Überall für alle
SPITEX

OST
Ostschweizer
Fachhochschule

Rollende Küche

Department of Computer Science
OST - University of Applied Science
Campus Rapperswil-Jona

Spring Term 2021

Authors:	Marc Scherrer & Philipp Bolliger
Advisor:	Prof. Frank Koch
Project Partner:	Spitex Wädenswil
External Co-Examiner:	Hansjörg Huser
Internal Co-Examiner:	Silvan Gehrig

This document was typeset using LaTeX
June 16, 2021

Task Definition

People

Graduate students: Philipp Bolliger and Marc Scherrer
Supervisor: Prof. Frank Koch, professor for business informatics
Customer: Hedi Zbinden, manager of the “Rollende Küche”, SpitexVerein Wädenswil

Initial Situation, Problem Description

The ”Rollende Küche” (RoKü) program is a lunch delivery service by the Spitex Wädenswil for elderly people, that do not want to go to a retirement home but cannot cook for themselves. Currently they work with a desktop application to save customers, customer preferences, orders and plan routes for the deliveries. As the customer base grew and the complexity of preferences, correspondence addresses and inventory management grew, it was clear that the existing application will soon be insufficient, due to its restrictions.

Task Definition

The resulting application should essentially be a more future proof replacement. Additionally, some functions that became essential, should be implemented.

Functional Requirements

- Customer management
- Order management
- Route management
- Menu management
- Statistics, billing information and daily plan printout

Non-Functional Requirements

- The application should be accessible for colour-blind people (deuteranopia and tritanopia)
- The manual should be complete
- The installer should be easy to use
- Error messages should be helpful
- A full data recovery should be possible
- Startup <3s & screen transitions <1s

- 80% should be unit tested
- The customer data should be encrypted

The full and more comprehensive list of functional and non-functional requirements can be found in the project plan.

Execution

Meetings with the supervisor and the customer are initiated by the students. The meetings should be prepared, lead, and documented by the students. The meeting protocol shall be found in the repository in the associated tickets.

For the execution of the project, a project plan must be be created. Attention should be paid to have continuous and visible progress. When a milestone is achieved, the associated artifacts and versions shall be delivered in advance.

Documentation and Submission

The documentation for the project plan and tracking is to be produced according to the guidelines of the department of computer science. The details for the documentation, the research and the results are defined by the concrete work plan.

The documentation shall be submitted digitally.

In addition to the documentation, the following has to be submitted:

- A A0 poster as PDF
- All results and data, to understand the thesis (Source code, build scripts, testing code, test data, etc.)

Dates

The following dates and deadlines apply:

- **22.02.21** Beginning of the bachelor thesis.
- Until **19.06.21** The abstract has been submitted in the online tool https://abstract.hsr.ch/ and the students have released it for the supervisor/examinator.
- Until **15.06.21** The abstract has been submitted by the supervisor/examinator, with the correct and complete brochure, for further processing by the course of studies office.
- **18.06.21** 12:00 Submission of the report to the supervisor.
- **18.06.21** 12:00 Upload of all related documents to archiv-i.hsr.
- Until **10.09.21** Oral Examination.
- Until **12.09.21** Submission of the final grade

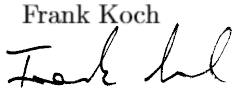
Evaluation

A successful bachelor thesis receives 12 ECTS points (1 ECTS point equals a work output of about 25 – 30 hours). The evaluation is based on the following criteria.

Aspect	Weight
1. Organisation, execution (project planning and tracking according to the project plan, autonomy, commitment, cooperation with the customer and supervisor)	1/6
2. Report (Content of the final project report, structure, presentation, language of the whole report)	1/6
3. Content	1/2
3.1 Problem analysis (Pre-study, literature study, requirements specification, requirements analysis, domain analysis)	1/6
3.2 Solution design (Solution variants and the assessment of those, decision, concept, draft)	1/6
3.3 Implementation and testing	1/6
4. Presentation and oral examination	1/6

Additionally, the rules and processes of the computer science degree program regarding bachelor thesis apply.
Rapperswil, the 22.02.21

Frank Koch



Abstract

The Rollende Küche (RoKü) is a service organized by the Spitex Wädenswil ZH, that delivers menus to elderly people. The service allows a customer to enjoy living and eating at home, without the hassle of cooking or ordering through conventional meal delivery services.

In 2001, the first step towards digitization was made in the form of an application that utilizes MS Access to store customer data, plan routes, and print reports such as daily delivery plans, a weekly overview of all orders and monthly invoices. Even though the application was updated in 2011, it still has limitations. The most significant are the route limit of five and that it is not possible to edit the orders.

The goal of this project was to replace the existing application. The new implementation should include all features of the current solution without its limitations.

To organize project management, a combination between Rational Unified Process and Scrum was used. A meeting with the key stakeholders was conducted at every sprint end, to show the progress of the project and get feedback.

A desktop application was created using the onion architecture to create an independent business layer. Dependency injection was used to decouple individual components. .NET 5 was used for the implementation. The data is saved in a simple SQLite database to make data backups easy and eliminate the need of an additional service that needs to run on the host machine. It further uses WPF as the user interface (UI) framework.

The resulting application has an intuitive design, is very flexible and the limitations of the existing application could be fully eliminated. Due to time constraints the scope had to be reduced.

Although the application is ready to be used in a productive environment, a few quality of life features are missing. Therefore the current state of the application is not able to fully replace the existing solution.

The new application is currently used in tandem with the current one. The software development will be continued outside of the scope of this project.

Management summary

Situation

Meal delivery planning for over 100 customers and five different delivery routes is not easy. With the help of a desktop application, the Spitex Wädenswil plans lunch deliveries since 2001. 20 Years later, the limitations of this application became more and more unbearable, and a renewal of the application was commissioned in the form of a bachelors thesis. The fact, that the routes are limited to five is the most severe limitation. With the growing customer base, more routes are soon needed. Other limitations are that separate billing addresses are not handled properly and that only one set of three routes can be permanently created, that serves as a template for other routes. Since the application is not well documented, functionality can not be easily extended.

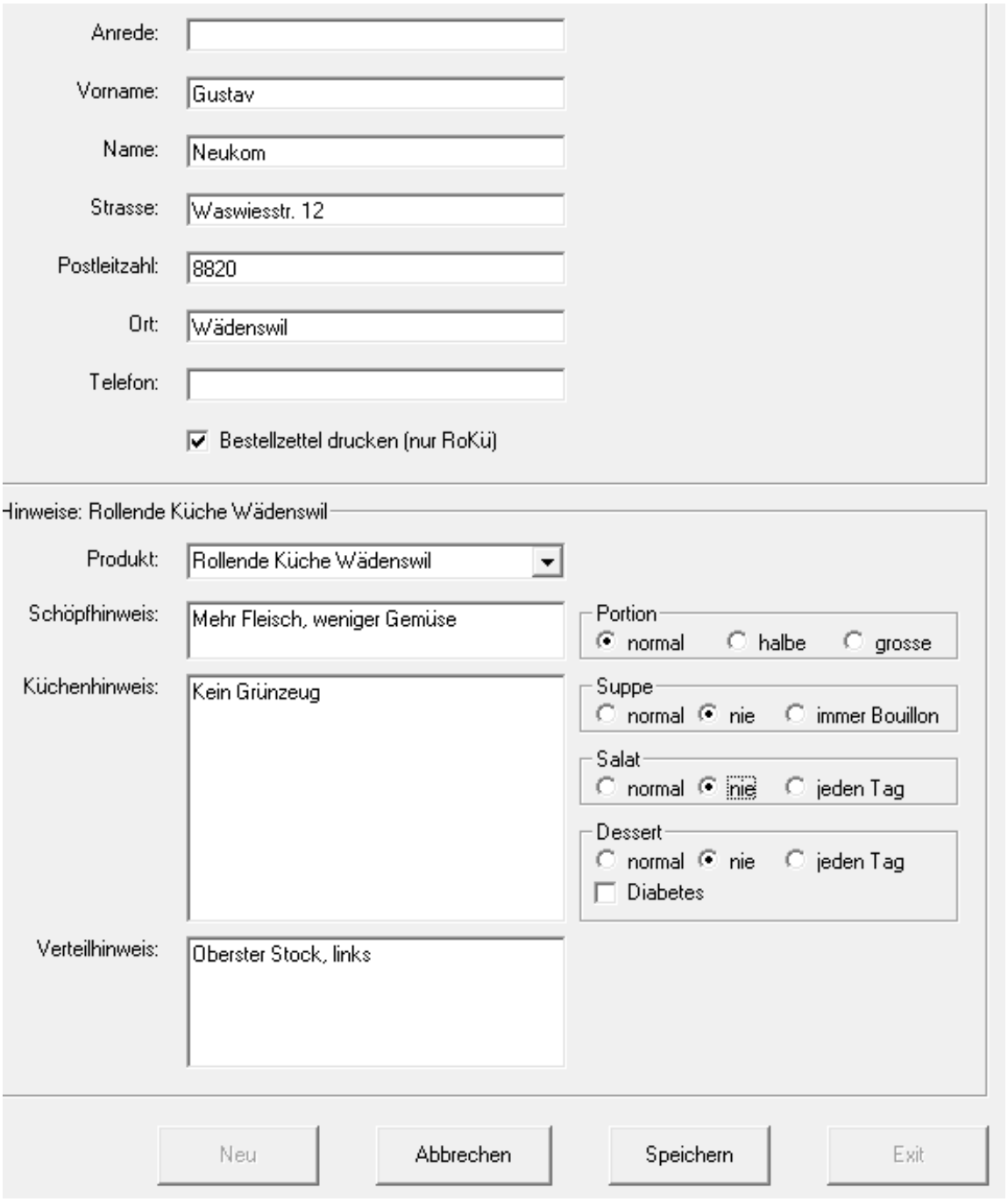


Figure 0.1.: Mask for entering a customer.

The renewed application should have all the functionality of the current one, but none of the limitations, and should be more flexible to give Spitex the possibility to change and evolve, without having to worry about the application.

Approach

First, the problem domain was carefully analyzed. Interviews with the RoKü administrator, the finance department and the kitchen chef were held to get to know the domain from different perspec-

tives. UI prototypes were created and reviewed with the key stakeholders. Functional requirements were discussed and prioritized. With all the gathered information, the evaluation for the technologies could be made. It was decided to create a WPF desktop application using .NET 5 and use SQLite to save data. The architecture of the software was designed, and a first architectural prototype was created. With the prototype up and running, more and more functionality was implemented and reviewed with the key stakeholders to get feedback and keep them up to date.

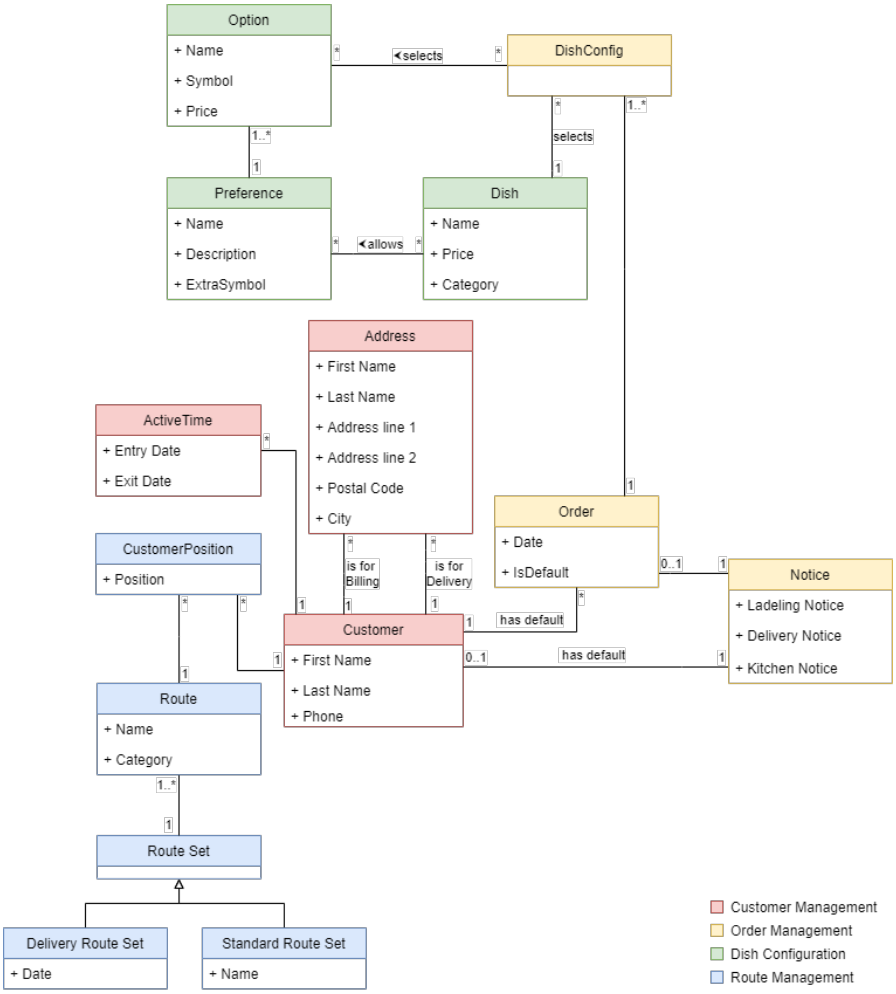


Figure 0.2.: Domain model.

Results

The resulting application is intuitive and more flexible. The limitations of the current version are not an issue in the new RoKü application. The administrator of the RoKü application can now define the available dishes, variations and several sets of predefined routes, that can be used as templates when planning deliveries. Orders and deliveries can be edited easily, and it is possible to assign customers default orders, that are automatically applied when creating new orders.

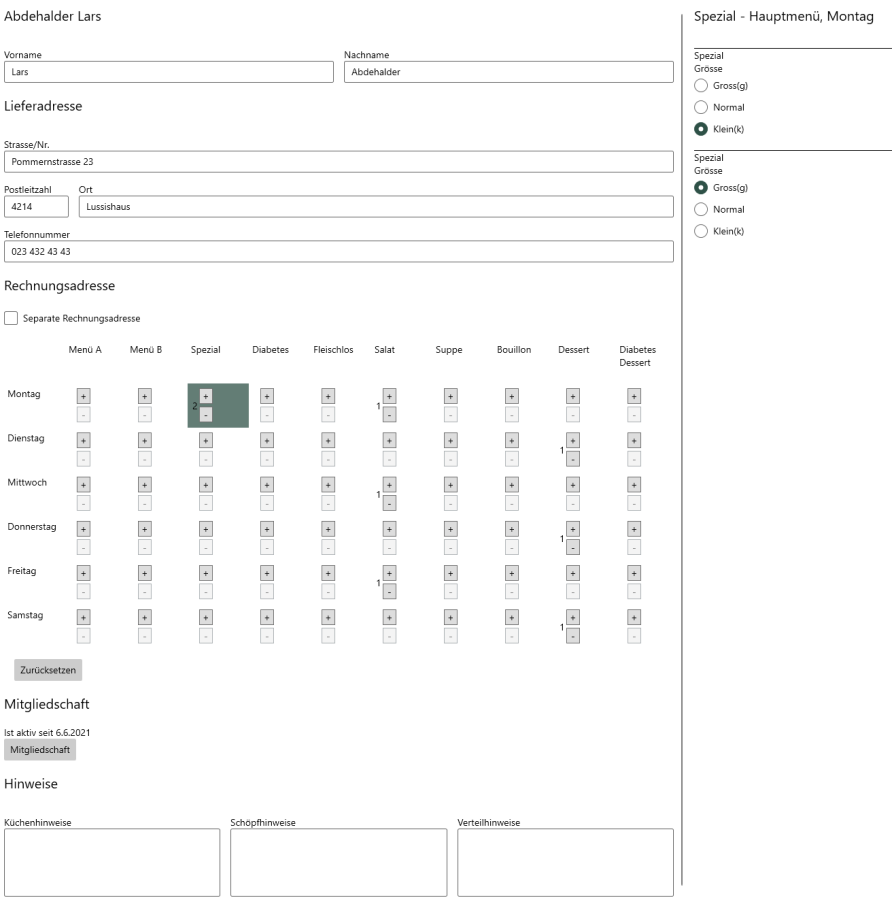


Figure 0.3.: Customer input screen.

Even though the application is in a usable state, some features and reports, that the current application has, could not be implemented in time. Therefore the application is not yet ready to fully replace the current solution.

Outlook

The development of the new RoKü application will be continued outside of the project scope until it is ready to fully replace the current application. A lot of additional features and changes came up over the project period. If and when these wishes will be implemented is unclear and depends on the decision of Spitex.

Contents

Part I.

Technical Report

1. Introduction

1.1. Disclaimers

- All data shown is sample data.
- This project is focused on solution creation.

1.2. Problem Description

The RoKü program is a service that provides customers of Spitex Wädenswil with a meal delivery service. The RoKü delivers lunch once a day with a salad and a dessert. It is intended for older people that do not have the energy to cook lunch every day. For this, Spitex works together with the elderly center Frohmatt Wädenswil. Menu plans and general information about the RoKü program are published on the website of Frohmatt Wädenswil and the meals are prepared in the kitchen of Frohmatt. Customers of the RoKü can order meals by calling and placing an order. Customers also receive an order sheet with their first delivery of the week. The customer can order food for the whole next week, by filling out the order sheet. The order sheet is picked up every Thursday.

In order to properly manage orders and plan routes, the administrator of the RoKü needs an application. New customers and address changes need to be reported to Frohmatt, the kitchen needs an overview of the orders for the whole week, the department of finance needs to know how much each customer needs to pay for the last month and the delivery personnel needs to know where they need to go and what to deliver. Doing all of this without an application would be way too much work and very error prone.

2. Current Solution

Spitex utilizes a program that helps them with customer management, order management and route management. The program was originally written in October 2001 and received some updates in January 2011. It stores data in a MS Access database.

2.1. Workflow

Every Thursday, the administrator waits for the filled out order sheets. With the order sheets the administrator can then update all customer details, enters the orders and plans the deliveries for the next week. Some customers have special wishes regarding cooking, scooping or delivery. Those wishes can be added to the orders. The orders can be added for each customer for the whole next week in a custom made mask in the program. The mask can be seen in figure 2.1.

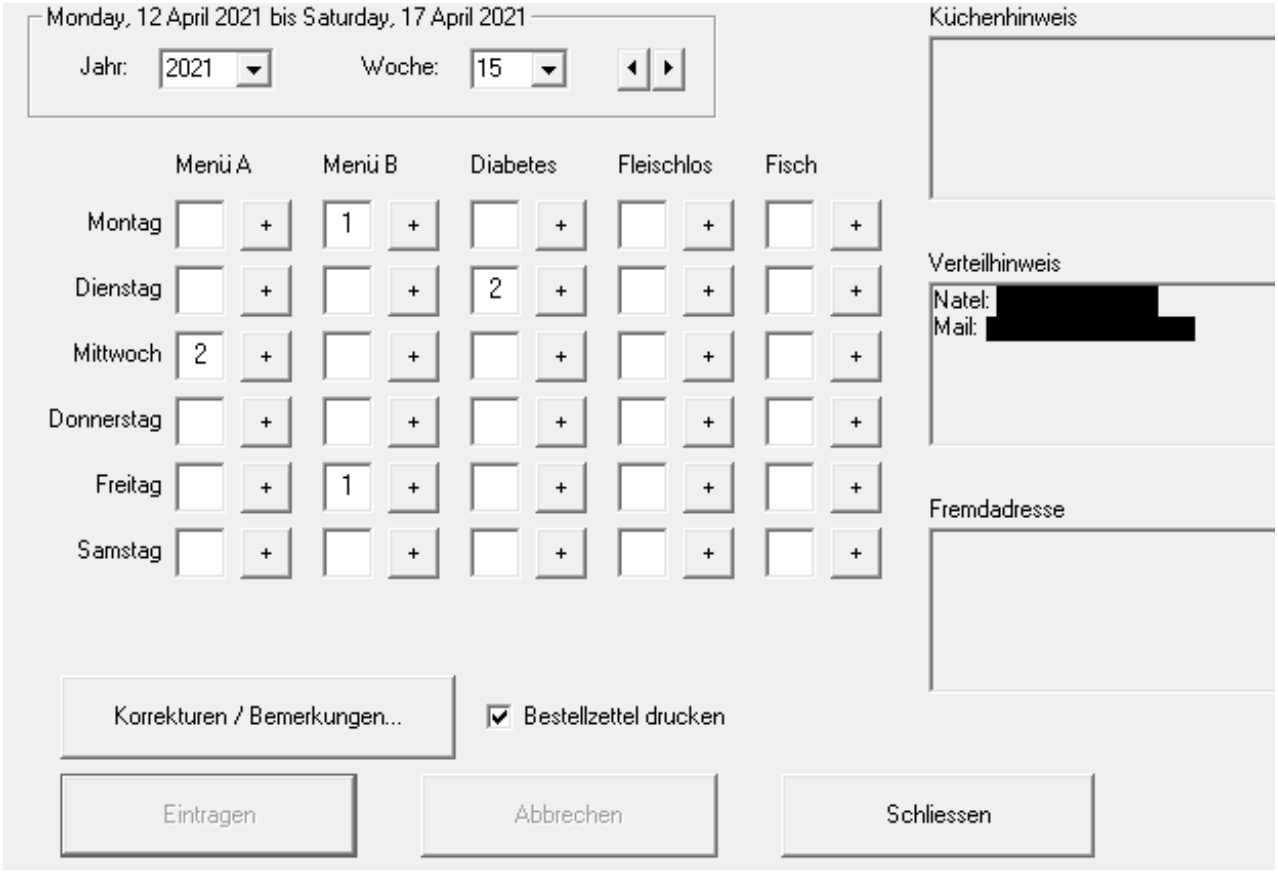


Figure 2.1.: Mask for entering the customers orders.

Customers are added to one of three standard routes. Customers within routes are in an order specified by the administrator. In which route and at which position a customer is, depends on his address. Because the delivery staff follows these predefined order during delivery, the administrator tries to order the customers in a way, that all meals can be delivered within one and a half hours and

that customers always get their meal around the same time. When planning a delivery for a specific date, the standard routes serve as a template for the delivery routes. The delivery routes only contain customers, that have an order on the day of the delivery. The delivery routes can be adjusted and an additional two routes can be configured for days with many orders. While the standard routes can be edited within the RoKü application, the delivery routes can only be edited in MS Access. The two masks are shown in figure 2.2 and 2.3. Once all delivery routes are planned, reports for the kitchen, delivery personnel and the finance department can be generated by MS Access.

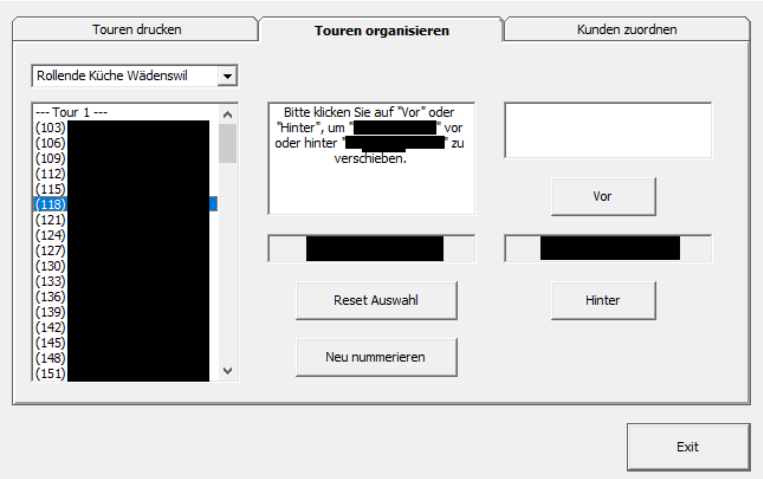


Figure 2.2.: Mask for planning standard routes.

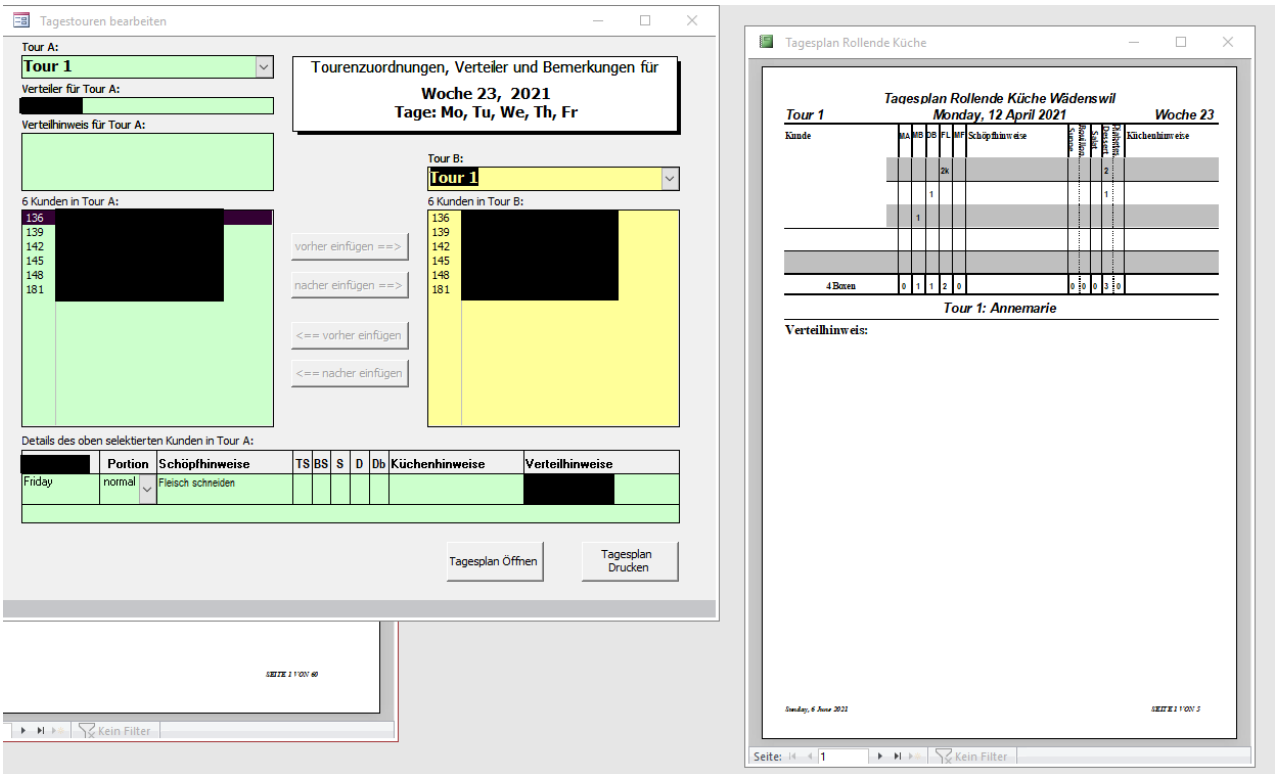


Figure 2.3.: Mask for planning delivery routes.

2.2. Limitations

The current solution has a lot of limitations. The program can only have three standard routes. This makes the delivery planning tedious when more routes are needed. But even then, only a maximum of five routes can be planned. With a lot of customers five routes may not be sufficient to deliver all meals on time. Further the program is counterintuitive and provides very little flexibility. Only the prices of the different menus can be changed without changing the source code of the program. People who have a separate billing address need to be handled differently in the financial report. The billing address should be printed, but it is simply ignored in the current solution.

3. Goals and Sub-Goals

Since the customer base is steadily growing, the route limitation of five will soon be insufficient and the current solution needs the be replaced. The goal of this project is to replace the current solution with a more flexible application that gets rid of all the limitations. In order to fully replace the current solution, the new application needs to implement the following functionality:

- Customer management
- Several sets of standard routes
- Order management
- Delivery planning
- Dish management
- Dish options
- Generation of reports

Additionally some sub goals were defined that can be implemented:

- Adding a dinner option
- A system to track warming boxes
- A fee for broken dishes
- A dish rental fee
- Fees for special wishes
- Order online

3.1. Conditions

The following conditions must be considered while developing the new solution.

- The application must be usable form the laptop of the RoKü administrator.
- The maintenance cost of the new application should be as little as possible.
- All decisions that either impact the administrator or the budget, must be approved by the current administrator which is the business partner and direct contact for the RoKü.

4. Solution Concept

Several interviews with different stakeholders were held to get to know the domain and determine requirements. The requirements were discussed with, and prioritized by the RoKü administrator. After a thorough evaluation it was decided to create a desktop application using WPF as UI framework, .NET 5 as software framework and SQLite as database. Entity Framework Core (EF Core) was used to access data in the database. UI prototypes were created, to make sure, that the problem domain was understood and the proposed solution would fit the administrators needs. A first architectural prototype was created as proof of concept. Dependency injection was used to decouple components and make testing easier. Unit and integration tests were written to make sure the code works as intended. The whole source code was managed with GitLab and every time changes were pushed to the repository, the unit and integration tests were automatically executed.

With the working prototype and the GitLab setup, the development of the new RoKü application could officially start. Every other week a meeting with the key stakeholders was held to present the progress and get feedback.

5. Results

The design of the new application is clean and simplistic. Especially in the screen, where the administrator can create new orders for the next week. The screen is packed with information, but offers a lot of quality of life features that make entering all orders more efficient. All other screens are less cluttered, and only important information is shown.

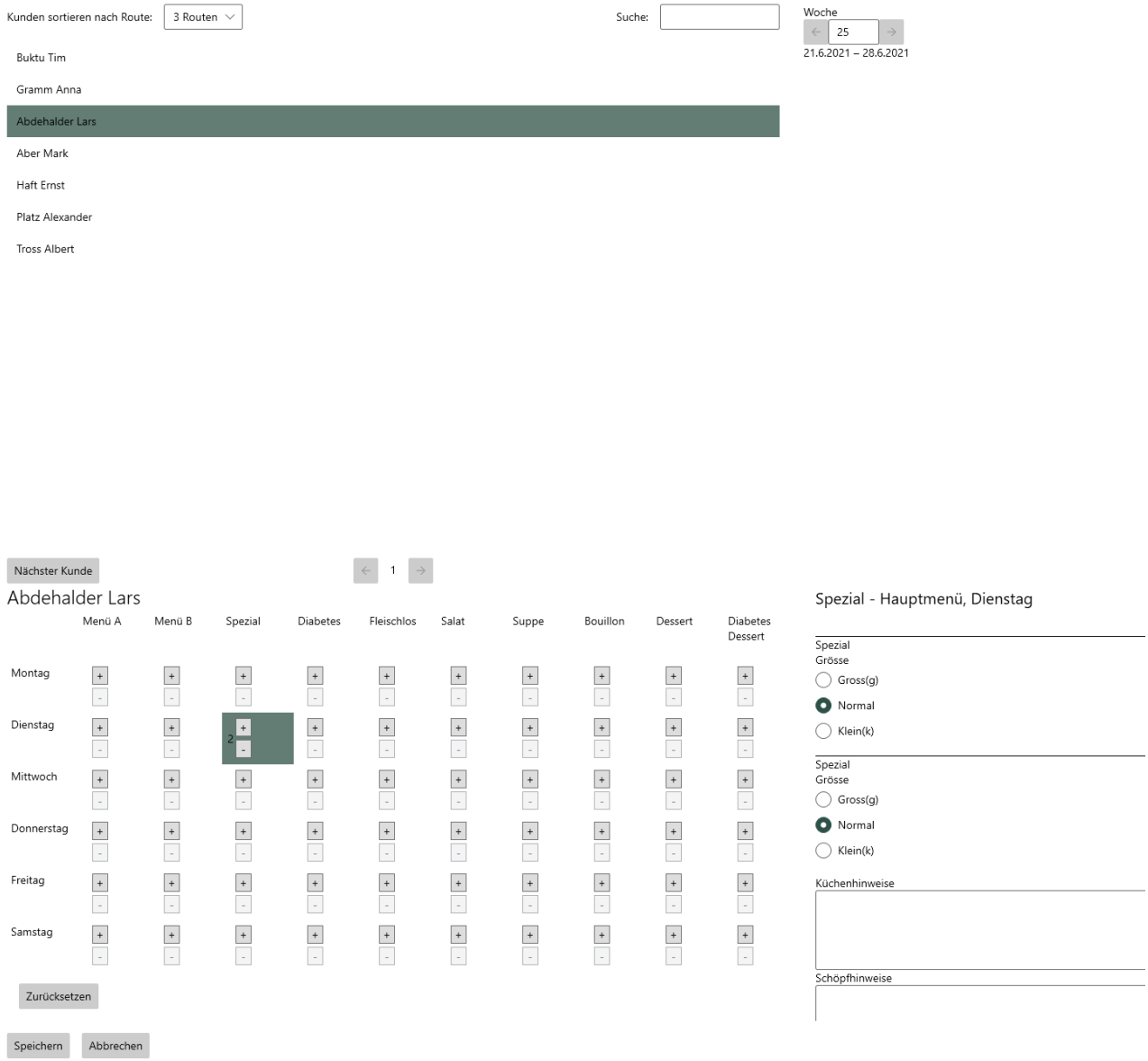


Figure 5.1.: New application.

The new application is much more versatile. The administrator can define the dishes that can be personalized with different preferences. Preferences can have several options which can have a

price difference that is added to the price of the dish. For example, some dishes can allow a "size" preference with the options "large", "normal" and "small".

Customers can have different default orders on different weekdays. Every order can consist of several different dishes and for each dish the preferences can be customized to match the customers needs. It is even possible to have more than one of the same dish per day, which is ideal for couples.

The administrator also has a lot more possibilities when it comes to route planning. It is possible to have several default sets of routes. When planning deliveries one of those sets can be chosen as template for the new delivery routes.

6. Conclusions

The new RoKü application is ready to be used. It offers a lot of new features that improve the daily workflow of the RoKü administrator. The new application solves all the limitation of the current solution. However, not all features of the existing solution could be implemented. A feature that allows administrators to assign default options to customers and some reports are still missing. The new application is therefore not yet ready to completely replace the existing solution.

The project got delayed due to underestimating the work to implement the architectural prototype and the individual features. Additionally, several smaller problems with the frameworks and libraries used further delayed the development.

All future use cases that came up during this project, are documented, in chapter 13. The development will be continued outside of this project.

Part II.

Software Project Documentation

7. Project Management

7.1. Project Plan

The project team had 17 weeks during the spring semester 2021 available. In order to successfully complete the bachelor thesis each member of the team needs to earn 12 ECTS, which is at least 300 hours according to the guidelines of the university council [8]. The total man hours available are therefore about 600 hours, which are equally distributed over the 17 weeks. This essentially means, that one week of work equals 36 man hours.

The process of developing the product is a combination of Rational Unified Process (RUP) and Scrum. The available time is divided into five phases. The four standard RUP phases and an additional documentation phase at the end of the project to finalize the documentation. The inception is only five days. Each other phase consists of one or more sprints. Every sprint starts and ends on a Friday and has a duration of two weeks.

Sprint	Period	Phase
0	22.02-26.02.2021	Inception
1	26.02-12.03.2021	Elaboration
2	12.03-26.03.2021	Construction
3	26.03-09.04.2021	Construction
4	09.04-23.04.2021	Construction
5	23.04-07.05.2021	Construction
6	07.05-21.05.2021	Construction
7	21.05-04.06.2021	Transition
8	05.06-18.06.2021	Documentation

Table 7.1.: Project plan.

7.2. Milestones

Milestone	Planned Date	Actual Date
Requirements	26.02.2021	27.02.2021
End of Elaboration	12.03.2021	21.03.2021
Customer Management	26.03.2021	07.05.2021
Route & Menu Management	09.04.2021	16.05.2021
Minimal Viable Product (MVP)	23.04.2021	30.05.2021
Functionality Freeze	21.05.2021	30.05.2021
Product Released	29.05.2021	11.06.2021

Table 7.2.: Milestones.

7.2.1. Requirements

The first milestone includes a set of functional requirements and non functional requirements (NFR). Every functional requirement should include:

- Role
- Use case
- Action
- A german version

7.2.2. End of Elaboration

The "End of Elaboration" milestone is significantly bigger and more important than the "Requirements" milestone. If the milestone can not be achieved, a transition into the construction phase is prohibited. The following criteria must be met in order to reach the milestone and proceed to the construction phase:

- Scope is set through use cases, a domain model and NFRs
- UI prototypes created and reviewed by Spitex
- Architecture is set
- Subsystems and interfaces are defined
- Architectural prototype created and working
- Developer tools, build server, testing and code analysis are set up
- Stories are created and estimated
- Risks are minimized

7.2.3. Customer Management

The milestone "customer management" is reached as soon as customers and preferences can be created, edited and deleted within the application and default preferences can be assigned to customers.

7.2.4. Menu Management

To reach the milestone "menu management", it is necessary to be able to do the following actions in the application:

- CRUD Dishes
- Default orders can be created for customers
- CRUD default route sets

7.2.5. Minimal Viable Product

The MVP is the first version the customer can work with. In addition to menu management and customer management, it also needs to include route management. The following features must be implemented to reach the MVP:

- CRUD Orders
- Plan deliveries
- Create daily plans as PDF
- Create order sheets as PDF
- Create weekly plan as PDF
- Create billing information as PDF

7.2.6. Functionality Freeze

After the MVP, time is used to implement further features and bug fixes. At the milestone "functionality freeze" all further features have to be implemented and tested. After this milestone, only bug fixes will be made.

7.2.7. Product Released

To reach the milestone "product released" the following criteria need to be met:

- A German instructions document that explains installation and operation of the product is in possession of Spitex.
- Spitex has the necessary files to install the program.
- Spitex has the source code of the product.
- The product is installed on at least one device of Spitex.

7.3. Meetings

On every first Thursday of the sprint, the review and the review evaluation are held. The retrospective, refinement and planning are held the day after the review. More information to the retrospective, planning and review can be found in the Scrum guide [9]. The meeting protocols can be found in the GitLab issues for the meeting.

7.3.1. Review Evaluation

The review evaluation takes place after the review but before the planning. It is necessary that this meeting is before the planning in order to take the new issues into consideration while planning. Based on the feedback from the review, the next steps are discussed and new issues are created if required.

7.3.2. Refinement

The refinement is a well known type of meeting among Scrum developers and is also mentioned in the Scrum guide, but what exactly needs to be done is not clearly defined. For this project, the refinement consists of estimating new issues, splitting large issues, into several smaller issues and discussing the relevance of existing issues which could lead to removing or adding issues.

When estimating issues, the developers discuss what needs to be done and use this information to guess its size. Each size specifies how long it takes to complete an issue. An issue is categorized into one of the following sizes.

Size	Time in hours
XS	1
S	4
M	8
L	16
XL	32+

Table 7.3.: Issue sizes and how long it takes to complete them.

Issues that were estimated with a size of XL need to be split down into smaller issues. The estimation is registered in GitLab as hours.

7.4. Stories

Stories are managed using GitLab. For each story one issue is created. The feature "task lists" is used to create tasks necessary to complete a single story [10]. Every issue is assigned to one or more labels to categorize the work that needs to be done. This is useful for the time reports. The full list of labels and their purpose can be found in the GitLab repository.

7.5. Roles

7.5.1. Advisor

Prof. Frank Koch is responsible for the evaluation and available to assist the team with any issues that they encounter.

7.5.2. Project Owner

Marc Scherrer will be the project owner as described in the Scrum guide [9].

7.5.3. Scurm Master

Philipp Bolliger will act as the scrum master as described in the Scrum gude [9].

7.5.4. Developers

Marc Scherrer and **Philipp Bolliger** will be the developers as described in the Scrum guide [9].

7.5.5. Stakeholder

A distinction is made between key stakeholders and normal stakeholders. The key stakeholders are the stakeholders who have a lot of power over the project. Key stakeholders will attend the review meetings.

Name	Role	Description
Prof. Frank Koch	Key stakeholder	Prof. Frank Koch is a very valuable stakeholder because he has a lot of experience which can help improve both the development and the resulting product.
Mrs. Hedi Zbinden	Key stakeholder	As administrator and director of the RoKü, she has the most experience working with the previous tool, which makes her a very important key stakeholder.
Mr. Manuel Keller	Stakeholder	Mr. Keller is responsible for the finances of RoKü. He receives a report from the RoKü application.
Mr. Roger Gräppi	Stakeholder	Mr. Gräppi is the person in charge of the kitchen. He receives several reports from the system.

Table 7.4.: Stakeholders.

7.6. Developing Process

GitLab is used as the version control tool. Instead of having one repository with both source code and documentation, two separate repositories "Documentation" and "Product" were created. This is because changes on the documentation would also invoke a continuous integration (CI) pipeline for the the product and vice versa, which would use up a lot of resources. Experience has shown, that this can hinder the development process if subsequent work is dependent on CI results. To have consistent commit messages throughout the project the commit messages should be written according to the following template:

If this commit is applied it should <commit message> #<related issue number>

7.6.1. Documentation Repository

The documentation repository includes the thesis as well as all other written documents and models. Changes are pushed directly on the master branch.

7.6.2. Product Repository

The product repository includes the source code for the RoKü application as well as source code for any tools and scripts necessary for the development. For each story the developers create a separate branch to work on. As soon as the story is finished according to the definition of done, the developer creates a merge request to merge the feature branch back to the master branch. The merge request is reviewed by the other developer and finally merged into the master branch. Minor changes can also be done directly on the master branch. The following activities are considered minor changes:

- Renaming
- Removal of unused using statements
- Removal of unused code

- Removal of comments
- Correcting spelling errors
- Fixing small visual or logic bugs
- Changes to the read me
- Changes to GitLab CI configuration

7.6.3. Definition of Done

The definition of done helps to unify the understanding of when a story is considered done within the developer team. It consists of a set of criteria. If all criteria are met, the story is considered done and a merge request can be created. Since there is no universal definition of done, the project team had to create one that is customized for the RoKü project. The resulting criteria are listed below.

- No ToDo's that reference the story remain in the source code
- There is no code commented out
- The build succeeds for the whole solution
- The total test coverage is above 80
- All tests are successful
- Sonarcloud does not report any
  - bugs
  - vulnerabilities
  - security hotspots

7.7. Risks

This chapter deals with the risks that could occur during project execution. Each risk has the following estimated properties:

- **Max. Damage**: The maximum damage a risk can cause if it occurs.
- **Probability**: The probability of the risk occurring within the duration of the project.
- **Weighted Damage**: The weighted damage is the multiplication of the probability and the maximum damage. Risks with a high weighted damage are the most critical ones, which should be minimized with measures.
- **Measure**: The measure is the precaution taken to either lower the probability of the risk occurring or lower the damage an occurrence of that risk causes.
- **Behavior**: The behavior describes what is done when a risk occurs.

1. Risk: Sickness of a team member
   - Max. Damage: 24 h
   - Probability: 20%

- Weighted Damage: 4.8 h
- Measure: -
- Behavior: Scope reduction

2. Risk: Implementation of wrong functionality or properties
   - Max. Damage: 40 h
   - Probability: 50%
   - Weighted Damage: 20 h
   - Measure: Detailed requirement analysis and reviews with the customer
   - Behavior: New Stories to alter functionality, scope reduction
3. Risk: Unrealistic timeline
   - Max. Damage: 20 h
   - Probability: 10%
   - Weighted Damage: 2 h
   - Measure: -
   - Behavior: Scope reduction
4. Risk: Continuous requirement changes
   - Max. Damage: 40 h
   - Probability: 10%
   - Weighted Damage: 4 h
   - Measure: Incremental development with reviews each sprint
   - Behavior: Alter backlog to reflect the changed requirements.
5. Risk: Bad usability
   - Max. Damage: 40 h
   - Probability: 20%
   - Weighted Damage: 8 h
   - Measure: UI prototype that is reviewed with the customer. Decouple presentation from business layer.
   - Behavior: Create new UI prototypes, let the customer review them and implement changes
6. Risk: Familiarization with software technology takes longer than expected
   - Max. Damage: 20 h
   - Probability: 20%
   - Weighted Damage: 4 h
   - Measure: Attend workshops/tutorials to gain knowledge of technology
   - Behavior: Scope reduction
7. Risk: Technical issues with development environment, both locally (PC) and remotely GitLab
   - Max. Damage: 10 h

- Probability: 5%
- Weighted Damage: 0.5 h
- Measure: Install development environment on one extra device
- Behavior: Change to backup device/Ask OST support for help withGitLab issues

Total risk to count for: **43.3 h**.

8. Requirements Specification

8.1. Requirements Elicitation

Several interviews with the administrator of the RoKü, the chef and the department of finances were part of the requirement determination. A lot of documents provided by the administrator of RoKü, which included a manual for the current RoKü application, helped to determine the functional requirements. These documents can be found in the appendix E. After the first set of functional requirements was defined, they were discussed with the RoKü administrator. In this discussion more requirements were added, and some were altered or even removed, because a new requirement replaced it. After all requirements were defined, the list of requirements was sent to the administrator of the RoKü. The administrator then ordered the requirements based on importance. All requirements are listed in section 8.2, ordered by the administrator. This order is seen as prioritization of the requirements. The requirements are written as user stories. The following template was used to define user stories:

As <role> I want to <use case> to <reason (if not obvious)>

All use cases up to and including Nr. 32 (print out/send statistics) are implemented in the old application.

8.2. Functional Requirements

1. As an administrator I want to back up all routes/customers/menu data.
2. As an administrator I want to assign customers to routes.
3. As an administrator I want to change the order of the customers within a route.
4. As an administrator I want to record entry and exit date. Note: There are customers that enter end exit multiple times.
5. As an administrator I want to assign separate billing addresses to customers.
6. As an administrator I want to CRUD customers.
7. As an administrator I want to CRUD default distribution notices.
8. As an administrator I want to CRUD default kitchen notices, to reflect wishes (e.g only green salad).
9. As an administrator I want to assign default starter days.
10. As an administrator I want to add different courses to the menus.
11. As an administrator I want to assign default menu modifiers to customers.
12. As an administrator I want to CRUD default ladle notices to customers to reflect wishes (e.g more meat less noodles).

13. As an administrator I want to print out/send weekly orders sheet for the customer so the customer can fill it out. Note: additionally empty forms are needed.
14. As an administrator I want to enter customers orders.
15. As an administrator I want to change customers orders (short-term).
16. As an administrator I want to CRUD menu modifiers (e.g lactose intolerance, big/small).
17. As an administrator I want to create temporary routes that are only valid on one day.
18. As an administrator I want to CRUD distribution notices on orders to reflect last minute changes to the order.
19. As an administrator I want to CRUD ladle notices on orders to reflect last minute changes to the order.
20. As an administrator I want to CRUD kitchen notices on orders, to reflect last minute changes to the order.
21. As an administrator I want to CRUD standard routes.
22. As an administrator I want to CRUD sets of routes.
23. As an administrator I want to print out/send daily plans (per route). Note: 1 Page per route.
24. As ladling personnel I want to see the scooping notices on daily plans.
25. As a Chef I want to see the kitchen notices on weekly plan.
26. As a distributor I want to see the distribution notices on daily plans.
27. As a ladling personnel I want to see the scooping notices on the weekly plans.
28. As an administrator I want to print out/send weekly plans that already reflect the routes.
29. As a Chef I want to see how many dishes per category need to be done each day on the weekly plan.
30. As a distributor I want to see the distribution notices on weekly plans.
31. As an administrator I want to print out/send billing information.
32. As an administrator I want to print out/send statistics (orders, customers, preferences).
33. As an administrator I want to change the prices for modifiers.
34. As an administrator I want to bill broken dishes. Note: additionally rentals of dishes.
35. As an administrator I want to CRUD course categories.
36. As an administrator I want to create dinner menus.
37. As a customer I want to order my order online.

8.3. Non-Functional Requirements

The NFRs were made as SMART as possible. Some requirements could not be made measurable (e.g space efficiency of a document) and are therefore only evaluated in a usability test.

8.3.1. Usability

Requirement	Verifying Condition	Outcome
Colours should be chosen such that a colourblind person (deuteranopia and tritanopia) could use it.	By Hand, critical colours are not used in the same mask to signal hierarchy or used for contrast (in usability test)	Passed
With the help of the manual, every administrator should be able to perform all UCs that are typically performed each week as described in "Rollende Küche Infos für Bachelorarbeit2021" and "Arbeitsablauf und Programm Doku" by Spitex.	Usability test	Passed
The application should be installable by following four (4) Steps: 1. Launch installer 2. Select install location 3. Select existing data or create new 4. Set/create or enter password	Usability test	Passed
The application and manual should be available in German.	Usability test	Passed
Wrong inputs should generate error messages.	Unit tests	Not Passed
Error messages on wrong inputs should include information for the user, so he knows what he did wrong or how he can solve the problem.	Usability test	Not Passed
All documents generated by the application should be compliant with the finance, kitchen and delivery actors.	Interview with the actors	Passed
All documents should have a layout that is space-efficient.	Usability test	Passed

8.3.2. Reliability

Requirement	Verifying Condition	outcome
A full data recovery should be possible with the backup data.	Usability test	Passed

8.3.3. Performance

Requirement	Verifying Condition	Outcome
The startup period of the application should not exceed three seconds (3 s).	Usability test	Passed
The transition between the masks should not exceed one second (1 s).	Usability test	Passed, except when saving all orders

8.3.4. Supportability

Requirement	Verifying Condition	Outcome
80% of the code should be unit tested. Excluded from this are views and generated code by frameworks.	Code analytics (Sonarcloud or JetBrains dotCover)	Not passed (Passed without fronted)
The CSC "remote database" should be achievable only through changes in the configuration file.	Architecture review	Passed

8.3.5. Security

Requirement	Verifying Condition	Outcome
The customer data should be encrypted.	Manual review	Not passed

9. Analysis

9.1. Domain Model

The domain model, shown in figure 9.1, shows the conceptual classes and their relations of the RoKü domain. The following sections discuss important concepts of the domain.

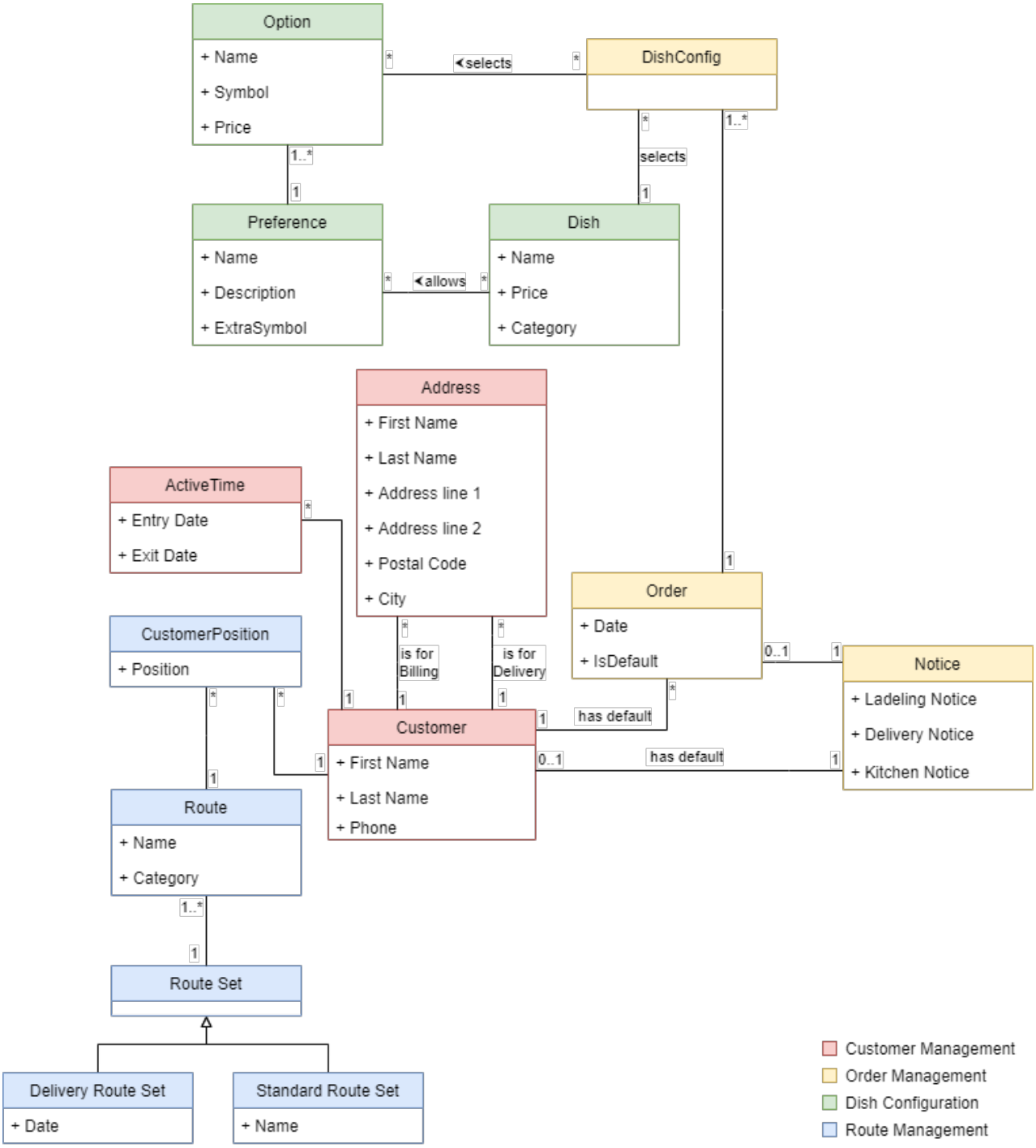


Figure 9.1.: Domain model of the business.

9.1.1. Route Planning

A delivery route contains a set of customers in a particular order. In order to effectively plan these routes, route set templates can be created and used when planing a delivery. These are called standard route sets. When planning a delivery, the delivery route set clones the routes and customer positions of a standard route set and removes all customer positions that reference a customer that does not have an order on the day of delivery. Since the routes and customer positions were cloned, the changes made on routes in a delivery route set do not affect the standard route sets.

9.1.2. Dishes

Currently, only menus can be ordered. With each menu the customer can choose from side dishes and desserts. So what the customer actually orders are several different dishes, with the restriction that one of those dishes is a menu. This is why menus, side dishes and desserts are all represented as a dish in the domain model.

9.1.3. Preferences

So far, the customer can order his menus in different sizes, can have a diabetes dessert and choose if he wants to have the soup of the day or a bouillon soup. These could all be represented as preferences. Not all dishes can be ordered in different sizes. Therefore depending on the dish, a preference may be chosen or not. A preference has options. In case of the size preference this could be small, medium and large. Each option may increase or decrease the price of the dish. The price of the option will be added to the price of the dish. If the price is negative, this will result in a reduction.

9.1.4. Orders

Each order consists of one or more dish configurations. A dish configuration combines one dish with zero or more options. Each order has its own notices. Every customer can have several default orders. It is possible to have a default order for every weekday. When creating an order for a customer with default orders, the default order for the specific weekday of the delivery can be cloned.

9.1.5. Customers

Customers can have two different addresses. One for the delivery and another for the bill. Not all customers are always active. Some may be inactive for a while due to vacation or because they are currently in the hospital. Some may just take a break from the RoKü and will eventually return. This is why customers can have several active times. An active time has an entry and exit date. If the exit date is not set, the customer is currently active. This allows administrators to check the history of customers.

9.2. Use Cases

The use cases were identified by working through the functional and non functional requirements. In figure 9.2 all use cases are shown. In this section only the most important use cases are discussed in detail.

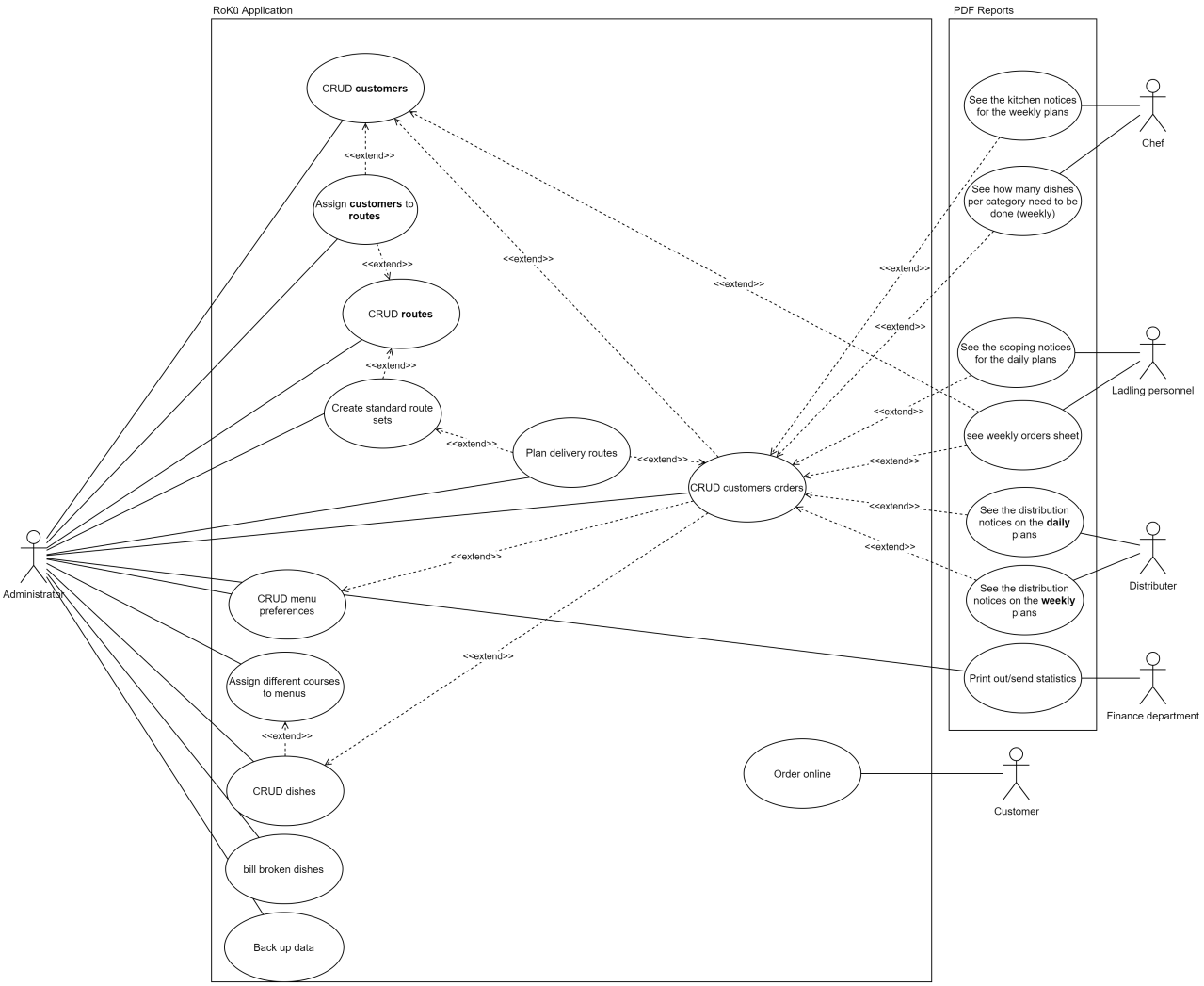


Figure 9.2.: Use case diagram.

9.2.1. Assign Defaults for Customers

In the same mask that the administrator uses to edit the customers, the customer preferences are assigned. Menu defaults can be chosen per weekday and dish. This means, that the administrator has a calendar-like overview of one week where the menu choices can be set. Additionally, all three (kitchen, ladling and distribution) notices can be edited as text. The preferences can be set by radio buttons. Preferences can be edited, created and deleted in another mask.

9.2.2. CRUD Customers Orders

The administrator enters the customers orders on a weekly basis. The process of entering one order consists of starting with the defaults and adapting them if necessary. Both menu and preference changes are on a daily basis. The administrator works through the list of customers in the same order as he/she gets the order forms in (which is typically in the order of the routes used on the same day). Already changed and saved orders can be edited later on.

9.2.3. CRUD Routes

Routes contain several customers in a specific order. The administrator must be able to change the order of customers within routes.

9.2.4. CRUD Standard Route Sets

A standard route set contains several routes. Typically all active customers are part of one route within a standard route set. The administrator can add and remove routes from the standard route set.

9.2.5. Plan Delivery Routes

The delivery routes are created from a default set of routes. All routes within the default set are cloned and customers without orders are automatically removed from the delivery routes. The administrator then can move customers to other routes and change the order of the customer within routes to optimize the routes. If necessary he also adds customers to a route that have orders but were not in the default set.

9.2.6. CRUD Menu Preferences

The menu preferences should allow a single dish to be further configured. The biggest contender right now would be the dish size. The administrator can then select, for each dish on each day, which size the dish should have. The modifiers can be enabled or disabled in a dish basis.

9.2.7. Create Route

This fully dressed use case is created with the template introduced in Software Engineering [19].

- Scope: RoKü
- Level: subfunction
- Primary Actor: Administrator
- Stakeholders and Interests: Administrator: Wants to create a default route, to have a template for delivery routes.

Preconditions

- The route set is already created.
- Customers are already created.

Success Guarantee

A new route has been created and saved with valid customers.

Main Success Scenario

1. The administrator navigates to ”routes”.
2. The administrator clicks on a default route set.
3. The administrator adds a new default route.

4. The administrator chooses the right customers.
5. The administrator orders the customers.

Extensions

1. The administrator creates a new set for the route.
2. The administrator creates an empty route to fill out later.

Special Requirements

All NFRs apply.

Technology and Data Variations List

-

Frequency of Occurrence

Initially for the setup, and when the customer base changes (estimated every month).

Miscellaneous

-

10. Design

Chapter 10 discusses technology independent design decisions and UI prototypes. Further architectural decisions and used technologies are discussed in chapter 11.

10.1. Application Type Evaluation

For the RoKü application two different approaches are suitable. A desktop application and a web application. First the advantages and disadvantages of both approaches are discussed. Then an evaluation is done.

10.1.1. Desktop Application

Once installed on a computer, there are no more costs to keep using the application. Since a desktop application does not need to communicate with components on other devices, an internet connection is not mandatory and the security risks are very low. Furthermore, the application is always available, unless the computer it is installed on is broken. The disadvantages of a desktop application are the scalability and the deployment.

10.1.2. Web Application

The web application shines with its scalability. The independence of a specific device is another benefit and updates are easily distributed. The availability of a web application should also be good. But if the server is down, there is nothing the user can do. Because the application is running on a server, it needs additional infrastructure or a suitable service provider, which generates costs. Since the client needs to reach the server over the internet, everyone can possibly access the server. This raises the need of an authentication service and a secured connection.

10.1.3. Evaluation

For Spitex the most important criteria is the cost and the security of their customer data. Since there are no plans to involve other Spitex associations, the scalability has little importance, and because the application is only used on one device, the deliverability is also not important. But since this project needs to replace an existing solution, it is relevant that the important features can be implemented in the limited time available for this project. The effort needed to implement the application is expected to be a lot higher for a web application because a web application needs to have additional security features.

Characteristic	Weight	Desktop	Web
Costs	10	5	4
Security	8	4	3
Availability	5	5	4
Workload	5	4	2
Deliverability	3	1	5
Scalability	1	1	5
Total		**131**	114

Table 10.1.: Evaluation of type of application.

In table 10.1 the full evaluation with the weighted criteria can be seen. Since the desktop application has a higher score than the web application, it was decided to create a desktop application.

10.2. Architecture

10.2.1. Deployment

The old application runs on a single machine, and so does the new. The resulting deployment diagram, which is shown in figure 10.1, is therefore very simple.

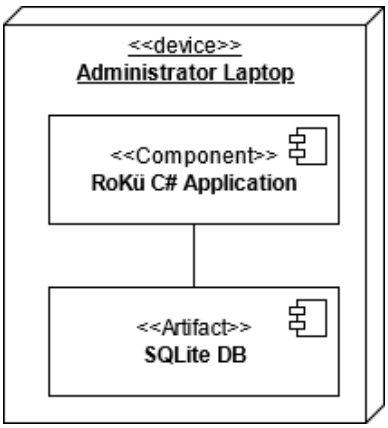


Figure 10.1.: Deployment diagram.

10.2.2. Layers

To keep the business layer independent from all other layers, the onion architecture is used. The onion architecture arranges the layers as shown in figure 10.2. In an onion architecture only dependencies from an outer layer to an inner layer are allowed. This architecture approach leads to more maintainable applications and relies on the dependency inversion principle [20].

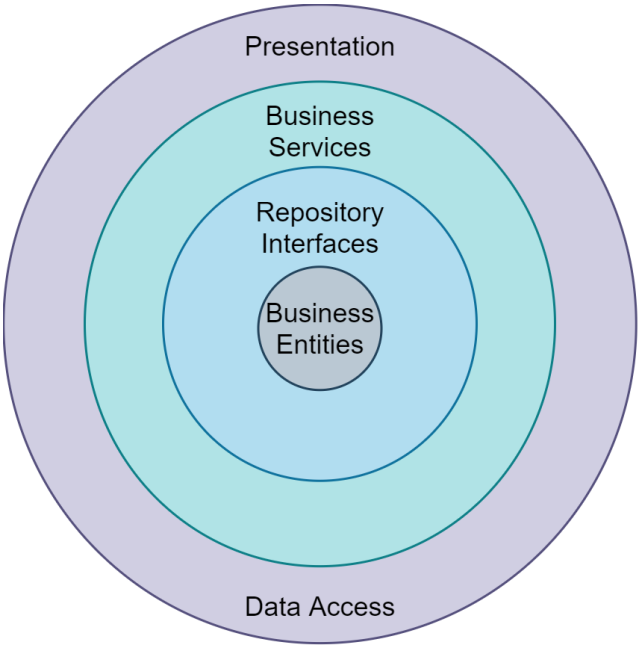


Figure 10.2.: Onion layer diagram.

The application is divided into presentation layer, business layer and data access layer. The presentation layer includes all code needed for the visual presentation of the application. The business layer has an additional three sub-layers. The business services, the repository interfaces and the business entities. The data access layer implements the interfaces in the repository interfaces. The implementations can then be injected into the business services using dependency injection. This makes the business layer completely independent from both the presentation layer and the data access layer, which makes it easier to exchange these layers in the future.

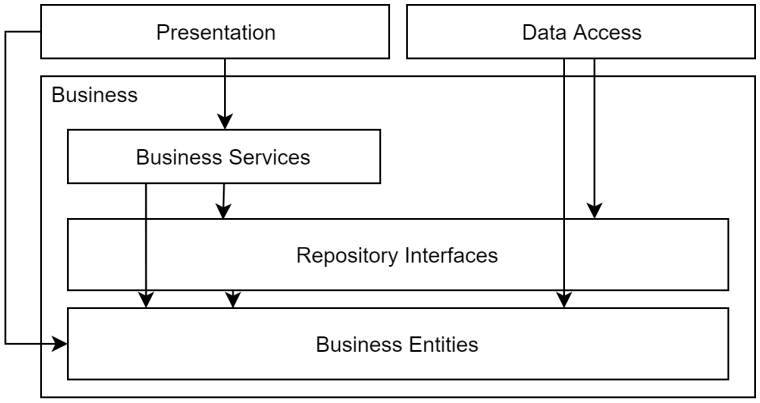


Figure 10.3.: Layer diagram.

10.3. User Interfaces

The first UI suggestion was made in Figma. The reception was good, and the discussion brought new insights.

The Navigation was held minimalistic, as were all lists, see figure 10.4.

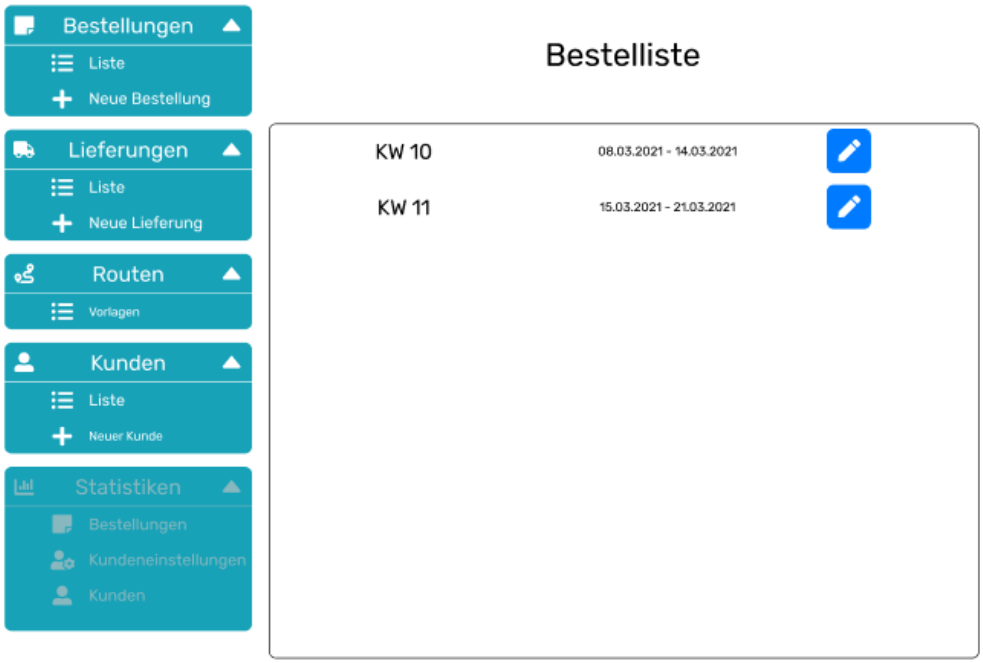
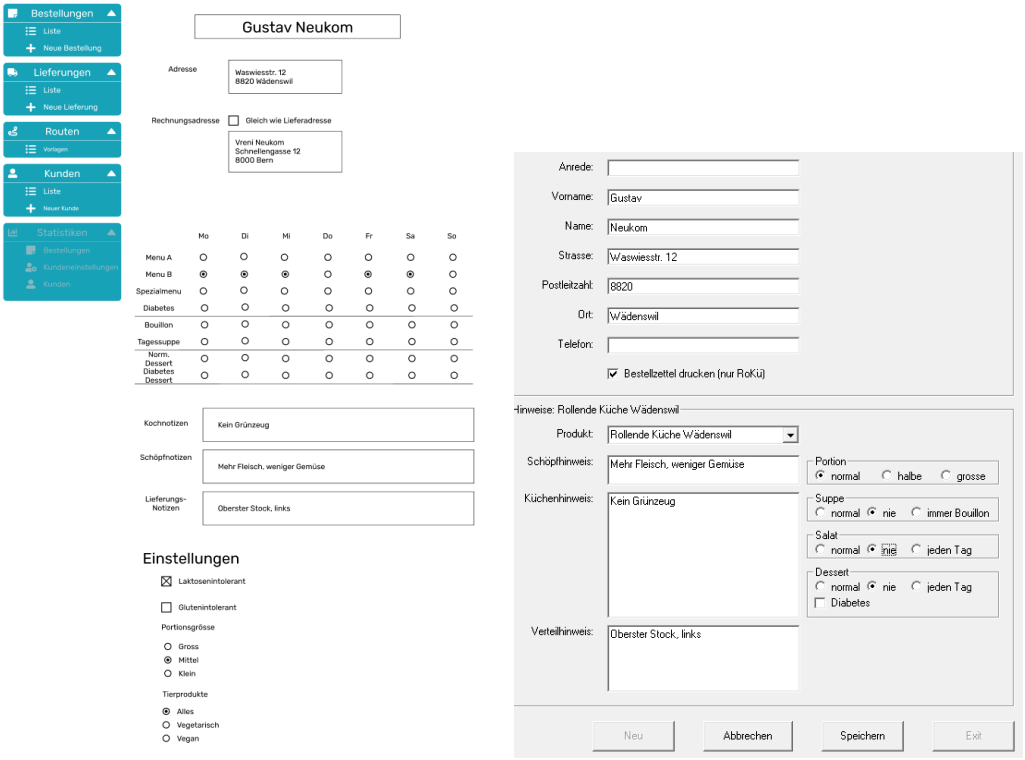


Figure 10.4.: Prototype screen: List of orders.

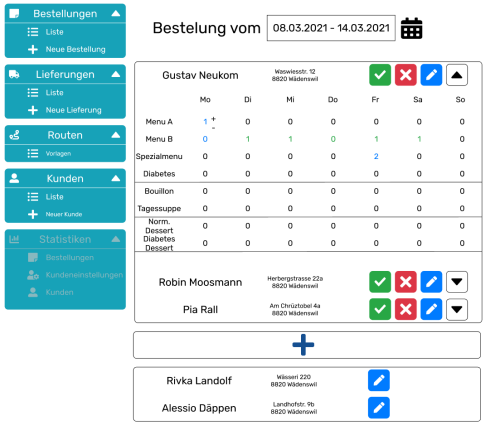
The customer input screen was an opportunity to show the advantage of the more abstract model. All data in figure 10.5a is the same as in figure 10.5b. New additions are that settings can be created by the administrator and are not locked by the application. Additionally, default orders can be created, which can be reused when creating the real orders.



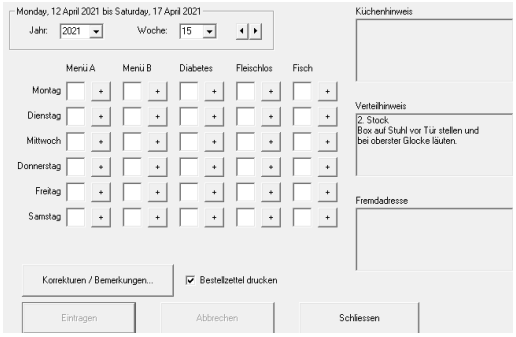
(a) Prototype. (b) Old.

Figure 10.5.: Screens: Customer input.

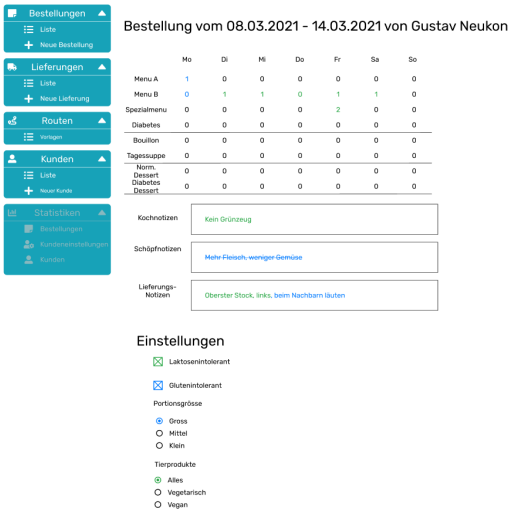
In the order input screen, seen in figure 10.6a, it is possible to assign options. This triggered a discussion on which basis such options should be assignable. Technically, a customer should be able to order the same menu twice on the same date, where both have different options. Therefore it was decided, that options are assignable per one dish selection. A similar discussion was held for the notices. In the old application, the notice was set for the whole week, but for the administrator it would make sense that the notices can be edited on a daily basis.



(a) Prototype.



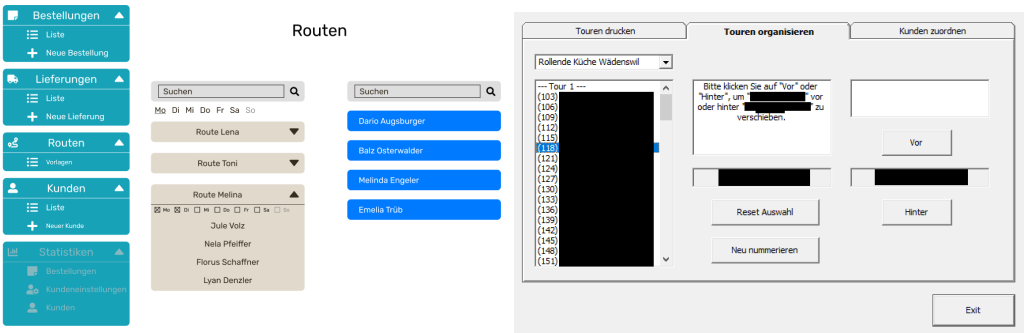
(b) Old.



(c) New - Details.

Figure 10.6.: Screens: Order input.

In the old application only one set of three predefined routes could be created (see figure 10.7b). In the new application, this should not be limited. It was intended to be able to create several route templates (see figure 10.7a), which can be used when planning a delivery. These route templates can be assigned to a weekday. The idea was that the customers regularly order on specific weekdays and it therefore would be nice to have different route templates on different days. The administrator stated that this idea is not correct, which is why it was decided that instead of route templates for specific days, sets of route templates can be created and named by the administrator.



(a) New. (b) Old.

Figure 10.7.: Screens: Route planning/overview.

Planning a delivery, means adjusting the routes to the current orders. In the old application this happened in MS Access. In the new application this will happen in the app itself. The administrator should be warned about conflicts and orders that would not be delivered. See figure 10.8.

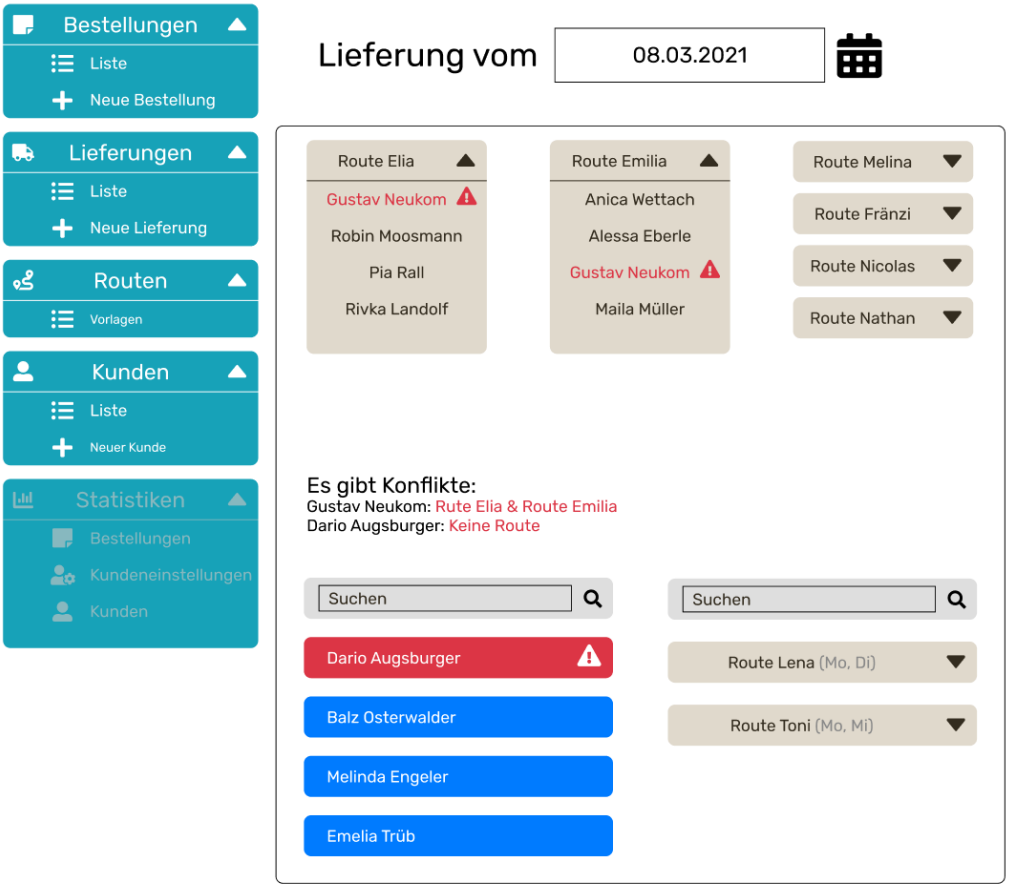


Figure 10.8.: Prototype screen: delivery planning.

11. Implementation

11.1. Technologies

Since the technology stack was not fixed, it was decided to go with the technology the project members have most experience with, in order to increase development speed. In this particular case that is C# with WPF as UI framework. Because WPF is not supported on operating systems other than Windows, this decision adds a limitation to the application. However, this decision does not affect Spitex since they use Windows computers.

11.1.1. Persistence

It was decided, that SQLite will be used to persist data. SQLite has several advantages compared to other options. First, it saves the data in a single file. This is important because the administrator wants to be able to create data backups on a memory stick. With SQLite they can simply copy the database file to create a backup without the need of additional functionality within the application. This keeps the project scope small and lets the developers focus on other features. Another reason SQLite is a good option, is that it does not need any additional database service. This removes a possible error source, which is important in an environment without any technical support on stand by. The fact that SQLite does not need additional services and is saved in a single file also makes testing easier. Additionally, SQLite is free to use and widely used. It is therefore fair to assume that it is stable and reliable. NoSQL databases were not included because the application does benefit from its advantages.

Object-relational mapping

It was decided to use EF Core as Object-relational mapping (ORM) framework. EF Core is a very powerful ORM framework that supports SQLite, is very widely used, and has a detailed documentation. In this project the approach "code first" is used, which means the database schema is created by EF Core according to the configurations made in the code.

11.2. Projects

The package diagram in figure 11.8 shows the projects and their dependencies [7]. The Launcher project is the actual application. All other projects are class libraries. The Launcher project bootstraps the whole application. The Presentation project contains all views and view models, as well as all styles, resources needed in the UI, dialogs, converters and other UI specific code. The presentation should only use services provided by the BusinessServices project. The BusinessServices include the whole domain logic of the application. It uses the interfaces in the DataAccessInterfaces to create PDF reports and access the data stored in the database. The ReportGenerator implements the interfaces within the DataAccessInterfaces to create PDF reports. Similar to the ReportGenerator, the DataAccessLayer implements interfaces within the DataAccessInterfaces. It implements all interfaces needed to access data stored in the database.

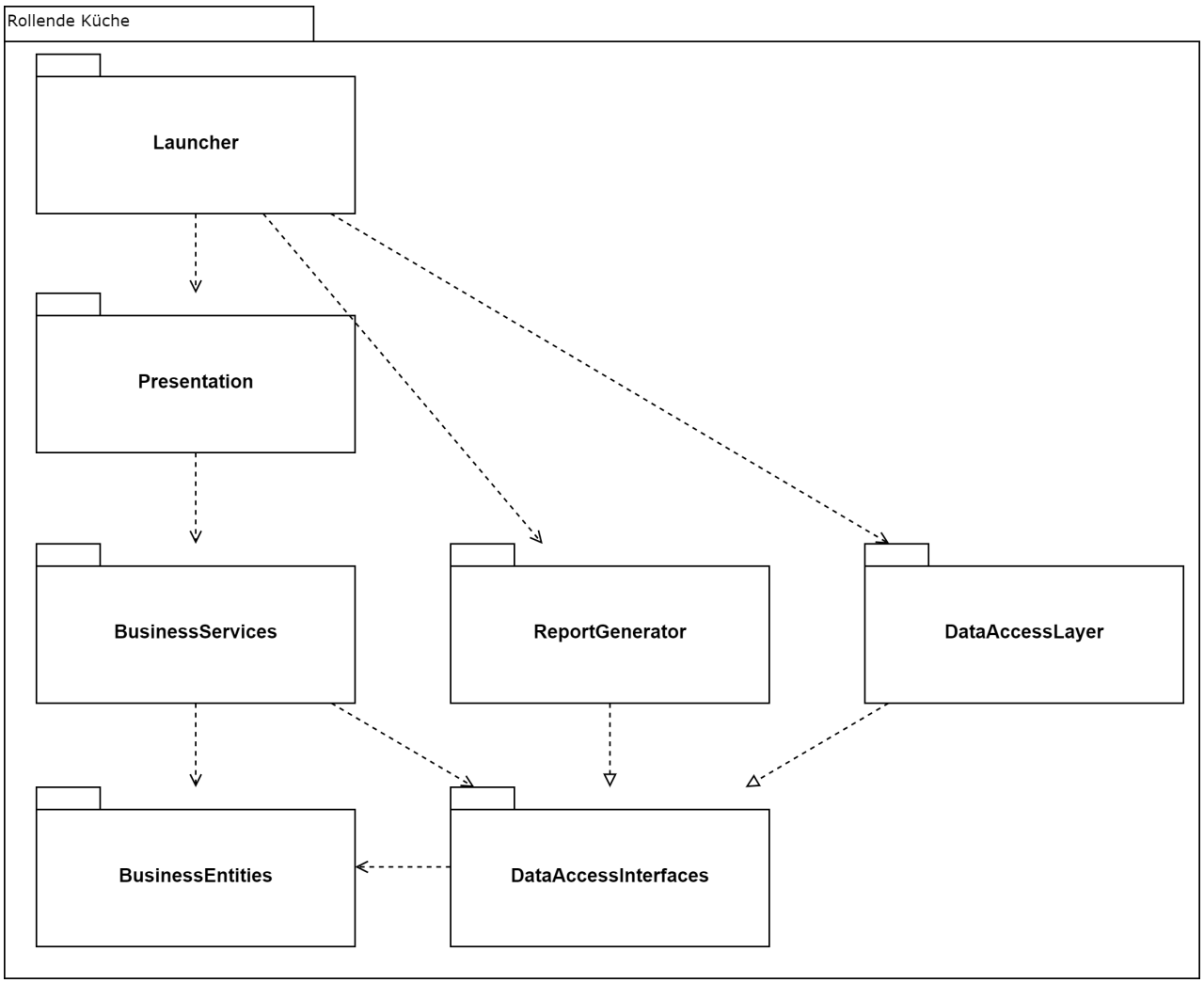


Figure 11.1.: Package diagram.

11.3. Concepts

In this section the most important concepts of the application are explained. The diagrams shown do not necessarily correspond to the implementation. Some methods or properties that are not important to understand the concept were not included in the diagrams.

11.3.1. Dependency Injection

The RoKü application uses dependency injection to decouple components and simplify testing. Autofac is used as an inversion of control container. With an additional extension it can be used for dependency injection. Autofac introduces modules where components can be registered on a ContainerBuilder. Every project in the solution provides a module. The module in the launcher project registers the modules of all other projects. At the start up of the application, the module located in the launcher project is registered and the Autofac container is created. The MainWindowViewModel, which was registered by the presentation module, will be resolved on project start up. Since all other component calls originate from the MainWindowViewModel, the container can handle the dependency injection of the whole application.

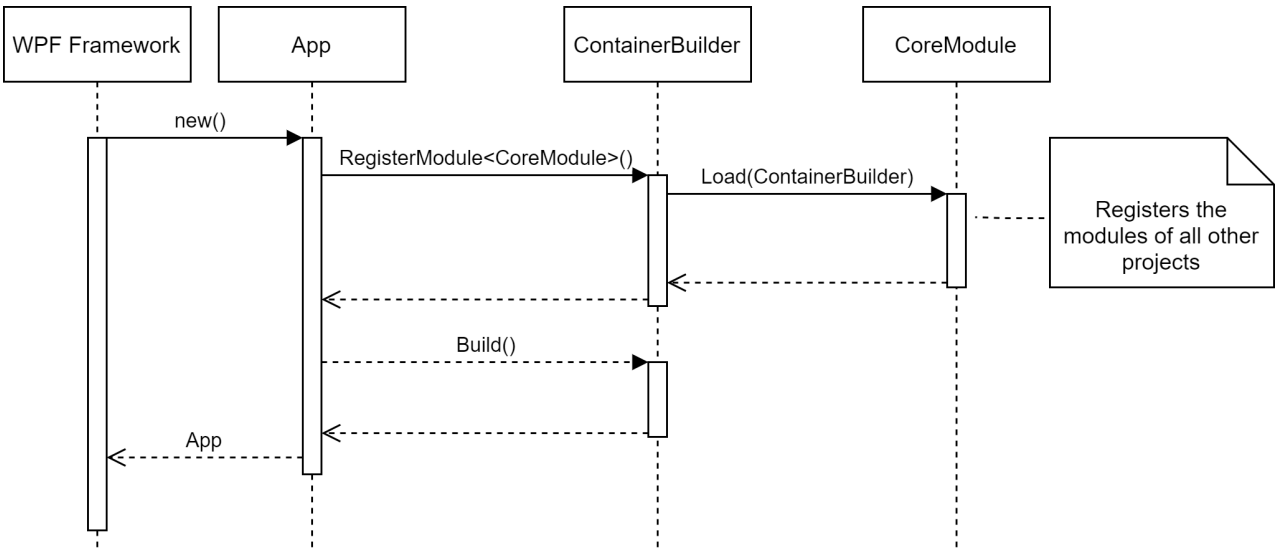


Figure 11.2.: Registering components to the Autofac container.

Components registered with Autofac can have different life time scopes, that need to be defined when registering a component. Most components are registered as "instance per dependency". This means, that Autofac will create a new instance of the demanded type every time the type gets resolved. But the MainWindow, MainWindowViewModel and the ScreenContainer are registered as "single instance". As the name already suggests, every time someone needs one of those types, Autofac will return the same instance. Single instance is used so the application does not open a new Window every time the screen changes.

11.3.2. MVVM

The Model-View-ViewModel (MVVM) pattern is used to write more reusable code. The MVVM pattern has three core components: the model, the view model and the view. The model is just an object that holds data. In the RoKü application, these are normal business entities. The view model implements properties and commands to which the view can bind to. The view is responsible for defining the appearance. This is done in XAML. Views should have very little to no code behind [5].

It was decided to use the mvvm light toolkit. It implements important MVVM concepts. Additionally, an AsyncRelayCommand was implemented. The AsyncRelayCommand stores a Func<Task> which will be executed when the command is executed by the WPF framework. This was necessary to make the view models testable. Without the AsyncRelayCommand a normal RelayCommand provided by the mvvm light toolkit would be used. This forces the method used as execute to be an async void method. When testing view models, this would be a big problem, because the test can not await the async void method.

11.3.3. Displaying Screens

The application has only one window, the MainWindow. The MainWindow has two sections. A section for the navigation and a section for the current screen. The data context of the MainWindow is set to the MainWindowViewModel at the start of the application. The MainWindowViewModel includes an instance of the interface IScreenContainer, which has a property Screen from the type ScreenBase. The MainWindow binds the section for the current screen to the Screen property from the screen container within the MainWindowViewModel. All screens therefore need to be derived from ScreenBase. Additionally, a screen needs a visual representation. This can be done with a data

template. A data template is a component of the WPF framework that can be used to provide WPF with information on how to represent certain types [17]. This data template needs to be created within a resource dictionary. The resource dictionary then needs to be included in the resource dictionary Views.xaml within the presentation project. The Views.xaml is registered in the app.xaml and makes sure, that WPF finds the data templates.

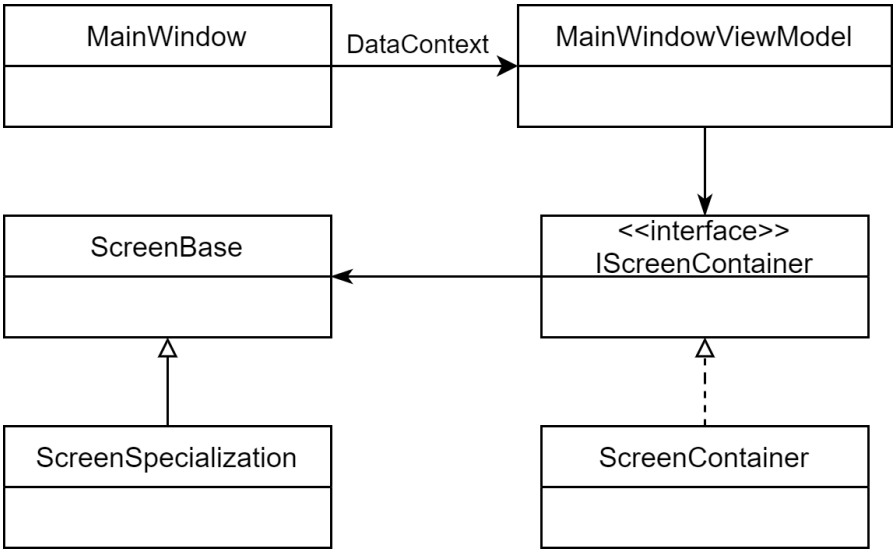


Figure 11.3.: Screen display class diagram.

11.3.4. Navigation

The navigator is responsible for the navigation within the application. The navigator is an implementation of the INavigator interface and provides a generic method NavigateTo with one type parameter. The type parameter needs to be a type that derives from ScreenBase because the Navigator simply resolves an instance from the Autofac container and assigns it to the Screen property of the screen container.

Since some Screens need some sort of initialization, the ScreenBase class has a virtual method called Initialize, that can be overwritten. The navigator calls the Initialize method before setting the Screen property. This makes the transition between screens smoother.

If a screen needs to get parameters from its caller, it can implement a generic IIntializable interface, which includes an overload of the Initialize method with a parameter. The type of this parameter is the same as the generic type parameter of the IInizializable interface, which needs to be an implementation of the IInizializeContainer interface. Developers can then use the NavigateTo overload on the navigator, that has a parameter. The type of the parameter needs to be specified in form of a type parameter, and also needs to be an implementation of the IInizializeContainer interface. The navigator then calls both Initialize overloads before setting the Screen property on the screen container.

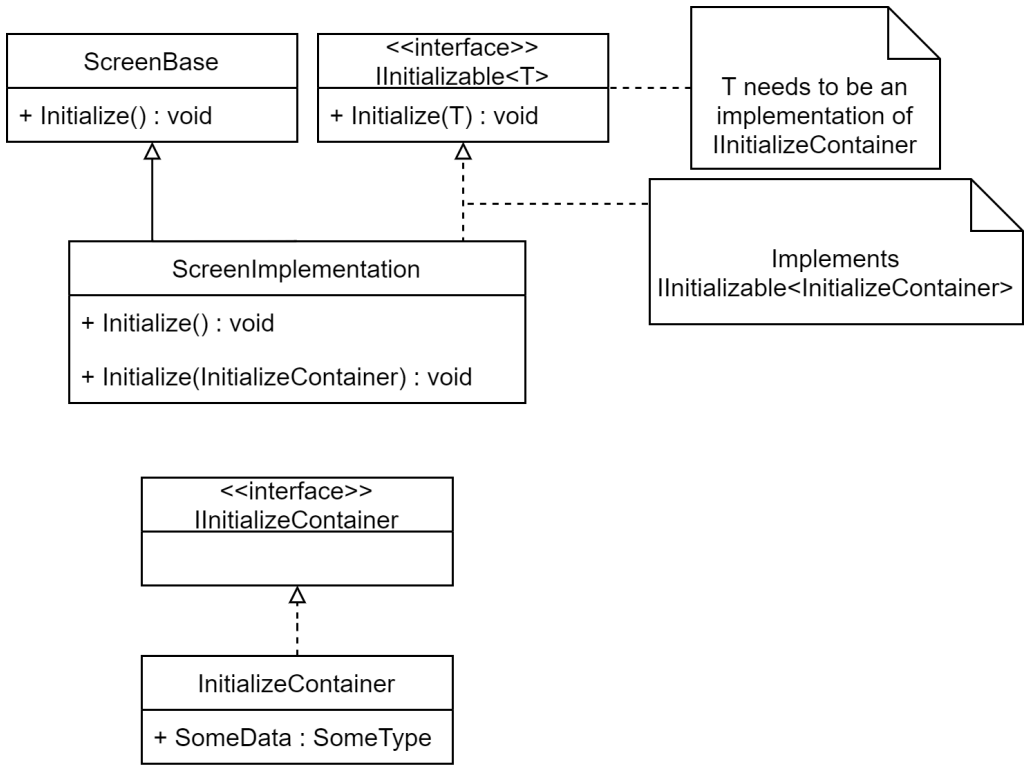


Figure 11.4.: Example of a screen that implements IInitializable.

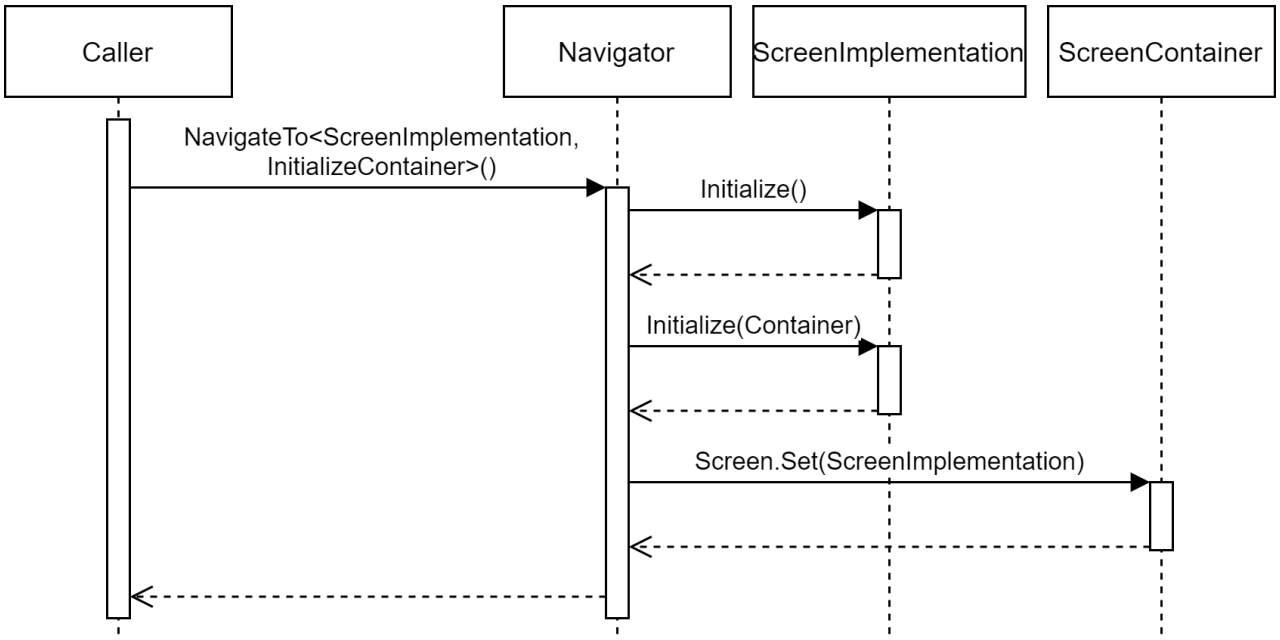


Figure 11.5.: Exemplary navigation to a screen that implements IInitializable.

11.3.5. Dialogs

The screen container introduced in section 11.3.3 has a dialog property beside its screen property. The MainWindow binds this property just like it does with the screen property, but only shows the content if the dialog property is not null. If the dialog property is set, the dialog is shown over the current screen and the rest of the screen is disabled as long as the dialog is shown. Similar to the screens, a dialog needs to have a visual representation in form of a data template that is registered in the Views.xaml. The dialog property has the type DialogBaseViewModel, which is an abstract class. It has an event called CloseRequest and a method RequestClose that invokes that event.

The navigator has a method ShowDialog, which takes a DialogBaseViewModel as parameter. The navigator sets the dialog property of the screen container and creates a TaskCompletionSource. TaskCompletionSource is a class provided by Microsoft that ends when someone calls the SetResult method. The navigator subscribes to the CloseRequest event of the dialog with a method, that sets the dialog property on the screen container to null and calls the SetResult method of the TaskCompletionSource. After subscribing to the event, the navigator returns the task of the TaskCompletionSource. This allows callers of ShowDialog to wait for the dialog to be closed before continuing.

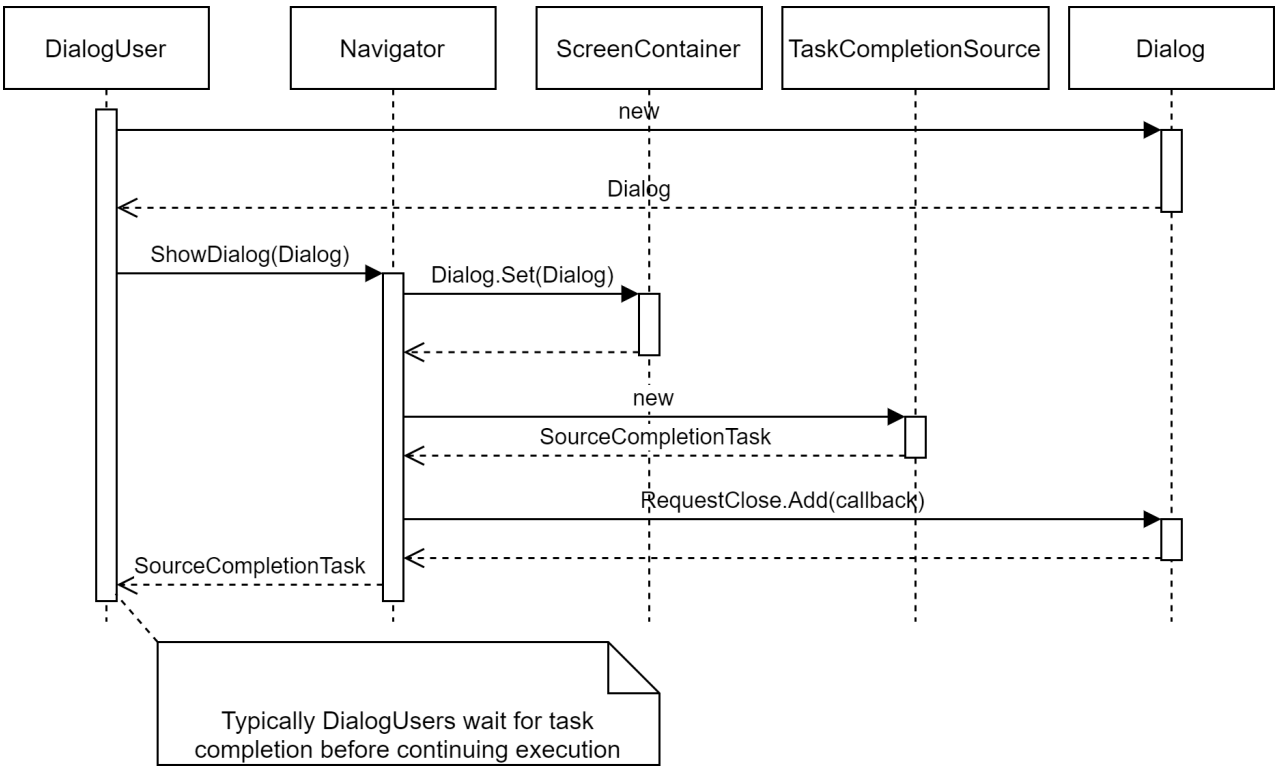


Figure 11.6.: Showing a new dialog.

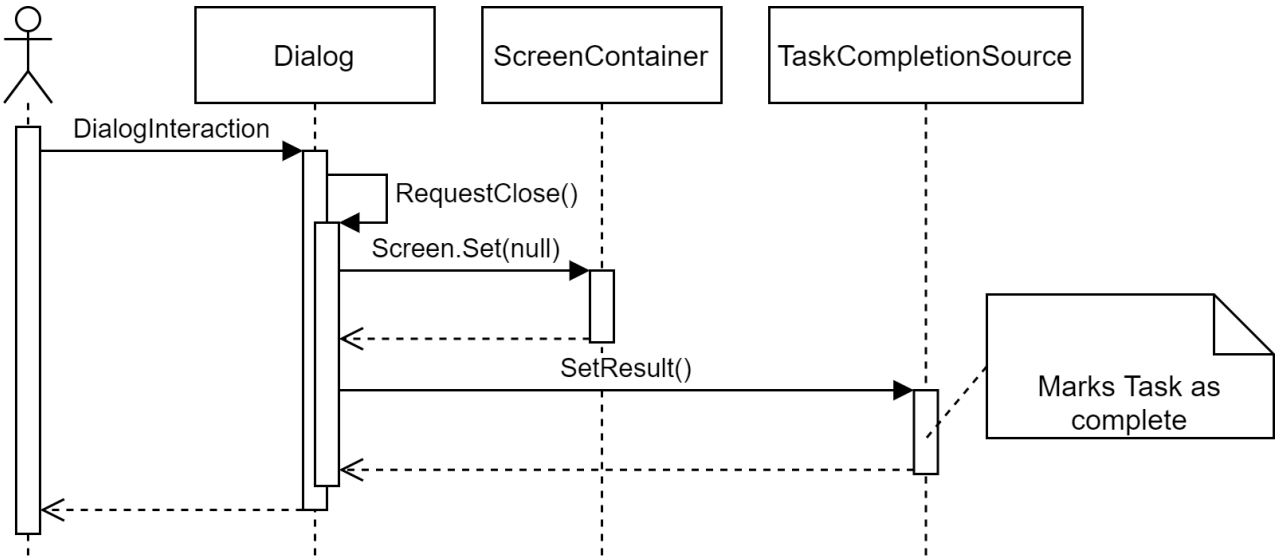


Figure 11.7.: Closing a dialog.

11.3.6. Repositories

The use of the repository pattern is necessary to make the business layer independent from EF Core. When using the repository pattern as intended, the repositories act like a collection-like interface [18]. Changes made to the repository would have no effect on the database. Only in combination with another pattern, such as the unit of work pattern, the changes made to the repository would be saved to the database.

This implementation of the repository pattern would require tracking of the repository to detect and save changes. But since the business layer and the data access layer implement different classes for the same conceptual class, such tracking would be hard to implement. An attempt was made to implement the repository in combination with the unit of work pattern, but then this idea was abandoned because the benefit would not justify the time needed for the implementation.

It was instead decided to make a compromise, and so repositories implement CRUD functionality and changes are written directly to the database [15]. As a consequence of this decision, it became necessary that the repositories don't save changes in navigation properties, which makes the services more complex. The problem with the navigation properties lies in the fact that they are not tracked. If an object with a navigation property gets updated and the navigation property is not tracked, EF Core will try to create a new one, which will fail because the ID is already set.

A generic IRepository interface handles the most common operations. If additional queries are needed, an interface that specializes the IRepository interface can be added to the DataAccessInterfaces project.

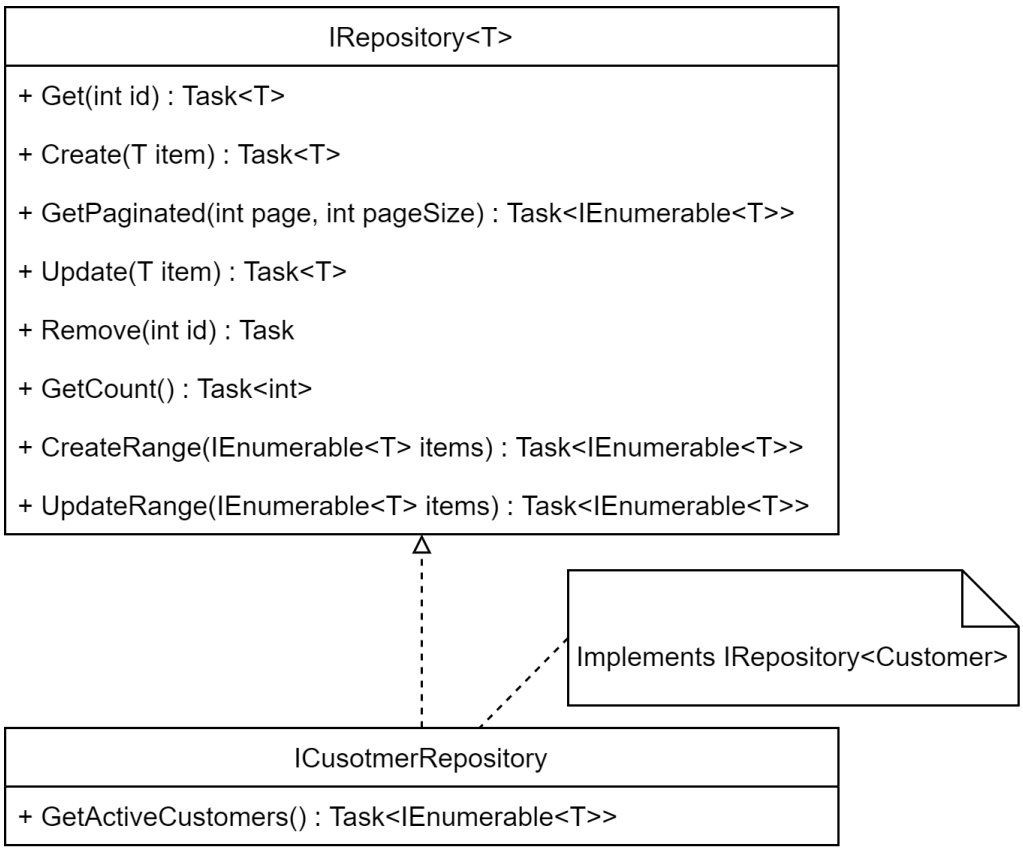


Figure 11.8.: The IRepository interface.

The implementations of the repository interface are located in the DataAccessLayer project. The generic RepositoryBase implements the IRepository. It uses the DatabaseContext to access the data in the database.

For each business entity a repository needs to be registered in the Autofac module of the DataAccessLayer. This is done in one of four ways:

1. If the entity has no references to other entities and no specific repository interface for that type is defined, the RepositoryBase can be registered as IRepository for the entity type.

2. If the entity has references to other entities but no specific repository interface for that type is defined, a specialization of RepositoryBase that overrides the GetImports method needs to be registered as IRepository for the entity type.

3. If the entity has no references to other entities but a specific repository interface for that type is defined, a specialization of RepositoryBase that implements the specific repository interface needs to be registered as that specific repository interface.

4. If the entity has references to other entities and a specific repository interface for that type is defined, a specialization of the RepositoryBase that implements the specific repository interface and overrides the GetImports method needs to be registered as that specific repository interface.

The GetImports method, mentioned above, is a virtual method of the RepositoryBase, that gets a queryable as a parameter and can be overwritten to eagerly load navigation properties.

11.3.7. Entity Mapping

Since there are a lot of similar classes in different layers, such as the Customer and the CustomerEnity classes, it is necessary to properly map objects from one type to another and back. In order to reduce development effort, it was decided to use AutoMapper. AutoMapper is a powerful library that provides functionality to automatically detect similarities between two types and map objects from one type to another. To be able to map an object to another type, the mapping needs to be configured first. This is done in one of two places depending on the type pair. If one type is from the DataAccessLayer, it needs to be done in the DataAccessAutoMapperConfigurationProfile. If one type is from the Presentation project, it needs to be done in the PresentationAutoMapperConfigurationProfile. The mapper can be resolved from the Autofac container. The mapper is registered to the Autofac in the AutoMapperModule located in the Launcher.

11.3.8. Generating PDF Reports

The PdfReportGenerator class implements the IReportGenerator interface. It provides all functions to generate the PDFs. All functions take a file url as the first argument and a form data object or a list of form data objects as second parameter. These methods then call the respective form generator with the same arguments.

The respective form generators, such as the BillingReport class, are subclasses of StandardPdf, which provides them with the graphics, some drawing helper functions, an advancement in pages and a function to add page numbers at the end.

The drawing is handled in a PdfHelper class that abstracts the PDF creation even further, allowing the caller to change the style. With the style, several properties can be adjusted, like which pen is used, which brush is used and where the anchor point lies. For an anchor point visualization, a test page can be printed. A sample of the test page is seen in figure 11.9.

The Package "PdfSharp" was used to generate the PDFs.

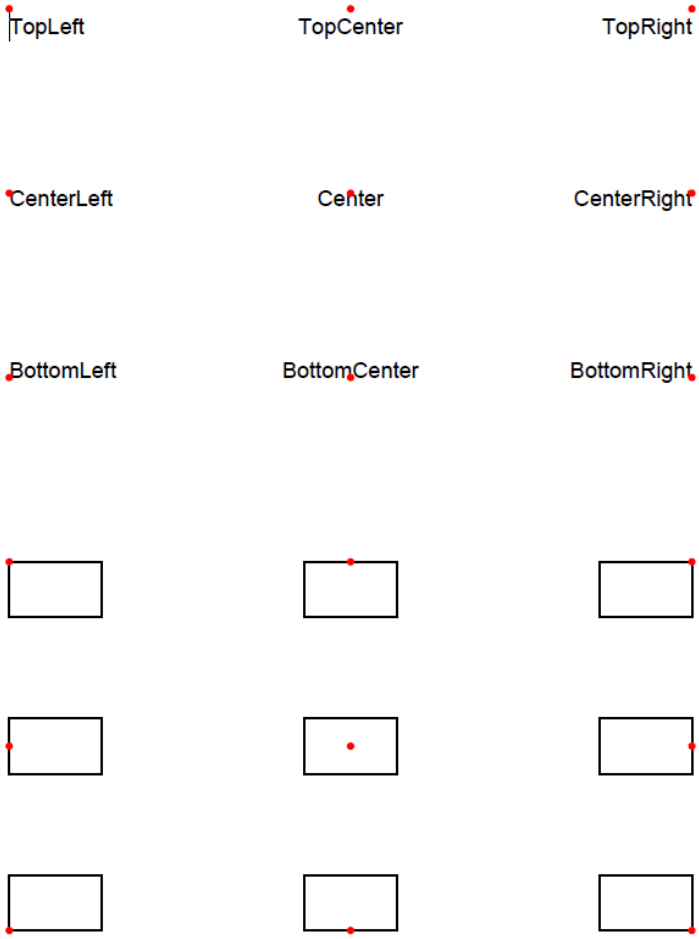


Figure 11.9.: Test page showing anchor points.

The form generators work sequential, from top to bottom, where a top cursor is advanced between the different sections of the page.

11.4. Important Classes

In this section some important classes are discussed that are not covered in the section 11.3.

11.4.1. App

The App class is the entry point for all WPF applications. It is a partial class, where one part is defined as regular C# code and the other part gets generated by WPF from the App.xaml. Within the App.xaml, the XAML resources of the whole application are merged and the Startup property is set. The Startup property defines what method WPF should call when the framework is ready. In the case of the RoKü application this method is the OnStartup method. In the constructor of the App class, the Autofac container is created. Also in the OnStartup method, the initialization of the MainWinow happens. First the MainWindow gets resolved from the Autofac container and shown. Next, the database context is resolved to make sure, that the database is created and finally data context of the main window is set to a instance of the MainWindowVieModel class that also gets resolved from the Autofac container. The start up is visualized in figure 11.10.

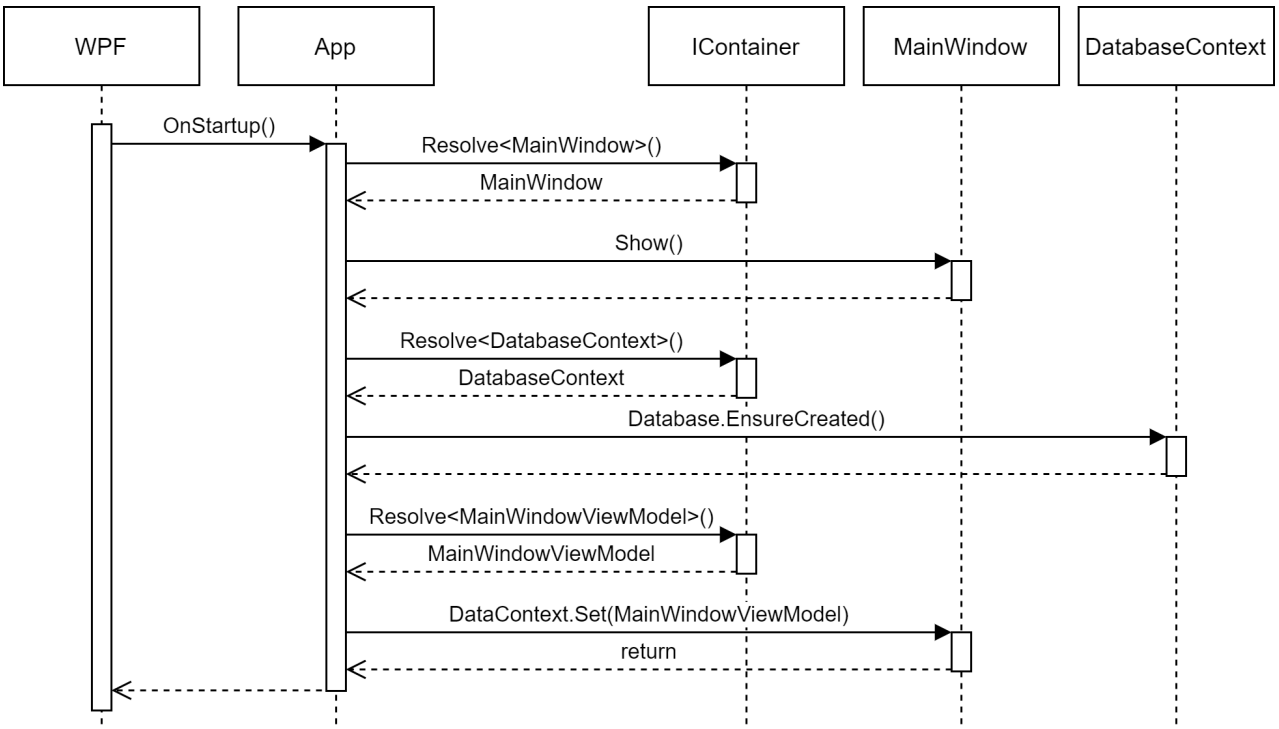


Figure 11.10.: OnStartup method as sequence diagram.

11.4.2. DatabaseContext

The DatabaseContext derives from DbContext, which is a class from EF Core. The DbContext represents a session to database that can be used to query and save data [3]. The DatabaseContext overrides the OnModelCreate method of the base type DbContext to configure the database schema, and has a DbSet for each business entity type. A DbSet represents the collection of all entities in the context of a given type [4].

11.4.3. Menu Selection Table

The menu selection table is an integral view model. It is used by the customer input screen and the order input screen. The table itself has one menu selection row per day. Each row has one menu selection cell per dish. The selected cell appears on the right side of the screen. The differentiation between the menu selection table within the customers for the default orders and the table for the orders for a specific week was solved by not showing fields that are null. Within the customer input screen the notices are null and therefore can't be seen. When creating orders for a specific week or editing orders in the order input screen however, the notices are not null and can be edited on the right side when a cell is selected. To get all orders that were entered, the menu selection table has a function called "GetCollectedOrders" that collects all orders from its rows, which in turn collect all orders from their cells.

11.5. Testing

11.5.1. Test Projects

In addition to the projects introduced in section 11.2, some projects for testing are necessary. Every layer has one test project to run automated tests. An additional test utilities project contains code

that can be reused in all test projects. Since most tests need some data to work with, a test data project was added. The test data project contains test data defined by the developers as well as a class, that creates customers with randomized values. The test data provider project is a console application, that prepares a fresh database with a certain amount of randomized customers. This is useful because in most scenarios it is necessary to have some customers already in the database when trying out new features.

11.5.2. Presentation Layer Testing

Tests in the presentation layer are by far the most time consuming to write because in the presentation layer unit and integration tests need to be made.

For each component under test, three classes need to be created. The tests are written within an abstract base class that derives from the TestBase class. The TestBase initializes the Autofac container. The module of the presentation and the AutoMapper are registered by default. An additional abstract AddRegistrations method lets subclasses add other services needed.

A test class for the unit tests and a test class for the integration tests derive from that abstract base class. Within those derivations the method AddRegistrations needs to be overridden. Since the idea of integration tests is to test the service in collaboration with the other services, the services need to be registered just as it is done when running the application. This means that the module of the business layer and the module of the data access layer have to be registered. Unit tests are written to only test the component under test. Therefore, the services the component needs while testing need to be mocked and the mocks are registered to the Autofac container.

Since the subclasses need to be initialized differently for unit and integration test, the TestBase class also includes an abstract Initialize method. This is where the database is set up in the integration test and the mocks are set up in the unit test.

When writing the tests in the base class, it can occur that some parts of the test need to be handled differently for the integration test and the unit test. In such cases, this part gets extracted into an abstract method.

An example of a class structure for unit and integration tests is visualized in figure 11.11.

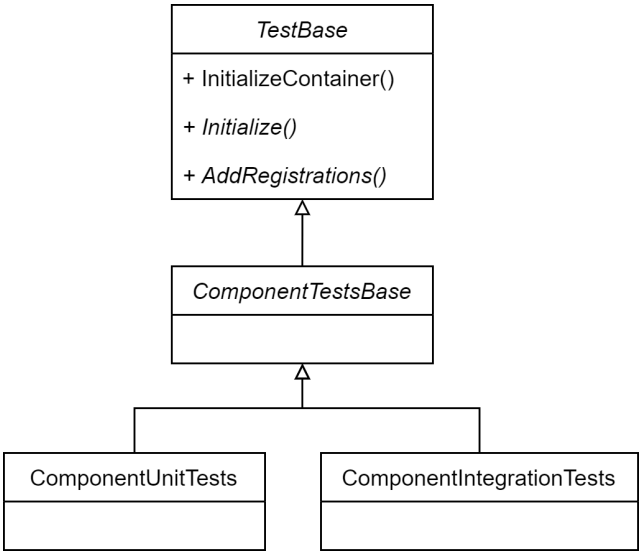


Figure 11.11.: Simplified class diagram of test structure.

11.5.3. Business Layer Testing

Testing in the business layer is way easier than in the presentation layer. Every test derives from the test base class. The test base class is not the same as the one used in the presentation tests. It creates the Autofac container and registers the module of the business layer and the AutoMapper per default. The virtual method AddRegistrations allows subclasses to add additional services to the container. Only unit tests are written in the business layer, which is why only mocks should be added as registrations.

11.5.4. Data Access Layer Testing

The test base in the data access layer creates the Autofac container and registers the module of the data access layer and the AutoMapper. An additional base class for tests that uses the database was added. This ensures that the database gets created and deleted properly.

11.5.5. Mocking

The mocking library Moq is used to mock dependencies. It is very easy to use and offers a lot of possibilities to setup mocks and verify method calls.

11.6. User Interfaces

The first thing that is obvious when comparing the UI prototype and the finished product is that the finished product is less colorful and more simplistic. Additionally, all the UI components are more consistent to aid the ability to navigate. The application implements the fluent design guidelines where appropriate. The navigation of the application can be seen in figure 11.12 as an example for simplistic the design.

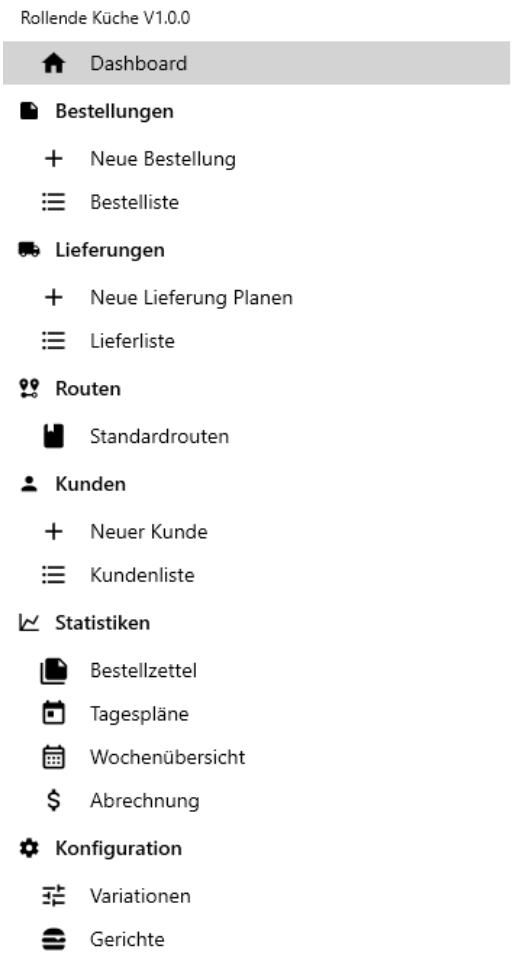


Figure 11.12.: Navigation.

The customer input screen is packed with information. The menu selection table was overhauled to allow multiple dishes on one day, and the options for the dishes were put on the right side of the screen, see figure 11.13.

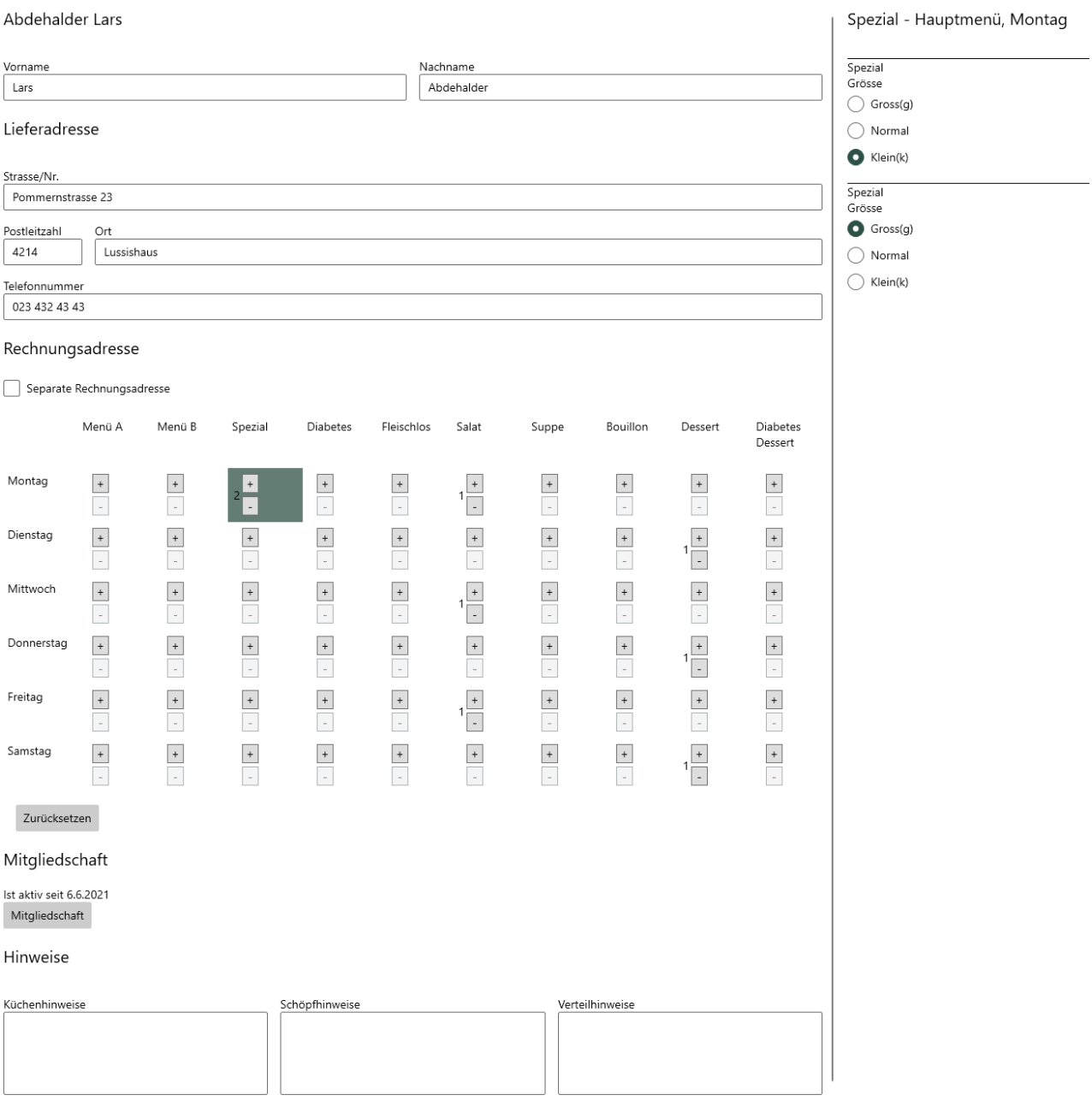


Figure 11.13.: New Screen: customer input.

The order input screen, seen in figure 11.14, went through similar changes. The menu selection table is on the left, and the options and notices on the right side of the screen.

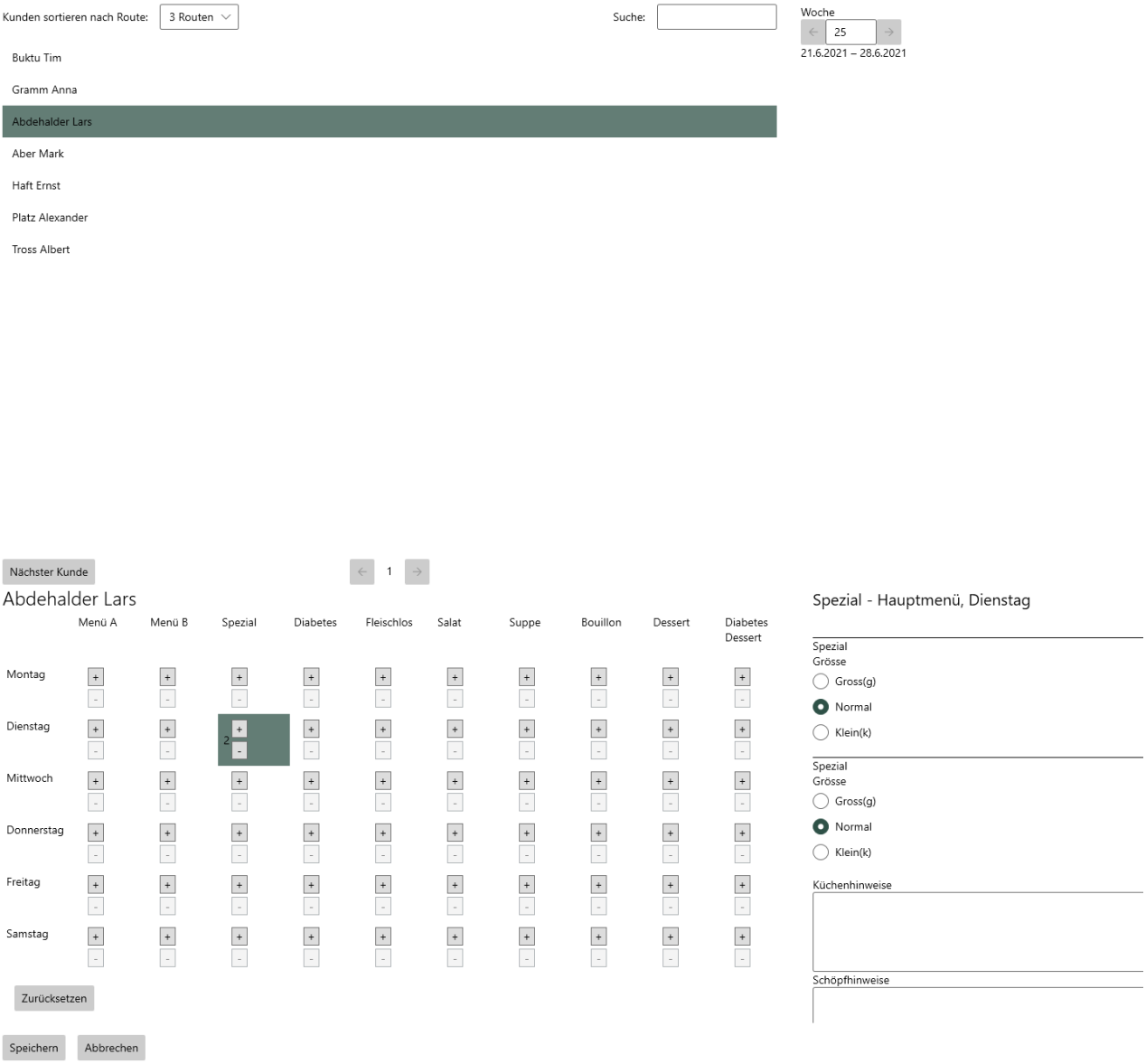


Figure 11.14.: New Screen: order input.

The route planning screens were more drastically changed. First, the list of routes, and the route with the list of customers, were separated into two screens. The list of delivery routes, shown in figure 11.15a, and a single delivery route, shown in figure 11.15b. The default routes and delivery routes are the same, except for the warning on the right when a delivery is planned but not every customer that has an order is in a delivery route.

(a) New screen: List of routes.

(b) New screen: Single route.

Figure 11.15.: New screens: Routes.

12. Quality Assurance

This chapter covers the tools and practices that were applied to ensure the quality of the product.

12.1. Unit and Integration Tests

In all layers unit test are written. All dependencies of a service under test are mocked. This can easily be achieved thanks to dependency injection. Additionally, integration tests are written in the presentation layer to make sure all components work together properly. Since SQLite is used as persistence solution, it is not even necessary to mock the data access, which makes the integration tests more true to production.

The testing framework used is MsTest. It is easy to use and is installed and fully supported by Visual Studio 2019. Moq comes into play to mock dependencies.

The goal is, to have at least 80% test coverage over the whole solution. JetBrains dotCover is used to measure the code coverage. Excluded from the test coverage are the following projects:

- All test projects
- The project "TestDataProvider"
- The project "TestData"
- The project "TestUtilities"

When it became clear that the scope of the project was to big for the given time, it was decided that no more presentation tests shall be written, to free time to work on functionality. This decision is reasonable because the presentation does not hold any logic and tests are very time consuming to write. The additional profit of those tests do not justify the time needed when confronted with such a tight time schedule.

12.2. Sonarcloud

Additionally to the unit and integration tests, a Sonarcloud project was setup. Sonarcloud analyzes the whole solution. It reports possible bugs, security hot spots, vulnerabilities, code smells and much more. The developers check the board regularly because it is part of the definition of done. This ensures that the health of the code stays good and not a lot of technical debt is introduced while developing.

12.3. Usability Tests

To find out how good the usability of the application is and find possible improvements for further development, usability tests were created. Since the usability tests were made very late, no additional adjustments were made based on the evaluation of the usability tests.

In the usability tests, the user is given certain scenarios. Based on the scenarios, it should be clear what the user needs to do. The user should provide as much information about his/her thought process as possible, so the observing project team member can try to understand where the usability

of the application needs to be improved. The observer should not help the user, but if necessary he can give small hints. The observer makes notes while the user works through the scenarios. After completion of all scenarios, the observer evaluates the results.

The completed usability tests can be found in the appendix.

12.4. Continuous Integration

The continuous integration helps developers to quickly notice when something breaks, which reduces the time needed to find errors and bugs.

Every time a developer pushes changes to the product repository, the continuous integration pipeline is activated. If one step of the pipeline fails, the pipeline is aborted and the result is reported to the GitLab site, where developers can check the status of the project. The first step of the pipeline is the build of the application. In the next step, all tests of the whole solution are gathered and executed. If any tests fail, the step counts as failed. Finally, the project gets uploaded to Sonarcloud.

13. Further Development

In this chapter, all further development ideas are discussed. For each idea a description and the procedure is discussed.

13.1. Architectural Development

13.1.1. Using the Range Repository Functions

In the early stages of the project, the repository interface did not have methods to create and update a range of entities. They were added very late in the project, when it became obvious, that the current solution for creating and updating a range of entities is very time consuming. In the current service implementation, the create or update method respectively are called for each entity, which creates a new database session for each entity. Because of the tight time schedule, it was decided that the services will not be refactored. The methods to create and update a range of entities were added, so they can be used when developing new services. A quick test showed that using these methods could reduce loading time in certain screens by a factor of ten or higher.

13.1.2. Implementing the Unit of Work Pattern

The unit of work pattern would reduce the complexity of the services significantly and would make operations on the repositories more natural. It would even reduce the coupling between business layer and data access layer. The easiest way to implement the unit of work pattern would probably be to use the unit of work functionality implemented by EF Core. To do this, a class needs to be added, lets call it a data session, that provides all repositories and has a method that saves all changes made to these repositories. The save method of the hypothetical data session could simply call the SaveAllChanges method of the database context. For this to work two things are important:

1. When changes are made on the business entity, the changes also need to be made on the entity tracked by EF Core.
2. The database context needs to be the same within one data session.

13.2. Missing Functionality

13.2.1. Reports

This part is pretty straightforward. The current version of the RoKü application can create some reports which the new RoKü application can't. The most important missing report for Spitex is the weekly pdf, which gives a good overview of the week. The following reports are missing:

1. Weekly PDF (customer information by route)
2. Customer notices for the kitchen
3. Statistics

For each new report another view, another service, another form data entity and another pdf report needs to be created.

Additionally, the billing report should include the amount of menus a customer has ordered in the month.

13.2.2. Assigning Options Directly to Customers

Currently, default orders can be created for each customer. Within a default order the options can be defined. But because the customer can choose from different menus every day, it is often not clear which menu he will order on which day. If those menus all allow the same preference, the customer could say that he always wants his menus with a certain option. A simple example for this would be a customer who always wants a big portion, no matter what menu he chooses. Therefore it would make sense, if the administrator could assign this option to the customer and, no matter what dish he orders, he will automatically get it the way he wants. To implement this it either requires one order that is not shown anywhere, or the model has to be adapted, which would be the cleaner solution. A dish configuration without a dish should be assigned to a customer. This default configuration must then be given to the menu selection table in order to have the defaults automatically selected.

13.2.3. Make Saving Cascades Clear

In some cases, it is unclear when the changes are written to the database. In the route sets, delivery routes and preference options only the save button of the underlying entity (e.g route set and preference) saves the whole configuration. Another way to help the user would be to rename the save buttons to something like "ok". Another step would be to warn the user that he loses the changes, when navigating somewhere else.

13.2.4. Make Entry/Exit Dates Viable for Orders and Routes

At the moment, entry and exit dates of customers can be assigned but are only evaluated on the current day. For example, a customer that will be active in a week can not order anything, even if the week chosen is next week. To change that, the order and route service has to be changed, and the views have to be adopted too. In the views themselves, each time a week or a date is changed, the customers have to be reloaded all over again.

13.2.5. Edit List Items with Double-Clicks

Another small quality of life improvement would be to allow users to double click on list items to edit them. This feature should not require much work, but it would be necessary to research how this can be done with WPF. Optimally, this would only require small changes to XAML code.

13.3. Additional Functionalities

13.3.1. Backup Through the Application

Currently, the user has to know where and what he has to copy to an USB stick to make a data backup. This could easily be automated. A new screen has to be added and a function that copies the "data.db" file to the backup location.

13.3.2. Create a CSV Billing Report

During the elaboration phase, the idea of generating a billing report that can be imported by the accounting department was discussed. This would not only require sharing customer ids between the accounting and the route planning department, but also changes in the accounting software. To implement this, further elaboration and discussion with the billing department is necessary.

13.3.3. Billing Broken Dishes

Sometimes, the customers accidentally break dishes or maybe even the warming box. So far, the customer receives an additional bill for the amount to replace the broken item. It would make it easier for both the customer and Spitex, if this fee could be handled by the RoKü application and be part of the monthly billing report. The billing of broken dishes was already discussed in the elaboration phase. Conceptually, this fee would belong to the order entity. So far, the order references many dish configurations. This could be replaced by a list of implementations of the IOrderable interface. The dish configurations could implement this interface and an additional class for broken dish fees that also implements the new interface could be added. These fees can then be added either in the associated order or at the end when printing the billing information.

13.3.4. Bill a Rental Fee for the Dishes

Spitex also mentioned a possible fee for the rental of the dishes. With the current solution this could only be done by increasing the prices of the dishes, but then the pricing would quickly become unclear, even to the administrator. With the solution described in section 13.3.3 another implementation of the IOrderable interface could be created. The fee could be automatically added when adding a dish configuration. This would also require additional screens to configure the rental fee.

13.3.5. Track Warming Boxes

Many customers of Spitex are forgetful and misplace warming boxes. Since Spitex has no tracking of the warming boxes, some never get returned. A simple tracking of who has how many boxes could solve this problem relatively easily. A possible solution could be to add an additional column to the daily plans, where the number of boxes the customer currently has is listed. In general, the program would assume that all boxes are returned on the delivery. If not all boxes are returned, this would have to be reported back to the RoKü administrator and the administrator would have to enter the amount of boxes, that the customer still has. On the next delivery those boxes are accounted for in the respective column. This problem should be elaborated further before implementing.

13.3.6. Installer

The current installation process is not well suited for personnel without much experience with computers. A simple installer that copies the files on its own and creates the short cut on the desktop would make the installing and updating of the application way easier. One option to achieve this would be to use the WiX toolset.

13.3.7. Last Minute Changes to Billing Information

It would make sense to allow the RoKü administrator to make last minute corrections to the billing information. To allow changes to the billing information, multiple files have to be changed. Before printing the billing information, an overview of all customers, what they ordered and how much they have to pay should be shown, where the administrator can change the total the customer needs to pay.

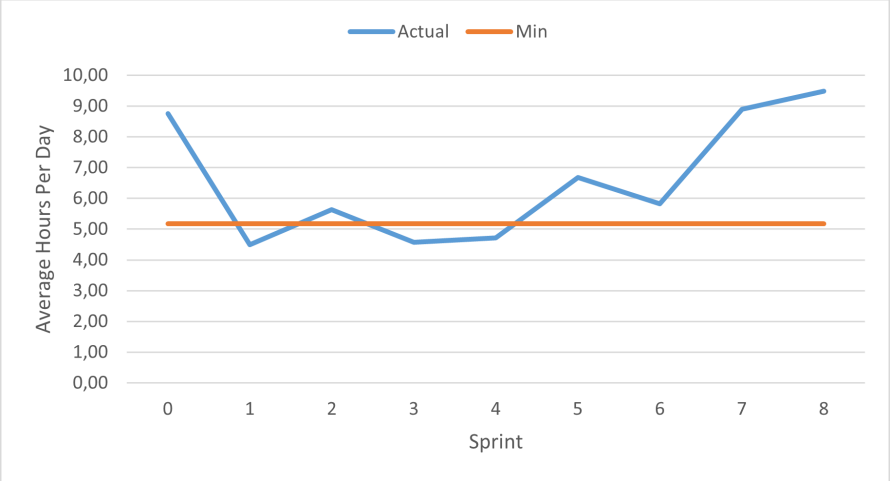
14. Project Monitoring

14.1. Time Statistics

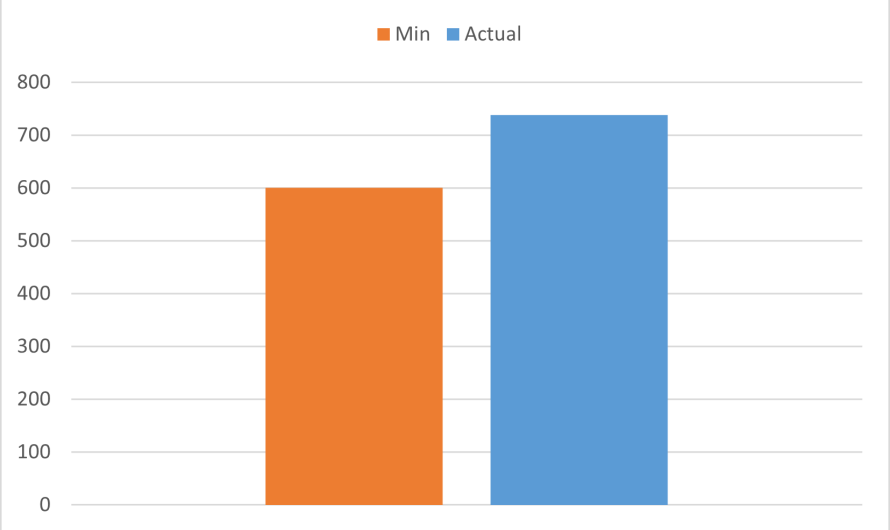
In general, the time needed by the team members was around what is expected for such a project. In the beginning, the time investment hovered around the minimum, but as the project progressed, more time was invested, see figure 14.1a. Some deviation is explained by the fact that in some weeks more work was done on the weekend. Since the end of the sprint was on a Friday, this could lead to time that is reported in the following sprint.

Sprint	Total time Spent	Days	Average Time Per Day	Minimum Time Per Day
0	43,75 h	5	8,75 h	5,17h
1	62,83 h	14	4,49 h	5,17h
2	78,83 h	14	5,63 h	5,17h
3	64,00 h	14	4,57 h	5,17h
4	65,92 h	14	4,71 h	5,17h
5	93,50 h	14	6,68 h	5,17h
6	81,50 h	14	5,82 h	5,17h
7	124,50 h	14	8,89 h	5,17h
8	123,25 h	14	9,48 h	5,17h
Total	**738,08 h**	**116**	**6,56 h**	

Table 14.1.: Work hours breakdown.



(a) Average hours per day per sprint.



(b) Total hours spent.

Figure 14.1.: Time comparison.

Since this was mainly a development project, and not a theoretical proof of work, the difference between the time spent per category, seen in figure 14.2 is reasonable.

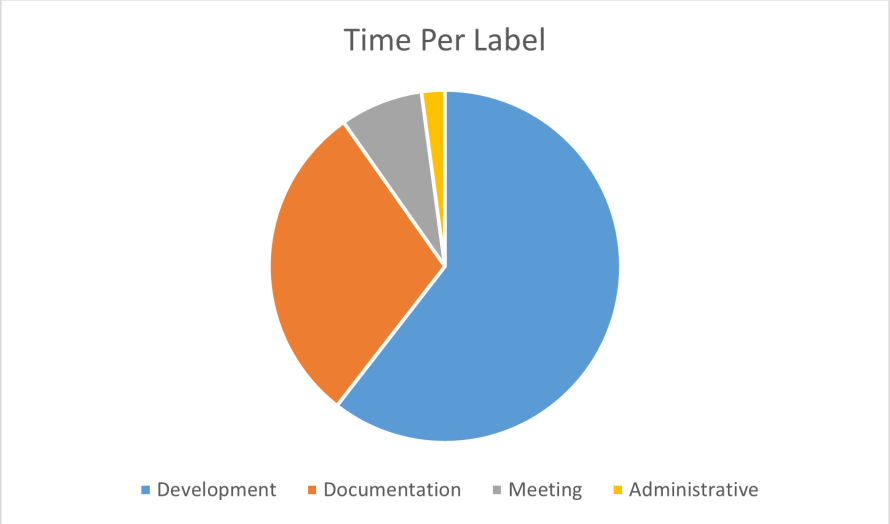


Figure 14.2.: Time breakdown by labels.

14.2. Code Statistics

As previously mentioned, Sonarcloud was used as the main tool to analyze code. It reports the following:

- 0 Bugs
- 0 Vulnerabilities
- 0 Security hotspots
- 3 h Technical debt (2 h 34 min are in the PDF generators)
- 17 Code Smells
- 0% Code duplication
- 16k lines of code

The code coverage was measured with JetBrains dotCover. The targeted 80% test coverage could not be reached, because the presentation project pulls the average down. The other projects have all reached the 80% test coverage. Further information about the missing test coverage of the presentation can be found in section 12.1.

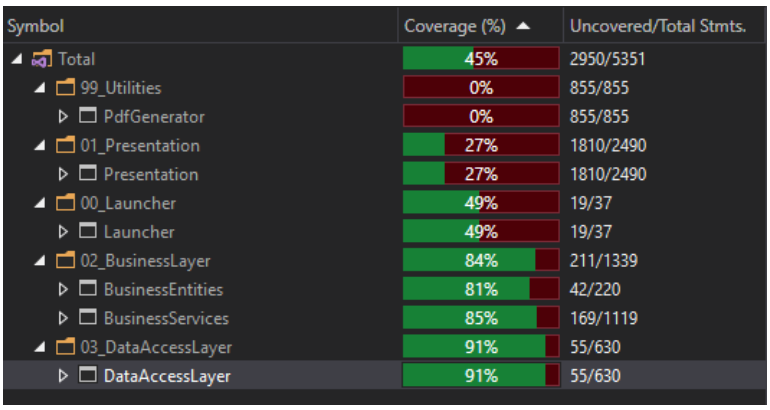


Figure 14.3.: Code coverage.

15. Software Documentation

The whole manual can be found in the appendix in chapter C. Since the manual needs is part of the deliverable, it needs to be written in German.

Acronyms

CI continuous integration. 16, 17

CRUD create/read/update/delete. X, 20, 21, 27, 28, 43

EF Core Entity Framework Core. 7, 37, 43, 47, 56, 72

MVP Minimal Viable Product. IX, 12, 14

MVVM Model-View-ViewModel. 39

NFR non functional requirements. 13, 21, 29, 65, 74, 76

ORM Object-relational mapping. 37

RoKü Rollende Küche. IV, VI, VII, VIII, 2, 4, 6, 7, 10, 16, 17, 20, 24, 26, 28, 30, 38, 39, 46, 56, 58, 105

RUP Rational Unified Process. 12

UI user interface. IV, VII, 7, 13, 18, 30, 32, 37, 49, 65

Glossary

Spitex Is an institution for elderly care. IV, V, VI, VIII, 2, 3, 13, 14, 30, 37, 56, 58

MS Access Microsoft Access, a database tool in the Microsoft Office packet. IV, 3, 4, 36

Rational Unified Process Is a software engineering process that provides a disciplined approach to assigning tasks and responsibilities within a development organization. [14]. IV, 12, 64

Scrum Is a agile, iterative framework for developing complex and unpredictable products [12]. IV, 12, 14, 15

.NET 5 A software framework. IV, VII, 7

SQLite A lightweight version of an SQL database. IV, VII, 7, 37, 54

WPF The Windows Presentation Foundation is a UI framework developed by Microsoft. IV, VII, 7, 37, 39, 40, 46, 57

Sonarcloud An online tool that allows code analysis. XI, 17, 23, 54, 55, 61

GitLab Is a Git repository manager, which provides time and issue tracking. 7, 14, 15, 16, 17, 18, 19, 55

SMART Specific, Measurable, Agreed-upon, Realistic, Time-based. A model adapted from psychology to have better NFRs [16]. 21

JetBrains dotCover A tool that runs all tests in the solution and analyzes the which lines are covered. 23, 54, 61

Figma A prototype design tool. 32

NoSQL NoSQL databases (aka "not only SQL") are non tabular, and store data differently than relational tables. [11]. 37

Autofac Autofac is an inversion of control container for .NET Core [1]. 38, 39, 40, 44, 45, 46, 48, 49, 69

XAML XAML is a visual markup language developed by Microsoft. 39, 46

mvvm light toolkit A toolkit that implements important mvvm components. 39

AutoMapper A convention-based object-object mapper [2]. 45, 48, 49, 72

Moq A mocking library that is simple to use, strongly typed and materialistic [6]. 49, 54

MsTest A testing framework by Microsoft. 54

Visual Studio 2019 A development environment by Microsoft. 54

WiX toolset The WiX toolset lets developers create installers for Windows Installer, the Windows installation engine [13]. 58

Git Is a version-control system, which helps collaborating. 65

Bibliography

[1] Autofac. https://autofac.org/. Accessed: 2021-06-10.

[2] Automapper. https://automapper.org/. Accessed: 2021-06-11.

[3] Dbcontext class. https://docs.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.dbcontext?view=efcore-5.0. Accessed: 2021-06-11.

[4] Dbset<tentity> class. https://docs.microsoft.com/en-us/dotnet/api/system.data.entity.dbset-1?view=entity-framework-6.2.0. Accessed: 2021-06-11.

[5] The model-view-viewmodel pattern. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm. Accessed: 2021-06-10.

[6] Moq quickstart. https://github.com/Moq/moq4/wiki/Quickstart. Accessed: 2021-06-11.

[7] Package diagram - uml2 diagrams - uml modeling tools. https://www.visual-paradigm.com/VPGallery/diagrams/Package.html. Accessed: 2021-06-06.

[8] Richtlinien des hochschulrates für die koordinierte erneuerung der lehre an den universitären hochschulen der schweiz im rahmen des bologna-prozesses. https://www.fedlex.admin.ch/eli/cc/2015/322/de. Accessed: 2021-06-06.

[9] The scrum guide. https://www.scrumguides.org/scrum-guide.html. Accessed: 2021-03-01.

[10] Task lists. https://docs.gitlab.com/ee/user/markdown.html#task-lists. Accessed: 2021-03-01.

[11] What is nosql. https://www.mongodb.com/nosql-explained. Accessed: 2021-06-10.

[12] What is scrum? https://www.scrum.org/resources/what-is-scrum. Accessed: 2021-03-01.

[13] Wix toolset. https://wixtoolset.org/. Accessed: 2021-06-11.

[14] R. Behrends, L. K. Dillon, S. D. Fleming, and R. E. K. Stirewalt. Rational unified process, best practices for software development teams. Technical Report TP026B, Rev 11/01, Rational the software development company.

[15] P.-E. Bergman. Repository design pattern. https://medium.com/@pererikbergman/repository-design-pattern-e28c0f3e4a30, 2017. Accessed: 2021-06-06.

[16] R. L. Bouge. Use s.m.a.r.t. goals to launch management by objectives plan. Accessed: 2021-03-10.

[17] A. S. D. George. Data templating overview. https://docs.microsoft.com/en-us/dotnet/desktop/wpf/data/data-templating-overview?view=netframeworkdesktop-4.8&viewFallbackFrom=netdesktop-5.0. Accessed: 2021-06-10.

[18] E. Hieatt and R. Mee. Repository. https://martinfowler.com/eaaCatalog/repository.html. Accessed: 2021-06-10.

[19] C. Larman. UML 2 und Patterns Angewendet. Massachusetts Institute of Technology Press, 2005.

[20] J. Palermo. The onion architecture : part 1. https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/. Accessed: 2021-06-06.

List of Figures

List of Tables

Part III.

Appendix

A. Personal Reports

Philipp Bolliger

The bachelor thesis was a very interesting but also very intensive time. Especially because I had to work two days a week for my company, and got reassigned to a much bigger project at the beginning of the bachelor thesis.

I am glad, that I could do the bachelor theses with Marc, because I know him very well and know how he works from previous projects. As always we worked almost perfectly together.

It was very motivating to work with an external customer, that needs the product and working with Mrs. Zbinden was a very interesting experience, because she was the first person I made a program for, that does not have a lot of experience with computers.

I was also happy to work with technologies I am familiar with. It was motivating to see how much I learned in the various projects I could work on at Reishauer AG. Although, after working for several years with the technologies we used in this project, I thought I have a good enough understanding, to setup a project without many problems. But it turned out to be a big difference if you work with those technologies, when everything is properly setup or want to set them up yourself. I wasted way too much time with EF Core problems and AutoMapper issues. This was very frustrating at times, especially when the solution turned out to be very easy.

Sadly we underestimated the scope of the project and could not finish all the features we wanted to. We were both very determined to finish what we have started, which lead to some classes I am not particularly happy with. Which makes it even more frustrating, that we couldn't deliver an application with all the features.

Marc Scherrer

This project, like all bachelors theses, was really tiresome for me. In addition to the current situation that forced us to work only from home, I also went through big personal changes, which was probably not a very good idea in hindsight.

Considering the project selection, I expected us to create a much better product than we actually have. The experienced with the engineering project and the study thesis did not help as much as I thought.

Our customer was really kind and motivating, to the point that I am frustrated that we could not deliver a better product to her. The Interactions with the supervisor were also really good. Our meetings with him were more productive than in the other projects. Having already done the study thesis with Philipp, we knew what to expect from each other. He was definitely more the expert in this project, since I was it more in the last.

Although we poured much time in this project, the product is a bit lackluster. However, I am satisfied with our performance and dedication. What I would have done, had I known the outcome, was split this project into a very detailed specification plus an architectural prototype, and an implementation. The domain was simply too complex to make a high standard requirement specification, technology evaluation, implementation and transition in one project.

All in all, I am happy with our performance, but not so much the product. As before, it was a pleasure working with Philipp.

Special Thanks

We also want to take the time to thank all people that have supported us. Frank Koch for overseeing this project and contributing helpful viewpoints, Mrs. Hedi Zbinden for being so supportive and motivational and last but not least, our friends who endured us during this time and helped us where they could.

B. Usability Test

This usability test is modeled after the usability test in Human Computer-Interaction Design.

Since this application is used by only one administrator, and this administrator works with an instruction, the usability test is carried out after the administrator read the instructions.

Knowledge Goals

1. Which "create" buttons are used (in the navigation or under the list)?
2. Are the words and concepts used, understandable?
3. Is the membership system clear?
4. Is the menu selection table clear?
5. Is the route planning (default and delivery route) clear?

Procedure

First, the administrator is asked to carry out the following tasks verbose, and provide commentary to what and why he/she does. The administrator is informed that no or very minimal help is given by the conductor of the test.

Next, the administrator is given the test scenarios one by one.

The Test conductor should keep an eye on the NFRs.

When the scenarios are finished, the administrator is asked about the general experience and arising questions are discussed. The evaluation is presented and if everything is okay for the administrator, the test is finished.

Test Scenarios

The details, like name and address of the customer, or the content of the order are not important.

Nr.	Instructions	Behavior	Knowledge Goal
1	Create a customer	A new customer was correctly created	1, 4
2	Enter an order	The order was entered, after the new customer was inserted in the route templates	1, 4
3	Plan a delivery	Delivery route was planned, but warnings were ignored	1
4	Create a new Variation	New variation was only clear AFTER it was explained	2
5	Customer X leaves in 1 week	New date was added	3
6	Customer Y enters in 1 week	New customer was added and directly added in the route templates	3
7	Customer X enters again in 2 weeks	New entry date was added successfully	3
8	A new order just came in	Orders were created separately, and thus overwritten	4
9	Plan a route from a to z	Was covered in previous tests	5

German Test Scenarios

1. Ein neuer Kunde (Max Mustermann) kommt hinzu. Er möchte immer Salat, aber nie Suppe
2. Eine Bestellung kommt rein. Max bestellt 5 mal Menu A, jeweils Montag - Freitag
3. Planen Sie eine Lieferung, in welcher Max irgendwo zwischendrin enthalten ist.
4. Das Spezialmenu kann jetzt auch fleischlos bestellt werden
5. Max verlässt die RoKü in einer Woche
6. Ein neuer Kunde (Peter Lustig) tritt in einer Woche bei
7. Max tritt in zwei Wochen wieder bei
8. Eine neue Bestellung für Max und Peter kam rein (Bestelliste wurde separat gesendet)
9. Sehen Sie Max und Peter als neue Stammkunden. Planen Sie eine neue Route mit Lieferung

Evaluation

Feedback:

- The saving mechanism, especially the cascading save in the routes was unclear.
- A bug made it impossible to print the daily order sheets.
- The search functionality was only used with the whole name, even when a part of the name would have sufficed.
- To edit something, an item was double clicked, before clicking the edit button.

- Generally, all light gray items were dismissed as inactive, even if they were table headers.
- There are many more possibilities now, but that means more effort.

NFR insights:

- Warning in delivery planning was dismissed.
- Installation and backup was explain intensive (drag-and-dropping files).

C. Manual

Spitex verein Wädenswil

Rollende Kueche 2021

Anleitung

Philipp Bolliger & Marc Scherrer
8.6.2021

Contents

Versionen

Datum	Autor	Beschreibung
06.06.2021	Philipp Bolliger	Dokument erstellt
08.06.2021	Marc Scherrer	Kleine Verbesserungen

Installation

Um das Programm zu installieren, müssen alle Dateien im RollendeKueche2021.zip in den Ordner C:\RollendeKueche2021 kopiert werden. Dazu einfach auf mit der rechten Maustaste auf die RollendeKueche2021.zip Datei klicken und dann im Menü «Alle Extrahieren» auswählen. Es öffnet sich dann ein weiteres Fenster.

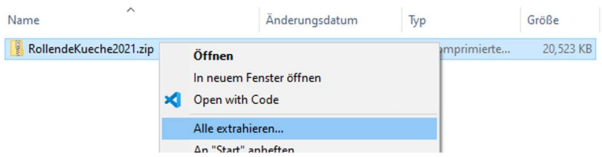


Abbildung 1: Schaltfläche im Kontextmenü

In dem neuen Fenster muss nun der Zielordner angegeben werden. Es kann einfach C:\RollendeKueche2021 ins Textfeld eingegeben werden. Falls der Ordner noch nicht existiert, wird er dann von Windows automatisch erstellt. Zum Abschluss muss nur noch auf die Schaltfläche «Extrahieren» geklickt werden.

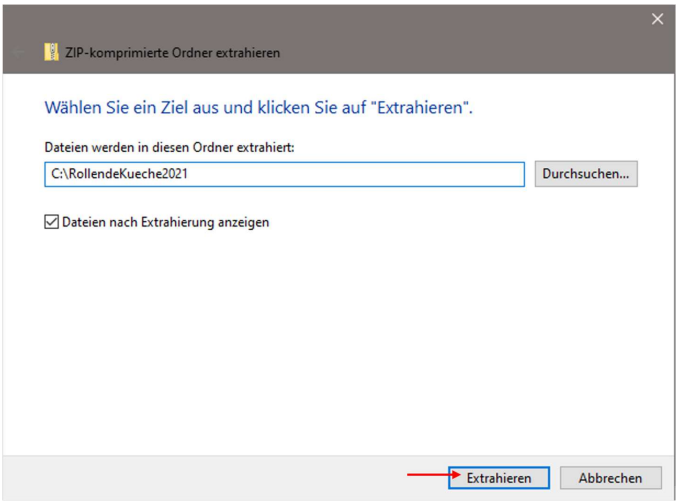


Abbildung 2: Neues Fenster, um Zielordner anzugeben

Sobald der Vorgang abgeschlossen ist, kann mit der rechten Maustaste auf die Anwendungsdatei «Launcher» geklickt und im Kontextmenü auf «Verknüpfung Erstellen» ausgewählt werden.

Achtung: Es ist wichtig, dass es sich bei der «Launcher» Datei um einen Dateityp «Anwendung» handelt!

Abbildung 3: Verknüpfung erstellen

Dies wird im Ordner C:\RollendeKueche2021 eine neue Datei mit dem Namen «Launcher – Verknüpfung» erstellen, welche nun zu «RollendeKueche2021» umbenannt werden sollte und auf den Desktop verschoben wird.

Update

Um das Programm zu aktualisieren, muss in einem ersten Schritt die Datei «Data» im Ordner C:\RollendeKueche2021 in einen anderen Ordner kopiert werden.

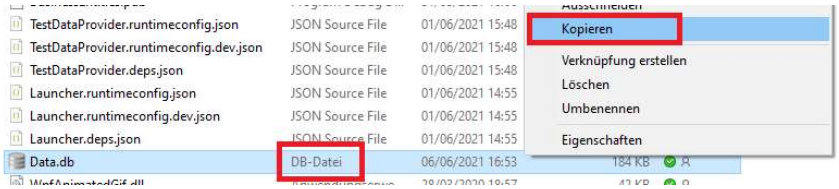


Abbildung 4: Data Datei kopieren

Dies ist notwendig, dass keine Daten beim Aktualisieren der Applikation verloren gehen. Sobald der Kopiervorgang abgeschlossen ist, kann der Ordner C:\RollendeKueche2021 komplett gelöscht werden. Danach muss die Installation wie oben beschrieben ausgeführt werden. Sobald die Applikation installiert ist, muss die «Data» Datei wieder in den Ordner C:\RollendeKueche2021 kopiert werden, damit die Daten wieder vorhanden sind. Danach kann die Kopie im anderen Ordner wieder gelöscht werden.

Backup

Um ein Backup zu erstellen, muss einfach die Datei Data im Ordner C:\RollendeKueche2021 auf ein USB-Stick kopiert werden. Dies Sichert alle Rohdaten der Applikation.

Navigation

Die Navigation der Applikation geht fast ausschliesslich über den Navigationsbereich auf der linken Seite des Fensters.

Abbildung 5: Navigationsbereich

Der grau eingefärbte Beriech signalisiert, wo man sich gerade befindet. Im Beispiel in der Abbildung 5: Navigationsbereich befinden wir uns im Dashboard, daher ist der Knopf «Dashboard» grau eingefärbt.

Konfiguration

Variationen

Variationen erlaubt es dem Benutzer eigene Einstellungen zu den Gerichten zu erstellen. Um eine neue Variation einzuführen, muss diese zuerst im Bereich «Variationen» erstellt werden. Dazu kann unterhalb der Liste aller Variationen auf «Erstellen» gedrückt werden. Um Variationen zu bearbeiten oder zu löschen die Variation zuerst in der Liste ausgewählt werden und dann kann der entsprechende Knopf gedrückt werden.

Abbildung 6: Eingabemaske einer Variation

In der Eingabemaske der Variation kann im oberen Bereich ein Name für die Variation und eine Beschreibung eingegeben werden. Die Beschreibung dient nur dazu, dass man später in der Variation nachschauen kann, was gemeint ist. Es kann zusätzlich angegeben werden, ob die Variation ein

separates Symbol hat. Eine Variation mit separatem Symbol wird im Tagesplan direkt in der Spalte das zusätzliche Symbol der ausgewählten Option dargestellt.

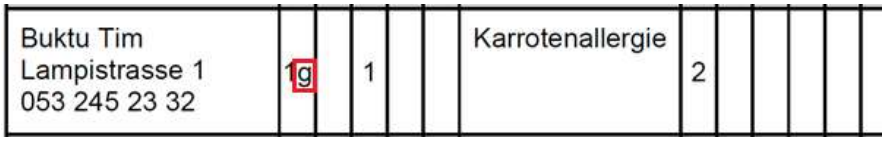


Abbildung 7: Darstellung im Tagesplan mit extra Symbol

Variationen ohne «Extra Symbol» werden zurzeit nicht benutzt.

Im unteren Bereich der Eingabemaske können die verschiedenen Optionen einer Variation erstellt und bearbeitet werden. Eine neue Option kann mit einem Knopfdruck auf «Erstellen» erstellt werden. Um eine Option zu bearbeiten, muss die Option zuerst ausgewählt werden und dann kann der Knopf «Bearbeiten» gedrückt werden. In Beiden Fällen wird man zur Eingabemaske einer Option geleitet.

Abbildung 8: Eingabemaske einer Option innerhalb einer Variation

Jede Option hat ein Name, ein Preis und ein Symbol. Das Symbol kann auch leer gelassen werden. Gerade bei Variationen ohne extra Symbol sollte das Feld leer gelassen werden. Der Preis ist ein Preisunterschied. Dies bedeutet, dass im gezeigten Beispiel in Abbildung 8 ein Gericht mit der Option Gross 2.- **mehr** kostet.

Sobald man alle Eingaben getätigt hat, kann man auf den Knopf «Speichern» drücken. Falls man nur aus Versehen eine Option hinzugefügt hat oder man die Änderungen an der Option verwerfen möchte, kann man auf «Abbrechen» drücken. In beiden Fällen wird man zurück zur Eingabemaske der Variation geleitet.

Achtung: Zu diesem Zeitpunkt sind die Änderungen an Optionen und die neuen Optionen noch nicht vollständig gespeichert. Erst wenn man in der Eingabemaske der **Variation** auf «Speichern» drückt, werden alle Änderungen endgültig übernommen! Vergisst man das oder drückt man auf «Abbrechen» gehen alle Änderungen an der Variation und den Optionen innerhalb der Variation verloren.

Wie die Variationen den Gerichten zugewiesen wird und benutz wird, wird in den Kapiteln Gerichtkategorien

Jedes Gericht **muss** einer Kategorie zugewiesen werden. Dies ist notwendig, um beim Tagesplan entscheiden zu können welche Gerichte in den linken und welche in den rechten Zeilen untergebracht werden. Um die Kategorien zu verwalten, muss man in der Navigation auf «Gerichte» klicken. In dem Bereich ist sowohl die Liste aller Gerichte als auch die Liste der vorhandenen Kategorien untergebracht. Um eine neue Kategorie zu erstellen, muss lediglich auf den Knopf «Erstellen» geklickt werden. Um eine Kategorie zu bearbeiten, muss zuerst eine Kategorie ausgewählt werden, bevor man auf den Knopf «Bearbeiten» drücken kann. Kategorien können auch gelöscht werden. Allerdings ist

das nur möglich, wenn kein Gericht mehr der zu löschenden Kategorie zugewiesen ist. Das Programm warnt den Benutzer, falls dies nicht möglich ist.

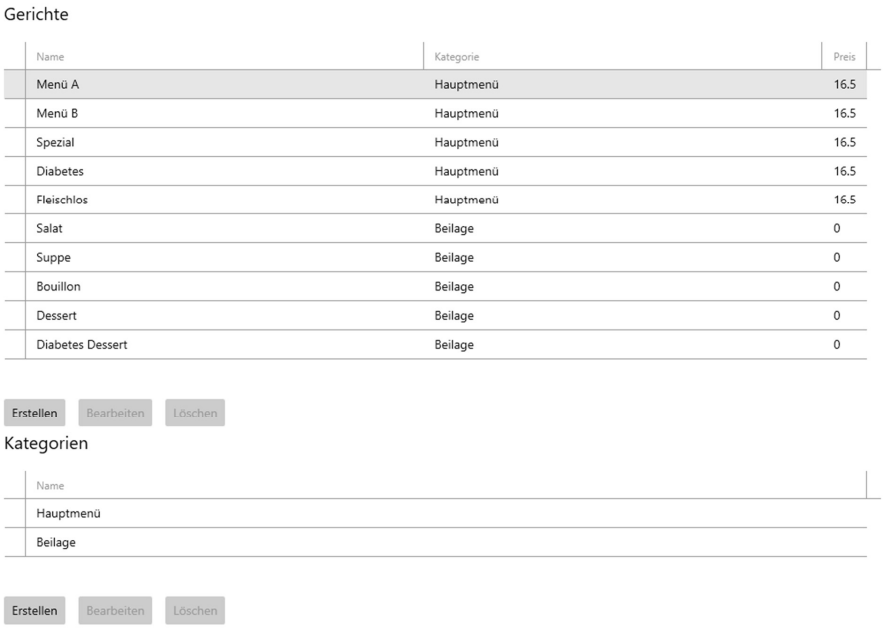


Abbildung 9: Liste aller Gerichte und Kategorien.

Beim Erstellen und Bearbeiten einer Kategorie wird der Benutzer zu einer separaten Ansicht navigiert, in welcher er den Namen der Kategorie bestimmen kann.

Gerichte, **Fehler! Ungültiger Eigenverweis auf Textmarke.** und Bestellungen eingeben erwähnt.

Gerichtkategorien

Jedes Gericht **muss** einer Kategorie zugewiesen werden. Dies ist notwendig, um beim Tagesplan entscheiden zu können welche Gerichte in den linken und welche in den rechten Zeilen untergebracht werden. Um die Kategorien zu verwalten, muss man in der Navigation auf «Gerichte» klicken. In dem Bereich ist sowohl die Liste aller Gerichte als auch die Liste der vorhandenen Kategorien untergebracht. Um eine neue Kategorie zu erstellen, muss lediglich auf den Knopf «Erstellen» geklickt werden. Um eine Kategorie zu bearbeiten, muss zuerst eine Kategorie ausgewählt werden, bevor man auf den Knopf «Bearbeiten» drücken kann. Kategorien können auch gelöscht werden. Allerdings ist das nur möglich, wenn kein Gericht mehr der zu löschenden Kategorie zugewiesen ist. Das Programm warnt den Benutzer, falls dies nicht möglich ist.

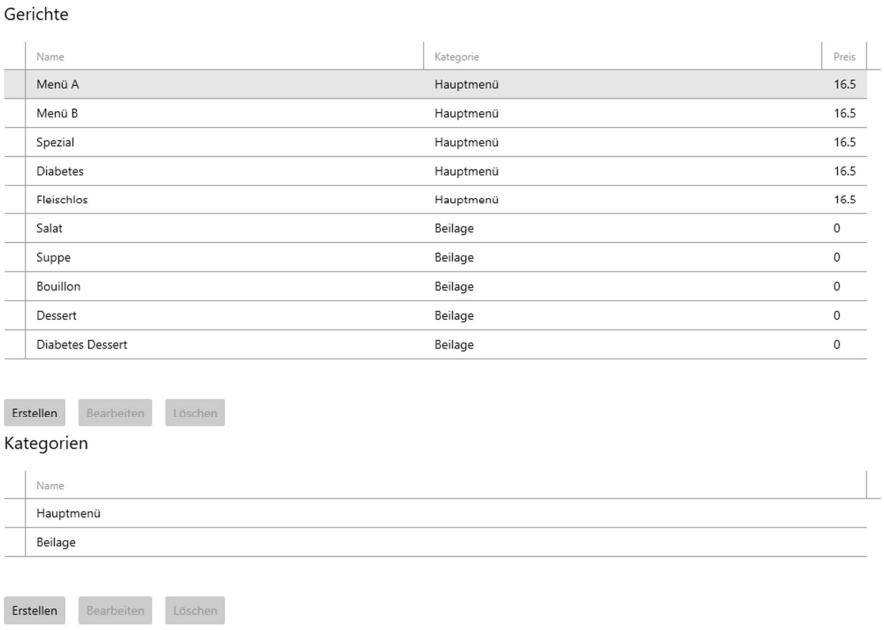


Abbildung 9: Liste aller Gerichte und Kategorien.

Beim Erstellen und Bearbeiten einer Kategorie wird der Benutzer zu einer separaten Ansicht navigiert, in welcher er den Namen der Kategorie bestimmen kann.

Gerichte

Im Bereich Gerichte werden alle Gerichte in einer Liste angezeigt. Dort können neue Gerichte erstellt werden und bestehende Gerichte angepasst und gelöscht werden. Um ein neues Gericht zu erstellen, muss lediglich auf den Knopf «Erstellen» direkt unterhalb der Liste aller Gerichte gedrückt werden. Um ein Gericht zu bearbeiten, muss das zu bearbeitende Gericht ausgewählt werden und dann auf den Knopf «Bearbeiten» gedrückt werden. Beim Erstellen und Bearbeiten eines Gerichts wird der Benutzer auf eine separate Eingabemaske navigiert. Dort kann man den Namen, den Preis, die Kategorie und die erlaubten Variationen angeben. Alle Variationen, die man bei dem Gericht erlaubt, können später in den Standard Bestellungen und den normalen Bestellungen konfiguriert werden.

Abbildung 10: Eingabemaske eines Gerichts

Achtung: Preisänderungen betreffen auch Bestellungen in der Vergangenheit! Wird also Mitte Mai ein Menü teurer, werden auch alle Bestellungen vorher teurer. Preisänderungen sollten daher nur nach dem Erstellen der Abrechnungen gemacht werden.

Kunden

Um einen neuen Kunden hinzuzufügen, kann man entweder direkt auf «Neuer Kunde» im Navigationsbereich, oder in der Kundenliste auf «Hinzufügen» klicken. Um einen Kunden zu bearbeiten, muss man ihn in der Kundenliste suchen und auswählen. Sobald ein Kunde ausgewählt ist, kann der Knopf «Bearbeiten» am unteren Bildschirmrand gedrückt werden. Kunden können gelöscht werden, indem man diese in der Liste auswählt und dann auf «Löschen» drückt.

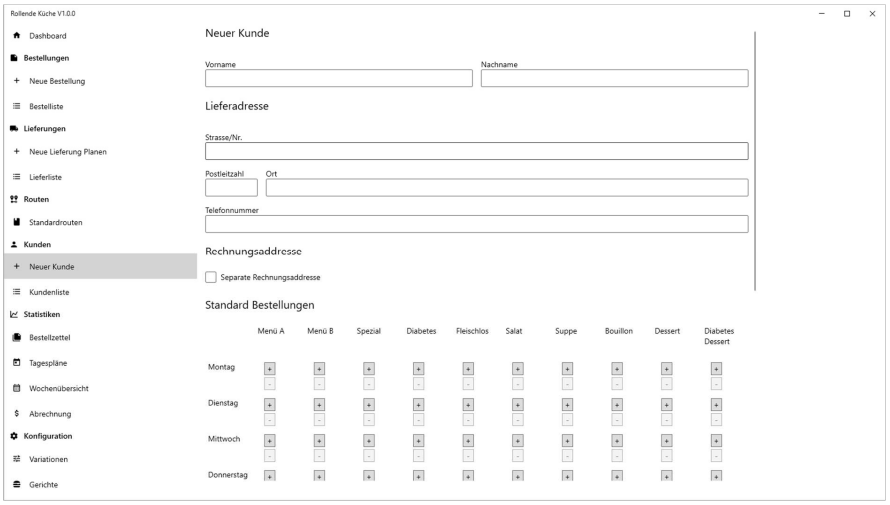


Abbildung 11: Eingabebereich Kunde

Kundendaten

Der Eingabebereich für das Erstellen und Bearbeiten eines Kunden fast identisch. Hier können verschiedene Daten der Kunden hinterlegt werden.

Separate Rechnungsadresse

Unter Rechnungsadresse kann optional eine separate Rechnungsadresse angegeben werden. Diese wird dann in der Abrechnung erscheinen.

Standardbestellungen

Falls ein Kunde immer die gleiche Bestellung möchte, kann dies in der Sektion «Standard Bestellungen» konfiguriert werden. Die Standardbestellungen werden beim Erstellen von neuen Bestellungen übernommen. Falls für ein Gericht Variationen vorhanden sind, kann für diese auf der rechten Seite direkt eine gewünschte Option ausgewählt werden. Dies muss aber für jedes Gericht einzeln wiederholt werden.

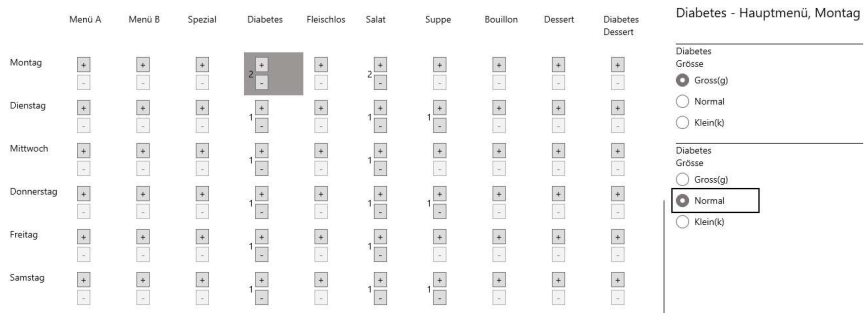


Abbildung 12: Ausgefüllte Standardbestellungen

In der Abbildung 12 sieht man ein Beispiel wie diese Standardbestellungen ausgefüllt werden können. Entsprechend möchte er jeden Tag ein Diabetes Menü. Auf der rechten Seite kann nun direkt angegeben werden, dass er gerne eine grosse Portion möchte. Der Kunde möchte jeden Tag Salat, aber nur dienstags, donnerstags und samstags eine Suppe. Zusätzlich hat er jeden Montag Besuch von einem Freund, der ebenfalls ein Diabetes Menu möchte. Entsprechend möchte er montags zwei Menüs und zwei Salate.

Mitgliedschaft

Dieser Bereich unterscheidet sich, ob man einen Kunden erstellt oder bearbeitet. Beim Erstellen eines Kunden, kann hier nur angegeben werden an welchem Tag der Kunde aktiv wird. Bis zu diesem Tag ist der Kunde dann inaktiv und es können noch keine Bestellungen für ihn erfasst werden.

Abbildung 13: Eingabefeld für das Eintrittsdatum beim Erstellen eines Kunden

Beim Bearbeiten eines Kunden, wird ein Knopf angezeigt, der den Benutzer zu einer separaten Eingabe führt, in der die Historie des Kunden angezeigt wird und sein Status von aktiv auf inaktiv, oder von inaktiv auf aktiv umgestellt werden kann. Dazu muss lediglich das Datum ausgewählt werden, wann sich der Status des Kunden ändern soll und dann auf «Speichern» drücken. Falls ein Fehler passiert, kann der letzte Eintrag in der Historie gelöscht werden. Allerdings kann dies nur gemacht werden, falls mehr als ein Eintrag in der Historie vorhanden ist.

Abbildung 14: Eingabemaske zum Bearbeiten der Mitgliedschaft eines Kunden

Achtung: Es sollte zurzeit vermieden werden, ein Datum in der Zukunft auszuwählen, da es im Moment zu Fehlern führen kann. Ein Kunde der erst nächste Woche aktiv wird, wird beim Planen der Bestellungen für nächste Woche noch nicht angezeigt! Dies ist ein Implementierungsfehler, der in der Zukunft behoben werden sollte.

Standardrouten / Touren

Im Bereich Standardrouten, können Routen-Sammlungen erstellt werden, welche beim Planen einer Lieferung verwendet werden können. Es ist möglich mehrere Routen-Sammlungen zu erstellen. Jede Routen-Sammlung enthält wiederum mehrere Routen. Die Anzahl Routen pro Routen-Sammlung ist nicht begrenzt.

Abbildung 15: Liste mit allen vorhandenen Routen-Sammlungen

Im Beispiel in Abbildung 15 wurden so verschiedene Routen-Sammlungen erstellt. So kann beim Planen der Lieferung eine passende Routen-Sammlung als Vorlage verwendet werden.

Um eine neue Routen-Sammlung zu erstellen, kann auf den Knopf «Hinzufügen» gedrückt werden. Um eine Route zu bearbeiten, muss man diese zuerst in der Liste auswählen und dann auf den Knopf «Bearbeiten» drücken. Selektiert man eine Routen-Sammlung und drückt dann auf «Löschen», wird zuerst nachgefragt, ob dies wirklich gemacht werden soll, bevor die Routen-Sammlung gelöscht wird.

Bearbeiten von Standardrouten

Abbildung 16: Eingabemaske für das Bearbeiten einer Routen-Sammlung

Hat man in der Standardroutenliste auf «Hinzufügen» oder «Bearbeiten» gedrückt, wird man zur entsprechenden Eingabemaske weiterführt, in welcher der Name der Routen-Sammlung festgelegt, neue Routen hinzugefügt oder bestehende Routen bearbeitet oder gelöscht werden können. Um eine neue Route hinzuzufügen, muss einfach auf den Knopf «Hinzufügen» gedrückt werden. Um eine Route zu bearbeiten oder zu löschen, muss die Route zuerst ausgewählt werden und dann kann man auf den entsprechenden Knopf drücken.

Wenn man eine Route bearbeitet, erscheint eine neue Eingabemaske, in welcher der Name der Route festgelegt, Kunden zu der Route hinzugefügt und von der Route entfernt werden können.

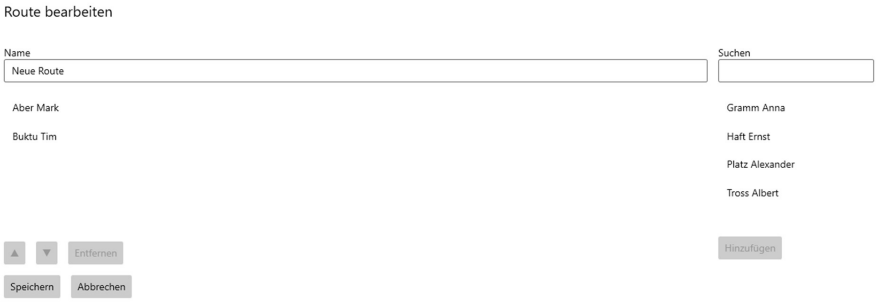


Abbildung 17: Eingabemaske einer Route innerhalb einer Routen-Sammlung

Auf der rechten Seite befinden sich alle aktiven Kunden, die noch in keiner Route eingetragen sind (in der Routen-Sammlung, in der sich die Route, die man bearbeitet befindet). Auf der linken Seite befinden sich alle Kunden, die bereits in der Route eingetragen sind. Kunden können zur Route hinzugefügt werden, indem man diese auf der rechten Seite auswählt und dann auf den Knopf «Hinzufügen» drückt. Der Kunde wird dann ganz unten in der Route hinzugefügt. Um Kunden wieder aus einer Route zu entfernen, muss der Kunde auf der linken Seite ausgewählt werden und dann auf den Knopf «Entfernen» gedrückt werden. Er taucht dann wieder auf der rechten Seite auf und kann auch wieder in anderen Routen innerhalb der gleichen Routen-Sammlung hinzugefügt werden.

Da die Reihenfolge der Kunden innerhalb einer Route eine wichtige Rolle spielt, wird die Reihenfolge gespeichert und kann auch verändert werden. Kunden können einfach mit der Maus gepackt und nach oben oder unten verschoben werden. Alternativ kann ein Kunde ausgewählt werden und mit den Tasten «▲» und «▼» nach oben beziehungsweise unten verschoben werden.

Mit dem Knopf «Speichern» werden alle Änderungen übernommen. Drückt man «Abbrechen» werden alle Änderungen verworfen. In beiden Fällen wird man zurück in die Eingabemaske für die Routen-Sammlung navigiert.

Achtung: Alle Änderungen an den Routen werden erst gespeichert, wenn man in der Eingabemaske der Routen-Sammlung auf Speichern drückt. Vergisst man das, gehen alle Änderungen verloren.

Abbildung 18: Korrektes Speichern

Arbeitsablauf

Der standardmässige Arbeitsablauf ist:

1. Bestellzettel für Kunden ausdrucken
2. Bestellungen von ausgefüllten Bestellzetteln ins Programm übertragen
3. Wochenplan für die Küche ausdrucken
4. Lieferungen Planen
5. Bestellungen kurzfristig anpassen
6. Tagesplan erstellen

Und jeweils am Monatsende muss eine Abrechnung erstellt werden.

Bestellzettel erstellen

Wählt man im Navigationsbereich «Bestellzettel» wird zur Eingabemaske navigiert, in der Bestellzettel konfiguriert und ausgedruckt werden können.

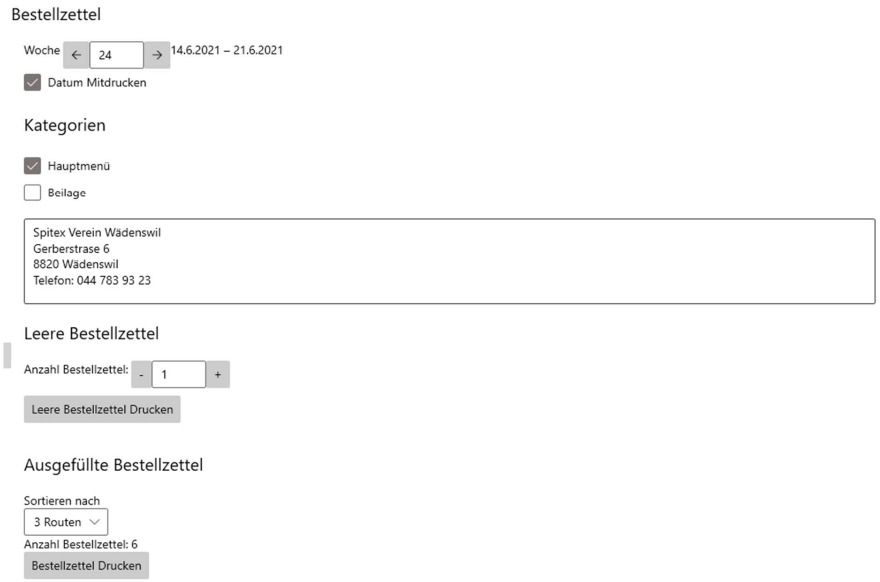


Abbildung 19: Konfigurationsmaske für Bestellzettel

Im oberen Bereich kann festgelegt werden für welche Woche die Bestellzettel sein sollen. Dies ist nur relevant, falls ein Haken bei «Datum Mitdrucken» gemacht wird. Dann wird die Woche und das Datum auf den Bestellzettel mitgedruckt. Lässt man dies weg, muss die Woche und das Datum vom Kunden eingefüllt werden.

Im Bereich «Kategorien» kann festgelegt werden welche Gerichtkategorien vom Kunden bestellt werden können. Es ist möglich auch mehrere Kategorien anzugeben. Es wird für jedes Gericht in den ausgewählten Kategorien eine Zeile auf dem Bestellzettel hinzugefügt.

Unterhalb der Kategorien kann die Fusszeile des Bestellzettels festgelegt werden. Dieser ist standardmässig die Adresse und Telefonnummer des Spitex Vereins Wädenswil.

Im Bereich Leere Bestellzettel ist es möglich Bestellzettel ohne Namen und Adresse auszudrucken. Drückt man auf den Knopf «Leere Bestellzettel Drucken» wird ein PDF generiert, in dem die angegebene Anzahl leere Bestellzettel enthalten sind.

Bei den ausgefüllten Bestellzetteln wird ein Bestellzettel pro aktiven Kunden erstellt. Alle Bestellzettel befinden sich im selben PDF. Bei ausgefüllten Bestellzetteln werden Name, Adresse, Telefonnummer und Bemerkungen bereits ausgefüllt. In den Bemerkungen werden die beim Kunden hinterlegte Notizen eingefüllt, damit der Kunde bessere Kontrolle über seine Daten und allfällige Änderungen hat. Es ist möglich die Bestellzettel im PDF anhand einer Routen-Sammlung zu sortieren.

Bestellung Rollende Küche

Menüs (Datum von/bis): 14.6.2021 - 20.6.2021

Woche: 24

Name ..... Buktu ..... Strasse ..... Lampistrasse 1 .....

Vorname ..... Tim ..... Wohnort ..... Gotrigen 7424 .....

Telefon ..... 053 245 23 32 .....

	Menü A	Menü B	Spezial	Diabetes	Fleischlos
Montag					
Dienstag					
Mittwoch					
Donnerstag					
Freitag					
Samstag					

Bemerkungen

Karrotenallergie

Hinterlgete Notizen

Spitex Verein Wädenswil
Gerberstrase 6
8820 Wädenswil
Telefon: 044 783 93 23

Fusszeile

Abbildung 20: Beispiel eines ausgefüllten Bestellzettels

Bestellungen eingeben

Sobald die Bestellzettel vom Kunden ausgefüllt zurückkommen, können die Bestellungen für die entsprechende Woche eingegeben werden. Dazu muss im Navigationsbereich auf «Neue Bestellung» gedrückt werden. Das Programm navigiert dann zu einer Ansicht, in der alle Bestellungen aller aktiven Kunden eingegeben und überprüft werden müssen.

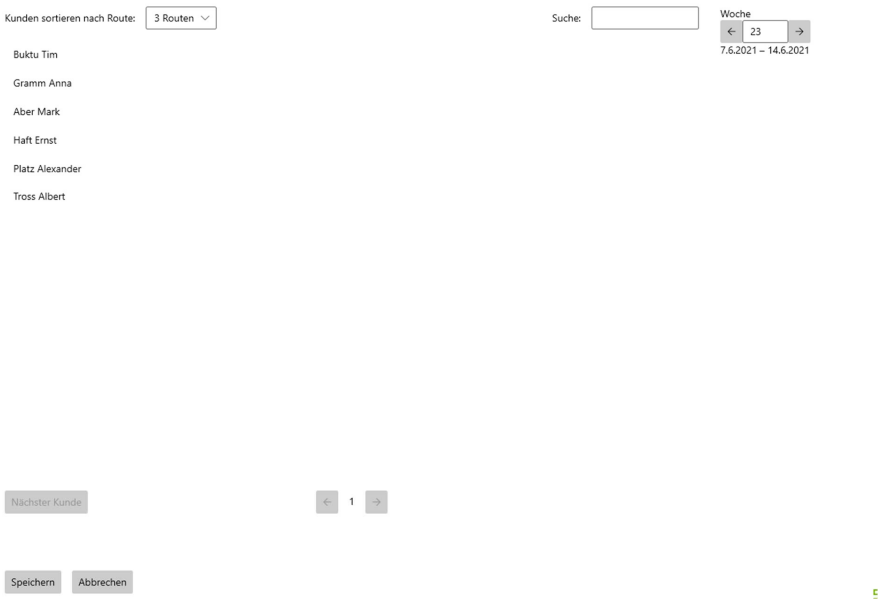


Abbildung 21: Eingabe der Bestellungen

Zuerst muss die Woche angegeben werden, für die die Bestellungen gemacht werden. Dies kann oben rechts eingestellt werden. Üblicherweise sind die Bestellzettel in der Reihenfolge der zuletzt abgefahrenen Routen. Entsprechend kann oben eine Standardrouten-Sammlung ausgewählt werden. Die Kunden werden dann entsprechend sortiert. Nun muss für **jeden** Kunden die Bestellung kontrolliert und angepasst werden. Im Idealfall startet man zuoberst, indem man den ersten Kunden in der Liste anklickt. Sobald man einen Kunden anklickt, ändert sich das Fenster. Es wird nun unterhalb der Kundenliste die Bestelltabelle für den selektierten Kunden angezeigt. Die Tabelle ist bereits vorbereitet mit den Standardbestellungen des ausgewählten Kunden.

Achtung: Falls es bereits Bestellungen für die ausgewählte Woche gibt, werden diese beim Speichern überschrieben!

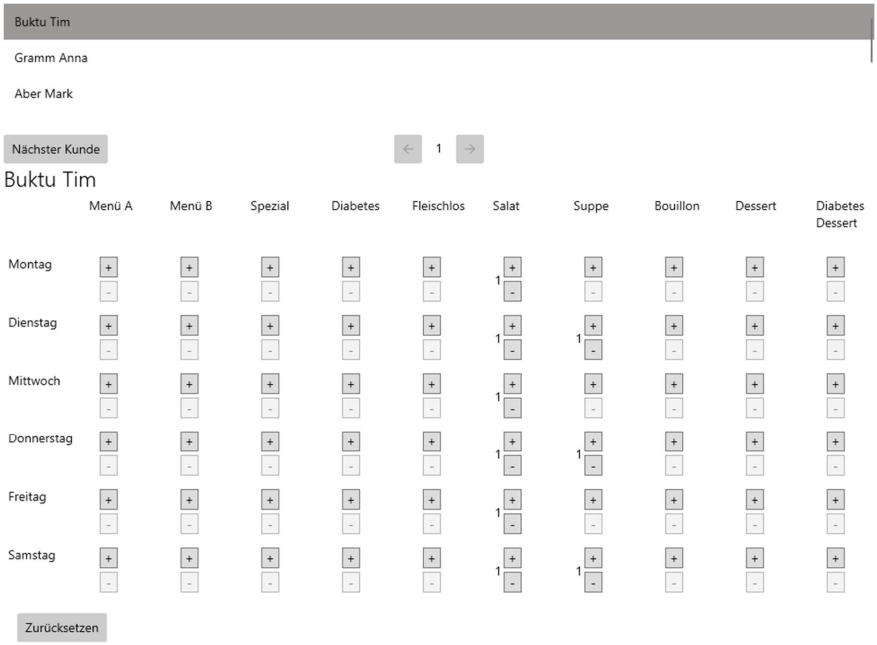


Abbildung 22: Bestelltabelle für ausgewählten Kunden

Im Beispiel in Abbildung 22 hat der ausgewählte Kunde in seinen Standardbestellungen jeden Tag einen Salat und dienstags, donnerstags und samstags eine Suppe. Dies wird entsprechend direkt in die Bestelltabelle übernommen. Am unteren Rand hat es einen Knopf «Zurücksetzen», der verwendet werden kann, um die Bestelltabelle des ausgewählten Kunden zu leeren. Dies kann beispielsweise verwendet werden, falls ein Kunde für eine Woche keine Bestellungen hat.

Achtung: Hat der Kunde an einem Tag keine Bestellung muss auch darauf geachtet werden, dass die ganze Reihe leer ist. Falls ein Kunde kein Menü bestellt aber trotzdem eine 1 bei Salat angegeben wird, wird eine Bestellung für einen Salat erstellt und der Kunde wird in der Routenplanung und dem Tagesplan auftauchen!

Wie bereits bei den Standardbestellungen im Kunden können bei der Bestelltabelle für das angewählte Gericht, Optionen ausgewählt werden und Hinweise kontrolliert und angepasst werden. Sowohl Optionen als auch die Hinweise werden von den hinterlegten Daten des ausgewählten Kunden übernommen, falls vorhanden. Die Variationen müssen für jedes Gericht und jeden **Tag** separat angegeben werden. Die Hinweise werden pro Tag erfasst.

Abbildung 23: Bestelltabelle mit Variationen und Notizen

Im Beispiel in Abbildung 23 wird für den Samstag das Menü A konfiguriert. In den hinterlegten Daten für den Kunden «Buktu Tim» hat es keine angegebenen Standardwerte für die Variation Grösse. Falls der Kunde eine grosse Portion möchte, kann dies nun auf der rechten Seite noch angegeben werden. Die Schöpfhinweise wurden von den hinterlegten Daten automatisch übernommen.

Sobald man die Bestellungen für den Kunden korrekt erfasst hat, kann man entweder in der Kundenliste über der Bestelltabelle den nächsten Kunden von Hand auswählen, oder einfach den Knopf «Nächster Kunde» drücken, welcher den nächsten Kunden in der Liste auswählt.

Sobald man alle Bestellungen für alle Kunden erfasst hat, muss man auf «Speichern» drücken. Erst dann werden die Bestellungen korrekt gespeichert.

Empfehlung: Mit dem Knopf «Nächster Kunde» arbeiten, damit sicher alle Kunden mindestens einmal kontrolliert werden.

Wochenübersicht für die Küche ausdrucken

Sobald die Bestellungen erfasst sind, kann die Wochenübersicht ausgedruckt werden. Um eine Wochenübersicht zu erstellen, muss man zuerst in der Navigation auf «Wochenübersicht» drücken, und in der dann angezeigten Konfigurationsmaske auswählen für welche Woche die Wochenübersicht erstellt werden soll und welche Kategorien von Gerichten dabei beachtet werden sollen.

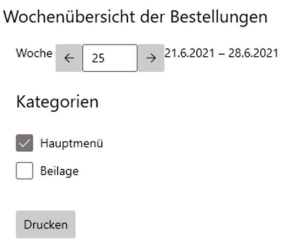


Abbildung 24: Konfiguration Wochenübersicht

Sobald die Konfiguration gemacht wurde, kann man auf den Knopf «Drucken» drücken.

Rollende Küche Wädenswil

Summen der Bestellungen der Woche 25: 21.6.2021 - 28.6.2021

	Menü A	Menü B	Spezial	Diabetes	Fleischlos
Montag	19	28	16	11	3
Dienstag	22	19	25	8	5
Mittwoch	15	27	27	12	7
Donnerstag	13	17	29	9	3
Freitag	35	21	13	7	4
Samstag	29	41	21	10	5

Abbildung 25: Beispiel einer Wochenübersicht

Lieferung Planen

Sobald die Bestellungen erfasst sind, kann eine Lieferung geplant werden. Dazu muss man in der Navigation auf «Neue Lieferung Planen drücken». Das Programm navigiert dann auf eine neue Eingabemaske. Dort muss das Datum der Lieferung angegeben werden und eine Standard Routen-Sammlung die als Vorlage für die neue Lieferung verwendet werden soll. Sobald man die entsprechenden Informationen angegeben hat, kann man auf den Knopf «Planen» drücken.

Abbildung 26: Eingabemaske, um eine neue Lieferung zu planen

Hat man «Planen» gedrückt erscheint eine neue Eingabemaske. In dieser wird oben das Datum der Lieferung angezeigt. Ausserdem werden die von der Vorlage erstellten Routen angezeigt werden. In den erstellten Routen befinden sich nur noch Kunden, die am gewählten Datum auch eine Bestellung haben. So kann es vorkommen, dass es Routen gibt mit deutlich mehr Kunden als andre Routen. Dies kann allerdings, wie im Kapitel Bearbeiten von Standardrouten beschrieben, angepasst werden. Änderungen beim Planen einer Lieferung haben **keinen** Einfluss auf die Vorlage. Falls es Kunden gibt, die eine Bestellung am gewählten Datum haben, aber noch nicht in einer Route sind, wird dies in Form

einer Warnung auf der rechten Seite dargestellt. Sobald alle Routen fertig konfiguriert sind und alle Kunden in einer Route sind, kann man den Vorgang mit einem Knopfdruck auf «Speichern» beenden.

Abbildung 27: Eingabemaske für das Planen einer Lieferung

Kurzfristige Korrekturen

Falls ein Kunde nach dem Erfassen der Bestellungen noch eine Änderung haben möchte, kann dies ohne Probleme angepasst werden. Dazu kann man im Navigationsbereich auf den Knopf «Bestellliste» drücken. In dem Bereich werden alle die Bestellungen pro Woche dargestellt. Nun kann man die Woche anwählen, in der eine Änderung gemacht werden möchte und unter der Liste auf den Knopf «Bearbeiten» drücken. Als nächstes kann man rechts über der Kundenliste den Namen des Kunden eingeben und den Kunden in der Liste selektieren. Es wird sich dann wieder die Bestelltabelle wie bereits in Kapitel Bestellungen eingeben vorgestellt öffnen, wo die Änderungen nun eingegeben werden können.

Falls ein Kunde seine Bestellungen an einem Tag abbestellt, muss dies zuerst in der Bestellliste angepasst werden und falls die Lieferung für den Tag bereits geplant wurde, muss der Kunde aus der Route in der Lieferung entfernt werden.

Kommt ein neuer Kunde spontan dazu, kann dieser ganz normal im Kundenstamm hinzugefügt werden. Dann kann die Situation wie eine Bestellungsänderung behandelt werden.

Tagesplan erstellen

Sobald die Lieferung für ein Tag geplant wurde, kann der entsprechende Tagesplan erstellt werden. Dazu muss man im Navigationsbereich auf «Tagespläne» drücken. Nun muss der Tag der Lieferung ausgewählt werden. Bei «Menüs Links» wird angegeben, welche Gerichte Kategorie in den Linken Spalten des Tagesplans dargestellt werden.

Tagesplan Rollende Küche Wädenswil

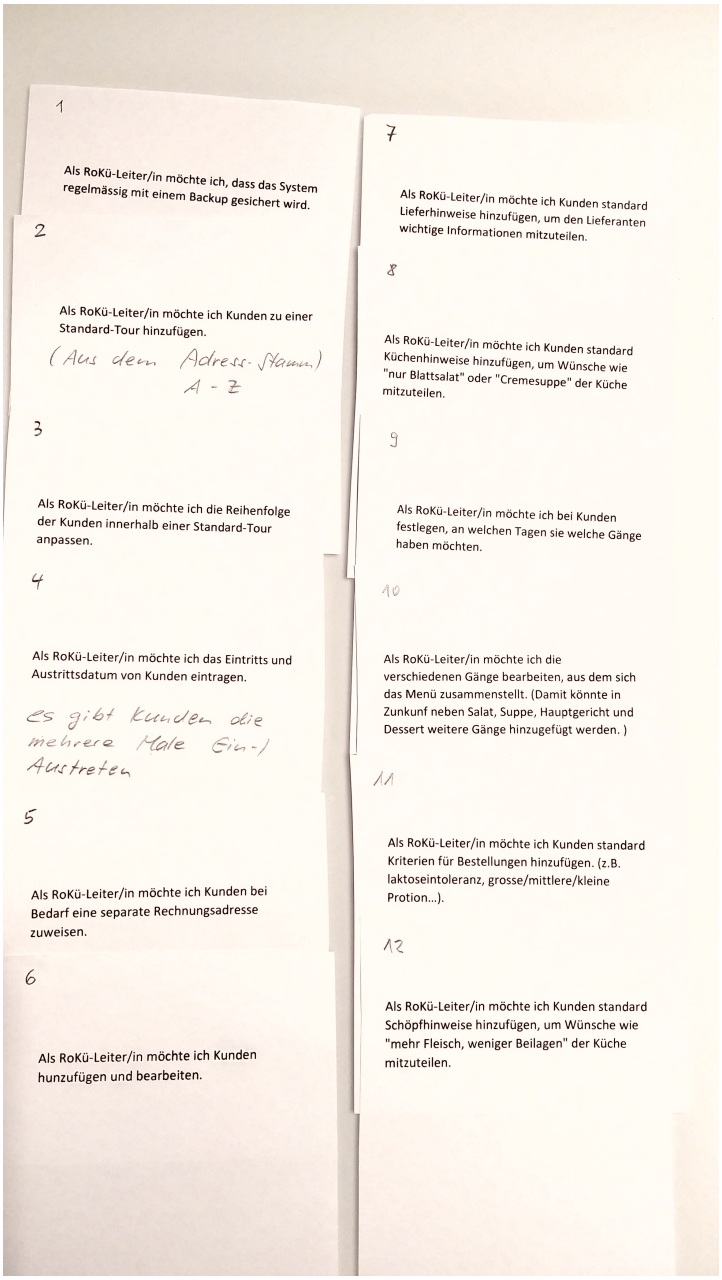
Route 1 - Lisa Spalten Links ***21.6.2021*** ***Woche 25***

Kunde	Menü A	Menü B	Spezial	Diabetes	Fleischlos	Schöpfhinweise	Salat	Suppe	Bouillon	Dessert	Diabetes Dessert	Küchenhinweise	Verteilhinweise
Buktu Tim Lampistrasse 1 053 245 23 32	1g					Karrotenallergie	1						
Gramm Anna Russlistrasse 2 023 343 12 32	1								1	1			
Total Boxen: 2	**2**	**0**	**0**	**0**	**0**		**1**	**0**	**1**	**1**	**0**		

Abbildung 28: Tagesplan mit Hauptmenüs in den linken Spalten

D. Use Case Prioritization

This is the prioritization, made by the customer.



1

Als RoKü-Leiter/in möchte ich, dass das System regelmässig mit einem Backup gesichert wird.

2

Als RoKü-Leiter/in möchte ich Kunden zu einer Standard-Tour hinzufügen.

(Aus dem Adress-Stamm)
A - Z

3

Als RoKü-Leiter/in möchte ich die Reihenfolge der Kunden innerhalb einer Standard-Tour anpassen.

4

Als RoKü-Leiter/in möchte ich das Eintritts und Austrittsdatum von Kunden eintragen.

es gibt Kunden die mehrere Male Ein-/ Austreten

5

Als RoKü-Leiter/in möchte ich Kunden bei Bedarf eine separate Rechnungsadresse zuweisen.

6

Als RoKü-Leiter/in möchte ich Kunden hunzufügen und bearbeiten.

7

Als RoKü-Leiter/in möchte ich Kunden standard Lieferhinweise hinzufügen, um den Lieferanten wichtige Informationen mitzuteilen.

8

Als RoKü-Leiter/in möchte ich Kunden standard Küchenhinweise hinzufügen, um Wünsche wie "nur Blattsalat" oder "Cremesuppe" der Küche mitzuteilen.

9

Als RoKü-Leiter/in möchte ich bei Kunden festlegen, an welchen Tagen sie welche Gänge haben möchten.

10

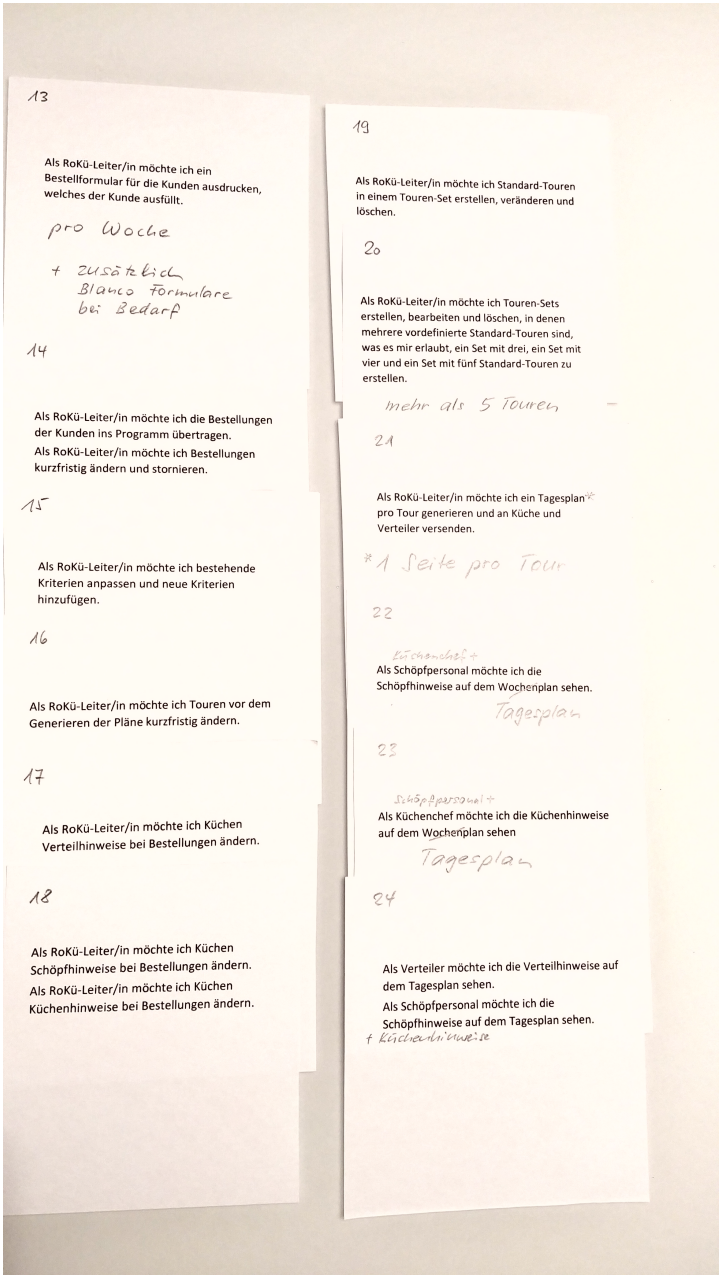
Als RoKü-Leiter/in möchte ich die verschiedenen Gänge bearbeiten, aus dem sich das Menü zusammenstellt. (Damit könnte in Zunkunf neben Salat, Suppe, Hauptgericht und Dessert weitere Gänge hinzugefügt werden.)

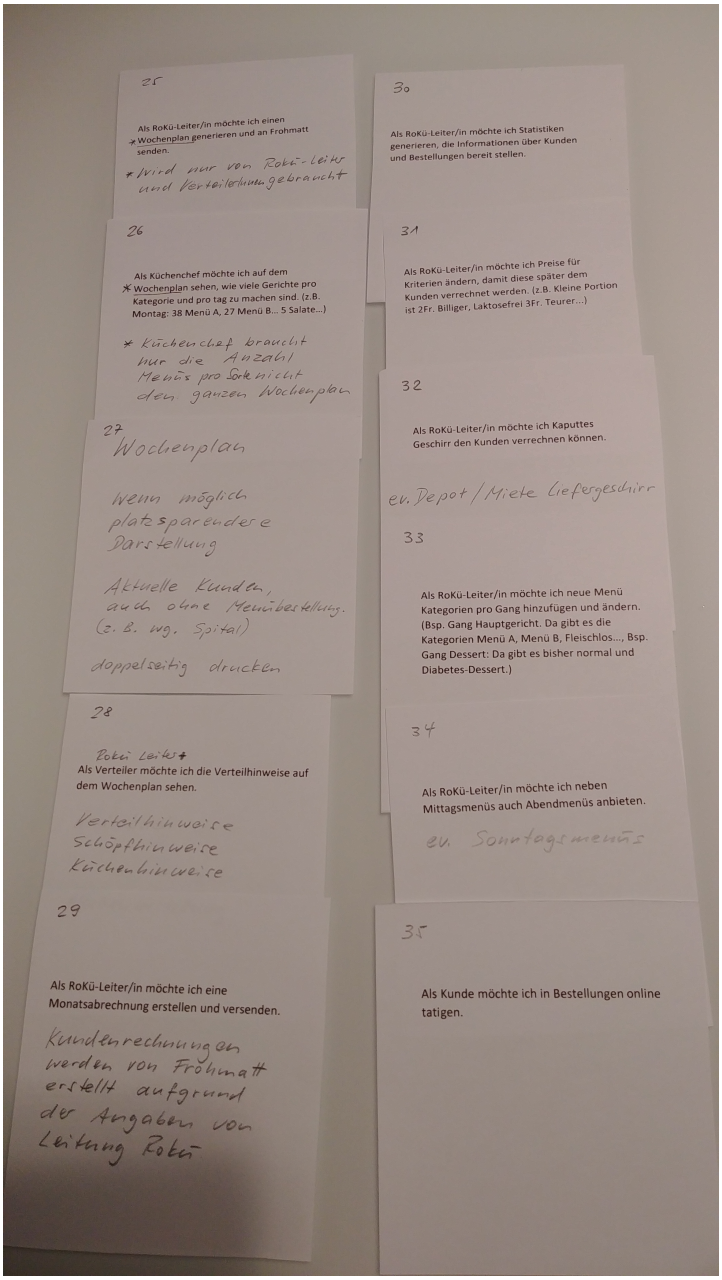
11

Als RoKü-Leiter/in möchte ich Kunden standard Kriterien für Bestellungen hinzufügen. (z.B. laktoseintoleranz, grosse/mittlere/kleine Protion...).

12

Als RoKü-Leiter/in möchte ich Kunden standard Schöpfhinweise hinzufügen, um Wünsche wie "mehr Fleisch, weniger Beilagen" der Küche mitzuteilen.


13
Als RoKü-Leiter/in möchte ich ein Bestellformular für die Kunden ausdrucken, welches der Kunde ausfüllt.
pro Woche
+ zusätzlich Blanco Formulare bei Bedarf
14
Als RoKü-Leiter/in möchte ich die Bestellungen der Kunden ins Programm übertragen.
Als RoKü-Leiter/in möchte ich Bestellungen kurzfristig ändern und stornieren.
15
Als RoKü-Leiter/in möchte ich bestehende Kriterien anpassen und neue Kriterien hinzufügen.
16
Als RoKü-Leiter/in möchte ich Touren vor dem Generieren der Pläne kurzfristig ändern.
17
Als RoKü-Leiter/in möchte ich Küchen Verteilhinweise bei Bestellungen ändern.
18
Als RoKü-Leiter/in möchte ich Küchen Schöpfhinweise bei Bestellungen ändern.
Als RoKü-Leiter/in möchte ich Küchen Küchenhinweise bei Bestellungen ändern.
19
Als RoKü-Leiter/in möchte ich Standard-Touren in einem Touren-Set erstellen, veränderen und löschen.
20
Als RoKü-Leiter/in möchte ich Touren-Sets erstellen, bearbeiten und löschen, in denen mehrere vordefinierte Standard-Touren sind, was es mir erlaubt, ein Set mit drei, ein Set mit vier und ein Set mit fünf Standard-Touren zu erstellen.
mehr als 5 Touren
21
Als RoKü-Leiter/in möchte ich ein Tagesplan* pro Tour generieren und an Küche und Verteiler versenden.
*1 Seite pro Tour
22
Küchenchef +
Als Schöpfpersonal möchte ich die Schöpfhinweise auf dem Wochenplan sehen.
Tagesplan
23
Schöpfpersonal +
Als Küchenchef möchte ich die Küchenhinweise auf dem Wochenplan sehen
Tagesplan
24
Als Verteiler möchte ich die Verteilhinweise auf dem Tagesplan sehen.
Als Schöpfpersonal möchte ich die Schöpfhinweise auf dem Tagesplan sehen.
+ Küchenhinweise


25
Als RoKü-Leiter/in möchte ich einen Wochenplan generieren und an Frohmatt senden.
* Wird nur von Roki-leiter und Verteilerkuche gebraucht
26
Als Küchenchef möchte ich auf dem Wochenplan sehen, wie viele Gerichte pro Kategorie und pro tag zu machen sind. (z.B. Montag: 38 Menü A, 27 Menü B... 5 Salate...)
* Küchenchef braucht nur die Anzahl Menüs pro Sorte nicht den ganzen Wochenplan
27
Wochenplan
Wenn möglich platzsparendere Darstellung
Aktuelle Kunden, auch ohne Menübestellung. (z. B. vg. Spital)
doppelseitig drucken
28
Roki Leiter+
Als Verteiler möchte ich die Verteilhinweise auf dem Wochenplan sehen.
Verteilhinweise
Schöpfhinweise
Küchenhinweise
29
Als RoKü-Leiter/in möchte ich eine Monatsabrechnung erstellen und versenden.
Kundenrechnungen werden von Frohmatt erstellt aufgrund der Angaben von Leitung Roki
30
Als RoKü-Leiter/in möchte ich Statistiken generieren, die Informationen über Kunden und Bestellungen bereit stellen.
31
Als RoKü-Leiter/in möchte ich Preise für Kriterien ändern, damit diese später dem Kunden verrechnet werden. (z.B. Kleine Portion ist 2Fr. Billiger, Laktosefrei 3Fr. Teurer...)
32
Als RoKü-Leiter/in möchte ich Kaputtes Geschirr den Kunden verrechnen können.
ev. Depot / Miete Liefergeschirr
33
Als RoKü-Leiter/in möchte ich neue Menü Kategorien pro Gang hinzufügen und ändern. (Bsp. Gang Hauptgericht. Da gibt es die Kategorien Menü A, Menü B, Fleischlos..., Bsp. Gang Dessert: Da gibt es bisher normal und Diabetes-Dessert.)
34
Als RoKü-Leiter/in möchte ich neben Mittagsmenüs auch Abendmenüs anbieten.
ev. Sonntagsmenüs
35
Als Kunde möchte ich in Bestellungen online tatigen.

E. Early Documents

RoKü Prospect

Unverbindliche Auskunft

Rollende Küche Wädenswil

Rufen Sie uns an! Wir beraten Sie gerne und besprechen gemeinsam alles Weitere:

SpitexVerein Wädenswil
Gerbestrasse 6
8820 Wädenswil
Telefon 044 783 93 23
Fax 044 783 93 24

info@spitex-waedenswil.ch
www.spitex-waedenswil.ch

SPITEX
Hilfe und Pflege zu Hause

stadt wädenswil
Frohmatt
Lebensqualität im Alter

Gesund und ausgewogen essen

Frisch zubereitet und warm nach Hause geliefert

Gesund und ausgewogen essen ist für alle wichtig, insbesondere für:

- Seniorinnen und Senioren
- Langzeitkranke und Behinderte
- Diabetikerinnen und Diabetiker
- Rekonvaleszentinnen und Rekonvaleszenten

Wir liefern Ihnen von Montag bis Samstag feines, warmes Essen direkt ins Haus. Die Menüs werden täglich im Alterszentrum Frohmatt frisch zubereitet. Der saisonale Menüplan bietet für jeden Geschmack etwas und gliedert sich wie folgt:

- Menü A
- Menü B
- Spezialmenü (je nach Saison)
- Menü für Altersdiabetiker
- Fleischloses Menü

Dazu servieren wir eine feine Suppe sowie einen Tagessalat oder ein Dessert. Bei Bedarf bieten wir das Essen fein geschnitten (gabelfertig) oder in pürierter Form an.

Ein massgeschneidertes Angebot

- Sie erhalten Ihre Menüs jeweils ab 11:15 Uhr bis spätestens 13:00 Uhr
- Die Anzahl der zu liefernden Essen und die Liefertage bestimmen Sie.
- Die Mahlzeiten werden in Isolierboxen transportiert.
- Unsere Mitarbeiterinnen und Mitarbeiter informieren Sie gerne.

Die Bestellung

Ihre Mahlzeitenbestellung

- Montag bis Freitag von 08:00 Uhr bis 11:30 Uhr nimmt eine Mitarbeiterin im Büro des SpitexVereins Wädenswil Ihre Bestellung entgegen.
- Mahlzeiten, die bis 11:30 Uhr bestellt werden, liefern wir bereits am Folgetag ins Haus. (Für die Lieferung vom Montag ist der Bestelltermin am Freitag bis 17:00 Uhr).
- Bitte melden Sie Ihre Änderungswünsche für die Mahlzeiten telefonisch bis 11:30 Uhr des Vortages oder wenden Sie sich an Ihre Verträgerin oder den Verträger. Verspätete Meldungen können nicht berücksichtigt werden, zu spät annullierte Lieferungen müssen verrechnet werden.

Preise der Mahlzeiten

Eine warme Mahlzeit, zu Ihnen ins Haus geliefert, kostet CHF 15.50.

Verrechnung der Mahlzeiten

Für die Mahlzeiten erhalten Sie eine detaillierte Monatsrechnung, zahlbar innert 30 Tagen.

Die Verträgerinnen und Verträger werden für ihre Arbeit bezahlt und dürfen daher keine Trinkgelder und Geschenke annehmen.

Porzellan-Liefergeschirr

Die Rollende Küche Wädenswil liefert die Mahlzeiten im Porzellangeschirr aus. Wir bitten Sie, folgende Punkte bezüglich der Handhabung des Porzellangeschirrs zu beachten:

- Porzellanteller und Porzellanschalen bitte ausspülen.
- Bitte verschliessen Sie nach dem Essen die Gefässe mit den Deckeln und stellen Sie diese wieder in die Isolierbox.
- Teller und Schalen **nicht** auf die heisse Herdplatte stellen.
- Isolierbox nicht auf die heisse Herdplatte stellen und nicht in den Backofen stellen.
- **Alle** Geschirrteile jedes Mal der Verträgerin, dem Verträger übergeben. Fehlende Teile werden verrechnet.
- Teller können mit Deckel in die Mikrowelle eingestellt werden.
- Die Geschirr- und Systemteile sind sehr wertvoll.

Sollten doch einmal durch falsche Handhabung Teile bei Ihnen zu Hause zu Bruch gehen oder unbrauchbar werden, müssen wir diese berechnen. Folgend eine Preisübersicht zu Ihrer Information:

Teller:	Fr. 35.00
Salat-, Dessertschale:	Fr. 15.00
Suppenschale:	Fr. 19.00
Kleiner Deckel:	Fr. 10.50
Großer Deckel:	Fr. 23.00
Isolierbox:	Fr. 108.00

Falls Ihnen die Handhabung noch unklar sein sollte, zögern Sie nicht, eine Verträgerin, ein Verträger zu Fragen.

Rollende Küche Wädenswil

Arbeitsablauf
und
Programm Doku

Access Datenbank
Version Nr. 2.1.29 ab Januar 2002
und Erweiterung ab Februar 2011

Inhaltsverzeichnis

Rollende Küche

Muster

Arbeitsablauf

Montag bis Samstag

Zwischen 13.00 und 13.30 Uhr Telefonbeantworter abhören / Mail Telefonzentrale

Neuanmeldungen / Abbestellungen bearbeiten

Tagespläne für den nächsten Tag erstellen und an Frohmatt mailen

Mail-Adresse: **Gruppe „Tagespläne“**

Donnerstag

Menüplan für die nächste Woche
Menüpläne werden von Küchenchef an Spitex und Rokü gemailt

Gedruckte Menüpläne werden von Küchenchef an die VerteilerInnen abgegeben und mit den Bestellzetteln zu den Kunden gebracht.

Bestellzettel für die nächste Woche
Die VerteilerInnen bringen die Bestellungen für die nächste Woche ca. um 13 Uhr ins Büro.

Neue Kunden
Die neuen Kunden im Rokü PC Programm aufnehmen und in entsprechende Tour einteilen.

Änderungen
Die Bestell-Änderungen spätestens vor dem Erstellen der Tagespläne im PC eintragen. (Menü-Abbestellungen z.B. wegen Spitaleintritt am gleichen Tag werden telefonisch in der Küche gemeldet).

Menübestellungen
Die Menübestellungen für die nächste Woche im PC eintragen

Formulare drucken
Wochenplan: 1 Plan für Büro Rokü
Pro Tour 1 Plan der betreffenden Tour für die VerteilerInnen, in die Tourenfächer in der Frohmatt legen

Letzte Seite des Wochen-planes:	Diese Seite ist eine Zusammenfassung der Menüs für die nächste Woche und wird von der Frohmatt benötigt, um die Menüs zu planen (im Frohmatt Küchen-Büro abgeben)
Bestellzettel:	für die nächste Woche (alle Kunden) ausdrucken und in die Tourenfächer in der Frohmatt legen.
Lieferliste:	wird nicht mehr verwendet.

Anfang Monat

- **Formular „Auswertungen für die Frohmatt“:**
  - Frohmatt Küche, R. Gräppi (mailen)
  - Frohmatt Büro, Manuel Keller (mailen) für Rechnungsstellung an Kunden
  - Büro RoKü (Ordner „Statistik“)
  - [redacted]

- **Buchhaltungsliste**
  - [redacted]
  - Frohmatt Büro, Manuel Keller mailen
  - Büro Rokü (Ordner „Buchhaltungslisten“)

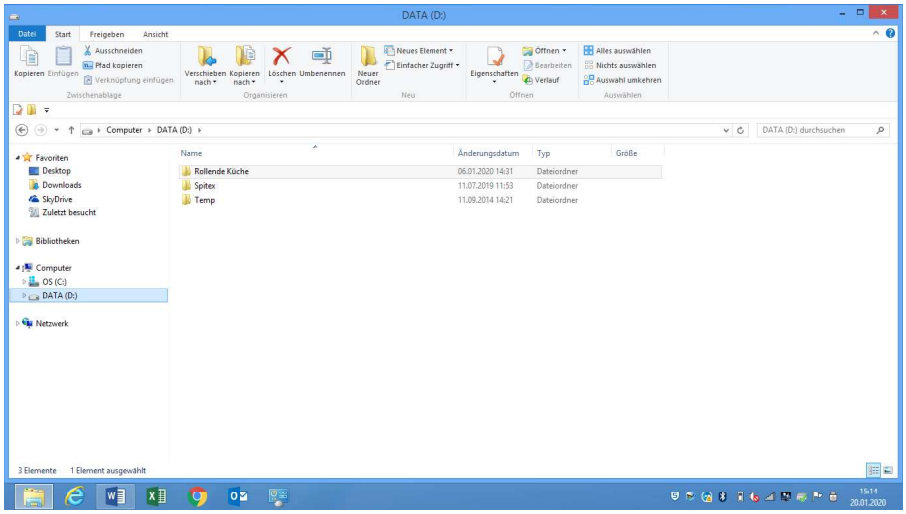
- **Formular „Anzahl Menüs“**
  - Büro Rokü (Ordner „Statistik“)
  - [redacted]
  - Frohmatt Büro, Manuel Keller (mailen)

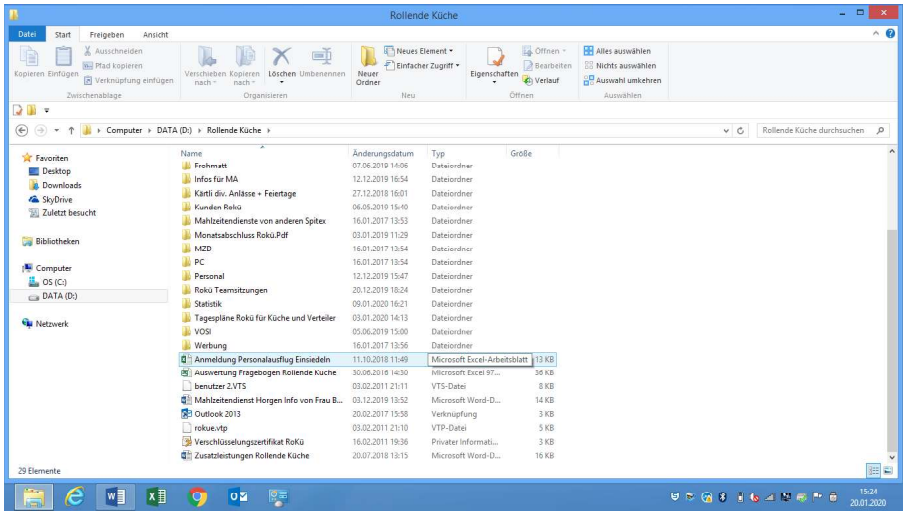
- **Formular „Kundenhinweise“**
  - Frohmatt Küche, Roger Gräppi (für Boxenkontrolle & Beschriftung)
  - Frohmatt Büro, Manuel Keller
  - [redacted]
  - Büro Rokü (Ordner „Statistik“)

- **Rechnungen und Einzahlungsscheine drucken (Frohmatt)**
  seit 1.1.2008 werden die Rechnungen von Frohmatt-Administration (aktuell Manuel Keller) erstellt und von den VerteilerInnen an die Kunden abgegeben.

Desktop Ansicht

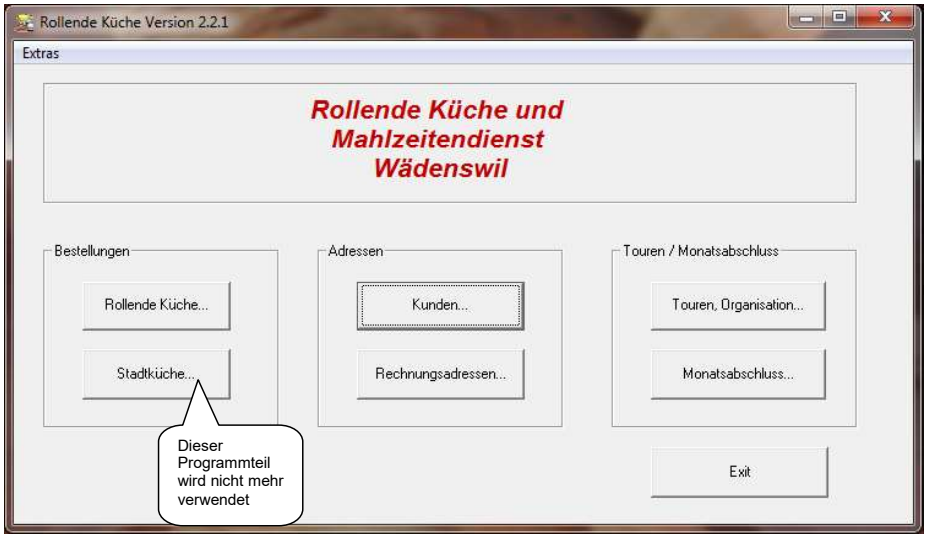
Explorer (= PC Daten-Inhaltsverzeichnis)

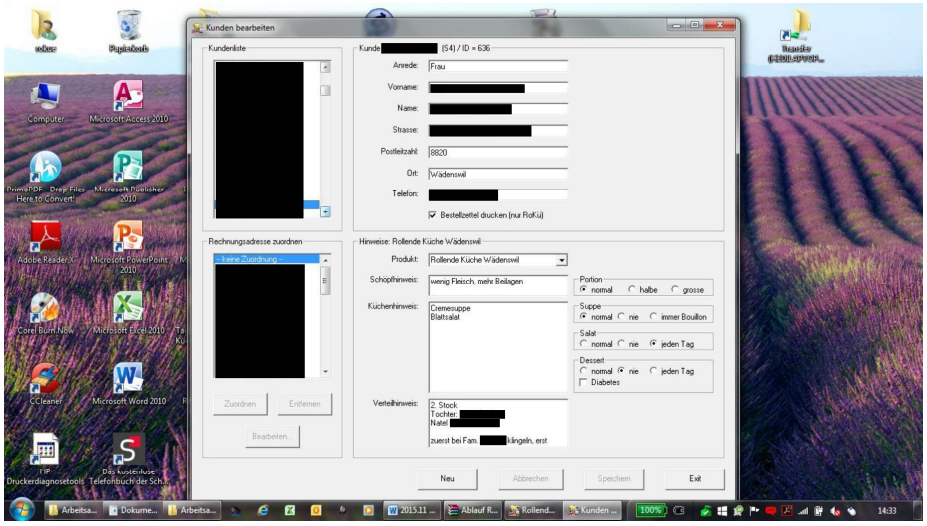




Kunden erfassen im Kundenstamm

(Programmteil Stadtküche wird nicht mehr verwendet)

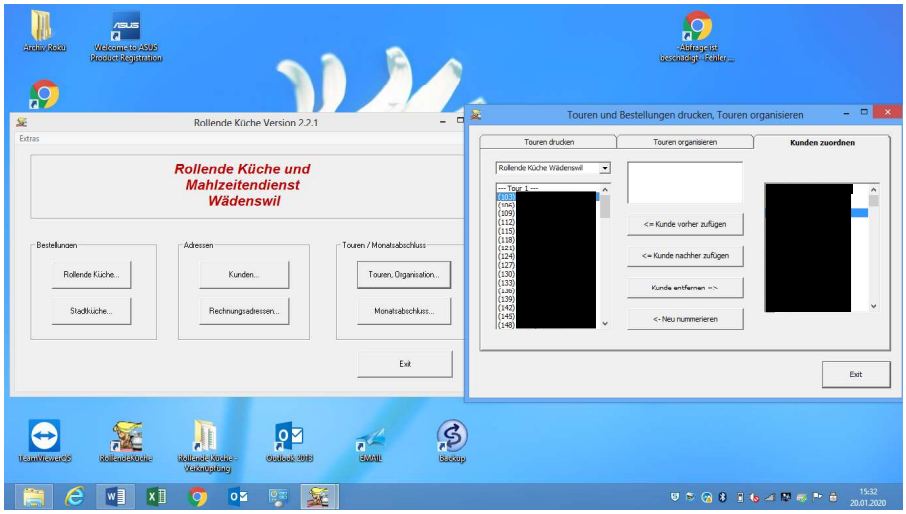




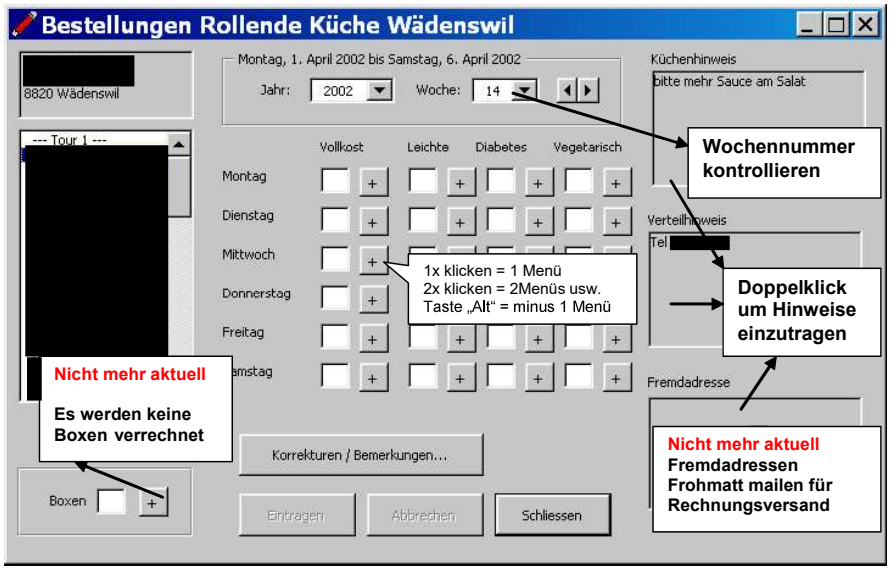
Rechnungsadresse erfassen: ist nicht mehr aktuell weil separate Adresse nicht auf den Rechnungsunterlagen für Frohmatt erscheinen. Falls Kunden die Rechnung nicht selber bezahlen muss die Rechnungsadresse mit den Unterlagen für die Rechnungen an Frohmatt, Administration, Manuel Keller gemailt werden.

Kunden in Tour einordnen

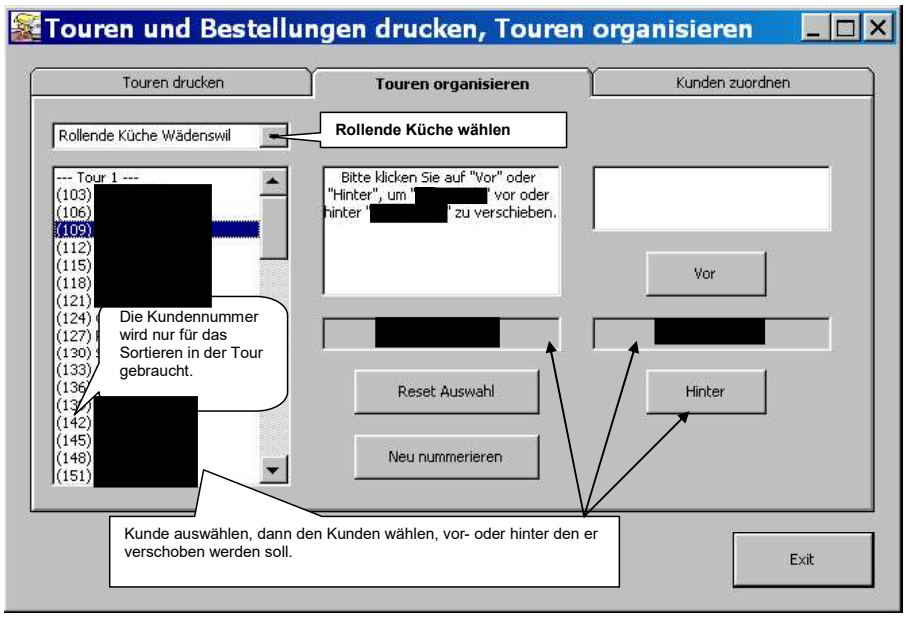
Schaltfläche Touren Organisation



Menübestellungen eintragen und Beilagen erfassen

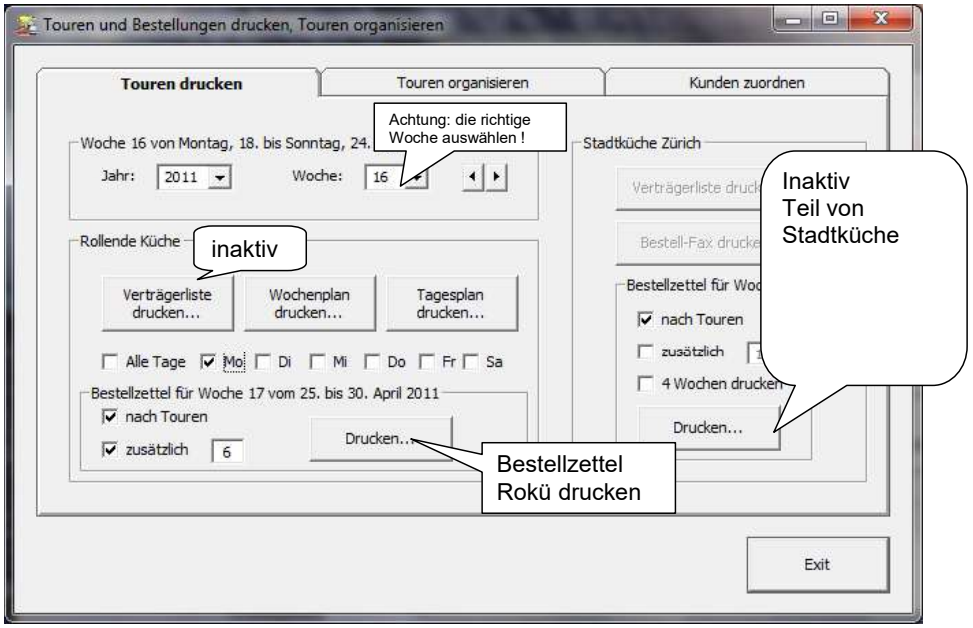


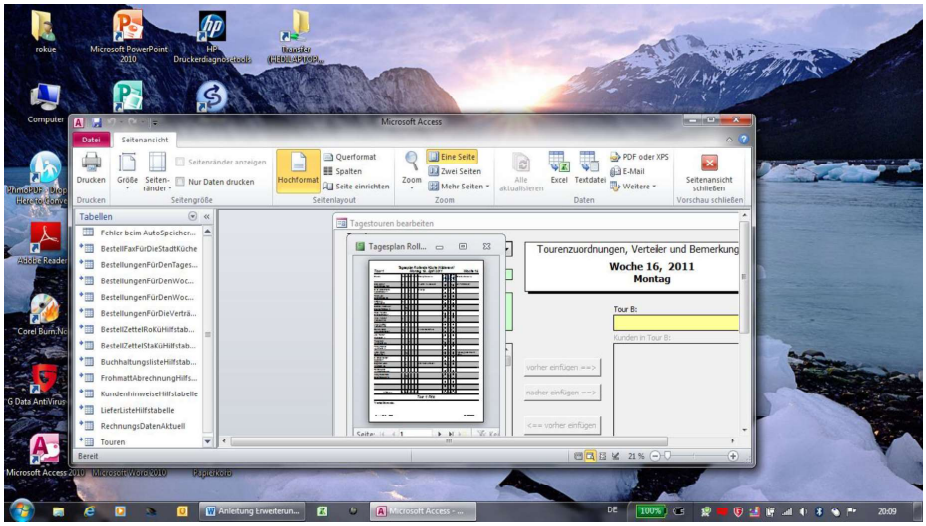
Touren organisieren, Kunden in andere Stammtour einordnen



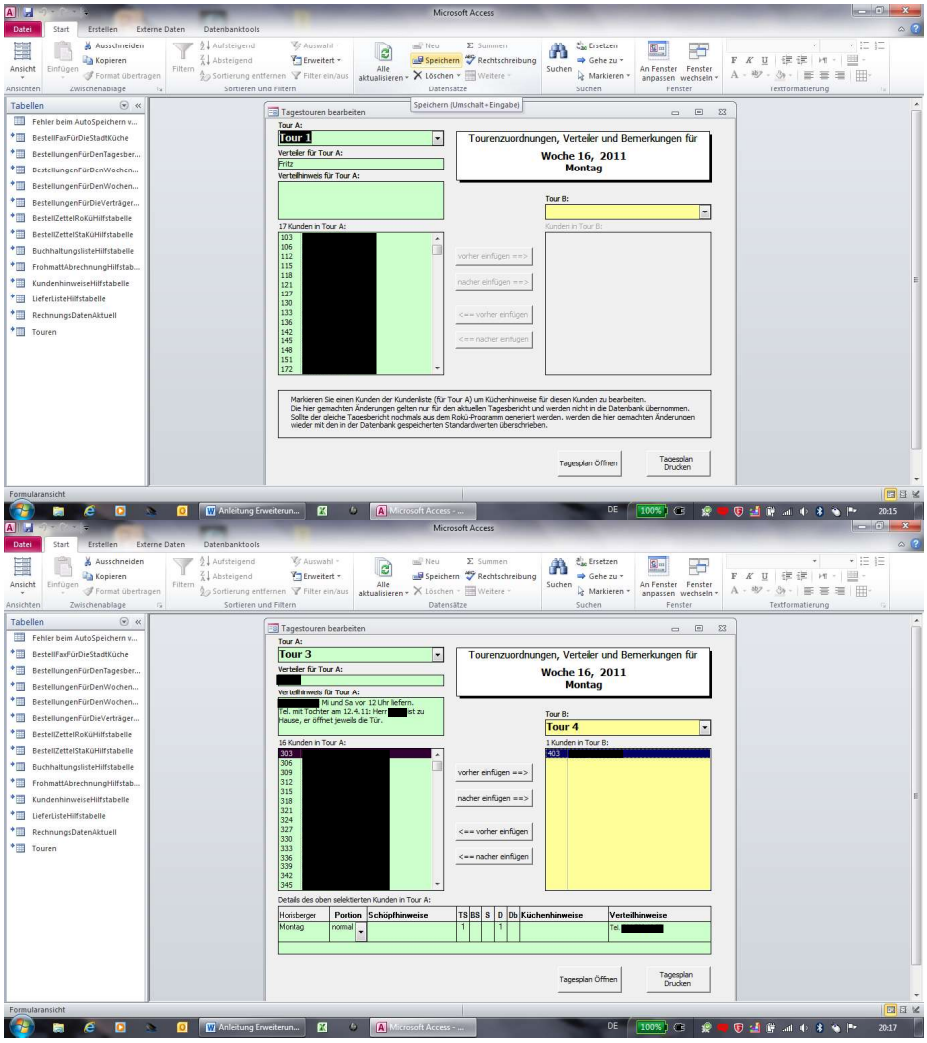
Tagespläne erstellen
Bestellzettel für Kunden drucken

Woche wählen / Tag wählen / Schaltfläche „Tagesplan drucken“ klicken





Im Hintergrund werden die Tagespläne geöffnet (mit Schaltfläche „Exit“ die Masken schliessen)
Es sind alle Kunden pro Tour aufgeführt die am gewählten Tag ein Menü haben.
Achtung: Verteiler und Verteilhinweise müssen kontrolliert und ev. angepasst werden.

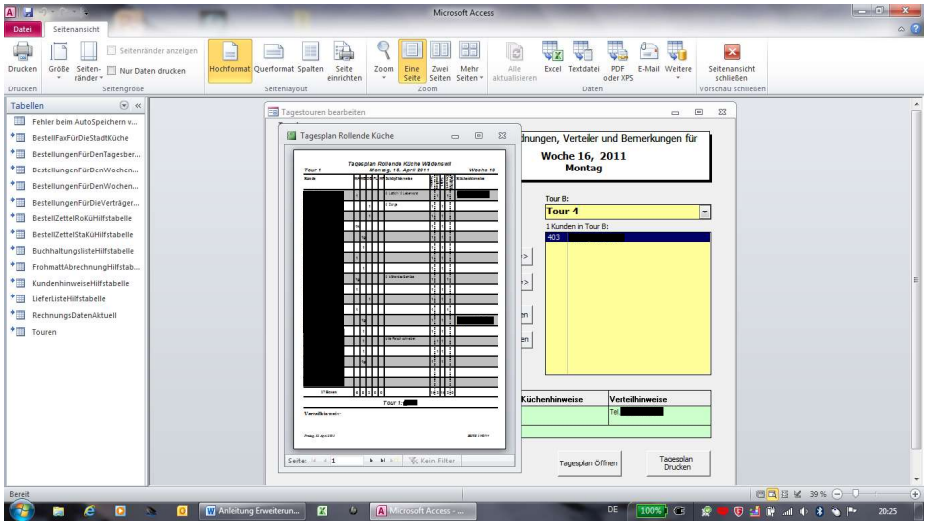


4. Tour erstellen: auf der rechten Seite Tour 4 wählen

Auf der linken Seite Tour wählen und Kunden auf Tour 4 einfügen.

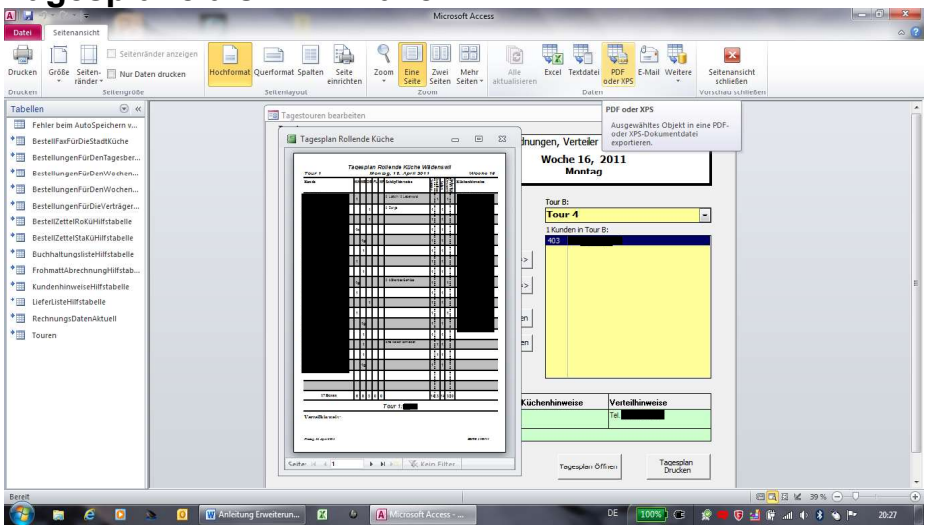
Falls die Reihenfolge der Kunden geändert werden muss kann auf beiden Seiten die gleiche Tour gewählt werden und die Kunden können in anderer Reihenfolge eingefügt werden.

Schaltfläche Tagesplan öffen: für Kontrolle, bei Korrekturen wieder schliessen.



Um die Tagespläne zu vergrössern muss der Cursor im Tagesplan geklickt werden. Am unteren Rand können die anderen Seiten gewählt werden.

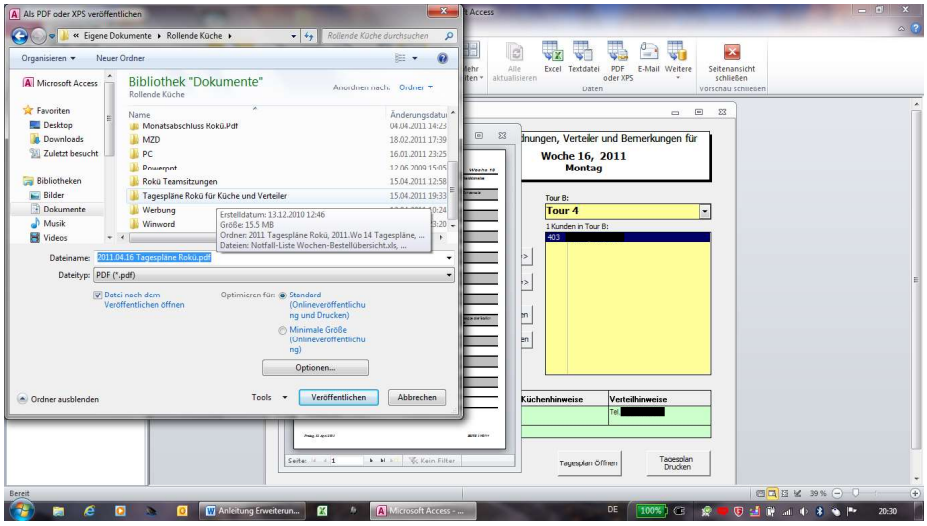
Tagespläne als PDF mailen:



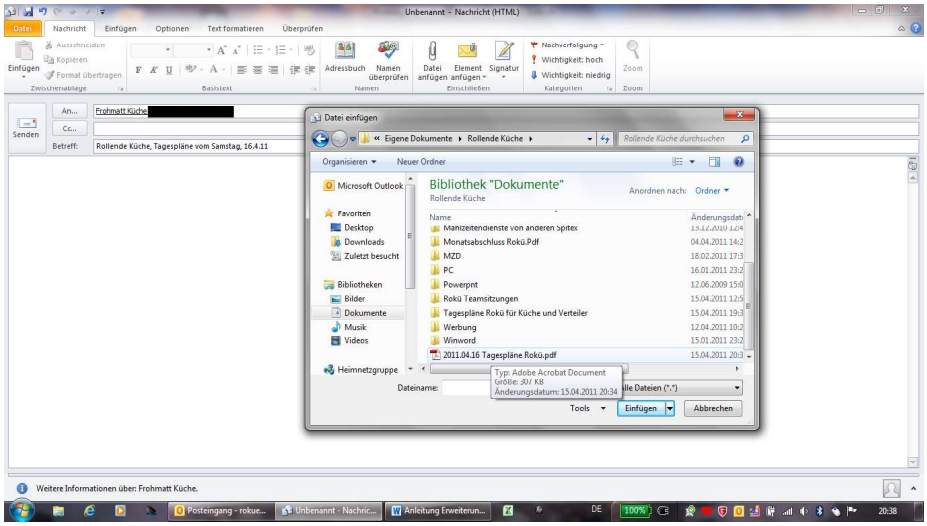
In der Menüleiste „PDF oder XPS“ anklicken

Dateiname eintragen / Ordner Tagespläne öffnen

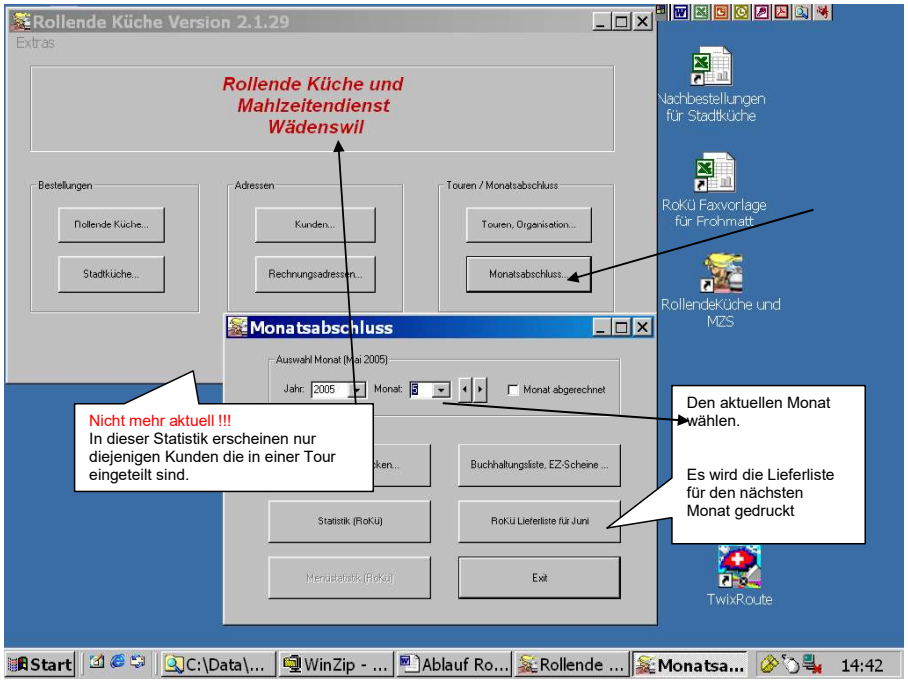
Optimieren für: „Standart“ wählen, dann Schaltfläche „Veröffentlichen“, dann oben rechts schliessen
und zurück zum Desktop.



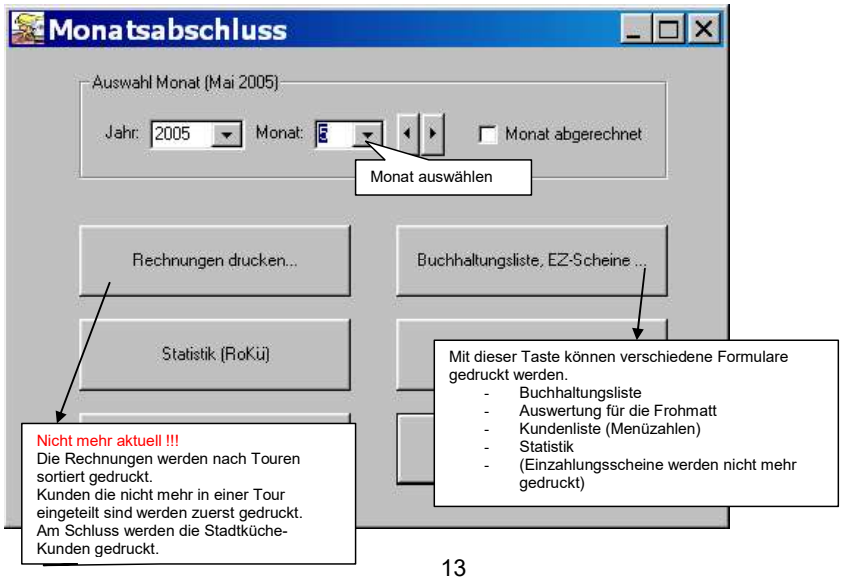
Auf dem Desktop „Outlook öffnen, Anhang (Büroklammer) anwählen und die Tagespläne einfügen.



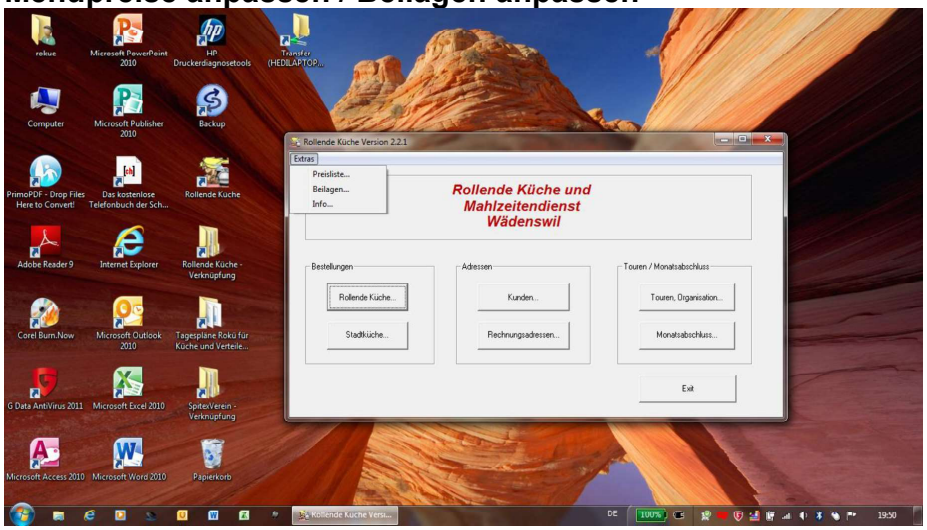
Lieferliste für den nächsten Monat erstellen
wird nicht mehr verwendet

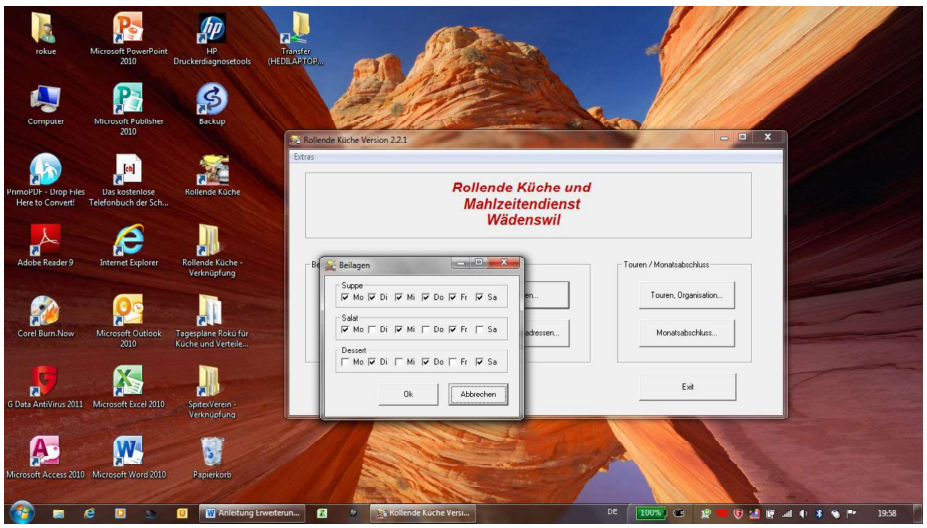


Monatsabschluss erstellen



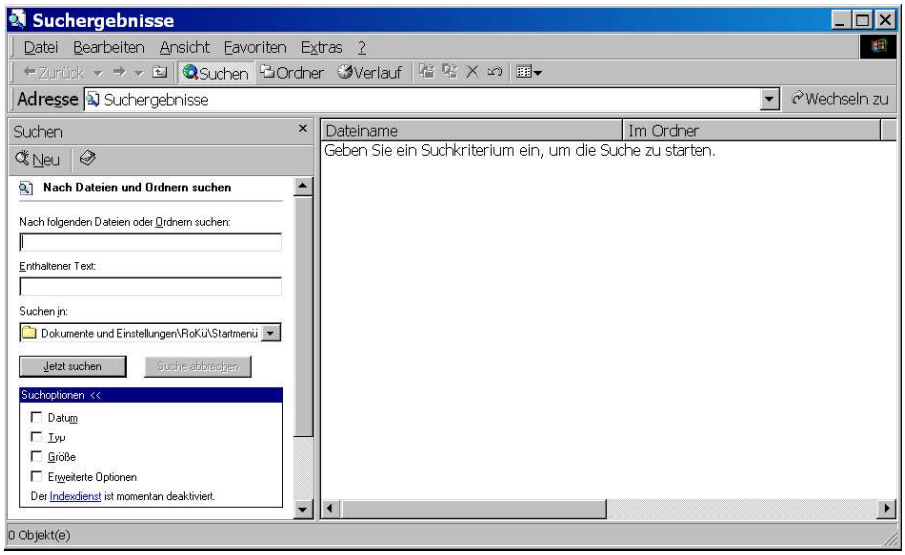
Menüpreise anpassen / Beilagen anpassen



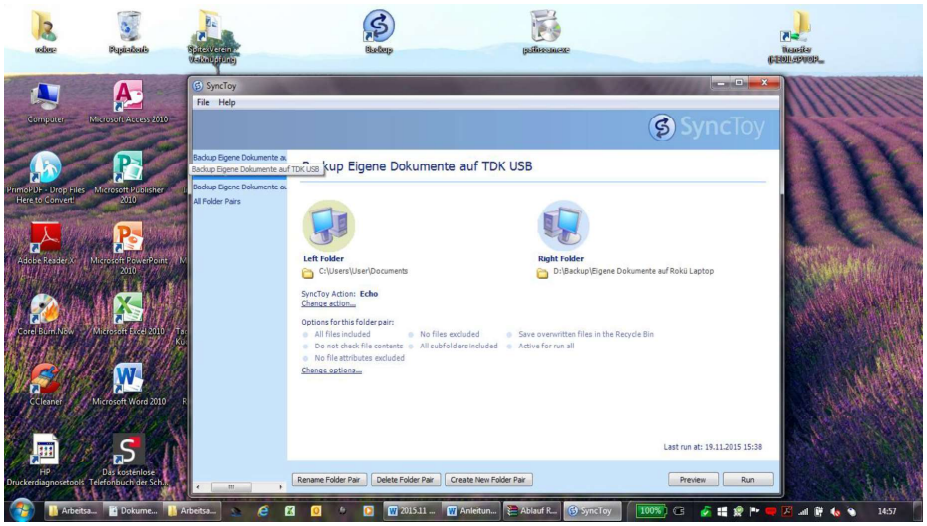


Dateien suchen

Rechtsklick auf den Startknopf, dann „Suchen“ auswählen



Daten sichern



- USB Stick anschliessen und die Daten von PC auf USB Stick übertragen
- USB Stick entfernen

Example Menuplan

Rollende Küche

	Menü A	Menü B	Spezialmenü	Alters-Diabetes	Fleischlos	Suppe/Salat/Dessert
Montag **01.02.2021**	Riz Casimir Pouletbrust-Streifen an Currysauce im Reisring Früchtegarnitur Salat	Cervelat gefüllt mit Käse und Speck umwickelt Risoléekartoffeln Fenchel Salat	Berner-Platte mit Rippli, Speck und Zungenwurst Salzkartoffeln Dörrbohnen Salat	Cervelat gefüllt mit Käse und Speck umwickelt Risoléekartoffeln Fenchel, Dörrbohnen Salat	Rassiges Gemüsecurry im Reisring gebratene Banane Früchtegarnitur Salat	Tagessuppe Salat
Dienstag **02.02.2021**	Rindsgulasch nach ungarischer Art Spätzli Wirzgemüse Nussschnecke	Pochiertes Dorschfilet an Kräutersauce Couscous Rüeblistäbli Nussschnecke	Zibelemärit-Chueche Knusprige Speck-Zwiebelwähe Gurkensalat mit Dill Nussschnecke	Rindsgulasch nach ungarischer Art Spätzli Wirzgemüse Rüeblistäbli Diabetiker Gebäck	Lauch- Käsewähe Gurkensalat Nussschnecke	Tagessuppe Nussschnecke
Mittwoch **03.02.2021**	Hausgemachter Hackbraten Rotweinsauce Stampfkartoffeln gelbe Bohnen Salat	Tortelloni mit Fleischfüllung an Käse-Rahmsauce Parmesan Cherry Tomatenragoût Salat	Berner Speck-Rösti mit Spiegelei Rahmspinat Salat	Hausgemachter Hackbraten Rotweinsauce Stampfkartoffeln gelbe Bohnen Salat	Vegetarischer Hackbraten Rotweinsauce Stampfkartoffeln gelbe Bohnen Salat	Bouillon Salat
Donnerstag **04.02.2021**	Pastetli gefüllt mit Fleischkügeli an Weissweinsauce Erbsli und Rüebli Marronicake	Wiener Kalbs-Rahmgulasch Semmelknödel Rotkraut Marronicake	Zanderfilet nach Berner Art mit Zwiebeln und Pilze Bratkartoffeln Kefen Marronicake	Wiener Kalbs-Rahmgulasch Semmelknödel Rotkraut Erbsli und Rüebli Diabetiker- Gebäck	Pastetli gefüllt mit einem sämigen Waldpilzragoût Erbsli und Rüebli Marronicake	Tagessuppe Marronicake
Freitag **05.02.2021**	Pangasius-Knusperli Remouladensauce Salzkartoffeln Blattspinat mit Pinienkernen Salat	Kasseler Braten Senfsauce gebratene Polentaschnitten Ratatouille Salat	Zibelemärit-Chueche Knusprige Speck-Zwiebelwähe Gurkensalat mit Dill Salat	Kasseler Braten gebratene Polentaschnitten Blattspinat Ratatouille Salat	Frittierter Broccoli Remouladensauce Salzkartoffeln Blattspinat mit Pinienkernen Salat	Bouillon Salat
Samstag **06.02.2021**	Schweins-Piccata nach Mailänder Art Tomatensauce feine Nüdeli Blumenkohl Schlorzichueche	"Schnitz und drunder" Eintopf mit Speck, Birnen und Kartoffeln Schlorzichueche	Berner Speck-Rösti mit Spiegelei Rahmspinat Schlorzichueche	Schweins-Piccata nach Mailänder Art Tomatensauce feine Nüdeli Blumenkohl, Spinat Diabetiker-Creme	Auberginen-Piccata nach Mailänder Art Tomatensauce feine Nüdeli Blumenkohl Schlorzichueche	Tagessuppe Schlorzichueche

Order Sheet

Bestellung Rollende Küche

Menüs vom 8. bis 13. Februar 2021

Woche 6

Name [redacted] Strasse [redacted]

Vorname [redacted] Wohnort 8820 Wädenswil

Telefon [redacted]

	Menü A	Menü B	Spezial	Diabetes	Fleischlos
Montag					
Dienstag					
Mittwoch					
Donnerstag					
Freitag					
Samstag					

Bemerkung

SpitexVerein Wädenswil
Gerbestrasse 6
8820 Wädenswil
Telefon Nr. 044 783 93 23

Bestellung Rollende Küche

Menüs (Datum von/bis):

Woche:

Name ..... Strasse .....

Vorname ..... Wohnort .....

Telefon .....

	Menü A	Menü B	Spezial	Diabetes	Fleischlos
Montag					
Dienstag					
Mittwoch					
Donnerstag					
Freitag					
Samstag					

Bemerkung

SpitexVerein Wädenswil
Gerbestrasse 6
8820 Wädenswil
Telefon Nr. 044 783 93 23

Weekly Plan

Rollende Küche Wädenswil
Woche 5 vom 1. 02. 21 bis 6. 02. 21 - Tour 1

Kunde	*Bestellungen*	*Küchenhinweise*	*Verteilhinweise*

103 [redacted]
8820 Wädenswil

Tag	MA	MB	SP	DB	FL	Box
Montag		1				
Dienstag			1			
Mittwoch						
Donnerstag			1			
Freitag		1				
Samstag				1		

Tel. [redacted]
2. Stock
Box auf Stuhl vor Tür stellen und bei oberster Glocke läuten.

---

106 [redacted]
8820 Wädenswil

Tag	MA	MB	SP	DB	FL	Box
Montag	1					
Dienstag	1					
Mittwoch	1					
Donnerstag	1					
Freitag						
Samstag						

0 Fruchtsalat

Tel. [redacted]
1. Stock
Tochter: Fr. [redacted]
Tel. [redacted]
Nate[redacted]

---

109 [redacted]
8820 Wädenswil

Tag	MA	MB	SP	DB	FL	Box
Montag	1					
Dienstag	1					
Mittwoch	1					
Donnerstag	1					
Freitag		1				
Samstag	1					

0 Rüeblisalat

Tel. [redacted]
Türglocke = [redacted]

Tochter Frau [redacted]

---

112 [redacted]
8820 Wädenswil

Tag	MA	MB	SP	DB	FL	Box
Montag						
Dienstag						
Mittwoch	1					
Donnerstag						
Freitag						
Samstag						

Tel. [redacted]

---

115 [redacted]
8820 Wädenswil

Tag	MA	MB	SP	DB	FL	Box
Montag		1				
Dienstag			1			
Mittwoch	1					
Donnerstag		1				
Freitag	1					
Samstag						

Tel. [redacted]

---

118 [redacted]
8820 Wädenswil

Tag	MA	MB	SP	DB	FL	Box
Montag		1				
Dienstag	1					
Mittwoch	1					
Donnerstag	1					
Freitag			1			
Samstag	1					

Tel. [redacted]
3. Stock
Natel [redacted]

---

124 [redacted]
8820 Wädenswil

Tag	MA	MB	SP	DB	FL	Box
Montag						
Dienstag						
Mittwoch						
Donnerstag	1	1				
Freitag						
Samstag						

Tel. [redacted]

Rollende Küche Wädenswil
Woche 5 vom 1. 02. 21 bis 6. 02. 21 - Tour 3

Kunde	*Bestellungen*	*Küchenhinweise*	*Verteilhinweise*

Summen der Bestellungen für Woche 5
vom Montag, 1. Februar 2021
bis Samstag, 6. Februar 2021

	Menü A	Menü B	Spezial	Diabetes	Fleischlos	Boxen
Montag	38	7	7	6	31	
Dienstag	32	14	5	4	28	
Mittwoch	36	11	7	3	25	
Donnerstag	28	32	5	5	16	
Freitag	26	21	6	6	14	
Samstag	29	14	6	13	10	

Daily Plan

Tagesplan Rollende Küche Wädenswil

Tour 1 **Dienstag, 2. February 2021** **Woche 5**

Kunde	MA	MB	SP	DB	FL	Schöpfhinweise	Bouillon Suppe	Salat	Diabetes Dessert	Küchenhinweise
[redacted]	1						1		1	
		1k					1		1	
	1k						1		1	0 Schokolade
			1				1		1	0 Tomatensalat
			1	1			2		2	
			1k				1		1	
		1					1		1	
			1				1		1	
		1g					1		1	
			1						1	0 Tomaten 0 Fruchtsalat
		1g				viel Gemüse wenig Kohlehydrate	1		1	
			1				1		1	
			1				1		1	
	1					viel Gemüse	1		1	
		1	1				1		2	wenn 2 Menüs nur 1 Suppe
17 Boxen	3	5	8	1	0		15 0	0	17 0	

Tour 1: Brigitt

Verteilhinweis:

Menu Count

Rollende Küche Wädenswil
Anzahl Menüs für Januar 2021

Kunde	RoKü	Boxen	MZD
-- Kunden ohne Tour --			
[redacted]	9		
	1		
Summen:	**Menüs: 10**	**Boxen: 0**	MZD: 0

Rollende Küche Wädenswil

Tour 1

[redacted]	18
	16
	22
	1
	20
	24
	8
	23
	16
	16
	19
	1
	20
	23
	14
	23
	5
	24
	30
	24
	4
	4
	12
	18
	4
	4
	11
	20

[redacted]	23
	23
	20
	19
	24
	48
	42
	9
	39
	24
	8
	24
	24
	40

Tour 2

[redacted]	3
	24
	13
	12
	13
	14
	24
	23
	24
	24
	19
	8
	24
	18
	24
	15
	8
	8
	24
	24
	20
	19
	24

Rollende Küche Wädenswil

Anzahl Menüs für Januar 2021

Kunde	RoKü	Boxen	MZD
[redacted]	12		
	6		
	24		
	44		
	4		
	24		
	12		
	8		
Tour 3			
[redacted]	12		
	24		
	16		
	24		
	24		
	24		
	34		
	23		
	16		
	20		
	20		
	20		
	24		
	22		
	1		
	14		
	24		
	8		
	10		
	7		
	24		
	12		
	24		
	15		
	23		
	12		
	6		

Summen: Menüs: 1817 Boxen: 0 MZD: 0

Billing Information

Rollende Küche Wädenswil
Rechnungsdaten für Januar 2021

Kunde	RoKü	MZD	Korrekturen	Gesamtsumme Rechnung
[redacted]	297.00			297.00
	115.50			115.50
	396.00			396.00
	379.50			379.50
	148.50			148.50
	132.00			132.00
	198.00			198.00
	330.00			330.00
	330.00			330.00
	264.00			264.00
	330.00			330.00
	66.00			66.00
	66.00			66.00
	99.00			99.00
	313.50			313.50
	396.00			396.00
	132.00			132.00
	198.00			198.00
	247.50			247.50
	181.50			181.50
	396.00			396.00
	379.50			379.50
	396.00			396.00
	396.00			396.00
	132.00			132.00
	396.00			396.00
	396.00			396.00
	396.00			396.00
	396.00			396.00
	396.00			396.00
	396.00			396.00
	363.00			363.00
	313.50			313.50
	726.00			726.00
	198.00			198.00
	396.00			396.00
	231.00			231.00
	49.50			49.50
	198.00			198.00
	132.00			132.00

Rollende Küche Wädenswil
Rechnungsdaten für Januar 2021

Kunde	RoKü	MZD	Korrekturen	Gesamtsumme Rechnung
[redacted]	379.50			379.50
	198.00			198.00
	396.00			396.00
	330.00			330.00
	231.00			231.00
	214.50			214.50
	643.50			643.50
	379.50			379.50
	396.00			396.00
	313.50			313.50
	99.00			99.00
	396.00			396.00
	379.50			379.50
	396.00			396.00
	396.00			396.00
	396.00			396.00
	379.50			379.50
	214.50			214.50
	379.50			379.50
	396.00			396.00
	198.00			198.00
	247.50			247.50
	198.00			198.00
	396.00			396.00
	148.50			148.50
	16.50			16.50
	792.00			792.00
	396.00			396.00
	330.00			330.00
	379.50			379.50
	330.00			330.00
	693.00			693.00
	16.50			16.50
	16.50			16.50
	330.00			330.00
	396.00			396.00
	330.00			330.00
	396.00			396.00
	66.00			66.00
	66.00			66.00

Rollende Küche Wädenswil
Rechnungsdaten für Januar 2021

Kunde	RoKü	MZD	Korrekturen	Gesamtsumme Rechnung
[redacted]	313.50			313.50
	16.50			16.50
	396.00			396.00
	264.00			264.00
	82.50			82.50
	165.00			165.00
	396.00			396.00
	363.00			363.00
	297.00			297.00
	297.00			297.00
	66.00			66.00
	264.00			264.00
	132.00			132.00
	132.00			132.00
	231.00			231.00
	495.00			495.00
	132.00			132.00
	396.00			396.00
	660.00			660.00
	561.00			561.00
	264.00			264.00
	264.00			264.00
Summen:	30145.50			30145.50

Customer Notices

Rollende Küche Wädenswil

Auswertung für die Frohmatt

Kundenhinweise für Januar 2021

Inaktive Kunden (letzte Bestellung im Dezember)

Name	Erste Bestellung	Letzte Bestellung	Rechnungsadresse	Verteilhinweis
[redacted]		10.12.2020	[redacted] 8824 Schönenberg	Natel: [redacted] mehrere Treppen hoch neben P Schwester: [redacted] Nachbarin Frau [redacted] Menü in Wohnung bringen im Spital ab 10.12.20 (1 Box in WG)
[redacted]		23.12.2020	[redacted] 8820 Wädenswil	1. Stock mit Lift, dann links ab Woche 53 keine Rokü mehr
[redacted]		30.12.2020	[redacted] 8820 Wädenswil	Wohnung Nr [redacted] Frau Krapf: [redacted] ab 1.1.21 keine Rokü mehr (Wegzug)
[redacted]		10.12.2020	[redacted] 8820 Wädenswil	Natel [redacted] ab 14.12.20 bis voraussichtlich Ende Dez. keine Menüs, meldet sich selber wieder.
[redacted]		05.12.2020	[redacted] 8820 Wädenswil	3. Stock, [redacted] Lift dann links, hinterste Wohnung Sohn: [redacted] ab 4.12.20 im Spital
[redacted]		31.12.2020	[redacted] 8804 Au ZH	1. Stock Türöffner = oberste Klingel mit leerem Feld Achtung: 2 x Mettler auf gleicher Etage Tochter: [redacted]
[redacted]		28.12.2020	[redacted] 8820 Wädenswil	Herr [redacted] ist vergesslich, immer Kontakt mit der Tochter Tel. [redacted] oder Schwiegersohn, Hr. [redacted] ab 29.12.20 im AH Sonnweid
[redacted]		01.12.2020	[redacted] 8824 Schönenberg	Natel Herr [redacted] Casa [redacted] Plus Fr. [redacted] Wohnhaus mit Scheune Falls Herr [redacted] nicht zu Hause bitte Menü vor dem Haus deponieren. Ab 2.12.20 keine Rokü mehr (Eintritt in Klinik)
[redacted]		29.12.2020	[redacted] 8820 Wädenswil	Tochter: [redacted] wohnt bei Tochter [redacted] ab 28.11.20. P neben Treppe vor Garage der Tochter ab Jan. 2021 keine Rokü mehr
[redacted]		31.12.2020	[redacted] 8820 Wädenswil	Natel: [redacted] ab 4.1.2021 keine Rokü mehr

Name	Erste Bestellung	Letzte Bestellung	Rechnungsadresse	Verteilhinweis
[redacted]		04.12.2020	[redacted] 8804 Au ZH	Klingel ohne Name= Türöffner Falls Hr. [redacted] nicht öffnet bei Nachbarn [redacted] oder von Allmen läuten, haben Schlüssel (Wohnungstür ist offen) Frau [redacted] Tochter [redacted] Natel: [redacted]
		19.12.2020	[redacted] 8820 Wädenswil	P vor Garage Nr. [redacted] 3. Stock mit Lift Fr. [redacted] geht an Stöcken, Menü in WG bringen. ab Wochr 52 keine Rokü mehr
		16.12.2020	[redacted] 8820 Wädenswil	7. Stock, [redacted] Schwiegertochter: [redacted] ab 9.2.21 wieder Rokü
		11.12.2020	[redacted] 8820 Wädenswil	3. Stock wenn möglich früh liefern im Spital ab 11.12.20

Anzahl Inaktive Kunden **14**

Neukunden (erste Bestellung im Januar)

[redacted]		26.01.2021	[redacted] 8820 Wädenswil	Natel: [redacted] Zufahrt von [redacted] vis à vis "[redacted]" [redacted] abwärts, linke Seite 2 Mehrfam.häuser mit [redacted]
		09.01.2021	[redacted] 8820 Wädenswil	Lift Stock [redacted] Natel [redacted]
		07.01.2021	[redacted] 8820 Wädenswil	sieht sehr schlecht
		15.01.2021	[redacted] 8820 Wädenswil	4. Stock, [redacted]
		27.01.2021	[redacted] 8820 Wädenswil	
		06.01.2021	[redacted] 8820 Wädenswil	2. Stock, [redacted] Sohn Herr [redacted]
		19.01.2021	[redacted] 8820 Wädenswil	
		27.01.2021	[redacted] 8820 Wädenswil	2. Stock, [redacted] Menüs von Tochter bestellt: Mi 27.1./3.2./10.2.21 ab 13.2.21 im [redacted] am Do 11.2.21 die leere Box zurückholen.

Anzahl Neukunden **8**

Monthly Statistics

Rollende Küche Wädenswil
Auswertung für die Frohmatt
Anzahl Menüs für Januar 2021

Liefertag	Menü A	Menü B	Spezial	Diabetes	Fleischlos	Boxen
4.	30	15	20	4	8	
5.	25	31	7	5	3	
6.	23	25	7	7	5	
7.	26	35	11	4	2	
8.	15	41	3	3	8	
9.	25	24	11	3	4	
11.	29	24	12	5	8	
12.	48	10	9	11	2	
13.	31	5	23	8	11	
14.	28	32	11	3	5	
15.	30	25	10	3	5	
16.	23	20	18	4	8	
18.	36	20	17	8	5	
19.	38	16	5	5	11	
20.	19	27	18	6	5	
21.	22	32	11	8	11	
22.	36	25	6	6	4	
23.	30	23	4	5	5	
25.	33	11	24	6	8	
26.	46	15	7	6	4	
27.	43	28	5	4	6	
28.	30	29	9	8	5	
29.	34	19	8	6	7	
30.	37	14	6	9	5	
Summen:	**737**	**546**	**262**	**137**	**145**	

Januar 2021 **Menüs: 1827** **Boxen: 0**

Samstag, 30. Januar 2021