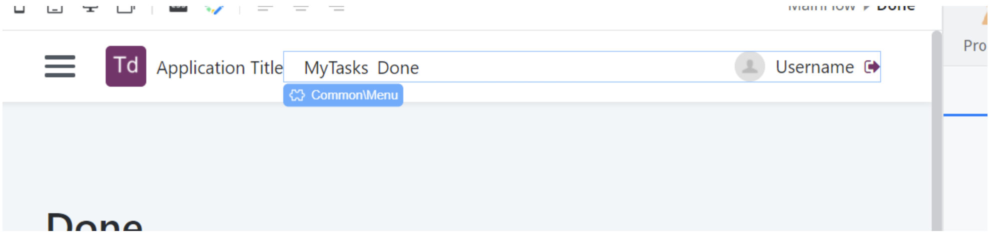
Is Mandatory
No

The process is then repeated for the Main screen by adding an otherwise similar filter to the aggregate. For the main screen, the condition should be `MyList.IsDone = False`. If this step is not taken, the Main screen will show all the tasks regardless of their status as done or not.

5.2.4 Finishing touches

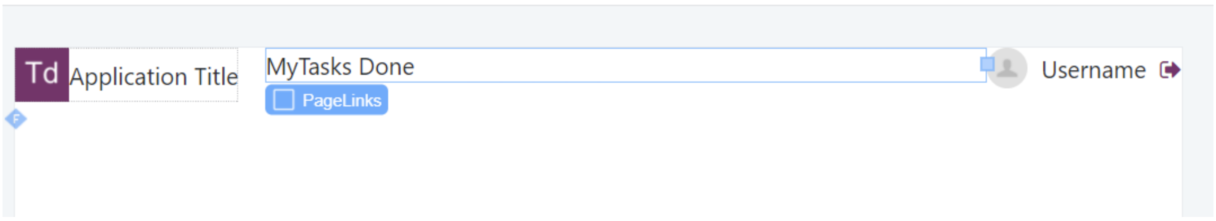
Lastly, a link to the “Done” screen made from scratch needs to be added to the Common menu. OutSystems created the menu during the making of the first two screens, but the new screen needs to be added separately. The Common menu can be accessed for editing by double-clicking the menu on any of the screens.

Figure 27 Common menu



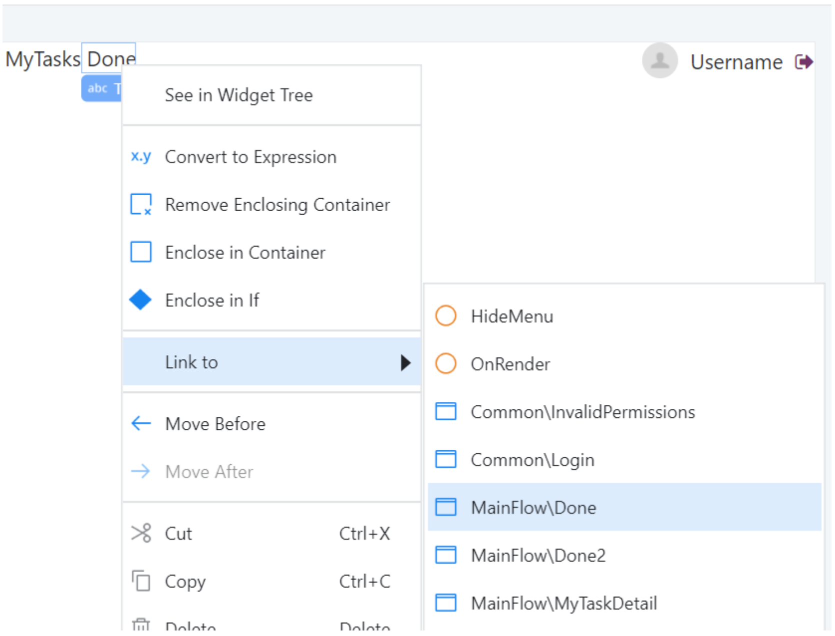
Then the link can be added by typing the link text, in this case, "Done," to the PageLinks-widget in the menu. The placement of the link text and PageLinks-widget is shown in Figure 28.

Figure 28 PageLinks widget



Right-clicking the added text opens a dropdown menu. From that dropdown, I chose "link to" and then the screen "Mainflow\Done," which was suggested by OutSystems. The dropdown menu and the suggested link destinations are shown in Figure 29.

Figure 29 Creating the link to Done-screen.



Now the link to the “Done”-screen is in the Common Menu and can be accessed from any screen.

Then the program can be published by clicking the green “publish” -button on the top of the screen. After Service Studio has deployed the application, it can be opened in the browser.

The basics of the app are now done. The budding OutSystems-developer is now able to develop a simple program and use it as a base to practice and test their development ideas.

5.3 The study materials

The study materials will consist of a PowerPoint presentation that will guide the students through the start of the development process and then the three development stages: setting up the database and entities, setting up the screens, and then using aggregates to control the data seen on the screens. One presentation will also be made about Service Studio, giving an overview of the platform. The contents of this report could also be used as guide for beginner developers.

I will also include the small and simple Excel file to be used in developing the To Do-app. It will contain the basic info for the database, and the Entity attributes that Service Studio will use to create them. Then that information will be further used to create screens and tables within Service Studio.

6 Summary

The goal of this thesis was to develop a simple application and to make the accompanying study materials. The study materials are based on the development process depicted in this thesis. HAMK needed study materials on the following subjects: custom validations and data modelling, user-interface, utilizing the built-in top menu, utilizing existing templates in their own application, and writing custom SQL queries. Apart from the custom SQL queries, the materials discuss all the needed subjects. Instead of custom SQL queries, aggregates were studied and used, as OutSystems suggests using them instead. (OutSystems, 2022m)

The development was straightforward and simple, but the challenge was keeping the scope of the application sensible. The point of Low-code development is that it is simple to do. So, a simple application really is that. It is easy for the scope to get out of hand; I was considering trying out integration with Outlook but ultimately decided that it would have taken out the simplicity of the application.

Overall, the application is working as it should. I left it relatively simple so the student could use it as a base to build on and test their ideas. One thing that the application lacks is the possibility to delete the tasks that are done. Nevertheless, that could be considered a challenge for the students. How would they develop it further? What would they do differently? There are so many choices in developing that we cannot think of the student-made application as clones but more like siblings or cousins of the original.

Low-code development is growing rapidly; some estimates are as high as 70% of all development made with low-code applications in 2025. Low-code will not likely grow by replacing traditional development but by making development possible for more people and increase overall total development. So, more development overall with the rise of citizen developers.

References

Bratincevic, J., & Koplowitz, R. (2021, May 11). *The Forrester Wave^{TM}: Low-Code Development Platforms For Professional Developers, Q2 2021*. https://reprints2.forrester.com/#/assets/2/228/RES161668/report

Hay, R. (2021, October 20). *What Are Low-Code/No-Code Platforms?* https://www.itprotoday.com/no-codelow-code/what-are-low-codeno-code-platforms

Kissflow. (2022, July 15). *Low-Code vs High-Code : Choose The Best For Your Business App Development in 2022*. https://kissflow.com/low-code/low-code-vs-high-code/

Liptak, J., & Horwitz, L. (2021, February). *What is citizen development, and what is a citizen developer?* https://www.techtarget.com/searchsoftwarequality/definition/citizen-development

nocode.tech. (2022). *NoCode - The history of no-code: from VisiCalc to Airtable*. https://www.nocode.tech/article/the-evolution-of-no-code-from-visicalc-to-airtable

Northcutt, R. (2021, August 16). *High Code vs. Low Code vs. No Code: Why Choose Just One? | Acquia*. https://www.acquia.com/blog/high-code-vs-low-code-vs-no-code-why-choose-just-one

obviously.ai. (2022). *The Short, Inventive History of No-Code and the Long Future Ahead | Obviously AI Blog*. https://www.obviously.ai/post/the-history-of-no-code

OutSystems. (2022a). *Entity Relationships - OutSystems 11 Documentation*. OutSystems. https://success.outsystems.com/Documentation/11/Developing_an_Application/Use_Data/Data_Modeling/Entity_Relationships

OutSystems. (2022b). *OutSystems Pricing | OutSystems*. https://www.outsystems.com/pricing-and-editions/?_gl=1*1356bm1*_ga*NjA1MjI5MDY4LjE2NTU4MDc2Njc.*_ga_ZD4DTMHWR2*MTY2NDkwODE5OS4xNC4xLjE2NjQ5MTA4MzguMTkuMC4w

OutSystems. (2022c). *What is AI-assisted development? | Evaluation Guide | OutSystems*. https://www.outsystems.com/evaluation-guide/what-is-ai-assisted-development/

OutSystems. (2022d). *What is OutSystems.ai? | Evaluation Guide | OutSystems*. https://www.outsystems.com/evaluation-guide/what-is-outsystems-ai/

OutSystems. (2022e). *What kind of suggestions can developers get from AI-assisted development? | Evaluation Guide | OutSystems*. https://www.outsystems.com/evaluation-guide/suggestions-developers-get-ai-assisted-development/

OutSystems. (2022f, June 29). *Application Templates - OutSystems 11 Documentation*. https://success.outsystems.com/Documentation/11/Developing_an_Application/Application_Templates

OutSystems. (2022g, June 29). *Choose the right app for your project - OutSystems 11 Documentation*. https://success.outsystems.com/Documentation/11/Getting_started/Choose_the_right_app_for_your_project

OutSystems. (2022h, June 29). *Design UI - OutSystems 11 Documentation*. https://success.outsystems.com/Documentation/11/Developing_an_Application/Design_UI

OutSystems. (2022i, June 29). *Implement Application Logic - OutSystems 11 Documentation*. https://success.outsystems.com/Documentation/11/Developing_an_Application/Implement_Application_Logic

OutSystems. (2022j, June 29). *Managing the Applications Lifecycle - OutSystems 11 Documentation*. OutSystems. https://success.outsystems.com/Documentation/11/Managing_the_Applications_Lifecycle

OutSystems. (2022k, June 29). *Service Studio Overview - OutSystems 11 Documentation*. https://success.outsystems.com/Documentation/11/Getting_started/Service_Studio_Overview

OutSystems. (2022l, June 29). *Understanding how to create an app - OutSystems 11 Documentation*. https://success.outsystems.com/Documentation/11/Getting_started/Getting_started_with_your_own_app_use_case/Understanding_how_to_create_an_app

OutSystems. (2022m, September 28). *SQL Queries - OutSystems 11 Documentation*. https://success.outsystems.com/Documentation/11/Developing_an_Application/Use_Data/Query_Data/SQL_Queries

OutSystems. (2022n, October 6). *High availability and scalability strategies - OutSystems 11 Documentation*. https://success.outsystems.com/Documentation/11/Setup_and_maintain_your_OutSystems_infrastructure/Setting_Up_OutSystems/Possible_setups_for_an_OutSystems_infrastructure/High_availability_and_scalability_strategies

Rubens, P. (2014, February 7). DIY apps and the rise of 'citizen developers'. *BBC News*. https://www.bbc.com/news/business-26052344

Wong, J., & Iijima, K. (2021, September 20). *Gartner Reprint*. https://www.gartner.com/doc/reprints?id=1-24AEKPVN&ct=201001&st=sb

Annex 1: Material management plan

The application developed for this thesis is hosted in my personal OutSystems environment, and I will also download a copy of it for backup. When the material gets used in the course, there will be several “siblings” for the application created by the students.

The PowerPoint presentations and the Excel file will be handed over to HAMK once the course teachers accept them. Copies will be saved in my personal OneDrive or Google Drive for at least a year, after which they will be removed. The backup copy of the developed application will also be saved in the same places and handed over to HAMK.

Attachment 1: MyTasks.xlsx

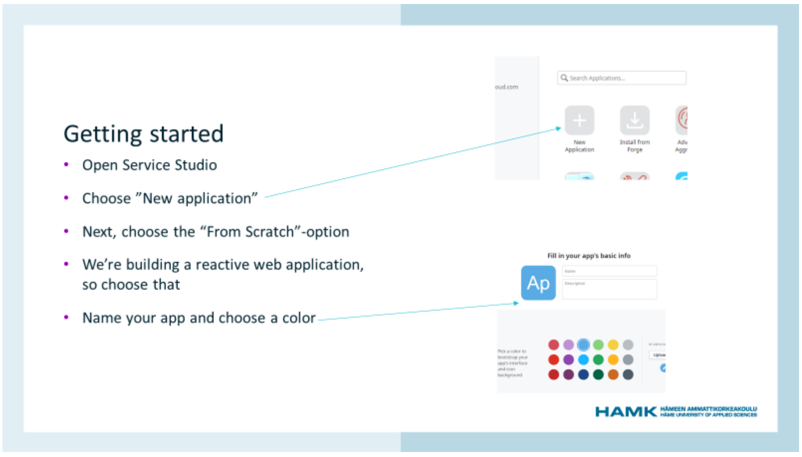
Description	DueDate	Comments
Start thesis	15/06/2023	
Return book	17/08/2023	

Attachment 2: PowerPoint presentations

Building an app with OutSystems

Build your first application

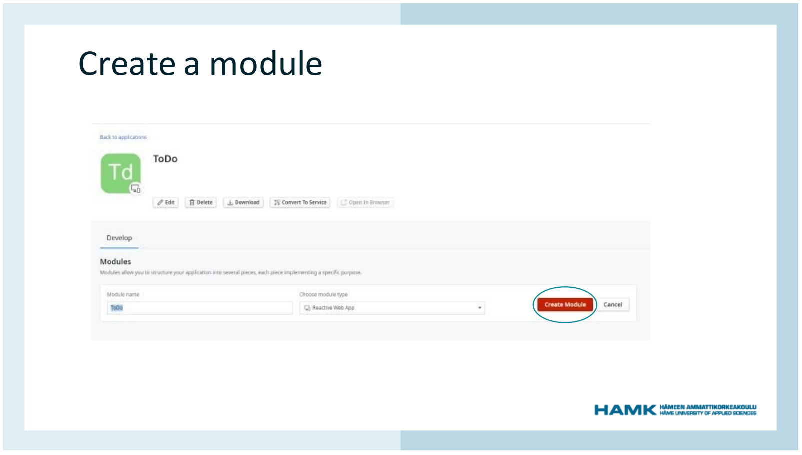
outsystems
HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES



Getting started

- Open Service Studio
- Choose ”New application”
- Next, choose the “From Scratch”-option
- We’re building a reactive web application, so choose that
- Name your app and choose a color

Search Applications...
New Application
Install from Forge
Fill in your app's basic info
Ap
HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES



Create a module

Back to applications

ToDo

Develop

Modules

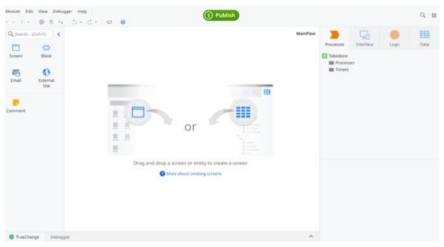
Modules allow you to structure your application into several pieces, each piece implementing a specific purpose.

Module name: ToDo

Choose module type: Reactive Web App

Workspace

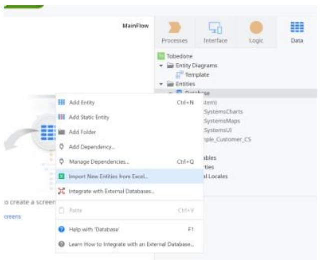
In the beginning, your workspace should look like this



HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Import Entity

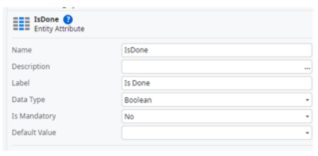
- Right-click on Entities in the Data-tab
- Choose “Import New Entities from Excel” and find the file you intend to use
- After importing, OutSystems creates the database and logic.
  - The database can be seen in the element tree after it is created.



HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Adding an Entity Attribute

- Right-click the Entity on the element tree
- Choose “add entity attribute” from the dropdown menu
- Check the properties and ensure that the Data Type is Boolean
  - The information needed here is whether the checkbox has been checked or not.

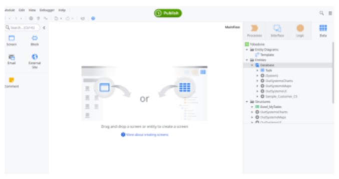

IsDone
Entity Attribute

Name	IsDone
Description	
Label	Is Done
Data Type	Boolean
Is Mandatory	No
Default Value	

HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Creating Screens using Entities 1/2

- Drag and drop the Entity you just created from the element tree to the MainFlow
- OutSystems will create screens and logic in the background

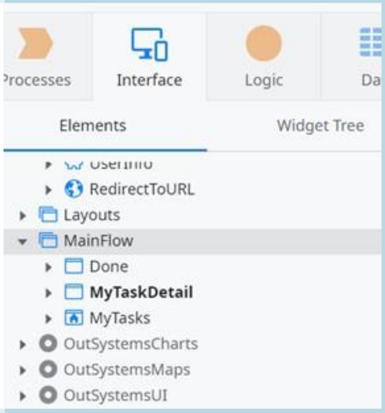


HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Creating Screens using Entities 2/2

- Service Studio will create two screens:
  - MyTasks - the main screen
  - MyTaskDetail - screen used for adding or editing the tasks on the list.
- The screens can be accessed and edited from the Interfacetabs Element tree.

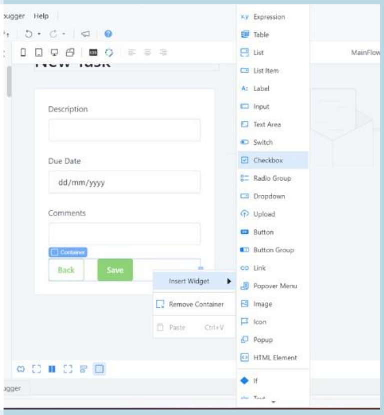
HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES



Adding the Checkbox

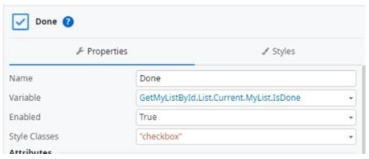
- In the MyTaskDetail screen, Add the Checkbox by rightclicking the position you would like to place it.
- Choose Checkbox from the dropdown menu
- You can also add the Checkbox by dragging and dropping it from the Toolbox

HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES



Configuring the Checkbox properties

- The checkbox needs an accompanying variable.
  - It is set in the properties box
- choose: "GetMyListById.List.Current.MyList.IsDone"
- By choosing this, the application will update the IsDone Entity Attribute as TRUE or FALSE, depending on whether the checkbox is checked.



Done

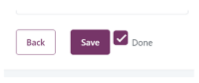
Properties | Styles

Name	Done
Variable	GetMyListById.List.Current.MyList.IsDone
Enabled	True
Style Classes	"checkbox"

HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Finishing touches to the Checkbox

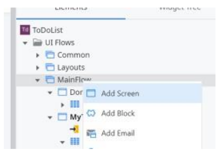
- Add a label with the text "Done."
- The label can be added by either dragging it from the widget toolbox or right-clicking and choosing "add widget" from the dropdown.
- Type "Done" into the text area



HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Adding a screen from Scratch

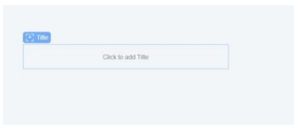
- Start by right-clicking MainFlow in the Elements tree of the Interface tab.
- Select Add screen from the dropdown menu
- Service Studio will show a popup where you can choose a template and name the new screen.
- Choose Empty screen and name it "Done."



HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Adding a Title to the screen

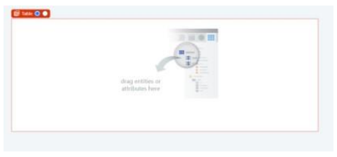
- Service Studio does give the developer some help with adding placeholders for the screen title and Main Content.
- Add the title by clicking the placeholder and typing the title
  - Since this screen lists out the tasks that are done, name it "Done."



HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Adding a Table widget to the screen

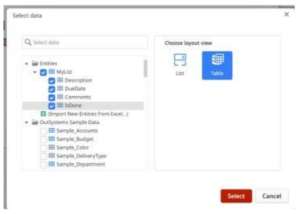
- Add a Table widget to the screen by dragging and dropping it from the Toolbox to the MainContent -area of the screen.
- As before, the widget can be added by right -clicking the MainContent -area and choosing the widget from the dropdown menu.



HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Configuring the Table widget

- Add the data to the table by right-clicking the red "Table" button.
- From the dropdown menu, choose "Select Source."
- Check all Entity attributes from the MyList Entity


Select data

Select data

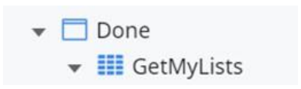
- Entities
  - ✓ MyList
    - ✓ Description
    - ✓ DueDate
    - ✓ Comments
    - ✓ IsDone
  - (Import New Entities from Excel...)
- OutSystems Sample Data
  - Sample_Accounts
  - Sample_Budget
  - Sample_Color
  - Sample_DeliveryType
  - Sample_Department

Choose layout view

HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Aggregates

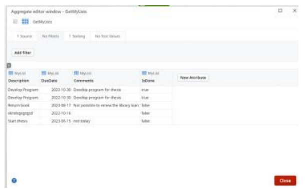
- For the Done screen to show only the tasks that have been marked as done with the checkbox, we need to edit the aggregate already created by Service Studio.
- The aggregate defines what data is shown in the table.
- An aggregate called GetMyLists has been created for the screen by Service Studio and can be accessed from the widget tree



HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

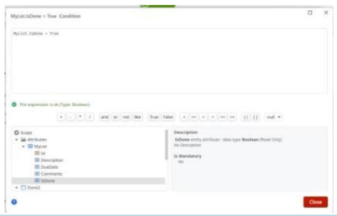
Aggregate editor

- In the editor, open the Filters tab (or No Filters tab if none have been made).
- After clicking on "Add filter," a Filter condition editor opens. There add the following expression: MyList.IsDone = True, and now the aggregate only shows items marked as done using the checkbox created earlier.
- The process is then repeated for the Main screen by adding an otherwise similar filter to the aggregate. For the main screen, the condition should be MyList.IsDone = False.


Aggregate editor window - GetMyLists

GetMyLists

MyList Description	MyList DueDate	MyList Comments	MyList IsDone
Develop Program	2022-10-30	Develop program for thesis	true
Develop Program	2022-10-30	Develop program for thesis	true
Return book	2023-08-17	Not possible to renew the library loan	false
skrsfsgsgsgsd	2022-10-16		false
Start thesis	2023-08-15	not today	false


MyList.IsDone = True Condition

```
MyList.IsDone = True
```

The expression is ok (Type: Boolean)

+ - * / and or not like True False = <> < > <= >= () [] null
Close

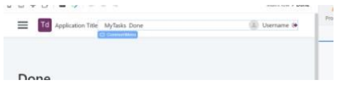
- Scope
- Attributes
  - MyList
    - Id
    - Description
    - DueDate
    - Comments
    - IsDone
- Done2

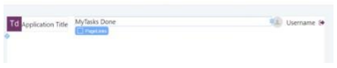
Description
IsDone entity attribute : data type Boolean (Read Only)
No Description

Is Mandatory
No

Editing the Common menu 1/2

- The Common menu can be accessed for editing by double - clicking the menu on any of the screens.
- Then the link can be added by typing the link text, in this case, "Done," to the PageLinks - widget in the menu.

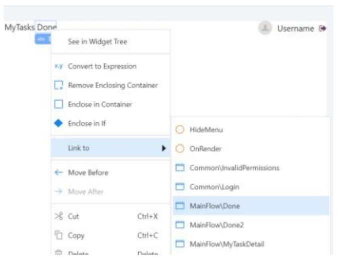




HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Editing the Common menu 2/2

- Right-clicking the added text opens a dropdown menu.
- From that dropdown, choose "link to" and then the screen "Mainflow\Done," which was suggested by OutSystems.
- Now the link to the "Done"-screen is in the Common Menu and can be accessed from any screen.



HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Publishing

- Then the program can be published by clicking the green "publish" -button on the top of the screen.
- After Service Studio has deployed the application, it can be opened in the browser.

HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Now what?

- The basics of the app are now done.
- Feel free to venture out and develop this little app further to fit your needs.
- Maybe add a delete button or more screens?

HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

OutSystems Service Studio 11

An Overview

outsystems

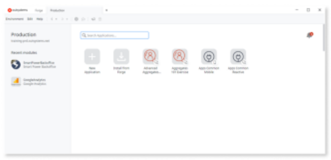
OutSystems Service Studio 11

- Applications are developed in Service Studio, the main IDE (Integrated development environment) in OutSystems. Service Studio is used to build, edit, and debug applications.
- To use Service Studio, the user needs access to an OutSystems environment.
  - The free option of OutSystems provides one runtime environment that can be used for training and development.
  - Two subscription options provide more runtime environments that are fully isolated.

HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Environment tab

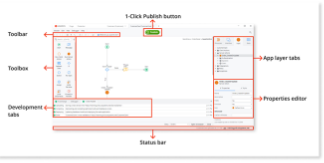
- Service Studio opens to the Environment tab showing users the applications currently on the server.
- It allows the user to view and access them, create new applications, and install application templates made by the community from the Forge.



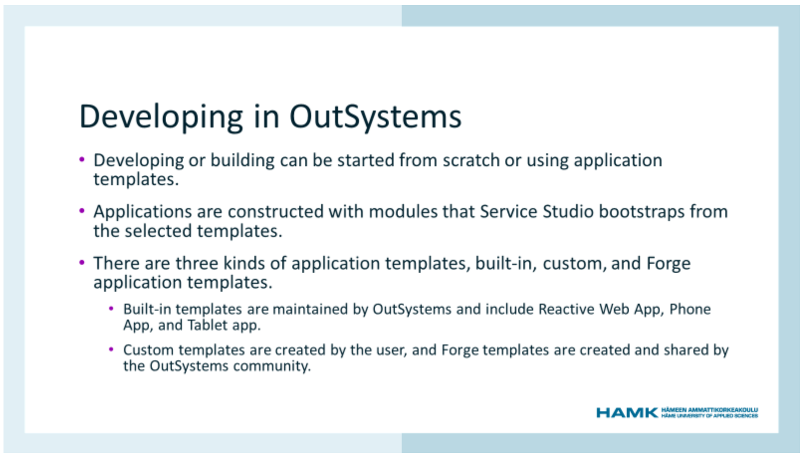
HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES

Workspace

- The Main editor is the area in the middle of the screen where the interface and logic are designed. Shortcuts to the most common operations can be found in the Toolbar, and the Toolbox contains the tools and widgets.
- The four development tabs consist of the TrueChange tab that displays errors and warnings, the debugger tab that is used for debugging as the name implies, the 1-Click Publish tab that shows the progress and result of the deployment, and the Search Results tab that shows search results for searches performed in the module.
- The other elements of the workspace are the Application layer tabs that contain the elements for the different layers (processes, user interface, logic, and data model), the 1-click publish button that starts the deployment process if the application has no errors, the Status Bar, and the Properties editor.



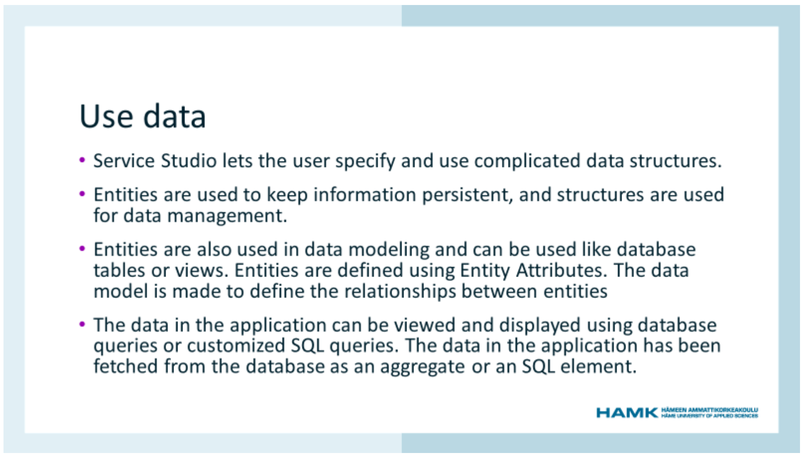
HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES



Developing in OutSystems

- Developing or building can be started from scratch or using application templates.
- Applications are constructed with modules that Service Studio bootstraps from the selected templates.
- There are three kinds of application templates, built-in, custom, and Forge application templates.
  - Built-in templates are maintained by OutSystems and include Reactive Web App, Phone App, and Tablet app.
  - Custom templates are created by the user, and Forge templates are created and shared by the OutSystems community.

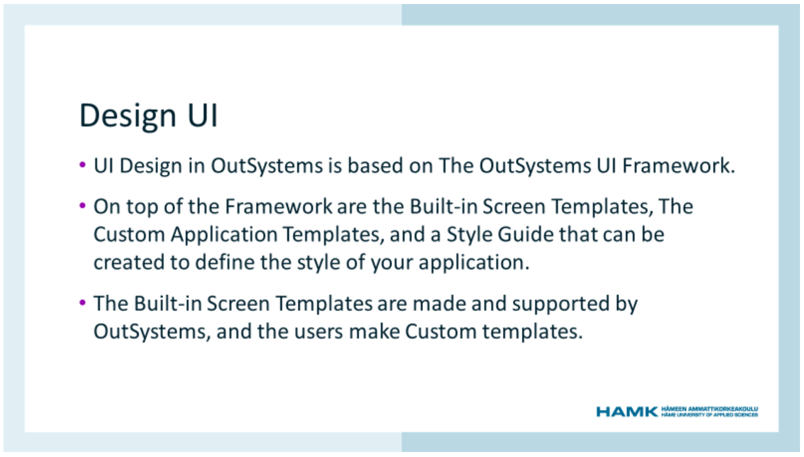
HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES



Use data

- Service Studio lets the user specify and use complicated data structures.
- Entities are used to keep information persistent, and structures are used for data management.
- Entities are also used in data modeling and can be used like database tables or views. Entities are defined using Entity Attributes. The data model is made to define the relationships between entities
- The data in the application can be viewed and displayed using database queries or customized SQL queries. The data in the application has been fetched from the database as an aggregate or an SQL element.

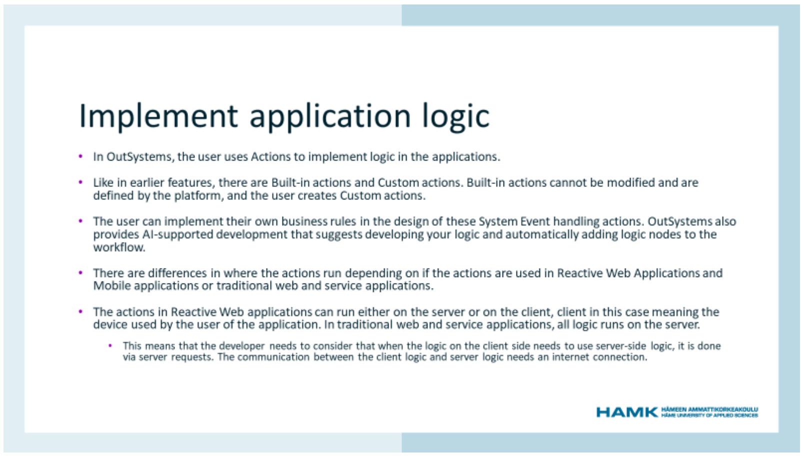
HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES



Design UI

- UI Design in OutSystems is based on The OutSystems UI Framework.
- On top of the Framework are the Built-in Screen Templates, The Custom Application Templates, and a Style Guide that can be created to define the style of your application.
- The Built-in Screen Templates are made and supported by OutSystems, and the users make Custom templates.

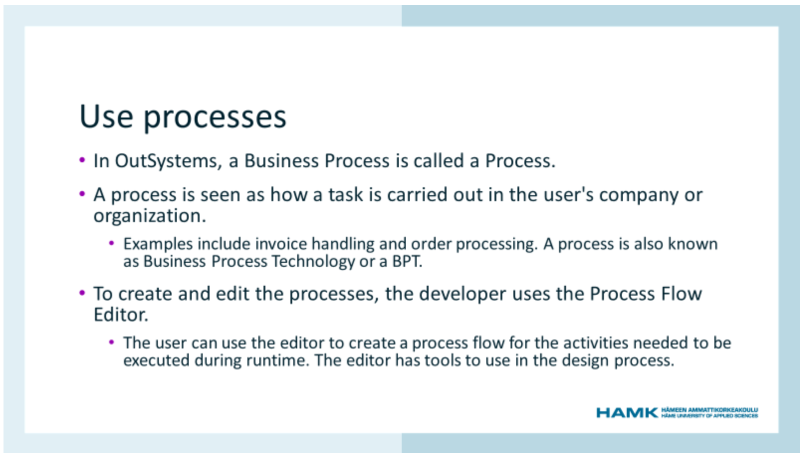
HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES



Implement application logic

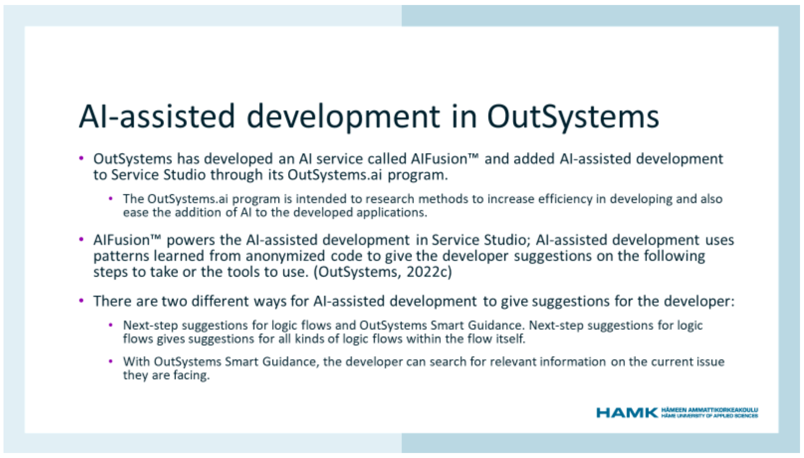
- In OutSystems, the user uses Actions to implement logic in the applications.
- Like in earlier features, there are Built-in actions and Custom actions. Built-in actions cannot be modified and are defined by the platform, and the user creates Custom actions.
- The user can implement their own business rules in the design of these System Event handling actions. OutSystems also provides AI-supported development that suggests developing your logic and automatically adding logic nodes to the workflow.
- There are differences in where the actions run depending on if the actions are used in Reactive Web Applications and Mobile applications or traditional web and service applications.
- The actions in Reactive Web applications can run either on the server or on the client, client in this case meaning the device used by the user of the application. In traditional web and service applications, all logic runs on the server.
  - This means that the developer needs to consider that when the logic on the client side needs to use server-side logic, it is done via server requests. The communication between the client logic and server logic needs an internet connection.

HAMK HÄMEEN AMMATTIKORKEAKOULU HÄME UNIVERSITY OF APPLIED SCIENCES



Use processes

- In OutSystems, a Business Process is called a Process.
- A process is seen as how a task is carried out in the user's company or organization.
  - Examples include invoice handling and order processing. A process is also known as Business Process Technology or a BPT.
- To create and edit the processes, the developer uses the Process Flow Editor.
  - The user can use the editor to create a process flow for the activities needed to be executed during runtime. The editor has tools to use in the design process.



AI-assisted development in OutSystems

- OutSystems has developed an AI service called AIFusion™ and added AI-assisted development to Service Studio through its OutSystems.ai program.
  - The OutSystems.ai program is intended to research methods to increase efficiency in developing and also ease the addition of AI to the developed applications.
- AIFusion™ powers the AI-assisted development in Service Studio; AI-assisted development uses patterns learned from anonymized code to give the developer suggestions on the following steps to take or the tools to use. (OutSystems, 2022c)
- There are two different ways for AI-assisted development to give suggestions for the developer:
  - Next-step suggestions for logic flows and OutSystems Smart Guidance. Next-step suggestions for logic flows gives suggestions for all kinds of logic flows within the flow itself.
  - With OutSystems Smart Guidance, the developer can search for relevant information on the current issue they are facing.