Hochschule Esslingen
University of Applied Sciences

DREAMHOME

Informationssysteme Projekt

im Studiengang
Wirtschaftsinformatik
vorgelegt von

Ranna Alemi
Matr.-Nr.: 760028

Tim Brecht
Matr. -Nr.: 759633

Anna Dobler
Matr.-Nr.: 760235

Sven Schuler
Matr.-Nr.: 759707

Timo Schwind
Matr. -Nr: 759823

am 30. Juni 2021
an der Hochschule Esslingen

Prüfer: Prof. Dr. Rer. Pol. Dirk Hesse

Table of contents

List of figures

List of tables

1 Group Structure

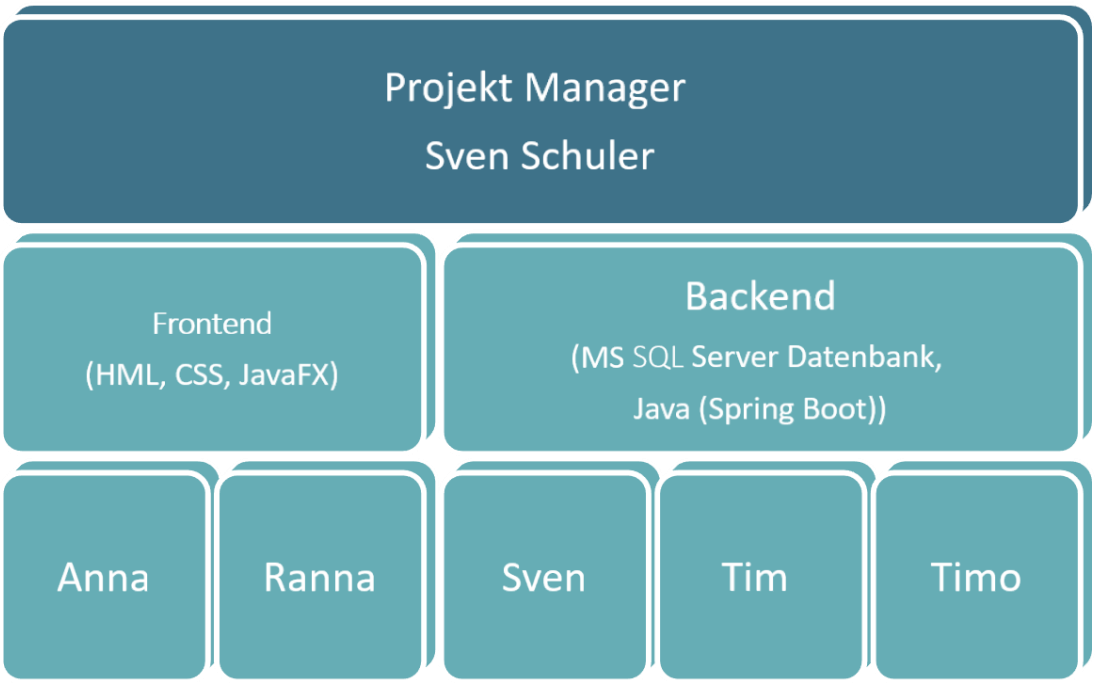


Figure 1: group structure

2 Mission

2.1 Mission statement

The property management/E-Ship Information System is used by employee of Dream House to manage the daily business regarding monitoring and handling incoming payment in relation to the managed properties.

The main focus will be on the automatic processing and validation of the data.

2.2 Mission objectives

- Display paid rents and open positions.
- Upload, decomposition and transformation of account detail.
- Calculate operating costs for each client.
- Partitioning of the operating costs for the clients and display them via PDF or CSV.

3 Time schedule

To coordinate the tasks in the team, we use the online project management tool “agantty”. This is a free online tool to organize tasks in a gantt chart. For each task it is possible to enter the responsible persons, a deadline and you can declare tasks as done. So every team member knows, which task he or she still has to do and sees the work of the others.

3.1 Milestone 1

Deadline: 05.05.2021

Goals

- documentation of group structure
- mission statement & mission objectives
- time schedule
- user views (use cases)
- list of requirements (structured, sorted, numbered)

Gantt-Chart

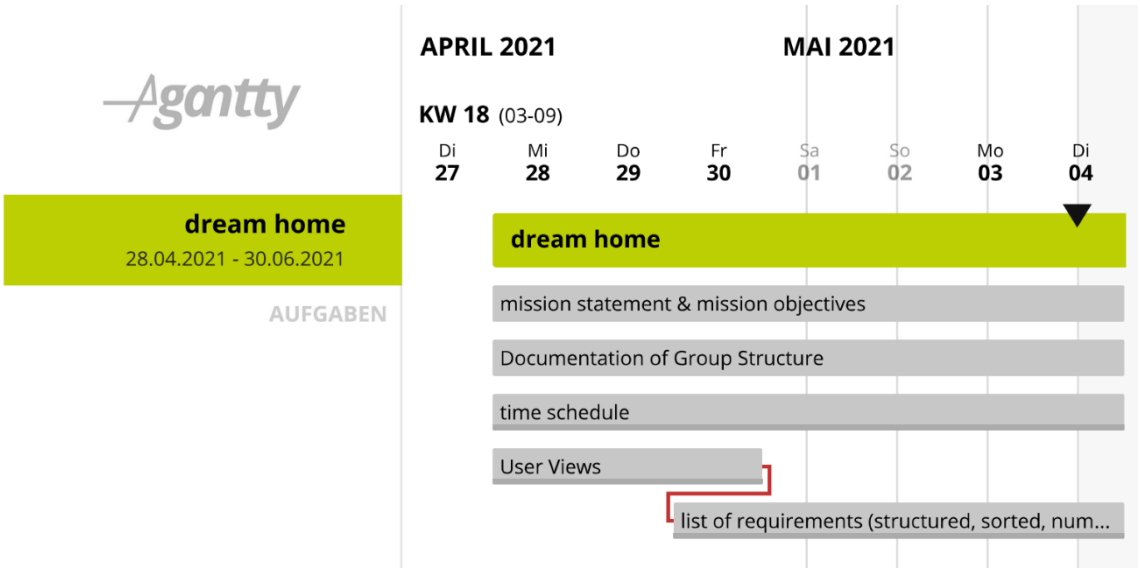


Figure 2: gantt-chart milestone 1

3.2 Milestone 2

Deadline: 19.05.2021

Goals

- conceptual data model on paper (Chen notation)
- data catalogue
- list of all relevant entities and relationships for each user view

Gantt-Chart

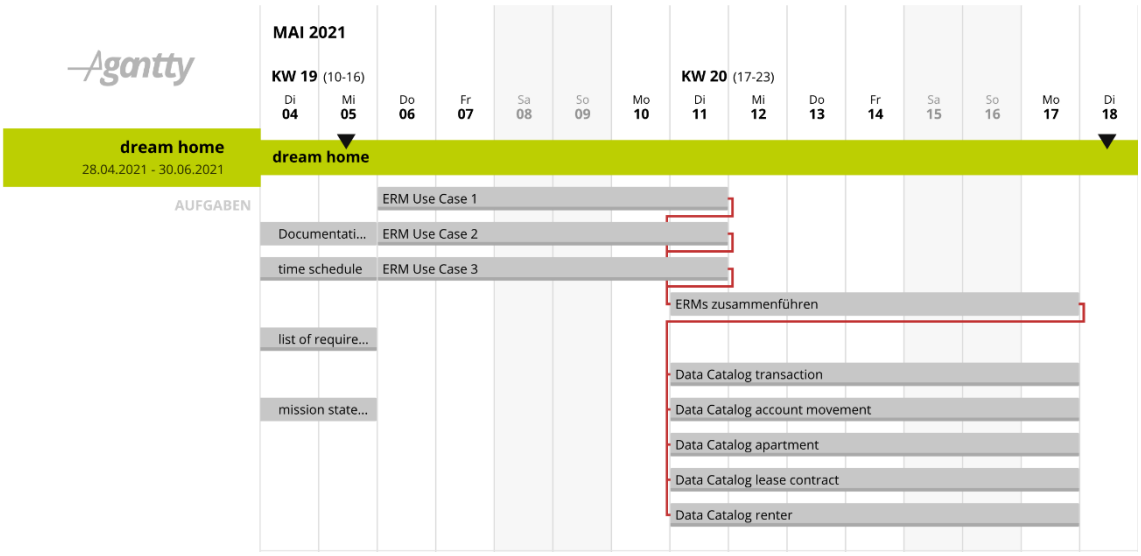


Figure 3: gantt-chart milestone 2

3.3 Milestone 3

Deadline: 02.06.2021

Goals

- conceptual data model in CASE-tool
- logical data model
- proof that all relations are in BCNF or explanation why not

Gantt-Chart

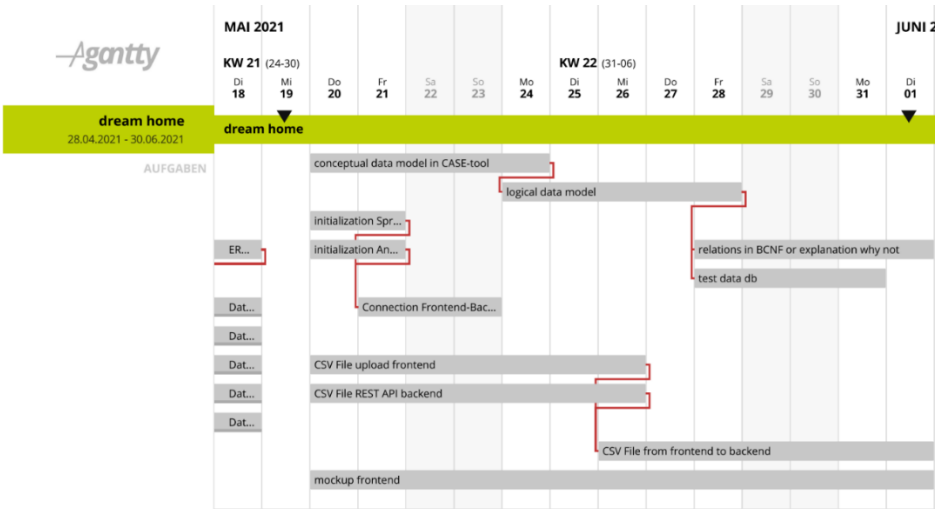


Figure 4: gantt-chart milestone 3

3.4 Milestone 4

Deadline: 15.06.2021

Goals

- All SQL code
- Java or source code of project
- Sql-code for data loading
- list of successful and failed test cases

Gantt-Chart

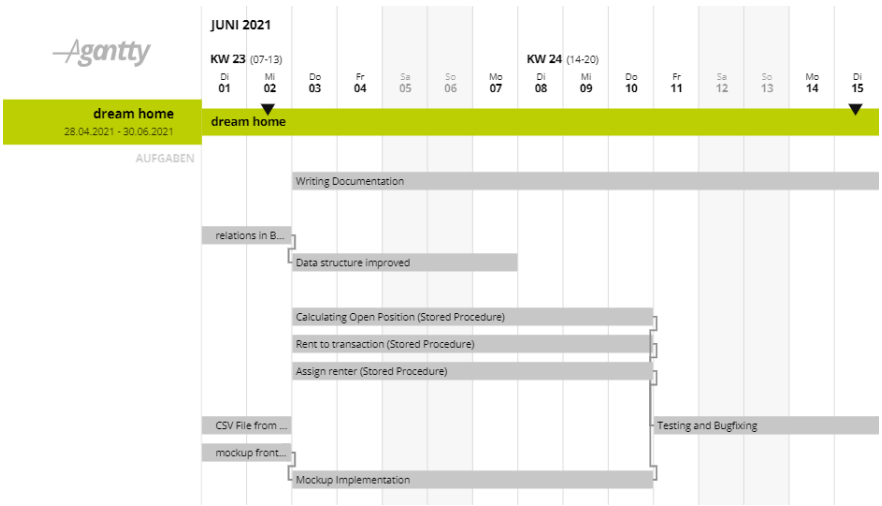


Figure 5: gantt-chart milestone 4

4 User views (use case)

4.1 Use Case 1: Csv file upload

Actor	employee of Dream House, responsible for Data Management
Preconditions	new monthly statement of bank account available
Trigger	Employee starts the upload manually
Basic Flow	1. Every month an employee of Dream House loads up the csv file to receive information about the bank account and it's transactions. 2. The system divides up the transactions to the renters and houses. 3. To divide up the operating costs the system uses several allocation keys (number of persons living in the flat – get from lease contract; Behälter – get from lease contract; Wohnfläche get from lease contract; Einzelabrechnung – get from number of flats)
Alternative Flow	If wrong file with wrong information, print error message for user. If no new information is available, print error message for user. If allocation keys can't be found, print error message for user.

4.1.1 ERM

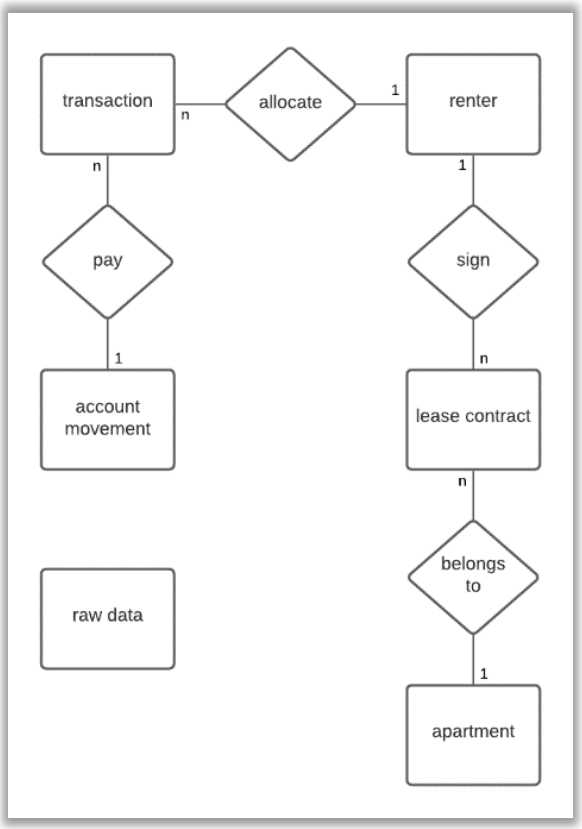


Figure 6: ERM use case 1

4.1.2 Relevant entities and relationships

- PostingDate
- PaymentReason
- AmmountOfPayment
- DebtorCreditor_ID
- Renter_ID
- NumberOfPersons
- Size
- Lease_Amount
- AdditionalCost

4.2 Use Case 2: Operating cost statement

Actor	employee/ specialist of Dream House, responsible for
Preconditions	Csv file uploaded

Trigger	new year; request of customer
Basic Flow	1. The system searches for specific customer figures (already allocated) 2. The system checks whether there have been some positive or negative transfers pre-payments of the specific renter and add these to the operating cost statement. 3. The system sums up the operating costs of specific renter and subtracts the sum of pre-payments of specific renter. 4. After this all operating cost statement are going to be send to renters.
Alternative Flow	If system can't information of customers, print error message for user.

4.2.1 ERM

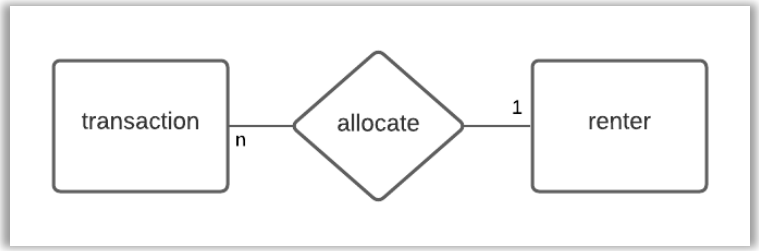


Figure 7: ERM use case 2

4.2.2 Relevant entities and relationships

- Renter_ID
- TransactionAmount

4.3 Use Case 3: Open positions

Actor	employee/ specialist of Dream House, responsible for
Preconditions	
Trigger	new month
Basic Flow	1. The system searches for specific customer figures 2. The system stores all done payments of specific customer and the figures of the payment he has to pay (triggered out of Stammdaten) in an accounting file. 3. Both figures are summed up separately. 4. The system shows the difference of the two sums.
Alternative Flow	If check (or calculation) of accounting file can't be done, print error message for user.

4.3.1 ERM

4.3.2 Relevant entities and relationships

- Renter_ID
- TransactionAmount

5 Requirements

5.1 Functional requirements

- The information system must verify the statement of bank accounts.
- The information system must be able to receive statement of bank account csv files.
- The information system must be able to print tables and information (operating cost statement; missing payments) as a pdf.
- The information system must be able to show operating cost statements and missing payments.
- The information system must calculate the operating cost statement.
- The information system must relate the cost positions to a building and distribute the costs to the different renters by a certain key.
- The information system must give the user the possibility to create, update and delete renters.
- The information system must not send the operating cost statements per e-mail.

5.2 Non-functional requirements

- The information system must use a MS SQL Server.
- The logic should (if possible) run on the database server.
- The information system needs a graphical interface.

6 Data Catalog

Table 1: data catalog

Data Catalog - Dream Home				
Entity Name	**Attributes**	**Description**	**Data Type & length**	**Constraints**
Transaction	AccountMovement_ID	FK from Table Account Movement	int	Foreign Key (composite key1)
	Renter_ID	FK from Table Renter	int	Foreign Key (composite key2)
	PostingDate	Date of the transaction	Date	Not Null
	PaymentReason	Explanation of the transaction	varchar	Not Null
	Amount	Amount of money	money	Not Null
Renter	Renter_ID	Unique identification	int	Primary Key
	rName	Last Name of the tenant	varchar	Not Null
	rSurname	First Name of the tenant	varchar	Not Null
	rPostcode	Adress of the tenant	int	Not Null
	rCity		varchar	Not Null
	rStreet		varchar	Not Null
	rGender	Gender of the tenant	char	Not Null
Account Movement	AccountMovement_ID	Unique identification	int	Primary Key
	PostingDate	Date of the transaction	Date	Not Null
	PaymentReason	Explanation of the transaction	varchar	Not Null
	Amount	Amount of money	money	Not Null
	DebtorCreditor_ID	Name of Payer	varchar	Not Null
	RawData_ID	FK from Table raw data	int	Foreign Key
Lease Contract	Contract_ID	Unique identification	int	Primary Key
	Renter_ID	FK from Table Renter	int	Foreign Key
	Appartment_ID	FK from Table Apartment	int	Foreign Key
	Lease_amount	Monthly Lease	money	Not Null
	DateOfSignature	Date the lease was signed	Date	Not Null
	NumberOfPersons		int	Not Null
	Additional costs	Information about add. costs	money	Not Null
	Security_Deposit	Amount of money	money	Not Null
Appartment	Appartment_ID	Unique identification	int	Primary Key
	Max_Renter	Number of maximum renters in this appartment	int	Not Null
	Size	Size of appartment in m^2	float	Not Null
	Appartment_number	number of appartments in house	int	Not Null
	Number_rooms	Number of rooms in appartments	int	Not Null
	House_Number	get number of appartments in house	int	Not Null

Table 2: description of relationships

Description of relationships				
Entity Name	**Multiplicity**	**Relationship**	**Multiplicity**	**Entity name**
transaction	n	allocate	1	renter
	n	pay	1	account movement
renter	1	sign	n	lease contract
lease contract	n	belongs to	1	apartment

7 Conceptual Data Model

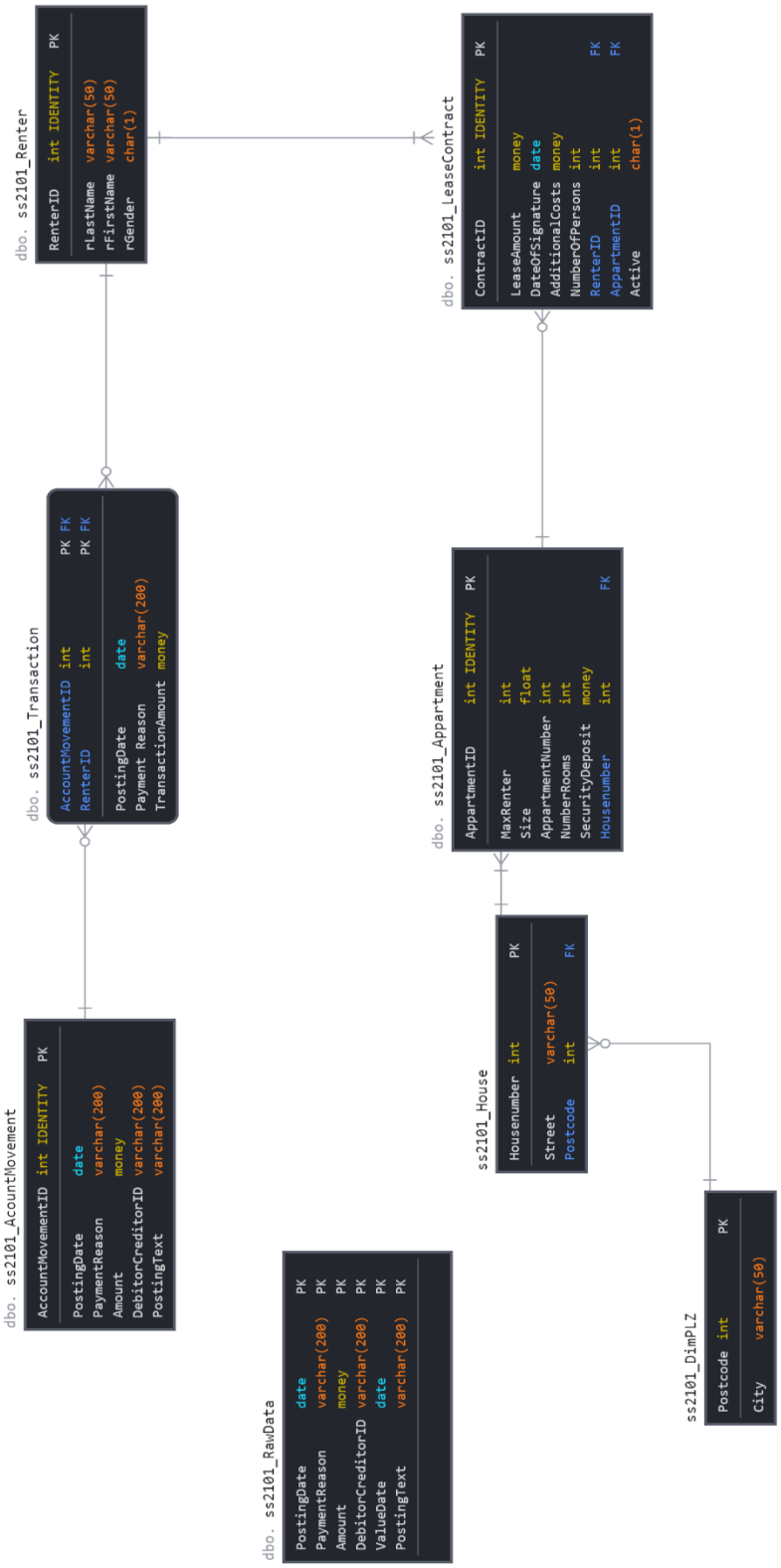
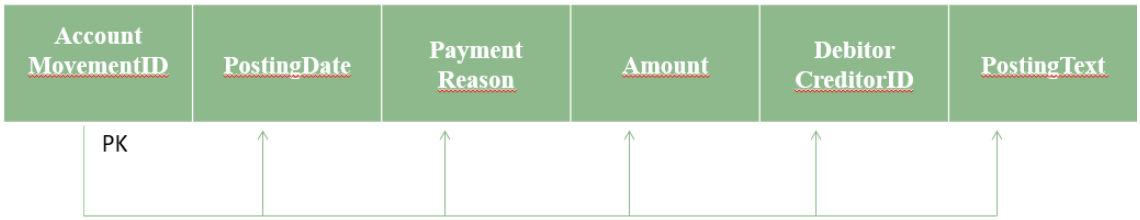


Figure 8: data model

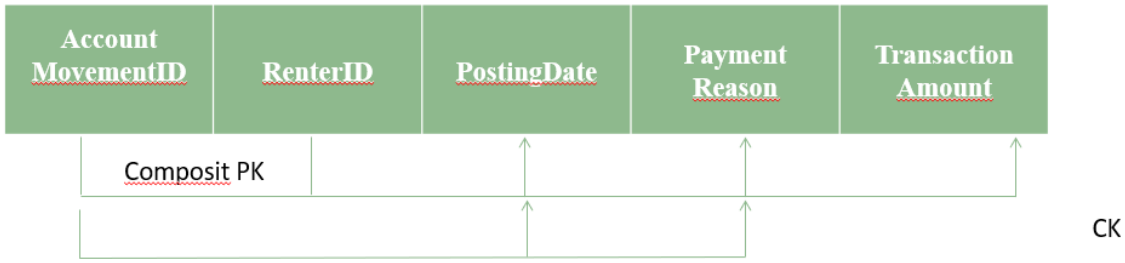
8 Relations in BCNF

8.1 SS2101_AccountMovement



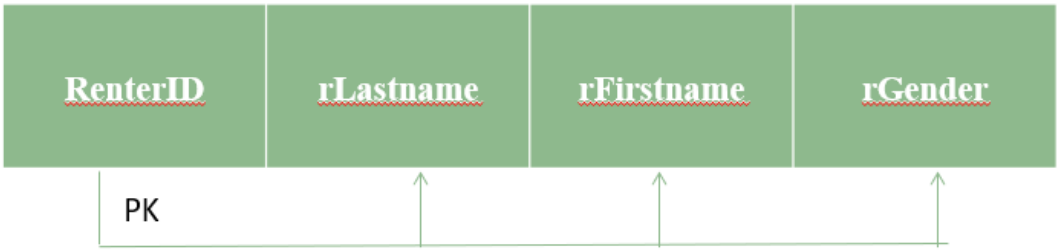
- No Transitive Dependencies (No additional Debitor-Creditor Info)
- No Partial Dependencies (Due to no composite PK)
- Every Determinant is a candidate key

8.2 SS2101_Transaction



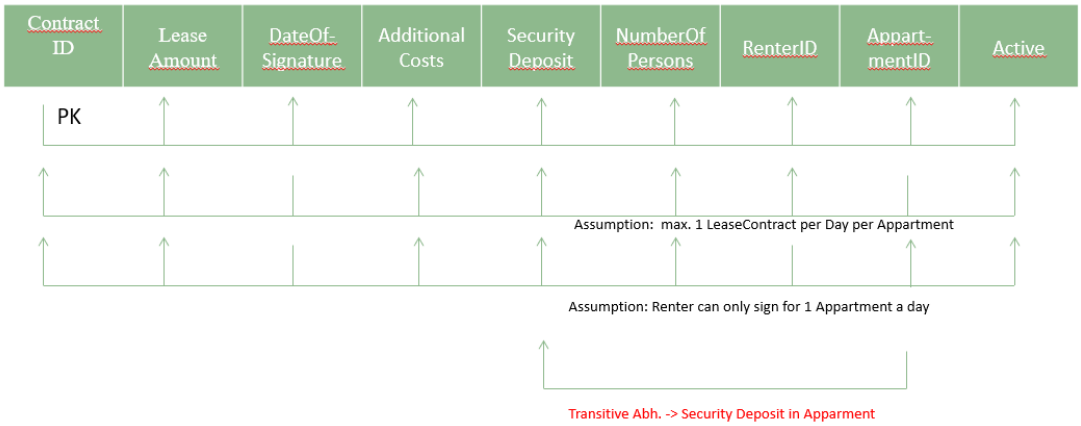
- No Transitive Dependencies
- 3 Partial Dependencies

8.3 SS2101_Renter

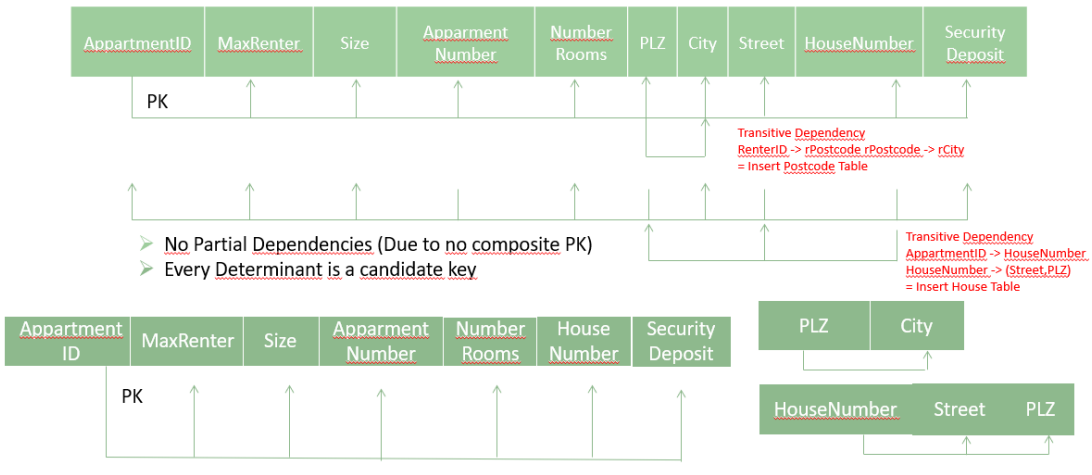


- No Transitive Dependencies (No additional Debitor-Creditor Info)
- No Partial Dependencies (Due to no composite PK)
- Every Determinant is a candidate key

8.4 SS2101_LeaseContract



8.5 SS2101_Appartment



9 Mockup vs. actual Implementation

Below is a comparison between the mock-up and the actual implementation.

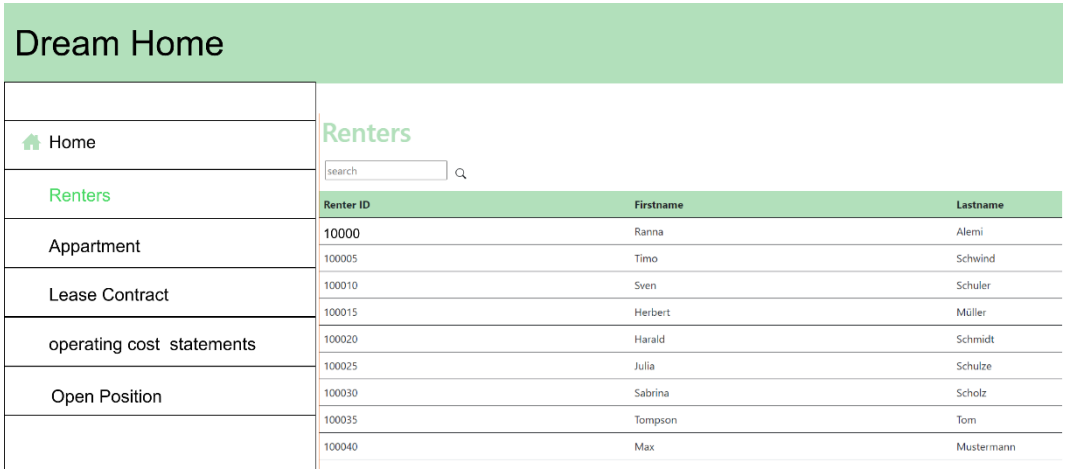


Figure 9: Renters Mockup

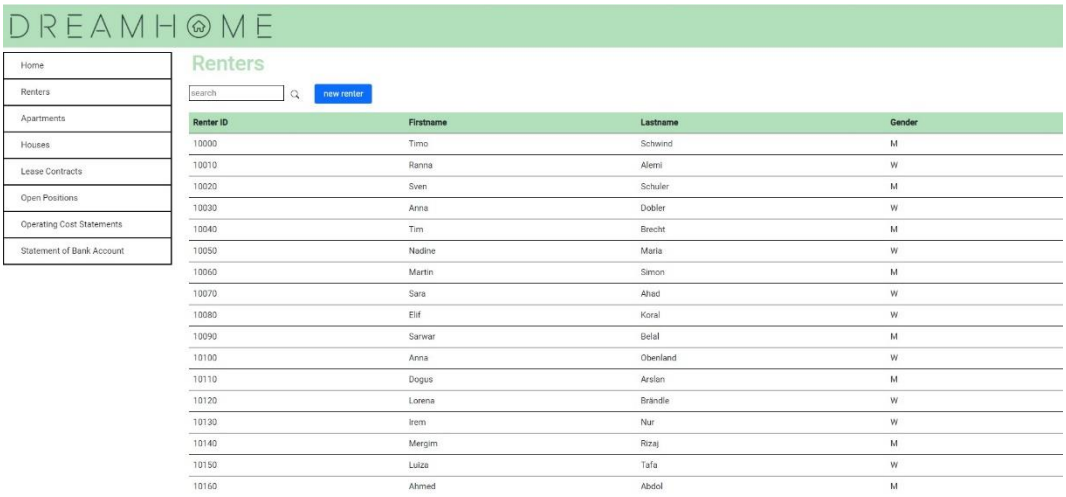


Figure 10: Renters Implantation

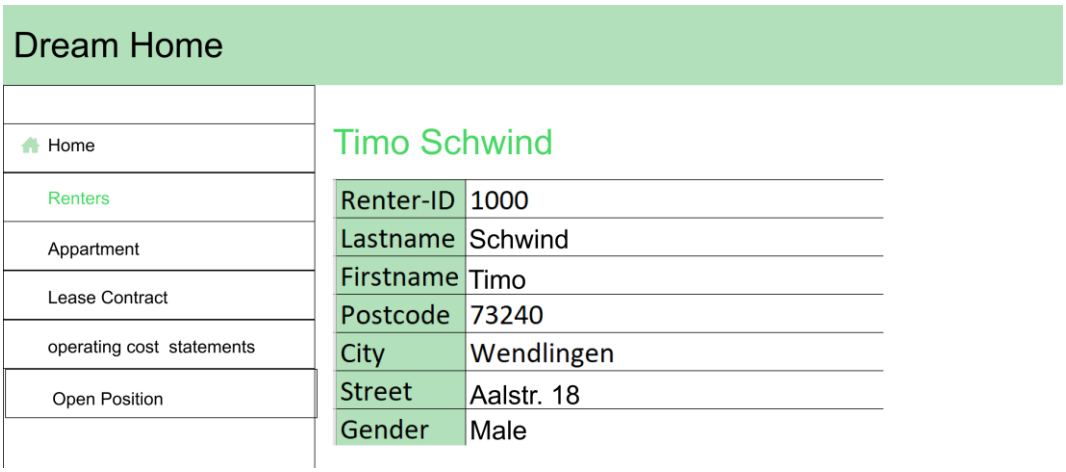


Figure 11: Renter Mockup

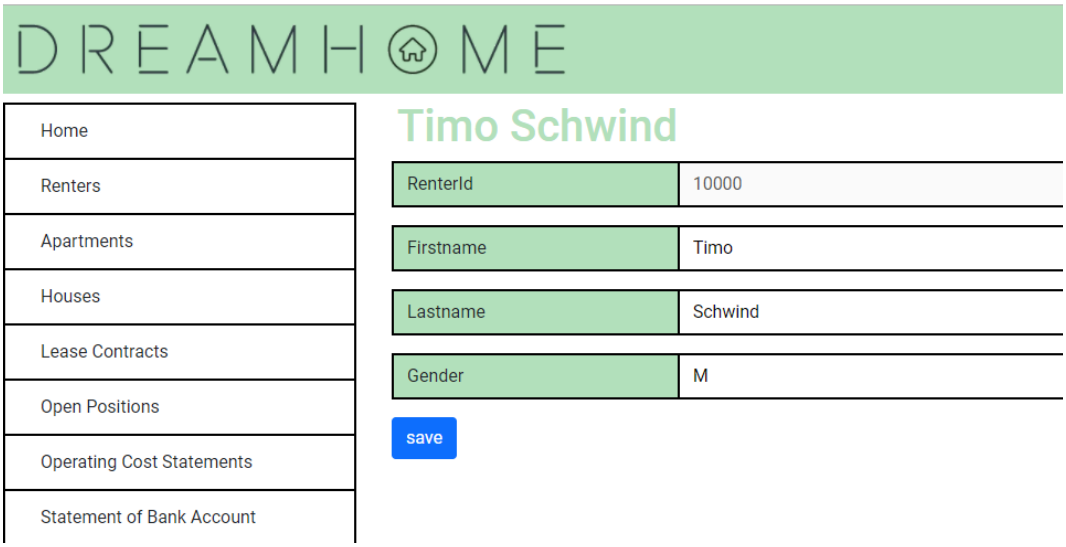


Figure 12: Renter Implementation

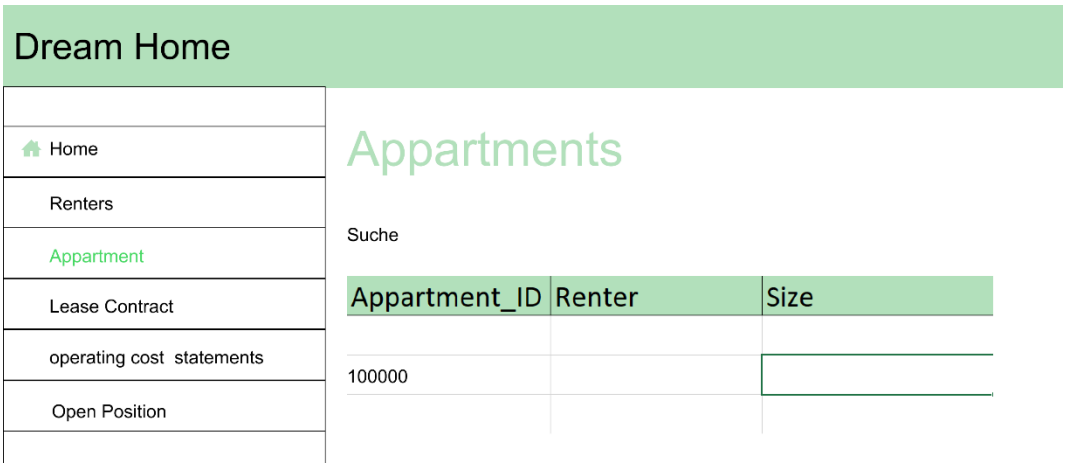


Figure 13: Apartments mockup

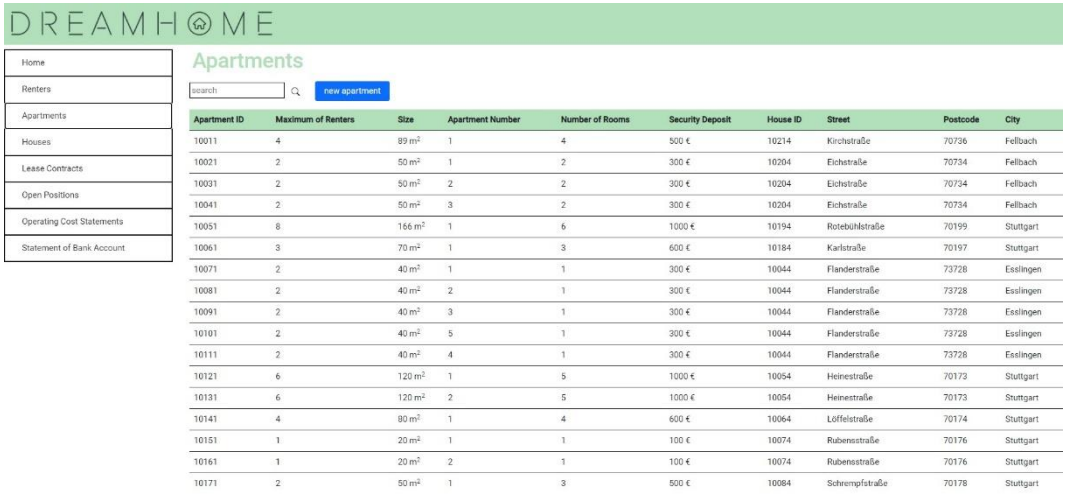


Figure 14: Apartments implementation

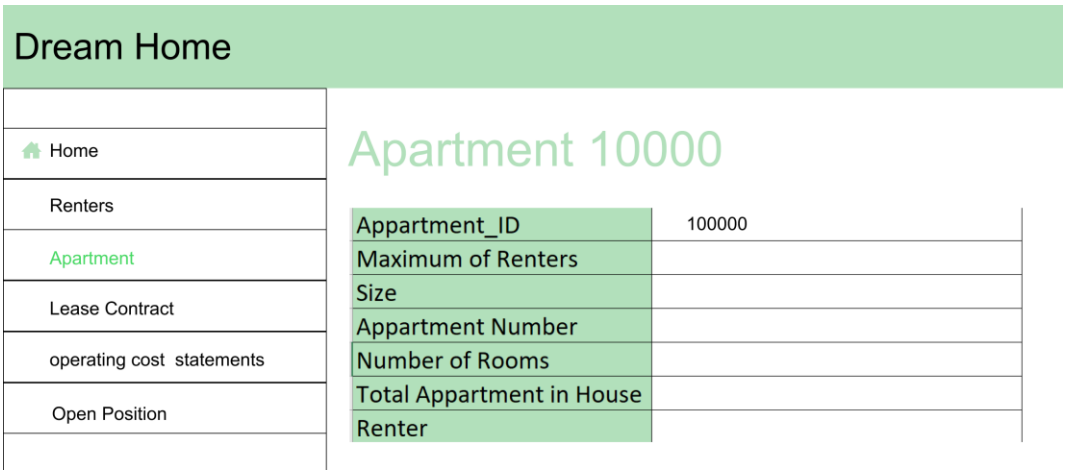


Figure 15: Apartment mockup

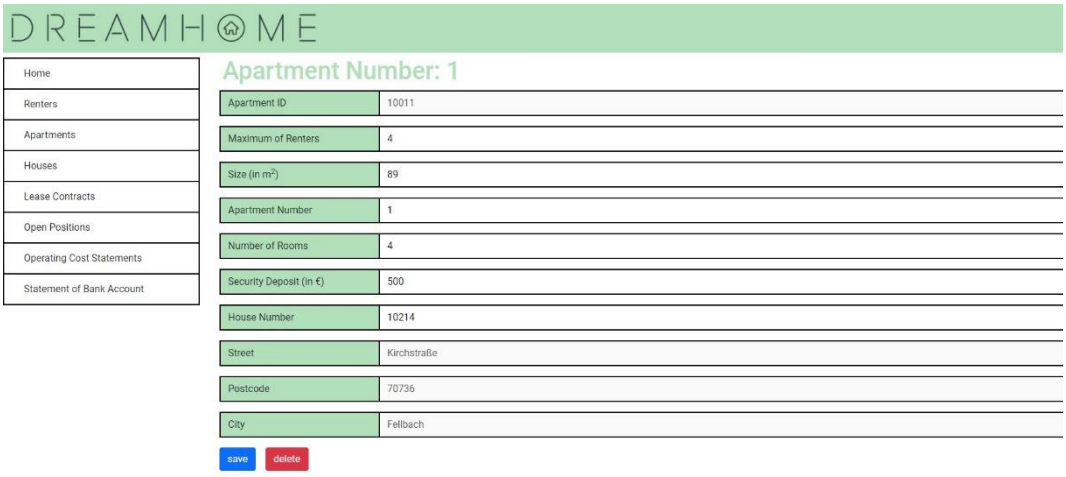


Figure 16: Apartment Implementation

Figure 17: Lease contract mockup

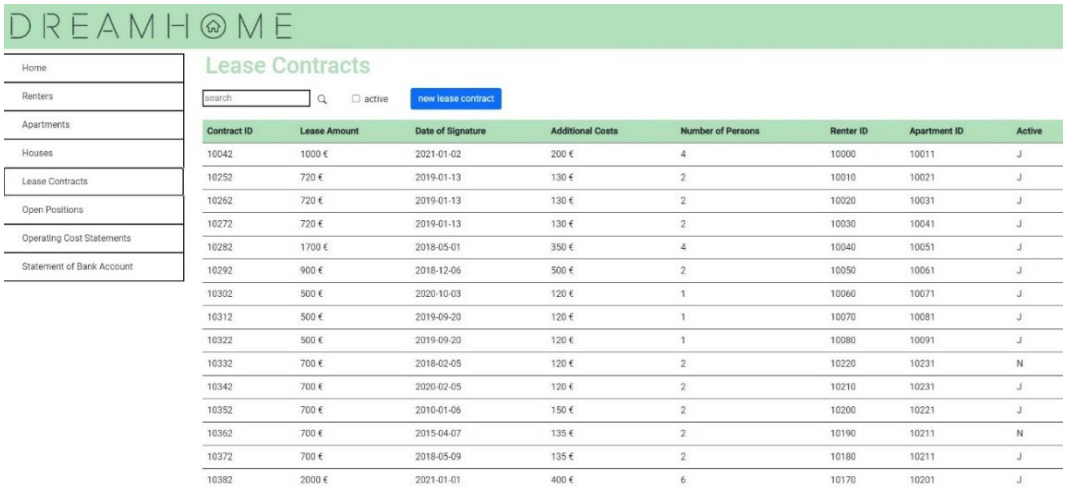


Figure 18: Lease contracts implementation

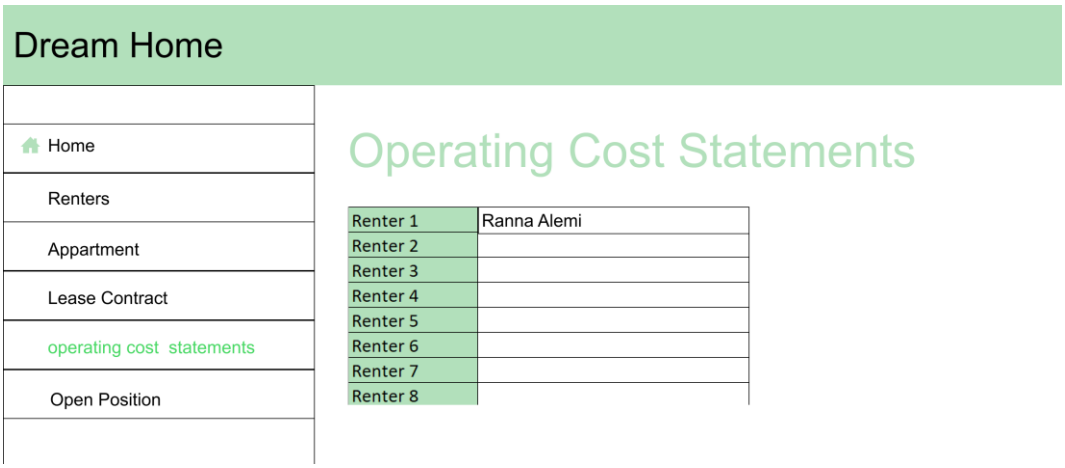


Figure 19: Operating cost statements mockup

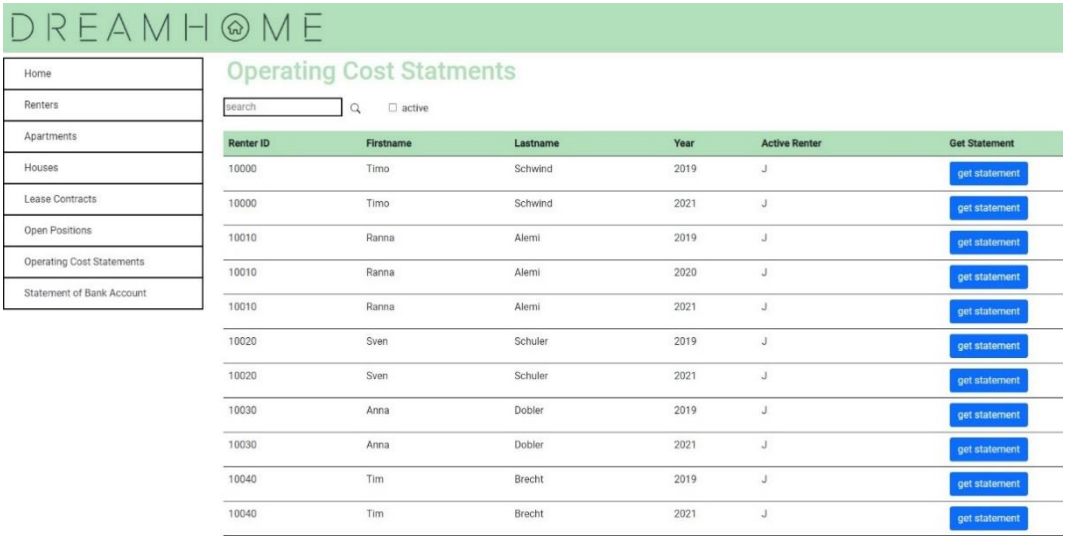


Figure 20: Operating cost statements implementation

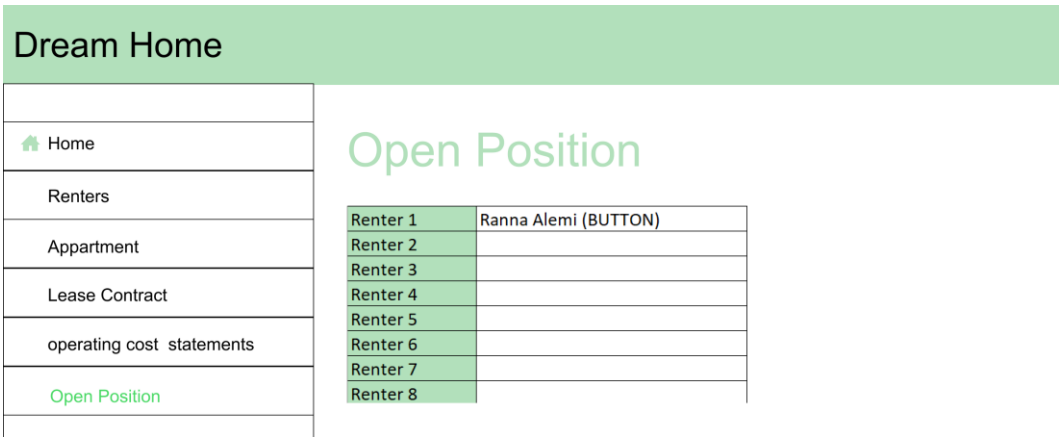


Figure 21: Open position mockup

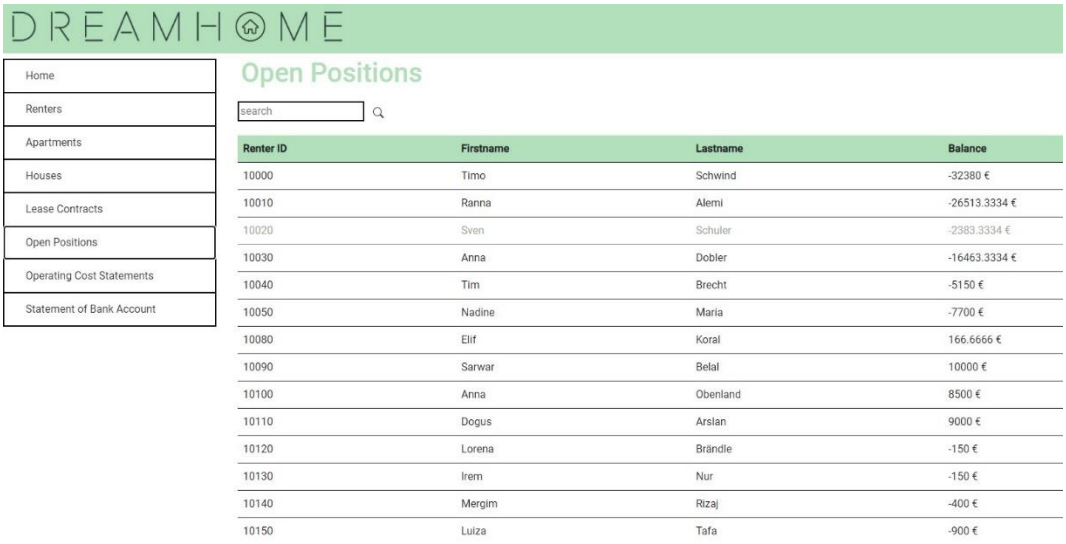


Figure 22: Open positions implementation

10 Lessons Learned

10.1 Ranna Alemi

For me it was the first experience to be from the beginning till to the end of such a project. I have learnt a lot from my colleagues. It was great to design a mock up und to implemented even better than what we expected. Even if the implementation in the frontend was a challenge.

The biggest problem was the time management. We invested more time than 60 hours of work (per person), because of the number of tasks. That is why it made it even difficulted to organize with all the other modules. Also, because we communicated only through Webex it made even harder. Because of that I learned how important it is to communicate very well in the team, so everyone can work in the short time, everyone has, effective and efficient.

10.2 Tim Brecht

I like the idea of designing and creating an information system from scratch. However, I had problems with taking so much time in the last weeks of the semester for such a lab. I think it would be worth considering this project instead of a written exam for the module grade.

However, I learned a lot and was able to gain practical experience around database conception and database creation. Furthermore, project basics were deepened during the collaboration with the team.

10.3 Anna Dobler

This project tought me even more how to work in teams. Most times there is not too much important stuff to do for each person in the beginning – it's more about discussions, in the end there is always a lot to do though. Therefore, it is very important that one starts immediately with planning and split work as soon as possible.

In our case we have had a lot of contact with each other even in the part where work was divided up. This was very good because we were able to talk to and help each other. This is what teamwork is about.

Furthermore, I was able to increase my knowledge in creating frontends with the help of angular. Discussions of the team also increased my knowledge and experience in database creation.

Additionally, I realized that writing protocols help to understand the project and the requirements even better. It allows you to think about it while writing the protocol.

This project also taught me how much work is behind all of the websites and databases we use in our daily lives. Although this was just a small project it took a while till it worked properly.

10.4 Sven Schuler

Through the project I improved my programming skills in java and typescript. For the first time I had to program a REST API which returns an excel file. Because of that I had to try a lot of different approaches to solve the requirement. Also, the implementation in the frontend was new, which was a challenge too.

The biggest problem was the time management. All tasks were split in the team, so everyone could work on his tasks by himself. But at the end of the project there were so many tasks, which could not be done early and with all the other modules and their exams and the other projects it was very hard to organize myself. Because of that I learned how important it is to communicate very well in the team, so everyone can work in the short time, everyone has, effective and efficient.

10.5 Timo Schwind

While working on the project I dramatically improved my T-SQL and SQL skills. I learned a lot about cursors and how to use them in the right way. I also worked a lot with the CASE-Tool SQLDBM which improved my understanding of data modelling and what is important to note during that process.

Timing the project wasn’t always that easy because I also had a lot of other things to do which often ended in stress, but in the end, everything worked fine, and I managed to do everything in time. Communication in our team was mostly very clear and on point, everybody had a good idea about what their tasks were so that everybody knew what to do. The stored procedure to assign the renter from the payment reason was by far the most difficult task to complete. But after I managed that too, my understanding of transact SQL even got better.