FH OBERÖSTERREICH

○ Hagenberg ○ Linz ○ Steyr ○ **Wels**

Projekt SCADA

Titel:

PROJEKTJOURNAL

Projektmitarbeiter:

Name

Projekt-Leiter:

Name

Inhaltsverzeichnis

1. Einleitung

Dieser Bericht dient als Sammelbericht zur Dokumentation der fortlaufend gemachten Tätigkeiten der Projektgruppe. Im Journalteil muss erkennbar sein, dass der verlangte Arbeitsaufwand in Relation zu Tätigkeiten und Ergebnissen für jede Einzelperson angemessen und nachvollziehbar ist. Granularität: i.d.R. Zeitraum vom letzten Termin bis zum aktuellen.

2. Abkürzungen

JM		Jäger Maximilian
BL		Brajdic Lukas
FL		Feichtmair Lauretta
NH		Naciri Hamza
KS		Kerschbaummayr Stefan
TS		Tollich Sebastian
DK		Dautovic Kenan

3. To-Do-Liste (optional)

3.1 Stand: 23.03.2023

- **Check the requirements before next meeting**

4. Journal

4.1 Sprint 01: Requirements (from 14.03 to 28.03)

4.1.1 Done

(JM, BL): requirements for subsystem database and model/business logic

(HN, KS, FL): requirements for subsystem PC Supervisor App

(DK, TS): requirements for subsystem Controller/Gateway

(BL,FL): requirements for subsystem App

4.1.2 Results

See section 2.3 “Technical Requirements” in Documentation

4.2 Sprint 02: Class Diagrams (from 28.03 to 11.04)

4.2.1 Done

(JM, BL, DK, FL, KS, TS): class diagram draft on OneNote, discussion of the needed classes as well as what functions are needed in the programm

(DK): class diagram on Figma

(FL, KS, HN): UI drafts for Storyboarding

(FL): Storyboarding as Animations in Figma and research on how Figma works and the methods of how the different UI elements can be implemented in Figma.

4.2.2 Results

(DK): link to Figma-Class-Diagram:

https://www.figma.com/team_invite/redeem/HEF2ppERYvliip5GN6mjaA

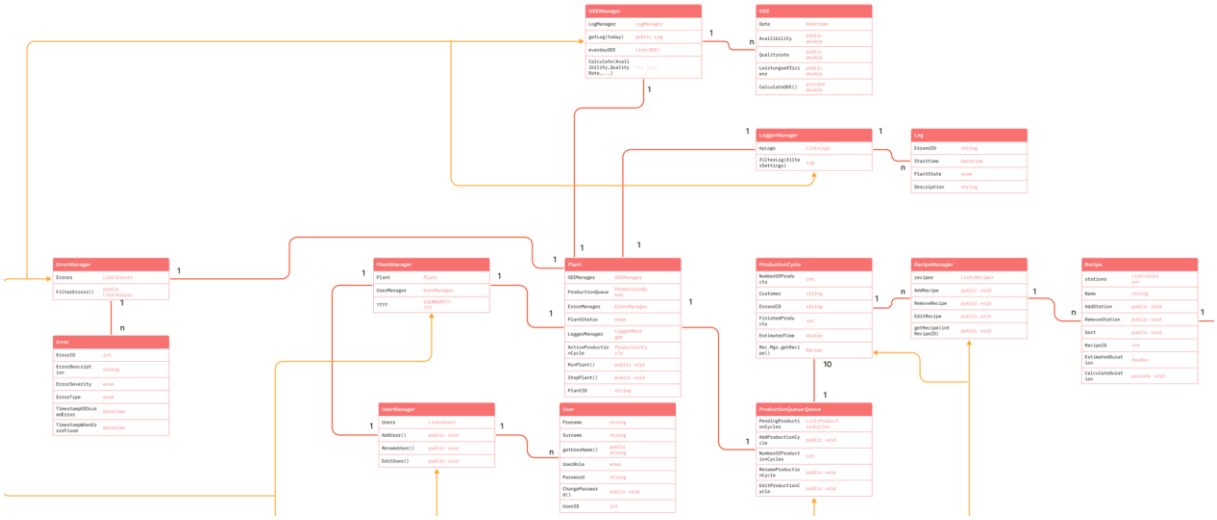


Figure 1: Excerpt of the whole class diagram on Figma

(**FL**): Link to Figma UI-Animations:
The UI has been tested in Figma and checked if the usability of the UI is given. The logic of the UI has been tested in Figma. The UI draft has been checked if all the requirements are implemented in the design proposal.

https://www.figma.com/file/1nfZN6XAy8Gz2jhuo4xYUy/Scada-LoFI-Design-V1?node-id=2-39&t=SE43WY5KcZVQBepi-0

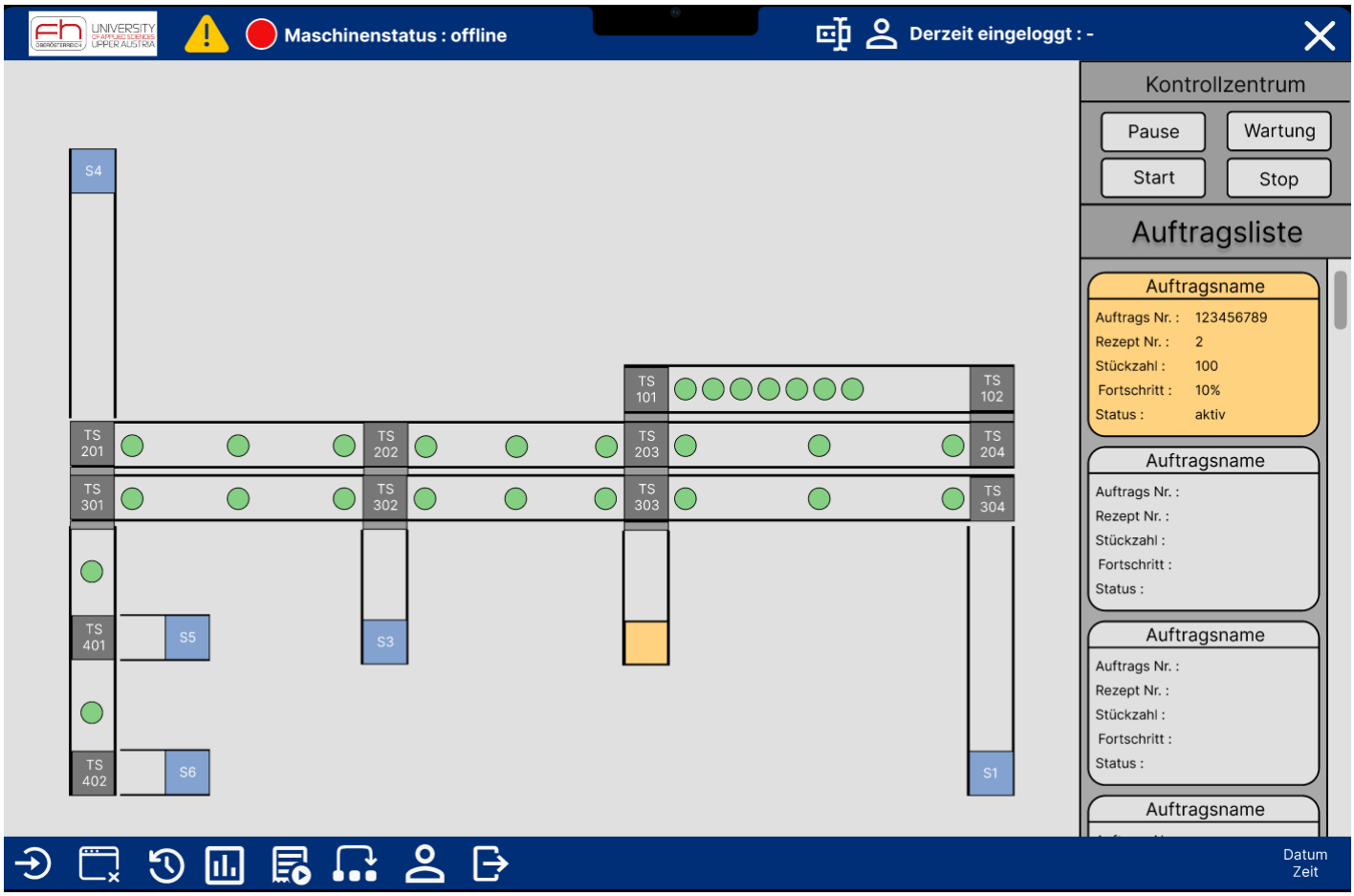


Figure 2: Excerpt of Desktop-UI-Animation on Figma

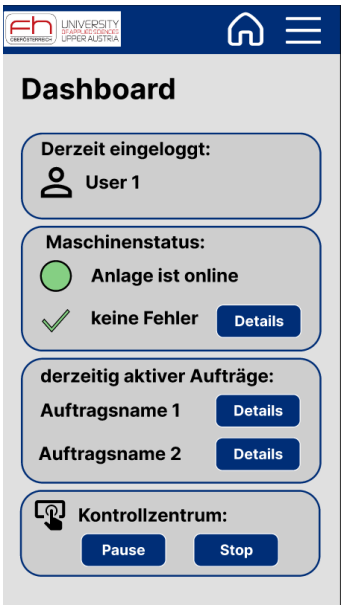


Figure 3: Excerpt of Mobile-UI-Animation on Figma

4.3 Sprint 03: RDM, Repo-Dummy-Classes (from 11.04 to 25.04)

4.3.1 Done

(DK, TS): More detailed introduction from Mr. Schichl to the simulation and OPCUA-Library play around.

(BL): Created a GitHub repository for the project and created the solution with the project files SimpleSCADA_Controller, SimpleSCADA_DesktopViewer and SimpleSCADA_SharedResources, kindly I invited all participants of this project to the repository. Furthermore, did some research for EF-Core how to create a database with EF-Core Code First and how to interact with the database.
Implemented Azure-SQL-Edge Server on Raspberry PI with docker.io and portainer.io, for this was excessive research for how to install a server on a docker on a Linux system necessary.

Docker.io and Portainer.io were installed as root user with the following commands:

- sudo apt-get update
- sudo apt-get upgrade
- apt install docker.io curl -y
- curl -fsSL https://get.docker.com -o get-docker.sh
- sudo sh ./get-docker.sh
- curl -SL https://github.com/docker/compose/releases/download/v2.17.2/docker-compose-linux-x86_64 -o /usr/local/bin/docker-compose
- docker volume create portainer_data
- docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:latest

Then the image of the AzureSQLEdge-Server can get installed on an on container with portainer.io or without a portainer.io as a root user with the following commands:

- sudo docker pull mcr.microsoft.com/azure-sql-edge:latest

- sudo docker run --cap-add SYS_PTRACE -e 'ACCEPT_EULA=1' -e 'MSSQL_SA_PASSWORD=yourStrong(!)Password' -p 1433:1433 --name azuresqledge -d mcr.microsoft.com/azure-sql-edge

(FL): Implementing of Test Project for Web-API, therefore research on how WEB-API's work and how they have to be implemented. (Source: Swagger website) Furthermore the Implementation of CRUD Methods (create, read, update, delete) has been researched and also tried on the User Management and Recipe Management. Also, the possible test methods including possible test software has been researched. Due to the User Management and the different access levels of the users the different methods of adding an access token to the CRUD calls has been researched.

(JM): Added hull for all Model Classes used in Shared Resources. Added all properties to model classes in Shared Ressources and added enums. Added constructors for all model classes except Plant. Configured references between namespaces (Projektverweise, using directives), filled out RecipeManager. Added properties, constructors and methods to all Manager-Classes except PlantManager

4.3.2 Results

(BL) Figure 4 Screenshot of Portainer and cmd where Azure-SQL-Edge Server is running

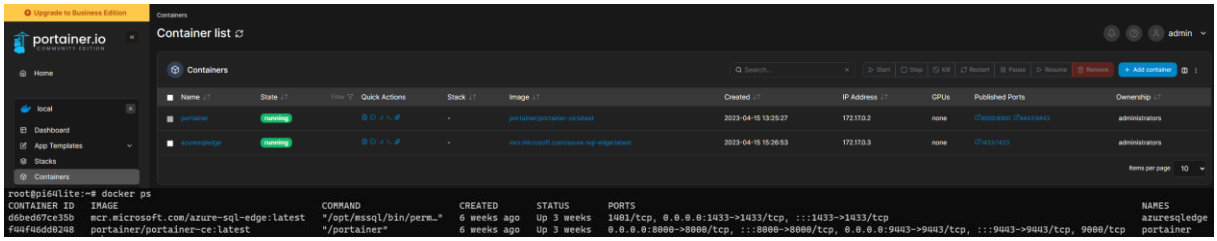


Figure 4 Screenshot of Portainer and cmd where Azure-SQL-Edge Server is running

(JM): All Model Classes and Manager Classes of Shared Resources now have the skeleton of the methods, Properties and constructors like configured in the Class diagram.

(FL): A first try of an API has been implemented and the already implemented CRUD methods have been tested with the Swagger UI.

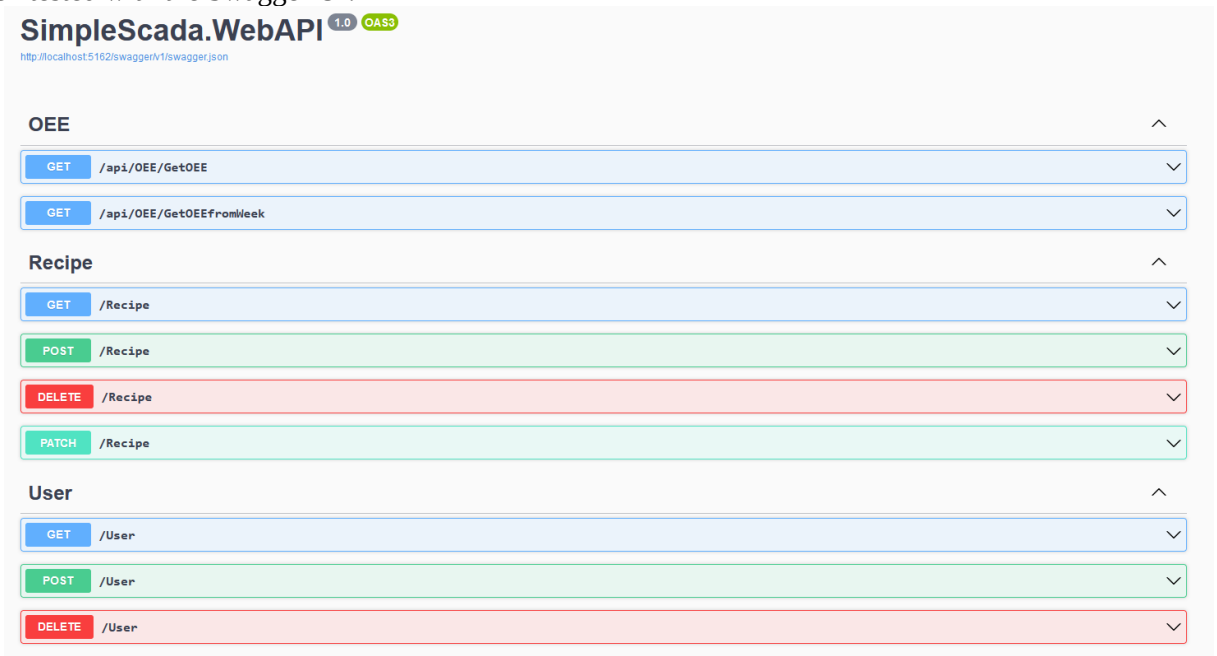


Figure 5 Screenshot of Swagger UI for the implemented CRUD methods

For the User access level the following best practices solutions have been found:

- API Keys (would protect the whole API) : The API key has been implemented and tested. The findings of the implementation are that this type of Authentication is not possible for the User Management.
- JWT: (Json Web Token) is a token which is send with the different requests to the server. An implementation with the JWT would be possible to use at the User Management.

4.3.3 Problems

(BL): Lack of Linux-Skills and due to renewed Software-Architecture the implemented server is unnecessary. Lot of research on the topic EF-Core was necessary.

(JM): Had to rearrange hull of the classe multiple times to fix it to updated view of Class Diagram. ProductionQueue already inherits from "Queue", but also has to inherit from "DbContext" in Future. Could be a problem.

(FL): Lot of research on the topic REST-API due to the lack of knowledge.

4.4 Sprint 04: Class Diagram, Implementing of UI Design (from 25.04 to 09.05)

4.4.1 Done

(all): Discussion of architecture of the different parts of the program. Checking the class diagram which has been prepared in Sprint 02 with Mr. Slabihoud.
Furthermore, the idea of using an API for communication has been discharged.

(DK): 90% of paths coded and tested, draft for continuous development (transfer, machine status).

(FL): Implementing of the draft version of the presenter for each window on the UI, and the empty windows for the UI. Start of Adding models and views to de different presenters according to MVP pattern. Adding design of Recipe Window from Figma into Soulution. Implementing of a Button with Round Edges. Furthermore starting the implementation of functions to open the Sub windows from the Main window as well as function and events the Main Window Presenter and Recpie Manager Presenter. Adding an Startup Method for the UI.

(BL): Created first test database for RecipeManager, after long trying, to see if the research was right and tried to interact with the database, interacting with database was only limited possible.

(JM): Updated gitignore to outcomment *.db once db is created. Added entity framework nuget packages to Desktop Viewer. Added a test-dbset in Controller project as a kind of playground. Powershell testing with appling migrations to the test.dbset.

(NH): Adding design of LogView, PasswordChangeView, JobProcessingView and UserView Window from Figma into Solution.

(KS): Implementing and designing of MainView from Figma into Solution. Visual functionality such as dropdown menu. Overall adjusting the custom UI for user friendly environments.

4.4.2 Results

(FL): A draft version for the different presenters has been created and where known already the model and the view have been added to the presenter according to the MVP pattern. Furthermore, an architecture for the presenters has been implemented. (One main presenter which has all the other presenters as a private member) The architecture of the UI program is outlined in the Figure xx below.

Skizze von der Programmachitecture auf UI einfügen

Implementing of Design for Recipe View according to UI Design in Figma.

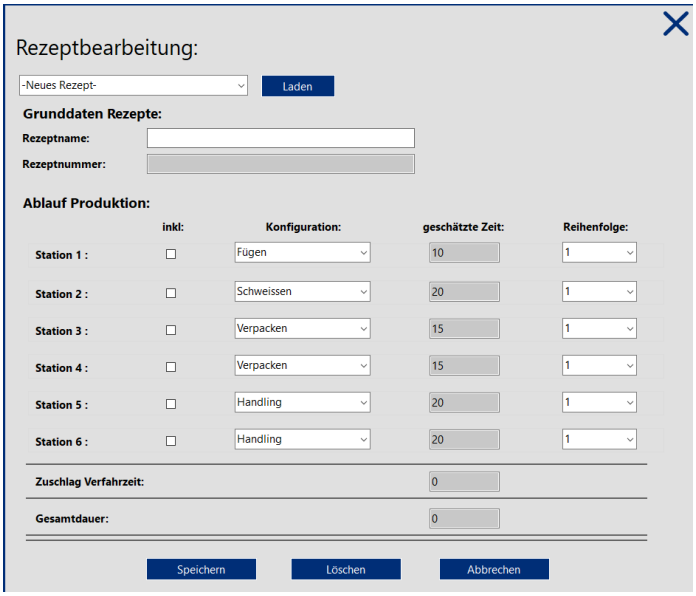
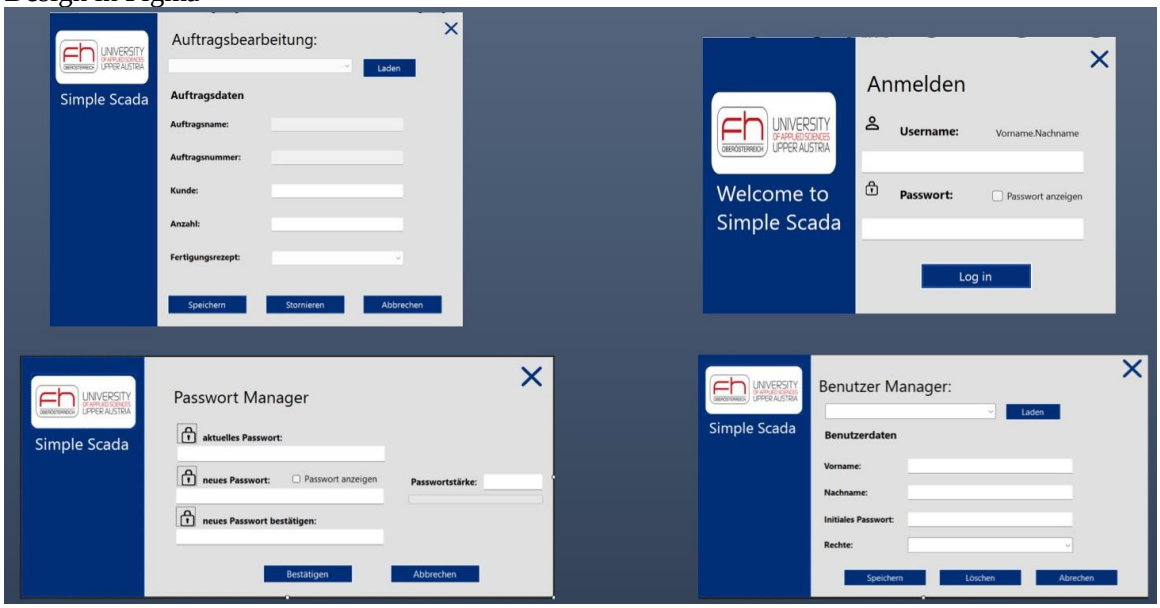


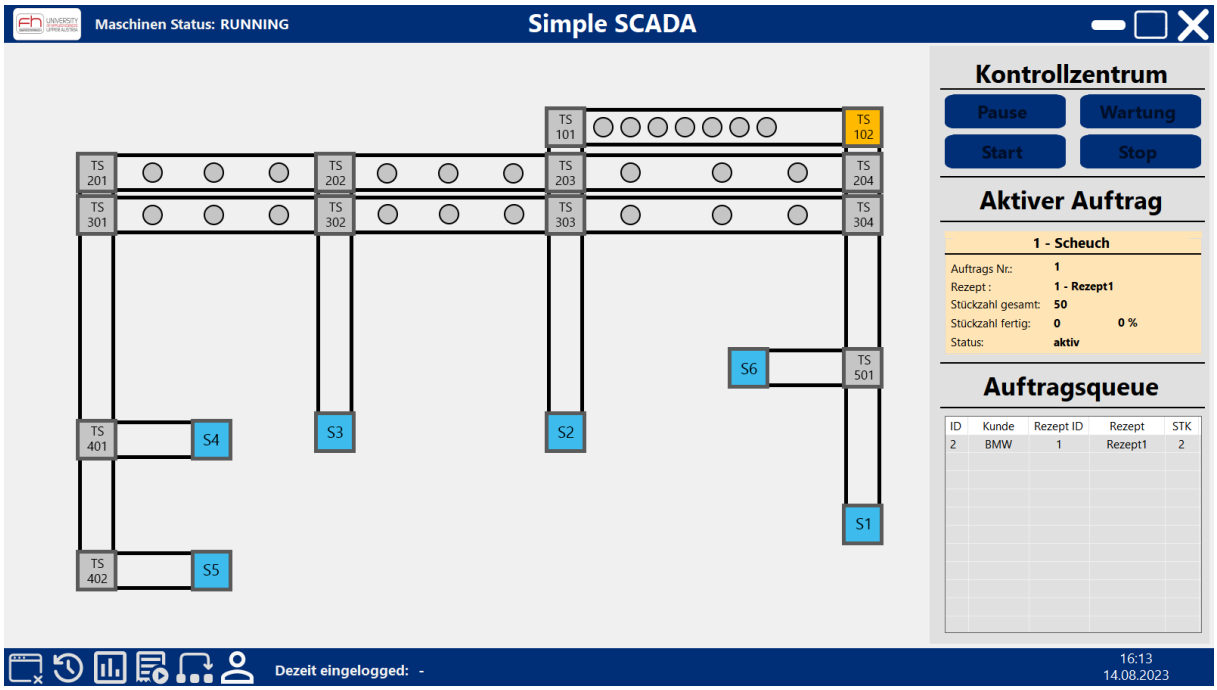
Figure 6 Screenshot of Recipe Manager Window

(JM): New gitignore that ignores all .db files.

(NH): Implementing of Design for LogV, PasswordChange, JobProcessing and User according to UI Design in Figma



(KS): Implementing of Design for MainView and User according to UI Design in Figma.



4.4.3 Problems

(all): Because of the variation of the knowledge level of the developers the idea of using an API for communication has been discarded. Maybe the API will be implemented in a later step after finishing the UI.

(DK): Simulation very buggy. Stucks randomly at some certain places.

(FL): The implementing of a Button with Round Edges for the Views wasn´t very successful because the design of the Button with Windows- Forms doesn´t look good (the border stile as well as the rounded edges are not looking visually appealing) Therefore the buttons and textboxes will be implemented with sharp edges.

(BL): Creating the database with EF-Core was in the beginning a little bit tricky, because it was very important to learn how to define the dbcontext and ids, for creating a proper migration and then apply the migration to a database.

(JM): Problem with ignoring .db files. To execute a migration on a database the database has to be in the repo. So we had to think about a change of plans for the gitignore for the development-phase.

(KS): Problems with Windows Forms to make it look good.

4.5 Sprint 05: Rework UI Design, Dummy Presenter (from 09.05 to 23.05)

4.5.1 Done

(DK): Showing Mr.Schichl some issues with the simulation and suggestions for improvement.

(FL): After merge of someone code has gone missing. Therefore the commit which has been has deleted code has been searched in the commit history and stashed from the Commit History. Adding draft code with NotImplementedExceptions to the User Manager Presenter ,Job Processsing Presenter OEE Presenter and Log In Window Presenter. Edit of the naming of the different UI elements on the Job Processing View, because the names haven´t been meaningful. (Names like button4 have been used)

(BL): The database for RecipeManager was improved by the right connections with the other classes (Station, Utility) 1 to many and many to many.

(JM): Updated gitignore to ignore all .txt files in /Databases directory (so that everybody can have their own absolute path to their own local database). Added entity framework nuget packages to Desktop Viewer.

(KS): Finalization of the design MainView. Working on optimizing the design in general of the pc application.

4.5.2 Results

(FL): Elements on the UI have meaningful names so that the coding can be faster and more efficient. Draft Implementation of Methods in different Presenters so that the functions can be implemented faster. The deleted code due to merge conflicts has been restored.

(JM): updated gitignore that only ignores the one part of the absolute path to each developer's database. Now migrations can be executed on each database so that it can be updated (new tables etc.)

4.5.3 Problems

(FL): Code has been deleted due to merge problems from somebody. The right commit had to be found in the commit history. The search has been very time-consuming. Due to the missing DB the programming of the functions hasn´t started yet.

(JM): Path has to be relative when applying new migrations in the Powershell, but absolute when running the program => switch had to be implemented but what implemented poorly in the beginning.

4.6 Sprint 06: User Managment (from 23.05 to 06.06)

4.6.1 Done

(DK): Finished and tested all Transitions and Paths. Started first drafts for Error-Handling.

(FL): Research to WebView.(for Production Queue on Main Window) Rework of the already existing GUI windows. Bug fixing of errors due to renaming from from collegaues. Adding method for changing the View according to the access right of the User. Generally the User Managment on the UI was programmed in this sprint. (The UI windows Password Change, Log In and User Manager have gotten the needed functionality

(BL): Created acceptable Database for RecipeManager which is able to perform all necessary transactions with the database except one problem. Implemented all necessary methods, at this moment, for RecipeManager, ErrorManager and UserManager. Also, Error handling was implemented in these Methods with try catch and adapted return values if needed. For this the method AddError in ErrorManager was changed to a static method.

(JM): Tested DbSet in Controller project. Added one main global dbcontext for temporary Db Creation via 1 migration context. Updated repo to contain seperate databases for each user for testing purposes (sqlite).
Created Database with all DbSets for now and splitted it to database for each developer. Updated Databases for all developers with new migrations to fit requirements for ChosenUtility (is a JunctionTable RecipeStation with a new Property "ChosenUtility" to save which Utility of the Station was chosen for the Recipe). Created Query to get all Recipes with all Stations (.Include) and all Utilites of each Station (.ThenInclude).

(NH): Creating the OEE interface and linking it with the corresponding button, along with implementing a functional close action to have the initial UI interactions.

(KS): Designing the Plant according to the latest schematic in Photoshop and implementing it into the MainView

4.6.2 Results

(DK): Recipes run solid 90% of the time.

(FL): The UI was reworked so that the buttons are the same size on each UI window. Furthermore the Text sizes and text format has been corrected so that the they are the same on each window. A general still overview can be seen in in the figure below.

Figure 7 Screenshot of Job Processing View

The naming of the UI elements has been checked on each window so that every component has a meaningful name.

The User Management has been fully programmed. The following functions have been added to the program:

- UI changes based on the logged in User and also show the Username of the Logged in User
- User is able to Log In with the Log in Window. The password is hidden with dots as char, but can be made visible with checkbox.
- The user is able to change his own password in the Password Change Window.
- The Super User is able to open the create, edit and delete users in the User Manager Window.

(JM): All conn-strings reference to AbsolutePath.PathGeneration() now, the developer only has to change one file to change path to all DbSets. Database for each Developer with new JunctionTable and newly filled up dummy data. query with all correct Recipe entries linked with all entries from the

junction tables. CreationDbContext with all possible table and relationships to add migrations to all databases and all tables.

```
public class DbCreationContext : DbContext
{
    #region Members
    0 Verweise
    private DbSet<Recipe>? Recipes { get; set; }
    0 Verweise
    private DbSet<Station>? Stations { get; set; }
    0 Verweise
    private DbSet<Utility>? Utilities { get; set; }
    0 Verweise
    private DbSet<Log>? Logs { get; set; }
```

Figure 5: The Cration Context holds all Tables

```
modelBuilder.Entity<Recipe>().HasKey(b => b.Id);
modelBuilder.Entity<Station>().HasKey(b => b.Id);
modelBuilder.Entity<Utility>().HasKey(b => b.Id);
```

Figure 6: The Creation Context also holds the Primary Key Reference of each Table

```
modelBuilder.Entity<ProductionCycle>()
    .HasOne(e => e.Recipe)
    .WithMany(e => e.ProductionCycles)
    .HasForeignKey(e => e.RecipeId);
```

Figure 7: The Creation Context also holds all relationships between the Tables

(NH): By the end of this sprint, I successfully established the OEE interface, enabling users to access it through the designated button. Additionally, the close action was implemented to have a common user experience along all the different Interfaces of our tool.

4.6.3 Problems

(DK): Simulation very buggy. Stucks randomly at some certain places.

(FL): Due to some changes in an other part of the program (renaming and transferring the classes to another place) there have been some bugs which needed to be fixed before adding more functions to the program. The method change user and change password doesn´t save the changes into the DB.

(BL): By fetching the recipes from the database, the stations and utilities weren´t appended to the recipe even though the relationship was right in database, after some hours of research the problem was unsolved, but MJ had solved the problem in fifteen minutes.

(JM): Relative paths not possible with sqlite, had to program a workaround for absolute-path-finding, that only works on Windows Machines but as Windows-Forms also only runs on Windows Machines, it's fine. Could not create Db from Multiple Contexts, one "parent"-Context only for Db-Creation was needed.

Without .Include and .ThenInclude, only Recipes without the entries from the Junction Tables were returned, had to get all the values again with new queries at first without knowing ".ThenInclude".

All up-to-this-point saved data in the databases of each developer was lost with applying a new migration, had to run a short script to fill up all databases with the dummy data again => always be prepared for needing a backup

4.7 Sprint 07: Recipe Managment (from 06.06 to 13.06)

4.7.1 Done

(DK): First tries of making everything work together so the simulation does something when a trigger from the user comes. Research and drafts on making some processes asynchronously. Providing functions and classes for the mentioned approaches.

(FL): Adding method so that the User Manager Window and recipe Manager Window is cleared when closed or after specific events. Adding functions to the Recipe Manager. Testing and Reworking of some Errors in the User Management.

(BL): Fixed the methods EditUser in Usermanager and EditRecipe in RecipeManager, because the changes weren´t saved in the database. Implemented the methods in LogManger with Error handling. Changed the names in the Enums ErrorType and ErrorSeverity to useful names and began to implement to save the plantstates in plant but due to lack of time this was finished by JM.

(JM): Created Methods for ErrorManager, came to a consens what properties the PlantManager should hold and added Methods to PlantManager. Created Methods for LogManager, ProductionCycleManager.
Added PlantStatus Class for the OPC UA to save the values of the palette there and for the UI to receive the value where the palette stands right now.
Built ErrorView and ProductionHistoryView (to view both Logs and FinishedProductionCycles) with the specific update-methods in the ErrorManger and LogManager.

(NH): Following the Model-View-Presenter (MVP) pattern to the project posed its own challenges as I aimed for cleaner code architecture and separation of concerns. Furthermore, linking the presenter with OEEManager functions to retrieve necessary data for OEE calculations.

4.7.2 Results

(DK): Nothings really working, that means more research and brainstorming ahead.

(FL): The following changes have been made in the User Manager Window:

- The window is cleared when closed an on specific events
- In the presenter some code has been fixed an is now working correctly (issue has been at the method edit User and change Password) The changes are now saved in the DB.

The following methods have been added to the Recipe window:

- All already existing recipes are shown in the drop-down menu.
- Method of loading utilities for each station (first the method didn´t work correctly -> after some rework the method is working)

(JM): Methods to manipulate Db in Error and Plant DbSets**.** Finished Windows, but for now only with "Refresh" Button, not asynchron.

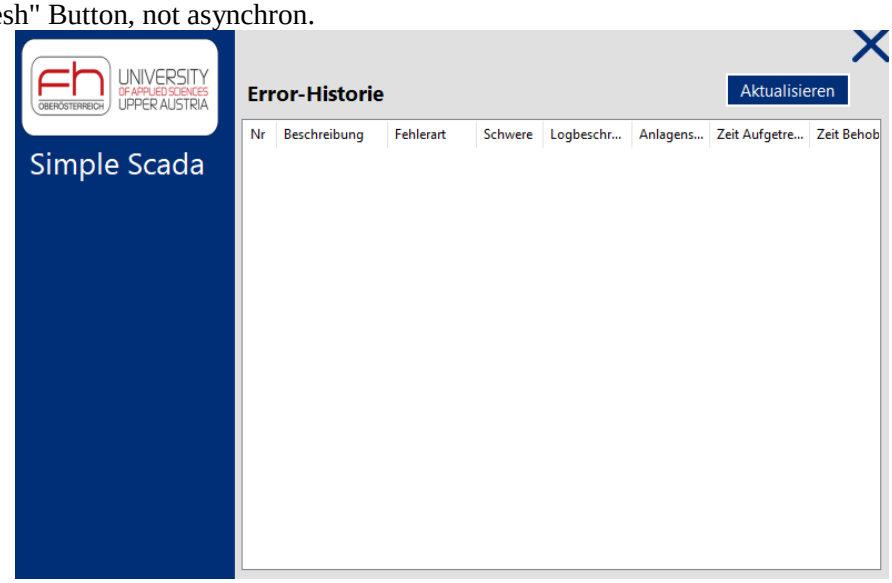


Figure 8: Screenshot Error-History

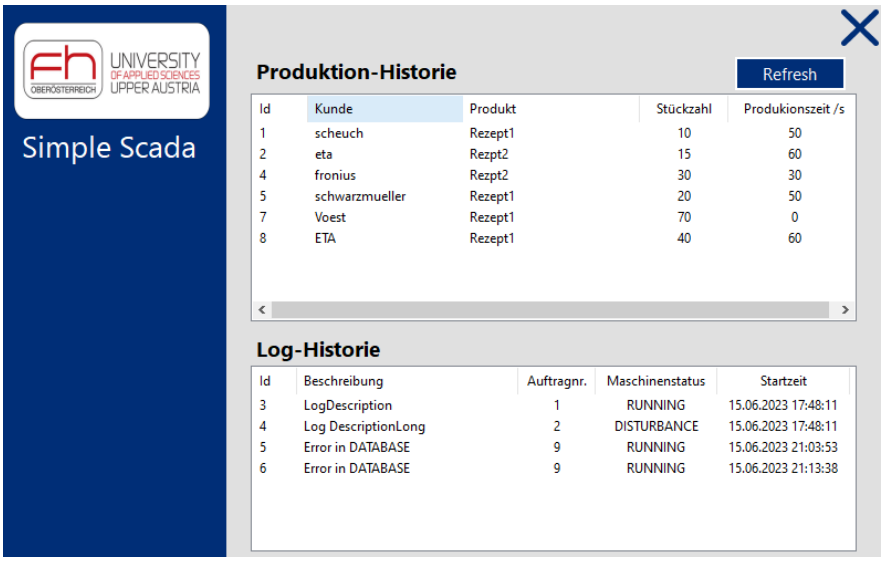


Figure 9: Screenshot Production-History

(NH): At the conclusion of this sprint, I had integrated the MVP pattern effectively, allowing for clearer code organization and maintainability. The OEE Presenter was successfully handling UI actions, contributing to a more streamlined user interface. The connection between the presenter and OEEManager facilitated the retrieval of data required for subsequent OEE calculations.

4.7.3 Problems

(FL): Further the Recipe window isn´t finished yet. Needs to be finished in the next sprint.

(JM): Implementation in OPC-UA, to provide the right methods for Team OPC-UA, didn't document the code right away which led to confusion later. Merges, had to give up inheritence of ProductionQueue from "Queue".

Many Nullable Objects bc of EntityFramework ForeignKey sometimes not required, last commit contained a critical error that i missed, program wasn't executable when pushed to repo => maybe run application a last time before pushing a large commit.

4.8 Sprint 08: Recipe Management, Cyclic query from UI to DB (from 13.06 to 20.06)

4.8.1 Done

(DK): Implemented an asychronnous approach to check for state changes in DBA and then executing it in the Simulation. Each defined state provides specific execution and works.

(FL): The recipe manager has been finished. The cyclic query of the active production cycle, production queue, machine state and sensor sate has been added with backgroundworkers.

(BL): Created the fully functionality of the JobProcessingView inclusive the JobProcessing Manager. This means to load, save and cancel a production cycle that wasn´t started yet. Also, created a View with Chuck Norris chokes that get loaded randomly from an API. The final design was made by FL.

(JM): Changed GetPlant() to update everytime even if it is called multiple times from the same PlantManager-Instance. Fixed Bugs of PlantManager to make the communication between GUI and OPC UA more resistent to errors.

Created Central SQLDatabase with Database server with a Student Account in the Azure Cloud, executed migrations on the Online-Database and filled it with the dummy data; in a seperate Branch the program now connects to the Online-Production-Database instead of the local Development-Databases.

(NH): The implementation of the OEE logic, involving calculations based on quality rate, availability, and efficiency, presented complexities. Developing various charts, including representation of the last 7 days and a latest OEE, posed challenges in terms of data retrieval, sorting, and synchronization, Additionally, the selection of a suitable chart library proved to be a challenge, given the variety of options available, and the evaluation of each library demanded a substantial investment of time.

(KS): Finalizing the custom UI across all Views. Designing and implementing of sensors and transfer stations.

4.8.2 Results

(FL): Plant does something when triggered state in DBA.
(FL): Finishing the functions of Recipe Manager:

- Recipes can be created, edited and deleted
- The time which is needed for the production is calculated based on the included stations

Backgroundworkers have been added for the cyclic query for the following functions:

- Backgroundworker active production cycle: Get the active production cycke from the DB and sets in on the UI. Gets the machine state from DB and sets it on the UI.
- Backgrundworker Production Queue: Gets the current production queue and sets it on the UI
- Backgroundworker Sensor Data: Gets the sensor data from the DB and sets it on the UI.

In the figure an overview of the Main Window with the cyclic data is shown.

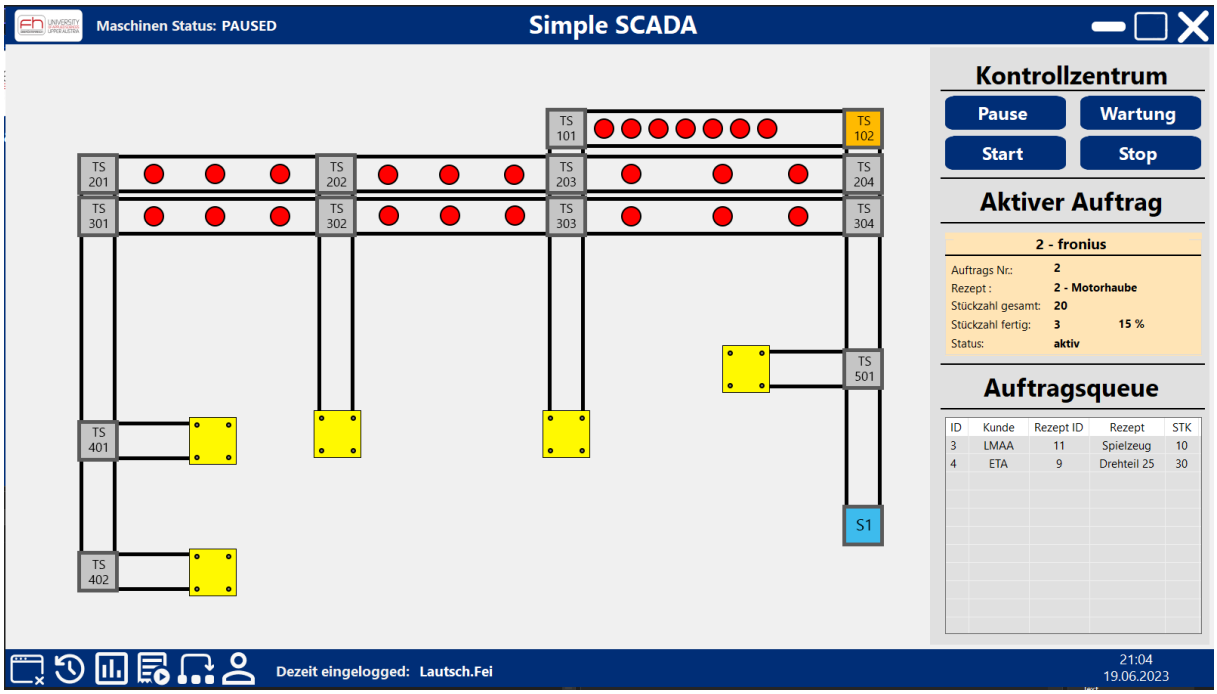


Figure 8 Screenshot Main View with cyclic data

(BL): Able to create, save and cancel production cycles in the JobProcessingView, see Figure 10.

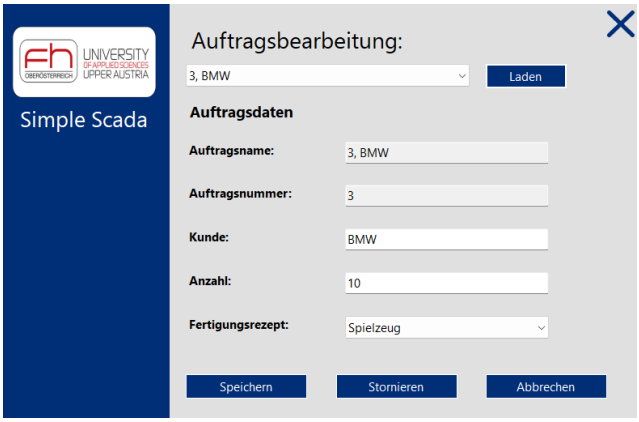


Figure 10 Screenshot JobProcessingView

Able to get Chuck Norris Chokes from an API in the JokeView, see Figure 11.

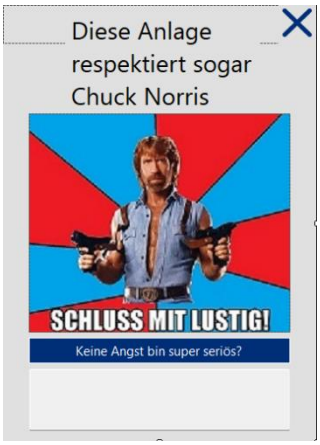


Figure 11 Screenshot JokeView

(JM): Always getting current data from db even if calling from the same PlantManager Instance multiple times. UpdatePlant is more solid and doesn't crash half the time now. Interaction is now with the Online-Production-Database in the Azure Cloud.

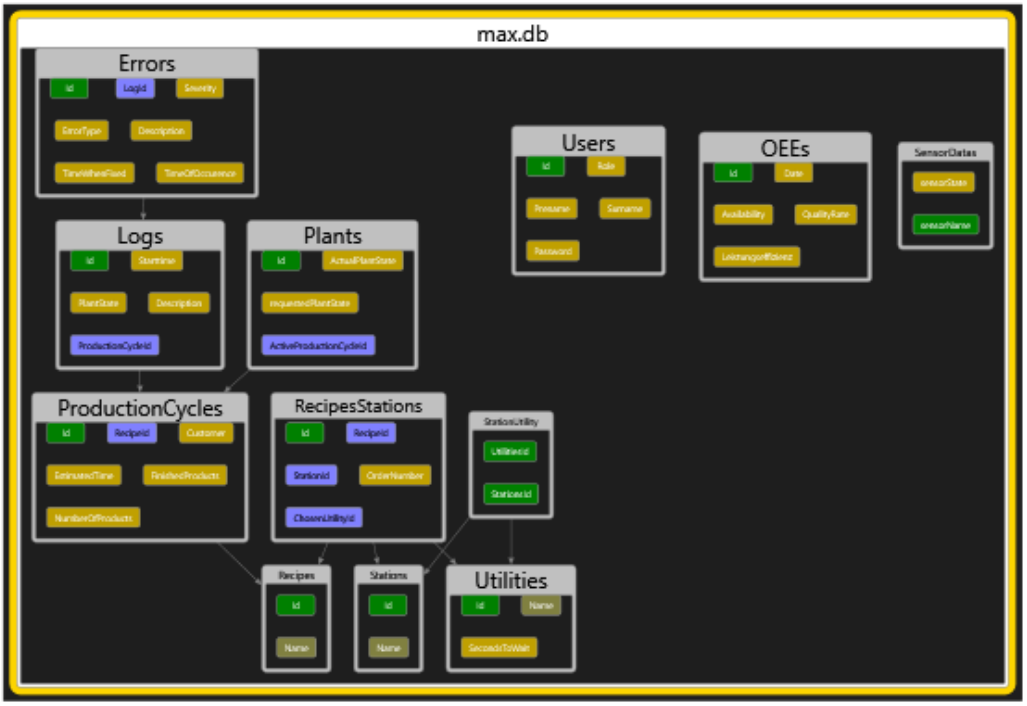


Figure 12: Final Database Table-Architecture

Name	Typ
simplescada	SQL-Datenbank
simplescadaserver	Computer mit SQL Server
simplescada	Ressourcengruppe

Figure 13: Database Server in Azure

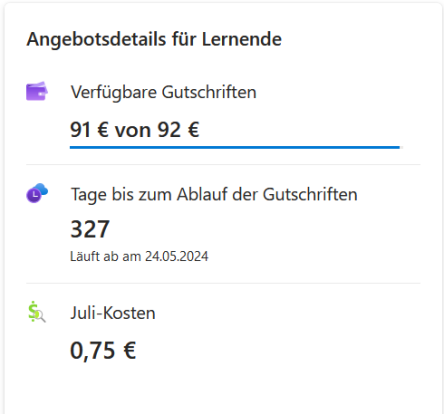


Figure 14: Database Hosting Costs

(NH): By the end of this sprint, I had overcome the hurdles surrounding OEE logic, resulting in accurate calculations based on quality, availability, and efficiency information. I successfully incorporated different charts into the interface, using dashboard library Syncfusion. These charts were effectively linked with corresponding data, providing users with insightful visualizations.

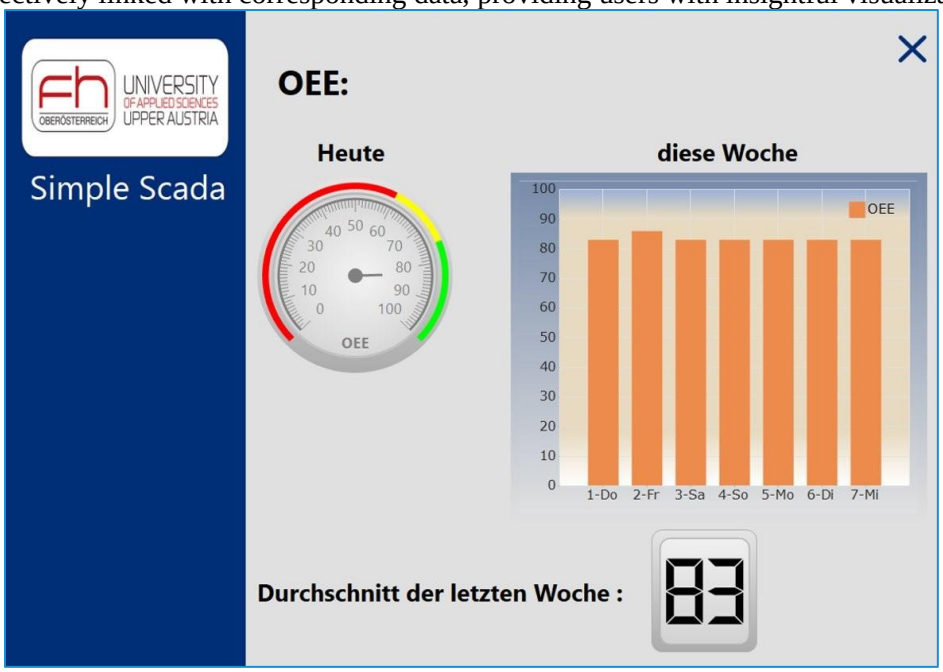


Figure 15: OEE-View

4.8.3 Problems

(DK): Most problems accrue because of simulation bugs that stayed unhandled to this point.

(FL): Problems at finding the right method on how to get the data cyclic from the DB. Solution has been found.

(JM): The OPC-UA and View did not get live updates when the other side changed something in the Db.

Put false comments above the methods in Class PlantManger, led to a lot of confusion when editing the PlantManager later => better comments above the methods ,bug where the CurrentProductionCycle wasn't updated in the GUI when changed by the OPC UA, had to build a static Method that gets called by a normal Method because I couldn't change it at that point because both GUI and OPC UA were already using the normal Method.

with multiple Simulations of the Production-Machine running locally the DbSet PlantStatus gets overwritten online all the time. This changes in Production of course because there only is one Plant running all the time.

4.9 Sprint 09: Presentation and Documentation of the Project (from 20.06 to 15.08)

4.9.1 Done

(ALL): The final presentation of the project has been prepared and presented on July 4th 2023.

(FL): Documentation of the Main View, Recipe View, User Management and Joke View has been made.

4.9.2 Results

(FL): Documentation has been partly finished (Documentation from the parts which have been programmed by me is complete.)

4.9.3 Problems

5. Recherchen (fortlaufend ergänzt)

5.1 Paper

5.2 Patente

6. Literaturverzeichnis

Kühnel, A. (2022). *c# 8 mit Visua Studio 2019*. Bonn: Rheinwerk.
Microsoft c sharp doucumentation. (2023). Von microsoft learn: https://swagger.io/ abgerufen
Swagger documentation. (2023). Von Swagger: https://swagger.io/ abgerufen