University of South-Eastern Norway

IIA1319: Software Engineering

Software Development

Isak Skeie, 245362

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

Contents

Contents 2
1 Introduction 3
2 Elaboration Phase 4
2.1 Requirements and Use Case Diagram 4
2.2 Iterations 5
2.3 System Architecture 5
2.4 Domain Model 6
3 Construction Phase 7
3.1.1 Fully Dressed Use Case Document *7*
3.2 Interaction Diagram, Class Diagram and implementation 8
4 Summary 11
5 References 12
Appendices 13

1 Introduction

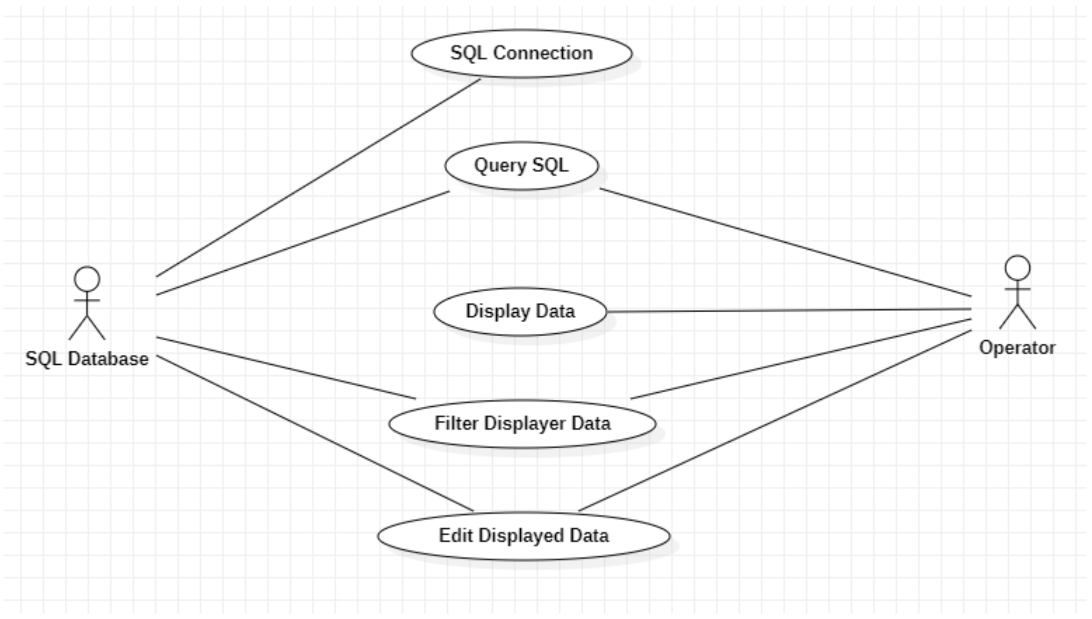
From the Software specification, a recipe reporting tool is developed for the company Kemira. Kemira produces chemicals with a batch production, using recipes to create the batches. The reporting application displays the batches in a table, which gives the operator oversight over previous batches together with analysis. The application is web based, making it accessible across several computers on the production network. Its going to be based on a Blazor server-side architecture. Making the application responsive and scalable. (Microsft, 2022)

2 Elaboration Phase

2.1 Requirements and Use Case Diagram

The requirements are based on the FURPS method with the operator using this for production purposes in mind. Based on the Functional requirements the Use Case Diagram is created, for further development the Use Case Diagram would likely be extended to fit more of the functionality the operator want. (Skeie, 2022)

1. Functional
    a. Connect to SQL, Be able to connect to th SQL server
    b. Display Data in a Table, Display data from SQL to the user
    c. Query data to SQL, Query both ways, update and fetch data
    d. Filter Columns in table, Filter columns in tables
    e. Filter number of rows, Filter rows in tables
    f. Filter by date, Filter table view by date
    g. Edit Table, Edit instances in table, this gets uploaded to SQL
    h. Reset Table, Reset filters applied to table
2. Usability
    a. Web Application
    b. Available across network, Several computers will need access
    c. Distinct grouping of table pages, Easier to navigate to correct table
3. Reliability
    a. Run 24x7, production doesn’t stop
    b. Website doesn’t crash when errors occur.
    c. Certificate for website is trusted
    d. Handles query exceptions
    e. Handle connection exceptions
4. Performance
    a. Insignificant loading time on user interaction
    b. Several users at once
5. Supportability
    a. GUI is made up of components, modular
    b. Runs on IIS



Figur 1: Use Case Diagram for application

2.2 Iterations

In total the could be 5 iterations for a functional product to be developed. The first iteration would be the elaboration phase, gathering requirements and creating diagrams for further elaboration. For each of the next iterations, the application is being constructed. The first construction iteration will focus on the most critical use cases, which are SQL Connection, SQL Query, and Data Display. For the next iteration, Data filter and filtered data display is in focus. For the 4th iteration, the use case Filter Data Display is in focus. The last iteration will focus on testing the system, to make sure it meets the requirements created in the first iteration.

With 5 iterations in place, it could be estimated that 5-7 weeks is needed for the project. Rougly one week per Iteration.

2.3 System Architecture

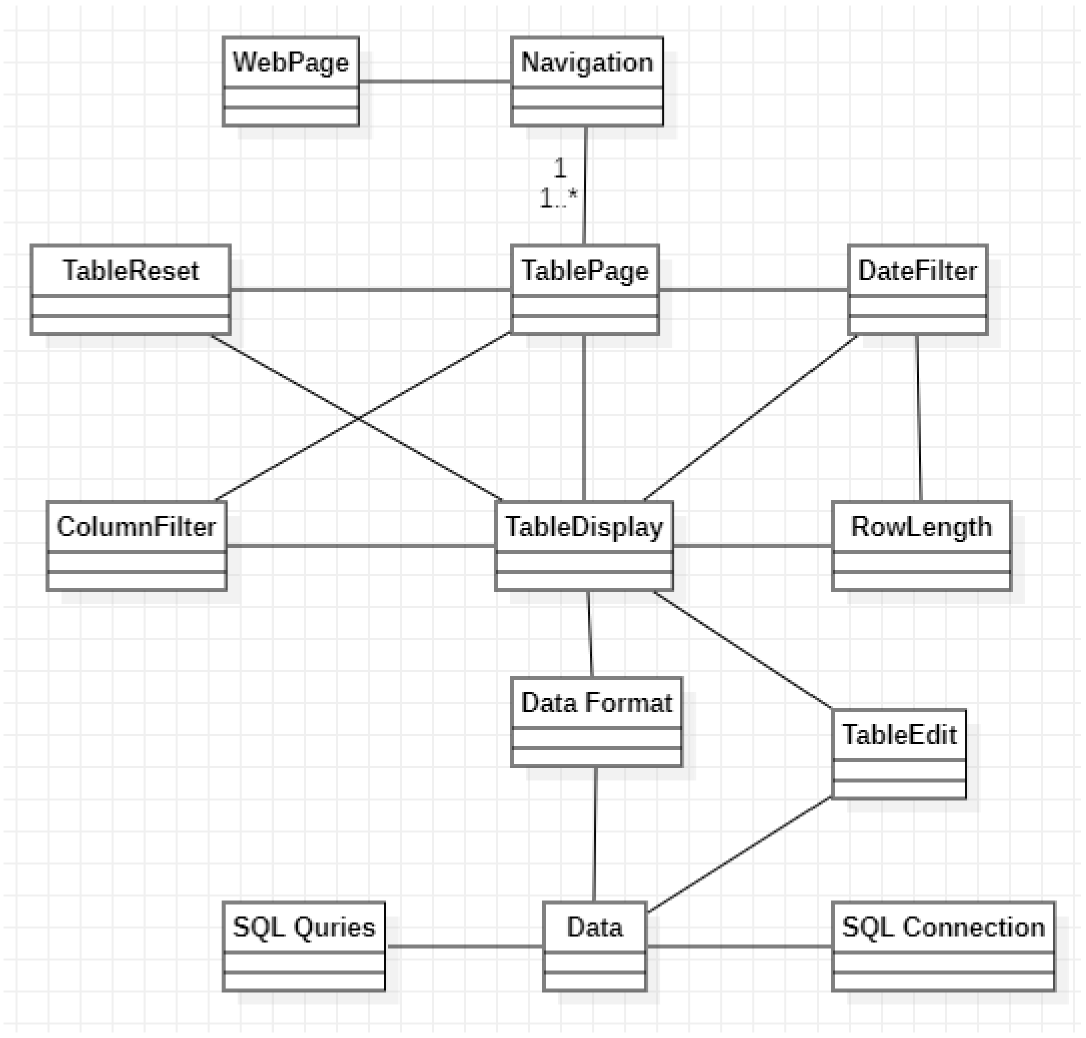
A two tier archtecture would in this case be to simple. The GUI layer would host HTML code for Data table and funtionality surrounding the data displayed, with some C# code being connected to this. The rest of the code would need to be placed in the Business Logic Layer, this includes code handling user input, as well as SQL quering.

A three tier architecture would be ideal in for this application. The GUI pages in the application resides naturally in the UI layer, the code handling user input would be placed in the Business logic, while the quering and management of incoming SQL data would be placed in the Data layer.

A four tier architecture would be unnecessary for this application, the three top layer woul be the same as for a three tier architecture. The OS layer wouldn’t contain anything. As the published application is created as a self contained service being run in IIS on windows.

2.4 Domain Model

When describing the functionality in the application from the requirements, Pseudo classes is created for the domain Model. These are then connected to each other based on the information flow.



Figur 2: Domain Model for the application

3 Construction Phase

From the use cases found in the Elaboration Phase, the SQL Connection is regarded as the most important one. The code from this use case have been collected from appendix. Using something that already exits with multiple people having tested it out is better and more robust than creating one from scratch. The use case Display Data will instead be analysed, as this is built from scratch, and goes into the UI layer, Business Layer, as well as the Data layer of the architecture.

3.1.1 Fully Dressed Use Case Document

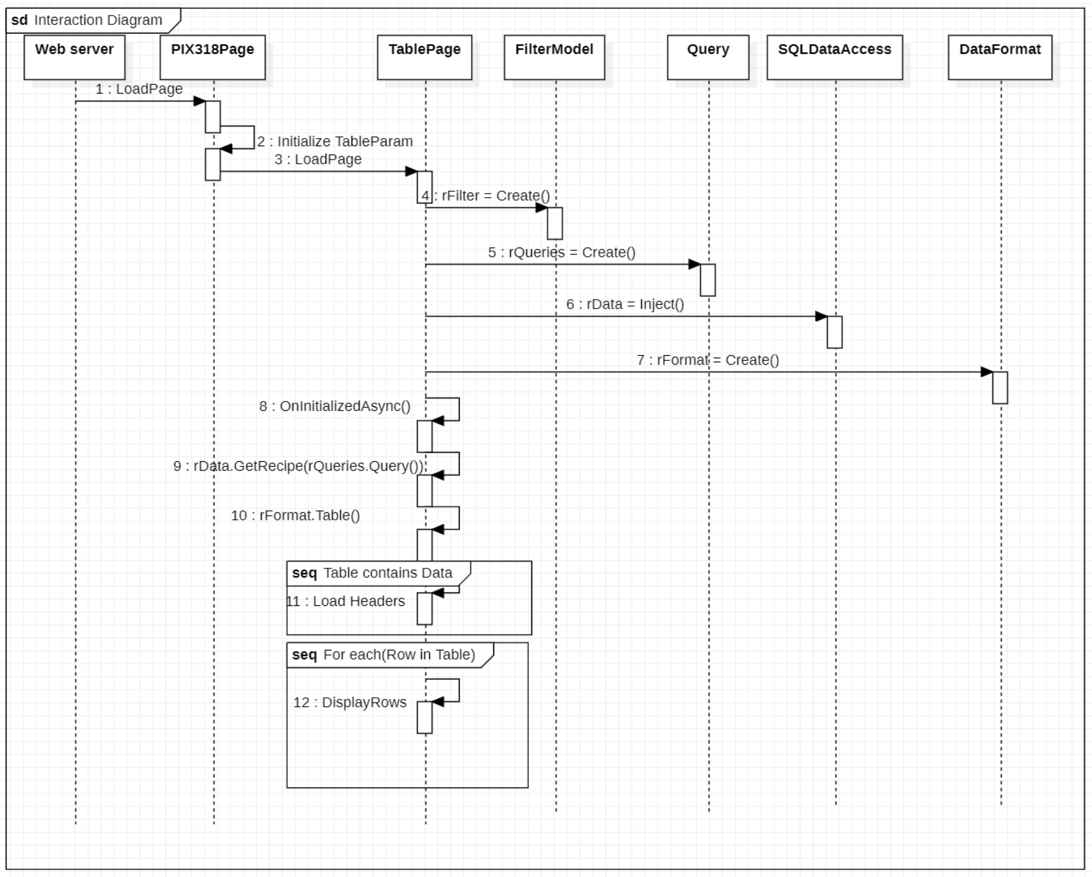
Display Data	
Scope	Recipe data presentation
Level	User Goal
Primary Actor	Operators
Stakeholders and Interests	Factory Manager wants an increase in traceability and efficiency, Operators wants a higher degree of automation, Global IT is concerned of the security.
Preconditions	SQL database, Automated data upload to SQL, IT infrastructure correctly set up
Success Guarantee	Data is displayed. Rows are sorted in the right order. Correct naming for Columns. Right data types and decimal numbers for floats.
Main Success Scenarios	1. Initialize filter 2. Set Table Parameters 3. OnInitializedAsync(): Filter columns from config (Config is already fetched from startup service) 4. Load table Component in page a. Initialize Queries b. Initialize TableList c. Initialize DataFormater d. OnInitializedAsync(): TableUpdate()

	i. GetRecipe ii. FormatRecipe e. Load Data in UI i. Places columns based on filter ii. Places rows on table
Extensions	4.ii:Query is wrong, SQL doesn't return valid data 4.i Data is formatted the wrong way
Special Requirements	
Technology List	
Frequency of occurrence	Every time Table page is loaded
Miscellaneous	

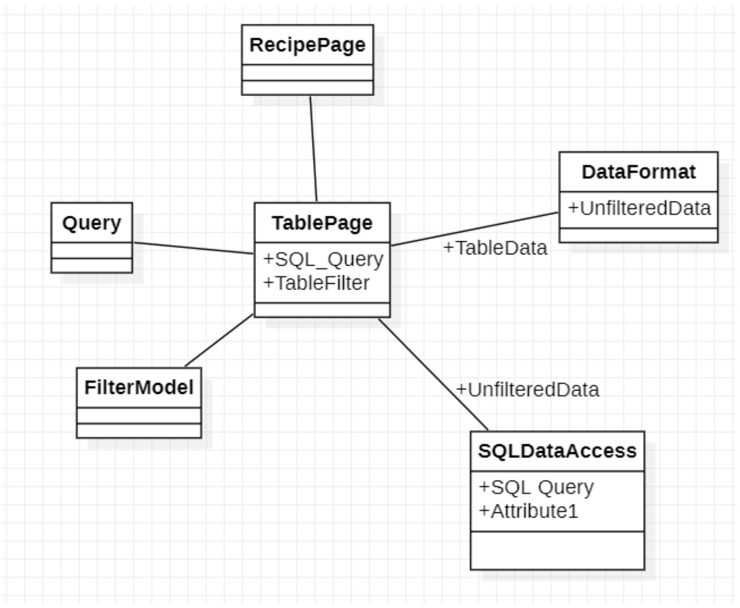
For this use case, there is no need of using Sub and Super classes. The only place where classes are inherited is with he SQL Connection.

3.2 Interaction Diagram, Class Diagram and implementation

Based on the Main Success Scenario of the Fully Dressed Use Case Document, the Interaction Diagram can be made. PIX318Page would be the controller pattern, instasiating the other UI components and classes needed to display data. The Table page is the information expert, this block decides which data that's going to be displayed. The classes in the class diagram is the same as for the Interaction diagram. The Class diagram shows the static information between the classes, and how the structure is made up.



Figur 3: Interaction Diagram for the specific Use Case

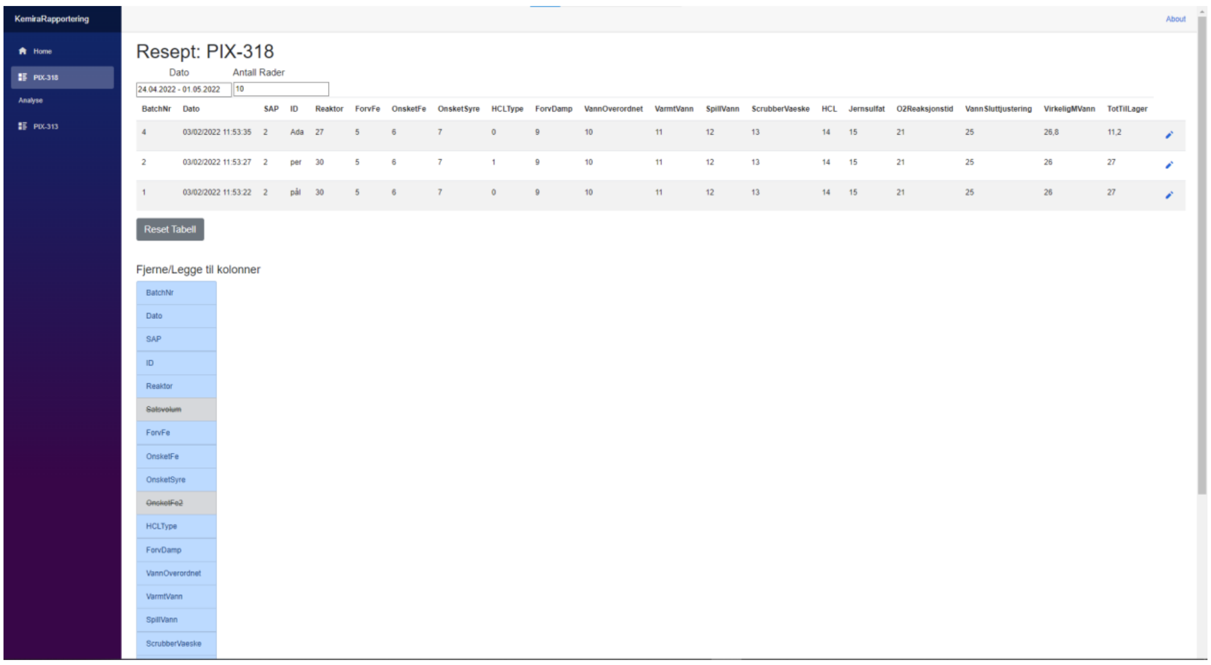


Figur 4: Class Diagram for Use Case

For this use case, there’s either one or very few use instances of the methods. This makes the need for an Object Diagram obsolete. As it would have been almost identical to the Class Diagram.

Based on the Interaction Diagram the use case is implemented. Without being able to connect to the SQL Server and display the data, further development would not have been possible, as the other use cases is based upon the data in the table being shown.

In the first iteration of the construction phase, the most important Use Case was found and elaborated. In this case the SQL Connection was the most critical one, but skipped, as code from the internet was used for this. The Use Case Table Display was chosen. A fully dressed Use Case Document was written, analyzing the Use Case, finding the steps needed for the Use Case to meet its requirements. An Interaction diagram and Class Diagram was drawn from. With the Interaction Diagram and the Class diagram in place, the Use Case was coded and implemented into the application.



Resept: PIX-318

Dato: 24.04.2022 - 01.05.2022
Antall Rader: 10

BatchNr	Dato	SAP	ID	Reaktor	ForvFe	OnsketFe	OnsketSyre	HCLType	ForvDamp	VannOverordnet	VarmtVann	SpillVann	ScrubberVaeske	HCL	Jernsulfat	O2Reaksjonstid	Vann Sluttjustering	VirkeligMVann	TotTilLager
4	03/02/2022 11:53:35	2	Ada	27	5	6	7	0	9	10	11	12	13	14	15	21	25	26,8	11,2
2	03/02/2022 11:53:27	2	per	30	5	6	7	1	9	10	11	12	13	14	15	21	25	26	27
1	03/02/2022 11:53:22	2	pål	30	5	6	7	0	9	10	11	12	13	14	15	21	25	26	27

Fjerne/Legge til kolonner

- BatchNr
- Dato
- SAP
- ID
- Reaktor
- ~~Satsvolum~~
- ForvFe
- OnsketFe
- OnsketSyre
- ~~OnsketFe2~~
- HCLType
- ForvDamp
- VannOverordnet
- VarmtVann
- SpillVann
- ScrubberVaeske

Figur 5: Screenshot of developed application

4 Summary

Having implemented the use case. It worked as it should. While writing this report, it is running 24/7 in a factory displaying recipe data for operators. The application runs on as a Internet Information Service on a Microsoft Server, Specifically a historian server, with the application accessed in an Web Browser from the control room computer. The operators has a button on the SCADA application that opens up the Developed application in a Browser. An issue with this, is that every time an operator want to open up the Developed application a new tab is created. A second issue has to do with certification. While the application runs a secure https connection, it doesn't have a certificate that's trusted by the website in the control room, giving the impression that the site isn't secure.

Based on the requirements from the elaboration a testing procedure can be made. That's useful for implementation in other locations, as well as a confirmation for the customer that it works as intended.

5 References

Microsft. (2022, 05 01). *ASP.NET*. Hentet fra Microsoft: https://docs.microsoft.com/en-us/aspnet/core/blazor/hosting-models?view=aspnetcore-6.0

Skeie, N.-O. (2022). *Lecture notes for object-oriented analysis,*. Porsgrunn: Nils-Olav Skeie.

Appendices

Appendix A: GithHub for code; IsakSkeie/KemiraRapport (github.com)

Appendix B Code for SQL connection with blazor; Intro to Blazor Server Side - Includes SQL Data Access and Best Practices - YouTube