Project Report FoodCoop
Klik-je-(k)eten

27-06-2014

Created by:
Martien Bonfrer, Roy Hoeymans, Bart Sunter, Dex Bakker
Bachelor students Information, Multimedia, and Management (2013-2014)
VU University, Amsterdam

Content

Executive summary

This project report was commissioned by the foundation FoodCoop NL. It mainly focuses on the problem of creating and providing insight into food chains by means of the provision of information and software. This way foodcoops are enabled to live up to their values, control where the food they buy is coming from and support the suppliers, processors and transporters they think deserve the attention.

The project goals are to give insight in the main problems FoodCoop NL has at this moment and will encounter in the future regarding the IT infrastructure, to create a data model that can be implemented in the system and provide a user interface mockup to visualize how the information and creation of chains can be displayed.

In the first deliverable we first zoom in on the current IT infrastructure with a complete analysis of Foodsoft and the data structure. The problems are divided into four sections in which we separate the issues. These consist of the manual entrance of supplier data, the manual synchronization of the databases, the dispersion of the location of data and the separation of supplying information and Foodsoft as services. For the previous problems we provide two main solutions: a central database and an API to access it.

The second deliverable focuses on the creation of a new data model for Foodsoft that makes it possible to link food chains together. Its objective is to provide a way to store all the information that is generated with the creation of a food chain. In addition, several real life use cases are elaborated to demonstrate how the data model could be used when implemented.

Finally the third deliverable provides a visual interpretation on how the process of linking chains can be practically modelled. We display this with a complete use case of a typical user that would create a chain via Foodsoft. The mock-ups that show this interpretation are meant as an inspiration of what the system could look like and does not focus on how information is retrieved. We end this final document with our final remarks and advice, which is split into several phases of implementation. This is an explanation on how we think FoodCoop NL should continue the development of their system in separate detailed steps.

The first phase would be setting up a central database and API to access the data. The second phase starts with creating a basic foodchain-system where organisations can create their own chains and visualize their own process. The third phase focuses on the extension of the system set up in the second phase where it will be possible to change existing chains and proposals will be send to organisations for approval. The final completion phase encompasses the addition of the ability to create a full chain by linking organisations together by foodcoop users.

1 Introduction

1.1 What's in this report?

This report serves as an accompaniment for the three deliverables that we (the project group) have provided. Section 2 of this report starts with a project description in which we define the objectives of our project. These objectives are reflected in the deliverables that we have created. Each deliverable can be treated as a separate report but together they relate to the goals and the questions that we've asked in the project description. This report closes with an advice section, which is serves as a guide for FoodCoop that explains how the deliverables can be used to realize and implement this project.

1.2 Terminology

We'd like to briefly clarify some of the terms we will be using during this report and in the deliverables.

What is a foodcoop?
A foodcoop is an abbreviation of the word food cooperation. A food cooperation is a group of people that collectively orders food directly from the suppliers. This way they make sure that they receive a good product and that the supplier gets a fair price.

Difference between foodcoop and FoodCoop
A foodcoop refers to a food cooperation, while FoodCoop refers to the organisation that this report is intended for.

What is Foodsoft?
Foodsoft is the software that is currently used by FoodCoop. Right now it is used as an order system for foodcoops to order directly from suppliers.

What is a "food chain"?
This question is one of the main subjects we focus on during this project. A food chain encaptures all the processes that a food product goes through from start to finish. The project description in this report and the data model specification (deliverable 2) give a much more in-depth explanation.

2 Project description

2.1 Description of Foodsoft

FoodCoop is an organisation that supplies software and information about suppliers to foodcoops, to support them with the process of selecting suppliers, composing a product assortment, and ordering products. They support secure payment with iDeal via their software or pin. This way, they aim to unburden foodcoops with tedious administration tasks, making it easier for them to focus on selecting products that match their values and ideals.

FoodCoop wants to expand their software with new functionality. They want to give foodcoops the ability to compose their own 'food chains' that matches their values and ideals. These values should be expressed by the choices that the foodcoops make when creating their food chain. These choices include selecting certain suppliers, transportation firms, food processors, and suppliers. This way, foodcoops can purchase food that matches the values and worldviews that they find important. This project focuses on this new functionality of creating food chains.

The following group of people will be using the new system:

- Suppliers - suppliers can make their processes visible to the foodcoops and offer their products via the software.
- Consumers - consumers can be part of a foodcoop. This allows them to collectively indicate from which suppliers they want to order and which values are important to them.
- Foodcoops - a foodcoop has a collective idea about their values, which influences which suppliers they choose to order from. Their is often one person or a group of person that represents the members of the foodcoop.

2.2 Scope of the project

The process of creating food chains is very complex and ambitious from not only an organizational, but also a logistic and technological perspective. In this project we will focus purely on how the concept of the food chain can be applied to the IT aspects of the organization FoodCoop, while also recognizing that there other important aspects that are important, like transportation, terms of delivery, rating systems for products or suppliers, and trading marks. We have kept this in mind while working on the project so that the system can be expanded to include these things.

2.3 Project objectives

- Giving an overview of IT problems that FoodCoop encounters or will encounter in the future.
- Designing a data model that can handle the food chain concept.
- Finding out how the large amount of information that comes with creating a food chain can be made visible to the possible users of the software.

By creating insight into the IT problems that FoodCoop copes with or will cope with, it can focus on other aspects of their organization. The goal of this report and the deliverables is to create a base for possible expansions and eventually as a reference for the implementation of this project.

2.4 Deliverables

Based on the three project objectives and questions we have asked ourselves, we have decided to divide this project into three deliverables. Below is a description of each deliverable along with related questions that we aim to answer in the deliverable.

- **Deliverable 1 - Analysis**

  The analysis of the IT aspects of FoodCoop will be presented in this document. This document consists of two parts: an objective analysis of the current system and a proposal for solving the problems that we've discovered in the analysis. This first part can be seen as a separate document that can be used as a reference for IT projects that FoodCoop might want to do in the future. The second part is focused on how solving the IT problems paves a way for the implementation of the food chain concept.

  Questions

  - What is the current infrastructure of the IT systems of FoodCoop?
  - Can this handle the 'food chain' concept that FoodCoop wants to implement?
  - If not, how does the infrastructure need to be changed so that it can?
    Which challenges related to IT does/will FoodCoop face when implementing the food chain functionality?

- **Deliverable 2 - Data model**

  Constructing a solid data model that can be applied to the food chain concept is perhaps one of the biggest challenges of this project. The data model is delivered in the form of a UML schema, accompanied by a document that explains the food chain concept in great detail, along with a few use cases that show how to apply the data model, and a documentation that describes each table.

  Questions

  - How can the many types of data that is available about products, suppliers, foodcoops etc. be generalized to a solid data structure?
  - How can the food chain concept be represented in the data model?
  - Which elements are part of a chain? How can these parts be linked?

- **Deliverable 3 - User interface**

  The last document presents a visual interpretation of how food chains can be composed with the use of the software. We have created a mock-up, along with a simple clickable model that can be accessed in a browser. Link: www.klik-je-keten.nl/mockup

  Questions

  - How can the food chain concept be represented visually?
  - How can we display the large amount of information concerning a food chain while not losing sight of the overall picture?
  - How can we show the consequences of the choices that the user makes while creating a chain on the world and environment?

3 Advice

In this section we would like to explain which steps FoodCoopNL can make toward a fully functional system in which the foodcoop's can create their own foodchains.

We think the project is too big to implement in one step, thats why we came up with a four step process. Each phase is a (small) step towards the end goal. It is very important that this end goal is kept in mind while developing the different phases, this way each new phase can be implemented with the least amount of issues.

3.1 Implementation phases

First phase
In the analysis (first deliverable) we talk about setting up a central information database and an API to access this data. This would be a great first step. This way you will have all the data at one location which makes the whole system much easier to maintain.

This step will take some time and effort because the Foodsoft system has to be reconfigured to work with the new database and API. But it is a vital first step in creating customizable foodchains.

The first deliverable explains how this phase can be accomplished.

Second phase
After laying the foundation it is time to start with the foodchain-system. The second step is implementing a very basic foodchain-system. We don't want to make things to complicated in this step. So we start with a system in which organisations can create a chain just to visualize their own process.

This will give the user insight in how a product is manufactured and choose a product accordingly. The other two deliverables (Data model and Mock-up) can be used as guidelines when developing this phase, as well as in the next phases. The Data-model provides information about which data is needed and how it is connected to each other. This model is based on a complete system this means that you do have to implement all it parts in the first phases yet. But it can be beneficial to implement as much of the structure as possible so you won't have to restructure your data between phases.

The Mock-up deliverable is also a great guide to use while developing this phase (and the next phases as well), the developers can use it to visualize the system which they are building. And its design can be used as inspiration for the real design.

Third phase
The third phase is enhancing the functionalities of the basic system. Now it will no longer be a system just to visualize the foodchain, it will also be possible to change an existing chain. This new chain will not work right out of the box, it is just an proposal to the organisations to collaborate. If the involved organisations approve the request they will start working together and then can create a new working chain.

The final phase
After implementing this final phase you will have a complete foodchain-system, where a user (foodcoop) can create a completely functional chain. The system takes care of all unnecessary

complicating processes. Another functionality of this final system is that it can advise a user to use a specific chain based on user's values and standards.

This last phase will probably be the hardest phase to implement. There are a lot of functions, stakeholders and organisational issues that have be overcome. We don't think it is impossible to implement this last phase but it will be very challenging.

Organisation
Implementing these phases while maintaining the current system will take a lot of time and effort. So much time that we don't think it is possible to accomplish with just one developer. Thats why we advise to incorporate other people in the process, this can be done by hiring new people or collaborating with other organisations.

Deliverable 1: Analysis

Content

Executive summary

This document gives an analysis of the current IT system that is used by Foodcoop, sums up the main problems that this system has or is going to have in the future, and offers solutions to fix or prevent these problems. The main issue that emerged from our analysis was the fact that the data that the current system uses is decentralized. Each foodcoop has its own database which has to be synchronized manually with a central database that contains supplier information and product lists. This dissemination of data is not a desirable situation for Foodcoop, especially from a technical point of view.

The vision of Foodcoop is that users of their software can construct their own 'food-chain', so they gain insight into all of the processes that their food goes through. This way people can fully support the way their end product is created from beginning to end. Constructing such a 'chain' requires a lot of interaction with the data. Having all this data stored in one central database improves the performance and decreases the complexity of the implementation of the software.

1 Introduction

This document contains an analysis of the current IT system that is used by FoodCoop. It can serve as a reference for developers as well as managers that need to make IT related decisions. Section 2 describes the current system by making a distinction between the application Foodsoft and the underlying data structure. Section 3 gives an overview of the problems that the system has or is going to have in the future, and section 4 gives solutions to those problems as well as some considerations of those solutions. The last section provides a set-up of the next phase of our project: creating a data model and a mock-up for the user interface.

To give clarity on the word usage, with FoodCoop (upper case) we imply the organisation and with foodcoop (lower case) we imply the cooperations of people who use Foodsoft.

2 Analysis of the current system

To describe the current system that is used by FoodCoop we will divide this section into two subsections: one concerning the application Foodsoft and one concerning the underlying data structure.

2.1 Description of Foodsoft

Foodsoft is an open-source software application. This means that all code is publicly available, and can be found on GitHub. Foodsoft is written in Ruby and makes use of the Ruby on Rails development framework. The main reason for FoodCoop to use this framework at the moment is the advantage of high flexibility. Ruby is a rather high level programming language which makes it easier to develop faster and modify more rapidly to the needs of the software. FoodCoop can, for instance, use it to migrate a different interface structure into Foodsoft rather simple when adjustments are required.

In the current situation, Foodsoft is software application that enables foodcoops (consisting of individual members) to order product fulfilling their needs. The suppliers are added to the system by sending their information to FoodCoop or filling in an Excel template. Some suppliers only produce orders when a certain amount is reached. For these situations Foodsoft includes a built-in function to complete order units with other members. An order cycle consists of four stages that are being carried out:

Preparation stage; foodcoops can update their supplier and product information and choose which articles can and cannot be bought. Vice versa suppliers can send in their own products and information to FoodCoop.

Ordering stage; Members of the foodcoop can place their orders for a limited amount of days after which the time period will be closed and the orders are send to the suppliers.

Receiving and dividing stage; The products will be delivered to a central location where they are counted, checked for defects and divided over the members of the foodcoop.

Closing stage; After the final check, the administration will be closed.

To give an idea of what Foodsoft looks like in the ordering stage, the picture below shows an example of a productlist in Foodsoft, which can be filtered on categories like ‘fruit’ or ‘vegetables’. The products show a name, price and number of units that can be chosen. For a full explanation and visualization of all steps and possibilities in Foodsoft, we refer to the Foodsoft order group and member manuals from FoodCoop.

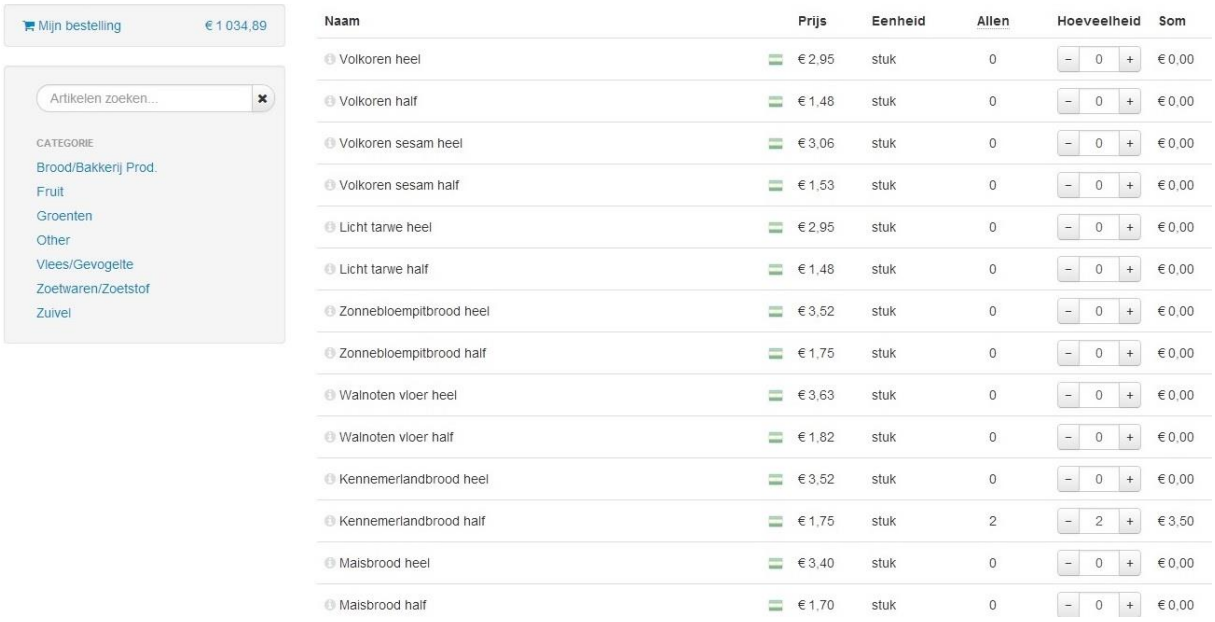


Figure 1: Foodsoft productlist in the ordering stage

The financial administration concerning the payments is mostly regulated by FoodCoop, but there are two options to fulfill the payments. In the first case, all the revenues are transferred manually by FoodCoop to the suppliers when the products are paid. So, all the income goes from the client, through FoodCoop, to the supplier. Finally FoodCoop returns a margin of the revenues to the foodcoops for all the costs they have made for realizing the event. Secondly, there is also the option for foodcoops to pay and handle the transactions to the suppliers themselves.

2.2 Description of the current data structure

The underlying data structure of Foodsoft mainly consists of two databases: the central database and a group of databases that each belong to a foodcoop. These databases interact with each other in the following way: every foodcoop has its own database containing every supplier, product information and orders (history). These local databases are synchronised with a central database that contains all product information from each registered supplier. This synchronisation is carried out through the Foodsoft application and in this way, the database is incorporated into the application. The foodcoops also have the possibility to add new suppliers and products to their own database, which are in that case not present in the central database. By using Foodsoft, foodcoops are able to show their product lists and place orders. A modelled representation of the data structure of the current system can be seen in figure 2.

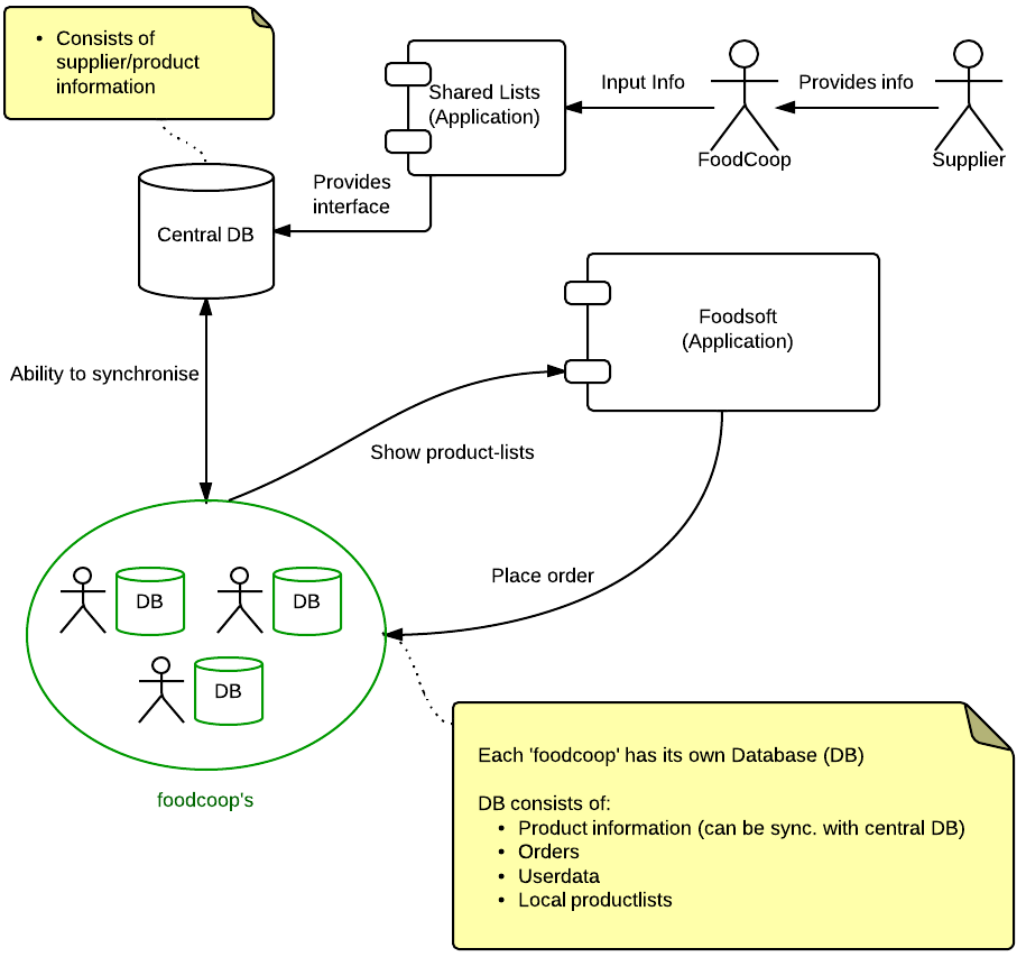


Figure 2. Data structure of the current system

2.3 Hardware infrastructure

The database and the application are currently hosted by Greenhost. Foodcoop doesn't have its own servers or hardware in its office.

2.4 Current functionality vs. future functionality

As described in section 2.2, the current functionality of Foodsoft consists of just an order system. Users can create an account, log into the application, select products from a product list, and place an order which they can then pay for. This works fine for now, but a lot of functionality needs to be added. The vision of Foodcoop is that users of their software can construct their own 'food-chain', which allows them to gain insight into all of the processes that their food goes through and support the way their end product is created from beginning to end.

Creating software that can do this is a complex task. In the next sections we will give an overview of the problems that Foodcoop will encounter, (possible) solutions to these problems, and an explanation of why these solutions are necessary to achieve the 'create-your-own-food-chain'-idea that is the core business of Foodcoop.

3 Problem description

While performing the analysis of the current system, we've identified four main problems or things that could become problems once the organisation grows:

- Manual entrance of supplier data; the more the system will grow, the more tedious and less feasible this becomes. It's also very error prone.
- Manual synchronisation between FoodCoop databases and the central database; can lead to invalid orders if a foodcoop doesn't have up-to-date information.
- All of the data concerning FoodCoop/Foodsoft is stored in more than one place; this makes it difficult to maintain and this will become even more apparent if the system grows.
- Providing information and Foodsoft are intertwined instead of separate services. Because of this coupling, the flexibility of the system is compromised.

A description of each problem and the reason why it's important to solve it is given in the following subsections.

3.1 Manual entrance of supplier data

At the moment there are three ways in which supplier information is added to or updated in the database. The first way is that a supplier sends his information (products etc.) to FoodCoop which in turn will, by hand, input all the data into the database. The second way is a bit more technologically advanced: the suppliers will receive a (Excel) template in which they can add their own information. When they are finished putting in their information in the template they will e-mail it to a FoodCoop address. FoodCoop then uses a self-made filter, based upon the current administration, that interprets the information and makes the necessary changes in the database. The third way is that foodcoops input supplier information and product lists into their own database and possibly add the extra option to share this data.

If we think about upscaling the whole system we see some major problems in this process, especially regarding the manual input of data into the system. It's not a feasible solution to input all data by hand, it will take a huge amount of time and is fairly error prone due to human errors. On the other hand forcing larger suppliers to use a template other than their own, can cause a decrease in their willingness to participate. Another problem regarding the use of a template is that the supplier can easily make a mistake when he fills in the template by hand. This can result in a system where the data is easily corrupted which in turn can result in a lot of (unnecessary) errors.

A solution for these problems could be an automatic wizard (a utility within the software) that recognizes particular sorts of data and fills in the database. This way the process of inputting data manually will be taken out of hand completely.

Finally when looking at the input of supplier information by foodcoops, their must be added an option to declare whether the owner of a product list is the supplier or the foodcoop.

3.2 Manual synchronisation

In the current system, users have to manually import product information from the suppliers database. If they don't do this, the information they act upon might be out of date. This incompleteness of data obviously leads to orders becoming invalid. It is therefore essential that the product information is being kept complete and up-to-date. The problem is that the information is decentralized. A central database can solve this.

Furthermore, some foodcoops may have specific arrangements with some of the suppliers. For example, they might order an X amount of a product and receive a discount. Agreements like this should be kept in a separate table (in the central database).

3.3 Decentralized data

Currently, all of the FoodCoop and Foodsoft data is stored at several locations, as stated in section 2.2 of the first part of this document. The way this data is stored right now could lead to several complications when the system would scale to a larger dimension. The biggest problem with storing information like this (in multiple databases) is the accessibility and maintenance of the data structure. The process of keeping the information about suppliers, products, orders and users up to date can be improved a lot when it comes to efficiency, ease of access and controlling an overview of all data points.

Combining this with the addition of an own API, the information can be controlled more easily, will be able to change faster and (just as important) automatically. A solution where an API is coupled to multiple databases is another possibility. However, this would be a far more difficult method of implementation. Also, when data from different databases are dependent, querying will be much slower than with a single data point.

3.4 Supplier information and Foodsoft

FoodCoop is an organisation that offers software that helps users to create their own 'food-chain', but also provides information about suppliers. In our opinion, these two should be separate services. In the current situation, the application and the underlying data structure are intertwined. This is not an ideal situation because it limits the flexibility of the system.

It also forces potential clients to use the software to obtain the supplier information, while not everyone necessarily wants to start or organize a foodcoop. The information in itself can be valuable for clients, especially once FoodCoop provides a structure that makes it easy to compare the suppliers.

Ideally, suppliers will be able to *choose* how much information is visible and to whom. This heavily promotes transparency. For instance, they can make their organisation information visible to everyone, but make their price lists visible to only foodcoops and/or business partners.
The new system has to account for this need. It should be clearly visible (in the database) who owns which data and who has access to it.

4 Solutions

This section contains the solutions to each of the problems described in the previous section. In our opinion, these problems can be solved by simplifying the current data structure to a structure with one central database, and an API to access and modify this data. The central database and the API complement each other, and this should be kept in mind when implementing them. For example, an API is much more complex and less effective if it has to work with multiple databases instead of one central database. Thus, this section describes one solution divided into two parts: the central database and the API, but they should not be implemented separately.

4.1 Central database

In our opinion the ideal system would consist of multiple individual parts. Each part has his own function and responsibility. Like we said before we want to split the two services FoodCoop offers to their customers. This split will result in the two main parts of the system: the Foodsoft software and the information database. There will be one central database containing all information the system requires. This central database contains user and account information, as well as supplier information, product lists, and order data.

As mentioned before we want to have one central database containing all of the required information. This means that individual foodcoops will no longer have their own database. Each foodcoop will logon to the central system, and the system will only show the information regarding that particular foodcoop. Each foodcoop can manage its own supplier information or it can use public supplier information stored in the central database.

Implementing a single central database solves a lot of the previously mentioned problems. Synchronisation isn't required anymore because the foodcoops don't have their own database. The data is all stored in one location which makes it easier to access. It also makes it easier to separate the software and the provision of information services that FoodCoop offers to its clients.

Accessing and modifying the data in the central database can be done via an API, which is described in the next subsection. A visual representation can be seen in figure 2.

4.2 API

The information in the central database can be accessed by an API (Application Programming Interface). An API makes it easier for (future) developers to design an application that uses the information stored in the central database. The API is not restricted to a particular programming language which makes the system much more flexible. Another benefit of using an API is that is acts like a 'gatekeeper'; it manages user privileges and prevents unintended damage to the information integrity.

The API can also be used to store new (or update existing) data in the central database. This will enable suppliers to automate the process of maintaining its data in the system. Which will lead to a more user-friendly and up-to-date system.

We divided the software that is used by the foodcoops and the suppliers into two seperate applications, because the suppliers and foodcoops both have different needs when it comes to the usage and functionalities of the software (e.g. the suppliers don’t need an ordering system). However, both applications are filled by one API that is connected to the central database. When it is set up this way, the API regulates the required information to the right application and consequently there will be less to no effect on the performance, because the API only functions as a single

doorway from the central database to the applications. So eventually, the data comes from the same location but they enable different methods of usage to separate applications. A visual representation of this can be seen in figure 3.

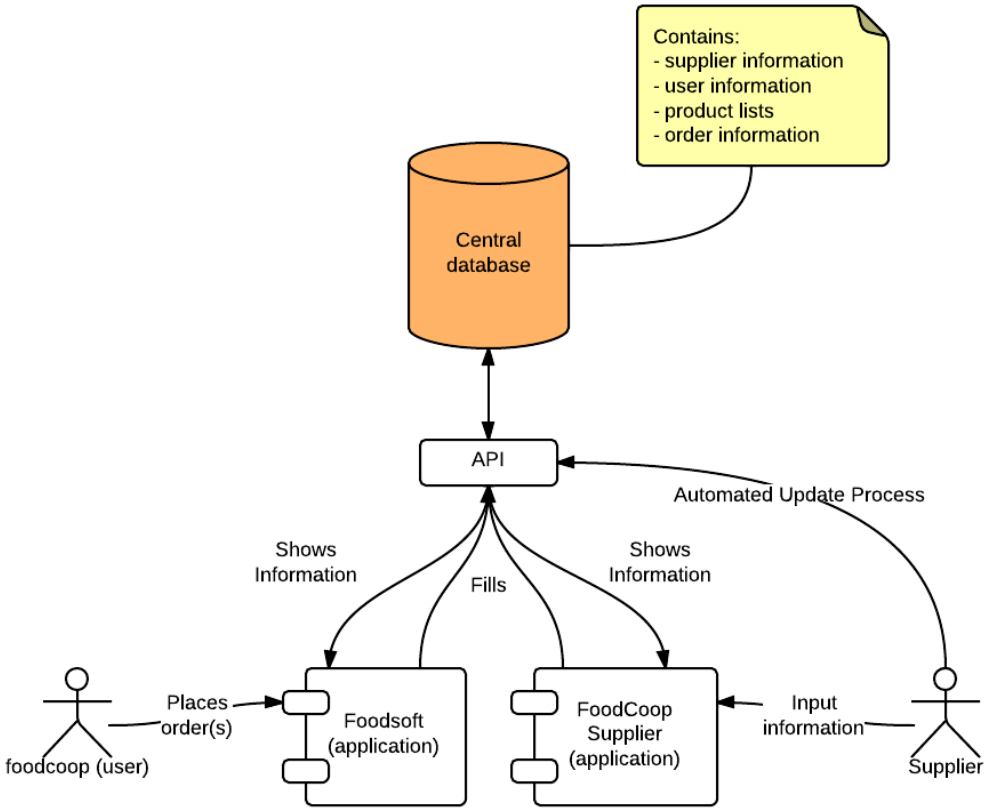


Figure 3. Data structure of the ideal system

4.3 Problem-solution relations

To summarize, the relation between the problems in the previous section and the solutions in this section can be visualized like this:

Problem	Solution
Manual entrance of supplier data	API
Manual synchronisation	Central database
Decentralized data	Central database
Intertwining supplier information and Foodsoft	Central database & API

Table 1. Problem-solution relation
The problems in this table are defined in section 2.1, 2.2, 2.3, and 2.4.
The central database and the API are defined in section 3.1 and 3.2.

4.4 Reflection

As we've described, working towards a system with a central database has many advantages and offers solutions to a lot of the problems that the current system has. However, perfect solutions don't exist so we also want to shed some light on the disadvantages of this solution.

First of all, the fact that the application depends on information which is all stored in one place can also be a disadvantage. If the database goes down, then so does the application. However, when implementing a system that gets its information from lots of databases, there is a higher chance that something goes wrong, especially if the data is not independent.

Another questionable perspective indicated by FoodCoop is the ethical point of view when it comes to implementing a single central database. The vision of FoodCoop, in the future, is that the information is gathered, controlled and maintained by all who want to contribute. In that way a single central database is in conflict with the idea of laying the data in the hands of the many, being the owners of the main location of the information.

However, especially from a technical point of view, we think that an implementation with a central database works a lot better than a decentralized approach for Foodcoop. It is a lot easier to implement the 'create your own food-chain'-functionality when all of the data is stored in one place, because a lot of the data is dependant on each other. A system that has to access all of the foodcoop's individual databases individually as well as the central supplier database is a lot more complex to create and will also eventually suffer from performance issues.

As pointed out in section 4.2 an API has many advantages, but you could argue that because of the nature of an API (being a extra layer between the database and the software) it has a negative influence on the performance. If the API is implemented in the correct way these performance issues will be kept to a minimal. We think the small decrease in performance does not outweigh all the previously mentioned benefits.

Deliverable 2: Data model

Content

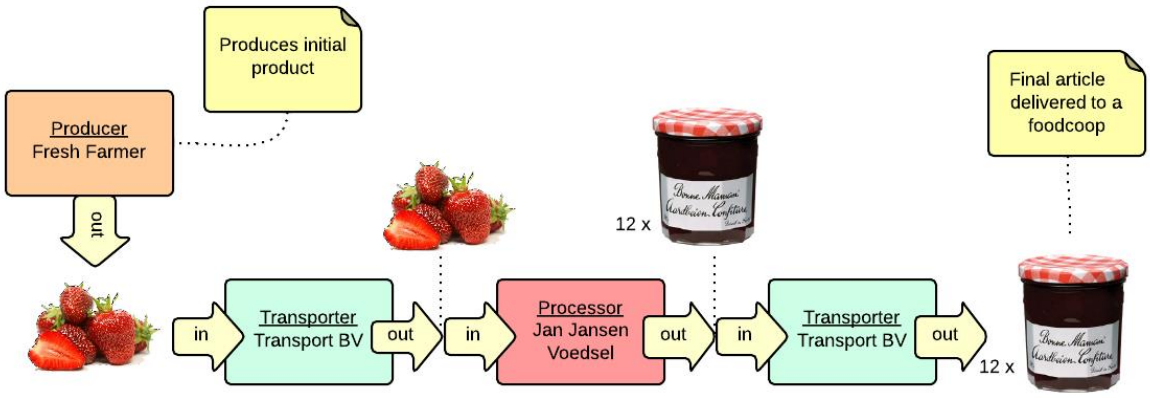
1 Introduction

This document contains the data model specification for Foodsoft that we have constructed for supporting the chain concept. This chain concept will also be explained in detail in this document, as well as some justifications for certain decisions that we have made during the development of this model. It also provides some use cases that show how the data model can be applied, some discussion points and a documentation section. The data model itself is intended for developers but this document might be useful for executives or other stakeholders of FoodCoopNL as well.

2 Chain concept

One of the most important functionalities of Foodsoft is going to be the ability to create, modify and/or see all the processes that your food product goes through. All these processes linked together can be seen as a so-called ‘food chain’ or simply ‘chain’. This data model focusses mainly on that aspect of Foodsoft.

A 'chain' is the full chain from initial product to end product, including processes and transport. A chain consists of service links. A service link belongs to a service provider, which can be a producer, a processor, or a transporter. Each service provider, with the exception of a producer, has an input and an output product. The input product of a service link is always the same as the output product of the previous link in the chain. An example of a simple chain:



Types of service providers:

Producer - produces the initial product. Always at the start of the chain, has no input product.
Processor - processes an input product into an output product.
Transporter - transports a product from A to B. Doesn't modify or add anything to the product. Its input product and output product are the same.

Matching inputs and outputs

One of the main challenges of creating a chain is to match an output of a certain process with an input of another process. To make this possible we have decided to make a distinction between a 'product' and a 'product-category'.

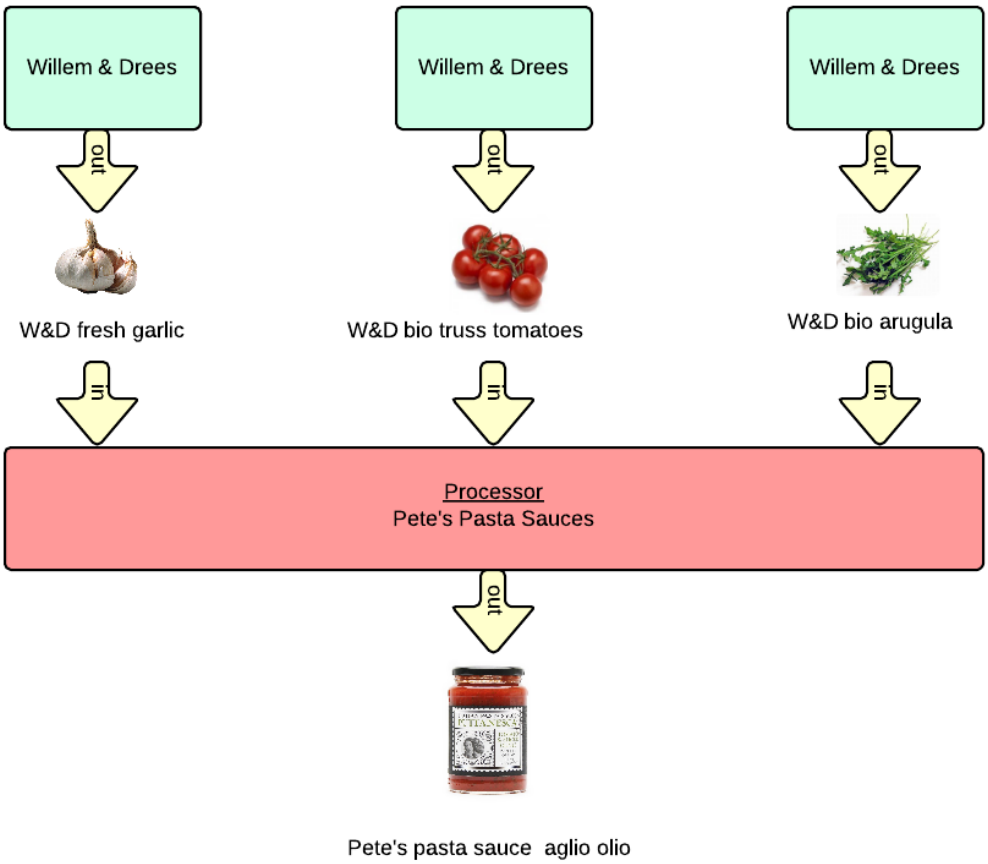
A product is the concrete product with attributes like price, unit, unit quantity, which producer has produced it etc. Every product, even semi-finished products like leaves of a beet or legs of a chicken , is a concrete product that can be ordered or chosen as input for a new service link.

Every product belongs to a product-category. A product category can be as specific as the supplier wants it to be. In most cases, the supplier will select the most specific category that is possible. A category can be a sub-category of another category.

An example of this idea in practice:
Say a supplier wants to make pasta sauce. One of the inputs he needs to select is some sort of tomatoes. If he specifically wants cherry tomatoes, he selects the product-category 'cherry-tomatoes'. Then he can chose from a list of every product that is in that category, for example: 'W&D biological cherry tomatoes', 'BioRomeo's cherry tomatoes', etc. If the supplier doesn't care what type of tomatoes to use he simply selects the category 'tomatoes' and then selects one of the products that belong to that category.

We will illustrate this idea with a few practical examples, starting on the next page.

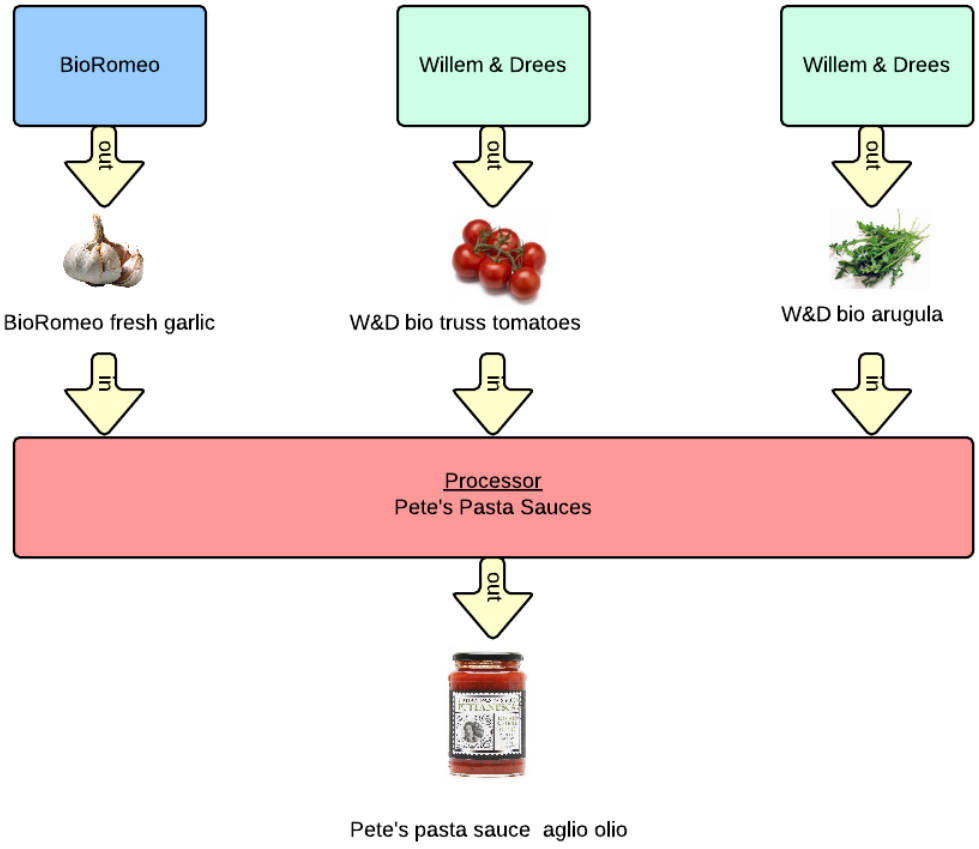
Example 1 - multiple inputs, one output



The above picture shows a visual representation of this idea. Pete's Pasta Sauces needs multiple input products to create its output product: pasta sauce. The input product-categories in this case are: truss tomatoes, garlic and arugula. Pete's Pasta Sauces can than select actual products belonging to those categories. In the above example the products that were selected all come from supplier Willem & Drees.

Product-category	**Product**
Garlic	W&D fresh garlic
Truss tomatoes	W&D bio truss tomatoes
Arugula	W&D bio arugula

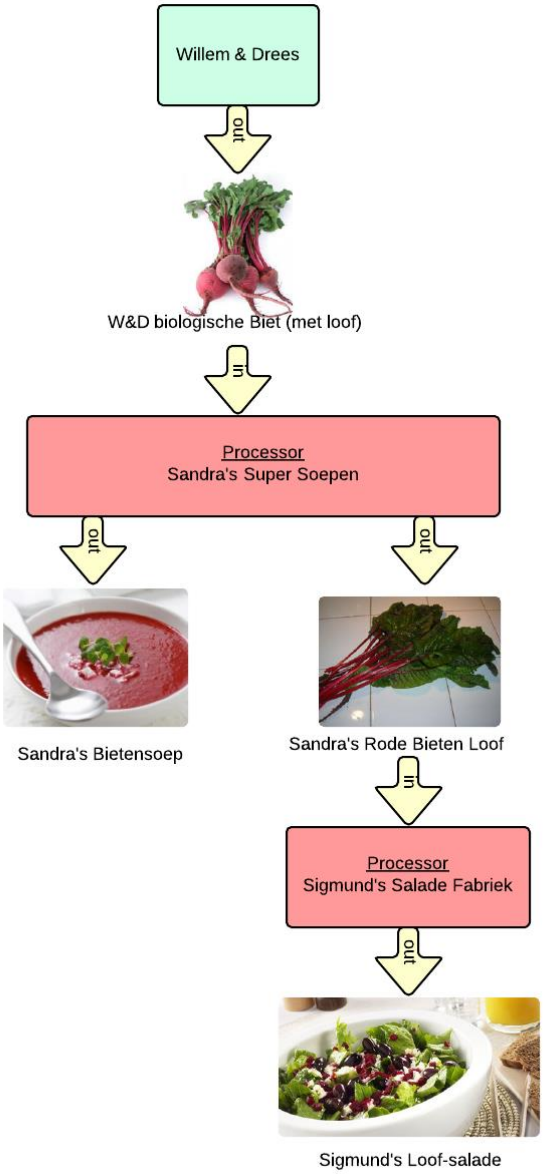
So a product can be selected from a product-category. It would also be possible to change the chain in the previous picture to this:



Here, an other product is selected from the category 'garlic', namely 'BioRomeo fresh garlic from supplier 'BioRomeo'.

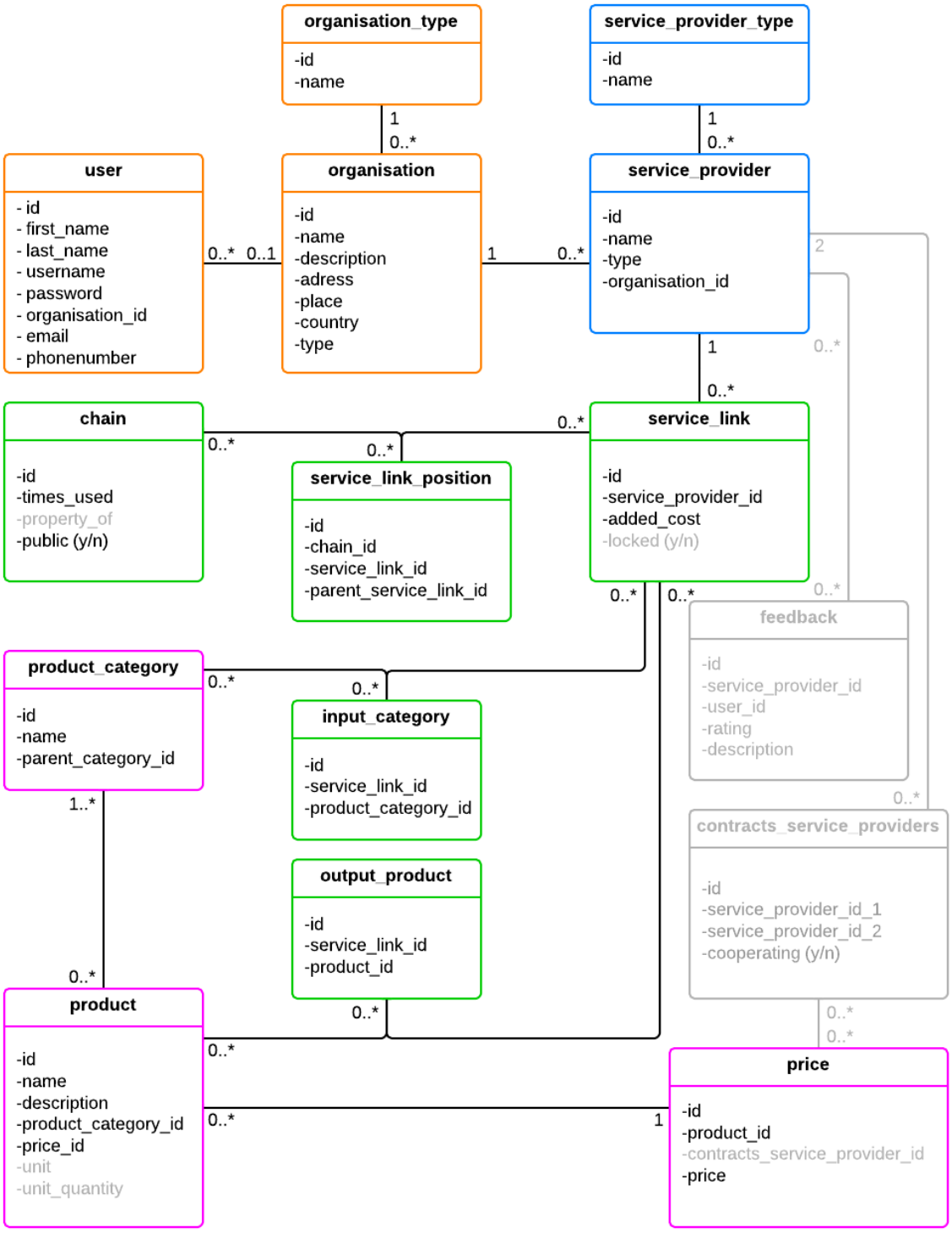
Example 2 - multiple inputs and outputs

Some processes have multiple inputs or outputs. For example: if you want to bake a cake you need multiple ingredients like flour, sugar and eggs. If you translate the baking of a cake into a process in the chain you would get a process with multiple inputs but with a single output product (the cake). Besides processes with multiple inputs there are also processes which have multiple outputs.
A process which has multiple outputs is shown in the figure below. The chain starts with beets which still contain the beet leaves. The beets are made into a soup but the beat leaves are unused. The user can choose to discard these leaves or he can choose to make something useful with it. In the figure you can see that the user chose not to discard the leaves but use them in another process to make them into a salad.



These idea's are represented in the data model by the service link table. As discussed before, a chain consists of service links. A service link contains a process with one or multiple input product-categories and one or multiple output products. The input product from a service link is the output product of the previous service link in the chain.

3 Data model



Notes: some parts of the data model are ‘greyed out’. These parts are important enough to include in the model but are not completely worked out because they are not part of the essential system concerning the chains. These parts need some extra attention when the system is developed/implemented.

4 Documentation

This section describes the function of each table in the data model. Keep in mind that the main objective of this data model is to show the relations between entities. We have only included the most basic attributes without going into much detail. Attributes can always be added, removed or modified when implementing the data model.

user
The user table stores account information. A user is linked to an organisation. The type of the organisation determines if the user belongs to a foodcoop or a supplier/producer.

organisation and organisation_type
The organisation table contains details about the organisation like name, address, country etc. An organisation can be of a certain type, which is stored in a separate table called organisation_type. An organisation can be a supplier or a foodcoop.

service_provider and service_provider_type:
As stated before, an organisation can provide multiple types of service. For example, an organisation can be a supplier but also a transport company. This is desirable if the organisation wants to produce it's own food as well as process or transport it. What type of service the organisation provides is stored in the service_provider table.

A service_provider is linked to an organisation and a type. The service_provider_type table looks like this:

id	description
1	Producer
2	Processor
3	Transporter

A producer has no inputs, only output(s).
A processor has at least one input (category) and one output.
A transporter only transports the product without adding or modifying it. A service_link with a service_provider with type transporter is not linked to an input and an output product. This is to prevent a data overflow in the tables 'input_category' and 'output_product'. For now we assume that the software can compute the input and the output product by looking at the previous service link in the chain.

Note: because an organisation can also be of type foodcoop, it is also possible for foodcoops to take on production or processing tasks themselves and be part of a chain.

service_link, service_link_position and chain
The service_link table contains a service provider ID. Service links are part of a chain. The positions of the service links in the chain are stored in service_link_position.

product_category and product
These tables contain basic information like name and description and such. The price of a product is stored separately in the price table (see price).

input_category, and output_product
These tables are just for linking inputs/outputs to a service_link. Input_category contains a list of category id's. Each one of these entries also contains a service_link id. This construction is similar to the output_product table's construction. This table contains a list of tuples (product id's and service_link id's). This way it it possible for each service_link to have a single or multiple inputs and outputs.

price
This table stores the price for a product at a certain point in time for a certain user (foodcoop). This table is also useful when you check the sales history, you don't want to see the price at this moment but rather the price of the product when the sale took place. There is also a possibility that a service provider offer the same product for different prices to different foodcoops. This can happen when a service provider and a foodcoop have some special price agreements (stored in the contracts_service_providers table). Or when the service provider gives a discount when a foodcoop orders a large quantity.

Additional (possible) tables:
These tables are grey in the data model and are just concepts. They can be applied later when more information about these concepts is available.

contracts_service_providers
This table can store a relation between two service providers. This information needs to be stored because not every service provider will want to cooperate with every other service provider. For example, a producer might not agree with the way a certain transporter transports their product, or a foodcoop may explicitly not want a certain supplier to show up in their chains because their values don't match. By storing this information we prevent that orders are not going to be delivered because there are disagreements between service providers in the chain. Other things that could be stored here are discount arrangements between a foodcoop and a supplier.

feedback
Stores feedback that an user can give to a service provider. What this feedback is going to look like exactly is not certain yet (rating system or text? etc.), and who gives feedback on what (products, service_providers?), but it could be stored here.

5 Use cases – applying the data model

This section contains two use cases that were designed to make sure our data model is actually practically applicable to the chain concept. Each use case contains a short description of its purpose, a visual representation of the chain that is constructed, and a description of what happens in the database tabels when a certain step in the process of creating the chain is performed.

Note that in these use cases, the chains are constructed by service providers. It would also be possible to let chains be constructed by foodcoops. How this will work in practice remains to be seen. Regardless, this data model is able to store information about these chains and will work for both cases.

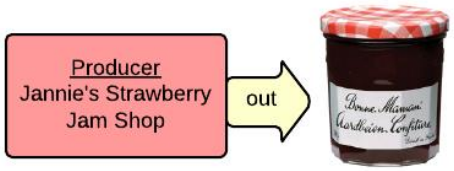
5.1 Use case 1 – Jannie's strawberry jam

Purpose

A very basic example that is used to show some basic interactions with the database. Shows the creation of a user account, how this account is connected to an organization and eventually a service provider. Also shows the creation of a very simple chain with no inputs and one output product.

Scenario

Jannie is a woman who has a very small business in Amsterdam North. She picks strawberries from her own garden and makes the most delicious strawberry jam herself. Jannie is the producer so she doesn't need any inputs. She has only one output, which is her product 'Jannie's strawberry jam', which belongs to the category 'strawberry jam'.



Product: Jannie's strawberry jam
Category: strawberry jam

Application of the data model

See next page.

Data use case 1 – Jannie's strawberry jam

Step 1

Jannie decides to create an account for the software with username 'jannie56'. *(Attributes that are not relevant or self-explanatory are omitted, for example in this case password, e-mail, phonenumber, etc.)*

user

id	first_name	last_name	username	organisation_id
1	Jannie	producer	jannie56	*NULL*

Step 2

Jannie fills in some information about her company. This information is stored in the organisation table, and the attribute 'organisation_id' is updated in the user table with the id of her organisation.

Rows that are added or modified during a step are colored green.

user

id	first_name	last_name	username	organisation_id
1	Jannie	producer	jannie56	101

organisation

id	name	place	country	type
101	Jannie's Strawberry Jam Shop	Amsterdam	Netherlands	supplier

Step 3

Now that Jannie's company is stored in the database, she can choose which services she can provide. She selects 'producer' because she doesn't have any inputs. *(In most cases the name of the service provider is the same as the name of the organisation. Exceptions are companies that provide multiple services, like production as well as transport.)*

user

id	first_name	last_name	username	organisation_id
1	Jannie	producer	jannie56	101

organisation

id	name	place	country	type
101	Jannie's Strawberry Jam Shop	Amsterdam	Netherlands	supplier

service_provider

id	name	organisation_id	type
201	Jannie's Strawberry Jam Shop	101	producer

Step 4

Now Jannie can enter the products she produces. For each product she can specify which inputs she needs and fill in other attributes like description, price, and category. She only produces one product, 'Jannie's strawberry jam', in the product category 'strawberry jam', and doesn't need any inputs. After she confirms all this, the product and its category are stored in the database.

Than, automatically a service link is created with one output product. Now, two things could happen depending on what Jannie choses to do. If she selects that the output product is the final product, which means that it can't be accepted as input by other service links, the service link is added to a newly created chain. If this is not the case, there is no chain yet, and the chain is not created. In this example we've chosen the first scenario.

user

id	first_name	last_name	username	organisation_id
1	Jannie	producer	jannie56	101

organisation

id	name	place	country	type
101	Jannie's Strawberry Jam Shop	Amsterdam	Netherlands	supplier

service_provider

id	name	organisation_id	type
201	Jannie's Strawberry Jam Shop	101	producer

service_link

id	service_provider_id
501	201

chain

id	times_used
701	0

service_link_position

id	chain _id	service_link_id	parent_service_link_id
801	701	501	*NULL*

input_category

id	service_link_id	product_category_id

product_category

id	name
301	Strawberry jam

output_product

id	service_link_id	product_id
601	501	401

product

id	name	product_category_id	price
401	Jannie's strawberry jam	301	1.20

5.2 Use case 2 – BioRomeo's biological beets

Purpose

This use case goes more in-depth in the chain concept. The chain that is created here contains all three possible types of service providers: a producer, a transporter, and two processors. This use case also has a service link with two outputs.

Scenario

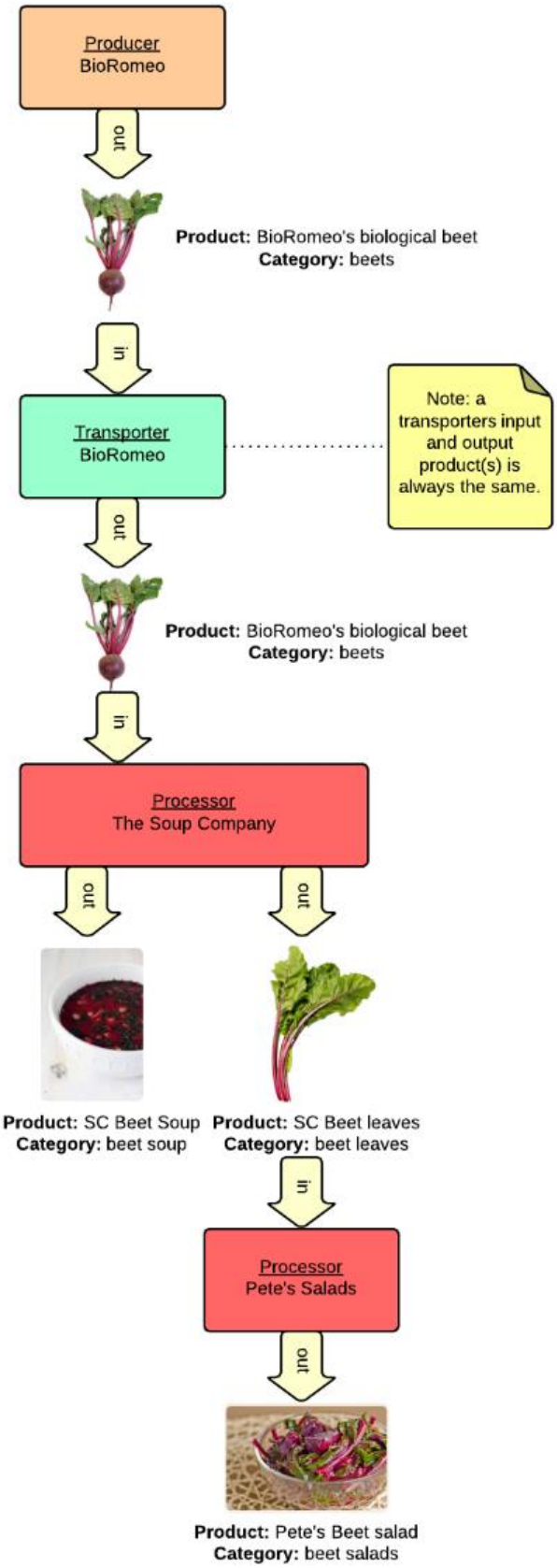
BioRomeo is a producer of many products. In this scenario, we focus on a chain that can be constructed from the beets produced by BioRomeo. BioRomeo also takes care of the transport.

The Soup Company is a processor that makes soups from all kinds of ingredients. To create beet soup, they obviously need beets. After looking through a selection of products in the beets category, they have decided to go with "BioRomeo's biological beets". Because the Soup Company doesn't process the beet leaves, it decides to sell them as another separate product.

Pete's Salads needs beet leaves for a beet salad. After looking through some chains that have beet leaves as output product, he decides to selects those from the Soup Company.

Application of the data model

See next page.



Data use case 2 – BioRomeo's biological beets

Step 1

For this example we have skipped the steps of creating a user and an organisation. Assume that the following service providers are created. As you can see, BioRomeo provides two services: production and transportation.

service_provider

id	name	type
1	BioRomeo	producer
2	BioRomeo transport	transporter
3	The Soup Company	processor
4	Pete's Salads	processor

Step 2

Now, BioRomeo adds a product: BioRomeo's biological beet. It is an output from the production process which requires no inputs, so a service link with one output product is added. This service link is stored in a new chain. A row is added to service_link_position to determine that the service link is the first in the chain, because it has no parent service link.

service_provider

id	name	type
1	BioRomeo	producer
2	BioRomeo transport	transporter
3	The Soup Company	processor
4	Pete's Salads	processor

Rows that are added during a step are colored green.

service_link

id	service_provider_id
101	1

chain

id	times_used
401	0

service_link_position

id	chain _id	service_link_id	parent_service_link_id
1	401	101	*NULL*

We assume that the software already has a list of product categories stored in the database.

input_category

id	service_link_id	product_category_id

product_category

id	name
301	Beets
302	Beet soup
303	Beet leaves
304	Beet salads

output_product

id	service_link_id	product_id
1	101	201

product

id	name	product_category_id	price
201	BioRomeo's biological beet	301	0.80

Step 3

The next part of the chain is the transportation step. This is another service link in the already created chain. No rows are added to the input_category or the output_product table, because the transportation step doesn't create or modify a product. The input and output product(s) are always the same in the transportation step. We assume that the system can compute what the input and output product(s) of this service link are by looking at the output product(s) of the previous service link in the chain. This way we prevent a duplication of data in the input_category and output_product tables.

service_provider

id	name	type
1	BioRomeo	producer
2	BioRomeo transport	transporter
3	The Soup Company	processor
4	Pete's Salads	processor

service_link

id	service_provider_id
101	1
102	2

chain

id	times_used
401	0

service_link_position

id	chain _id	service_link_id	parent_service_link_id
1	401	101	*NULL*
2	401	102	101

input_category

id	service_link_id	product_category_id

product_category

id	name
301	Beets
302	Beet soup
303	Beet leaves
304	Beet salads

output_product

id	service_link_id	product_id
1	101	201

product

id	name	product_category_id	price
201	BioRomeo's biological beet	301	0.80

Step 4

In this step, a processor called The Soup Company adds his product: SC Beet soup. However, it needs an input: beets. This needs for beets as input is stored in the input_category table which is linked to a new service link. He decides to select the product from BioRomeo: BioRomeo's biological beet. He doesn't need the leaves of the beet, so he decides to sell it as another product, which is thus another output product of this service link: SC Beet leaves.

service_provider

id	name	type
1	BioRomeo	producer
2	BioRomeo transport	transporter
3	The Soup Company	processor
4	Pete's Salads	processor

service_link

id	service_provider_id
101	1
102	2
103	3

chain

id	times_used
401	0

service_link_position

id	chain _id	service_link_id	parent_service_link_id
1	401	101	*NULL*
2	401	102	101
3	401	103	102

input_category

id	service_link_id	product_category_id
1	103	301

product_category

id	name
301	Beets
302	Beet soup
303	Beet leaves
304	Beet salads

output_product

id	service_link_id	product_id
1	101	201
2	103	202
3	103	203

product

id	name	product_category_id	price
201	BioRomeo's biological beet	301	0.80
202	SC Beet soup	302	2.00
203	SC Beet leaves	303	0.40

Step 5

This step concerns another processor that makes salads: Pete's Salads. Pete adds his product: Pete's beet salad, which is the output product of a new service link. He also needs an input from the product category: 'Beet leaves', which is stored in a new row in the input_category table.

Pete selects the product 'SC Beet leaves', which means the service link is added to the already created chain in the table service_link_position.

service_provider

id	name	type
1	BioRomeo	producer
2	BioRomeo transport	transporter
3	The Soup Company	processor
4	Pete's Salads	processor

service_link

id	service_provider_id
101	1
102	2
103	3
104	4

chain

id	times_used
401	0

service_link_position

id	chain _id	service_link_id	parent_service_link_id
1	401	101	*NULL*
2	401	102	101
3	401	103	102
4	401	104	103

input_category

id	service_link_id	product_category_id
1	103	301
2	104	303

product_category

id	name
301	Beets
302	Beet soup
303	Beet leaves
304	Beet salads

output_product

id	service_link_id	product_id
1	101	201
2	103	202
3	103	203
4	104	204

product

id	name	product_category_id	price
201	BioRomeo's biological beet	301	0.80
202	SC Beet soup	302	2.00
203	SC Beet leaves	303	0.40
204	Pete's beet salad	304	1.30

6 Discussion

Does a service link have to be part of a chain?
No, not if the input products aren't specified. At this point the service link exists in the database with an input category and an output product. It is added to a chain when a specific input product is selected.

Why is the beet salad part of the same chain as the beet soup in example 2?
We want to keep track of everything that happens in a chain, so we also want to see what happens with other outputs of certain processes that lead to a certain product. When writing the code for the software, it would also be possible to only show the chain up the beet soup by only querying the service link that belongs to it, along with its parent service links in service_link_position.

Who creates the chains?
We will have to see how this will work in practice. Our idea as of now is that suppliers create their own service links by specifying their input categories and output products, which can be connected to a chain by either other suppliers or foodcoops.

Deliverable 3: User-interface mock-up

Content

1 Introduction

Coming up with a technical solution to facilitate the creation of food chains is one thing, visualizing the complete process of putting those chains together is another challenge. In this section we will draw out a complete start-to-end use case of how a chain might be constructed in Foodsoft. The main goal is to paint the big picture and to give future developers inspiration and a starting point.

There are three main components the Foodsoft user interface needs to encapsulate for the chain construction process:

- **Flow:** during the process the user needs to be completely aware where he or she is and where he or she is going. Furthermore, he or she needs to gain a sense of achievement during the process; a feeling that he or she is actually doing something and working towards an end goal.
- **Consequences:** what consequences do the choices the user makes have on not only the process, but also the world/environment? For example, if the user selects Transport Company A to transport products; what are the advantages it has over Transport Company B? These consequences should be clearly shown to the user.
- **Values:** FoodCoop stands for delivering food according to your personal values and ideals. How does the system capture these values and make a recommendation based on them? We think this is a different and challenging problem altogether. We will touch briefly upon some ideas on how you might approach this problem, but for the purpose of simplicity we will assume that the system knows about the user's values.

2 Use case

In the sections below we present a complete process of a single user putting together a food chain. This use case is an extended version of the simpler clickable mockup we present in the domain: www.klik-je-keten.nl/mockup. This use case has the purpose of making the process of clickings chains together as clear as possible.

Imagine a 30 year old woman called Sophie. She is divorced and has two little children to take care of. She is discovering a new lifestyle and wants her children to grow up thinking about durability and to take good care of the world they are living in. That is why she starts to think more about where the food she is buying is coming from. She discovers a foodcoop and learns about a cooperation following the values in life they care about in buying food. Via the software Foodsoft she is able to search for the food she wants by composing her own food chain from start to end. She decides that she wants to buy strawberry jam from a supplier that fits her needs and values.

Assumptions

- The system makes use of a wizard, in the initial step of the process, that enables the creation of a starting selection of suppliers to begin with. The wizard has made a selection of strawberry suppliers who are of most interest to the user. The elaboration of the wizard does not fall into our domain. For this mockup we assume that the wizard is able to generate the output we stated above.
- The transporters and processors that are made available after your first choice are dependent on the supplier the user selected and again a selection will be made at the hand of the output of the wizard.
- The order size of the users fits to the desired minimum abatement of the supplier, transporter and processor.
- The destination (drop-off point) is a fixed location that the foodcoop must specify beforehand. We assume that the foodcoop has done this.

Step 1 – Logging in

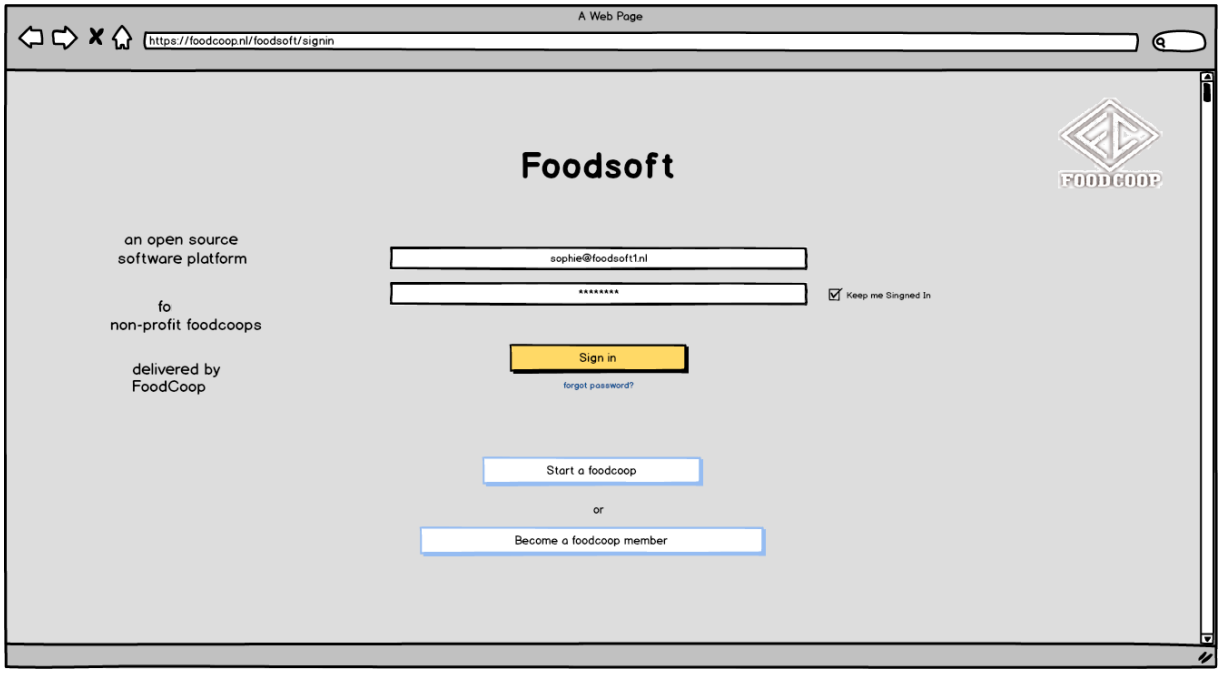


Figure 1: Login Screen

The process begins by logging in to the Foodsoft software. Sophie fills in her username email address (here: sophie@Foodsoft1.nl), types in her password and clicks on '*Sign in*'. Now she enters the next screen with a menu where she has access to a few tabs (depending on her user role and access permissions).

2. Starting a new chain

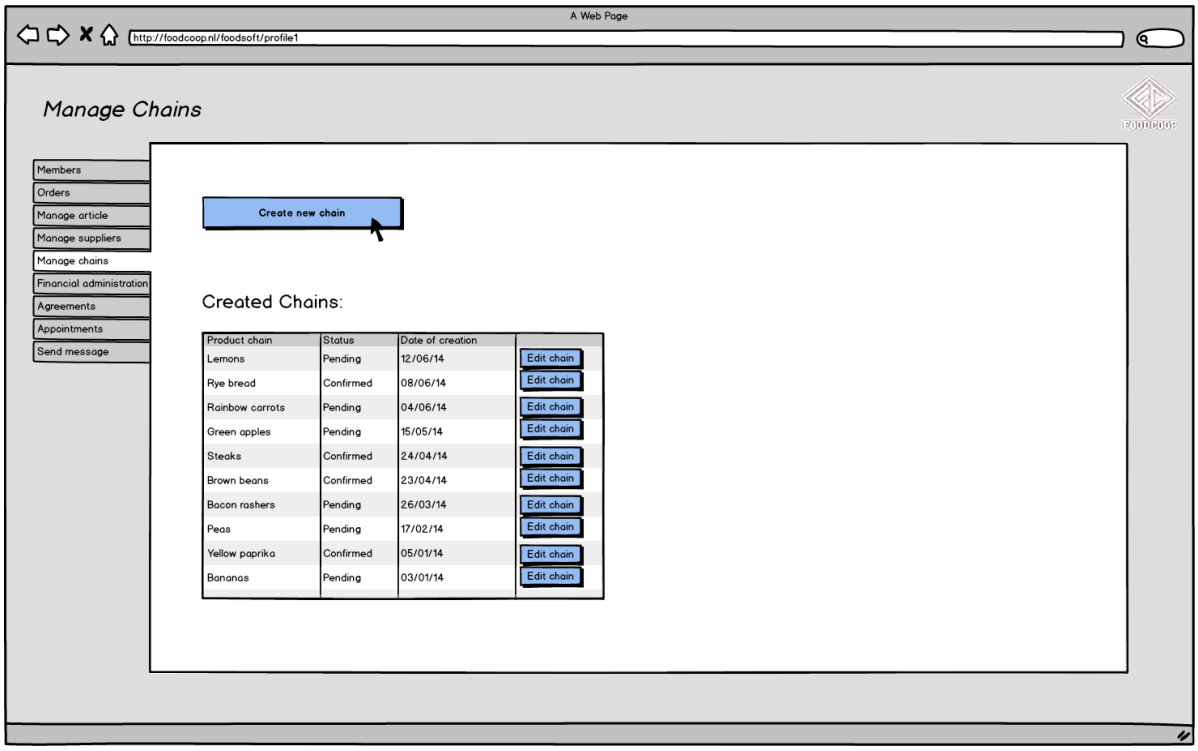


Figure 2: "Manage chains" gives an overview of existing chains and the ability to create a new chain

After having logged into Foodsoft Sophie continues by selecting the tab ‘Manage Chains’. On this screen, she sees an overview of all the chains available within the foodcoop.
It also gives her the option to create a new chain by clicking on ‘create new chain’.

3. Selecting a product category

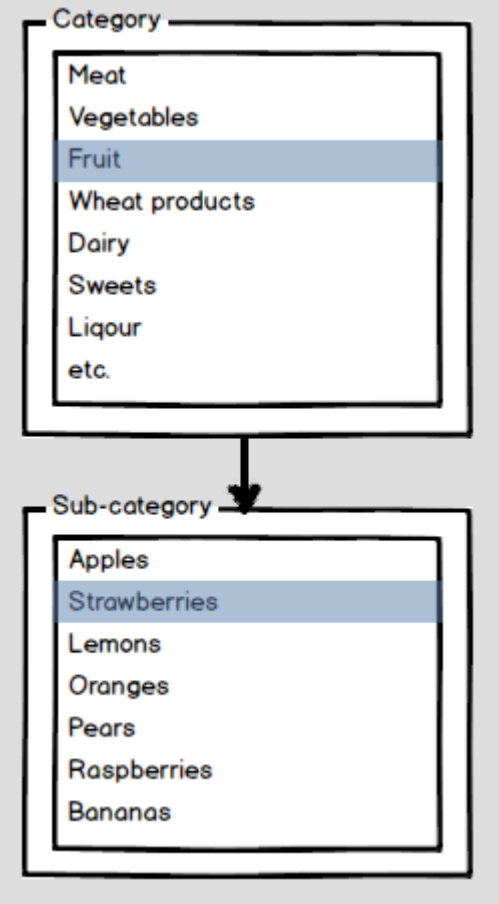


Figure 3: Category selection the user makes

Sophie has started the creation of a new chain. The first step in the process is the selection of a product category. Sophie decided to look for strawberries. She wants strawberry jam so she clicks on “Fruit” and then “Strawberries”. The number of available products is dependent on which suppliers are connected to the foodcoop.

4. Chosing a supplier

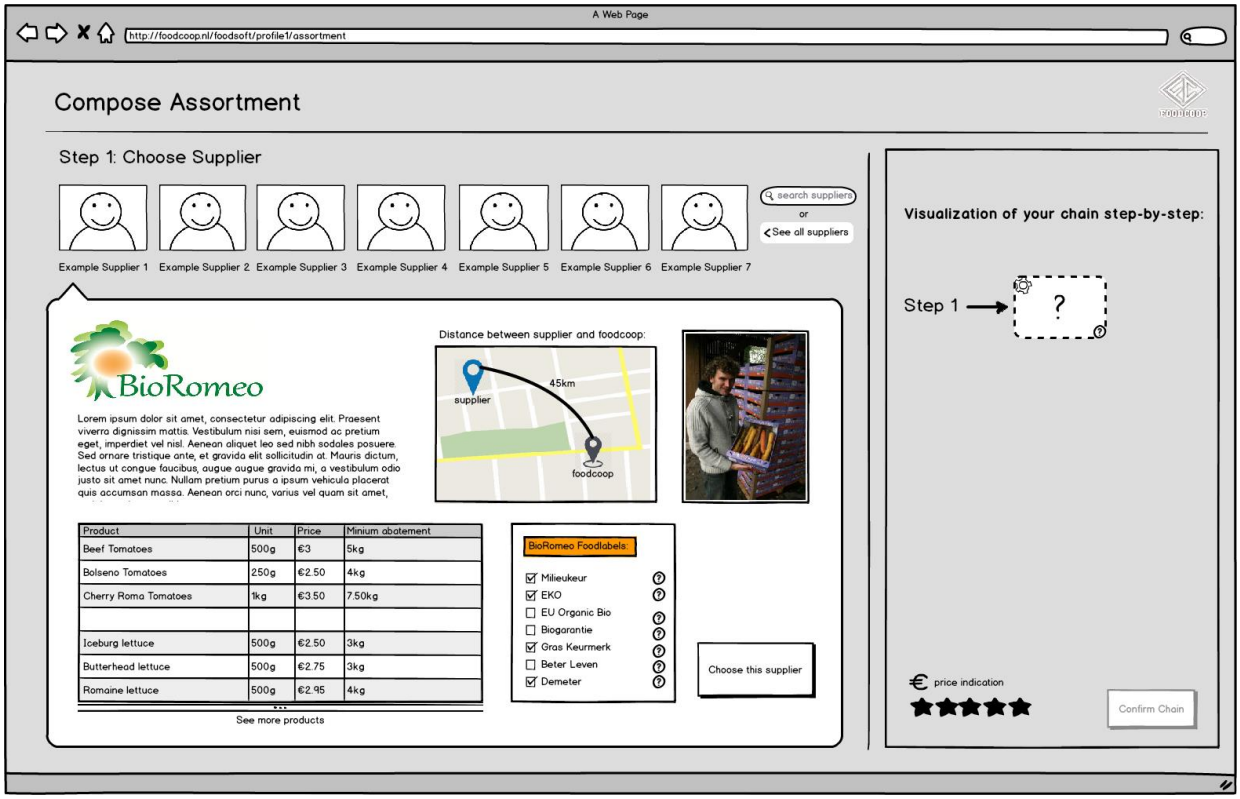


Figure 4: Supplier selection after having chosen the product category (step 1 in the process)

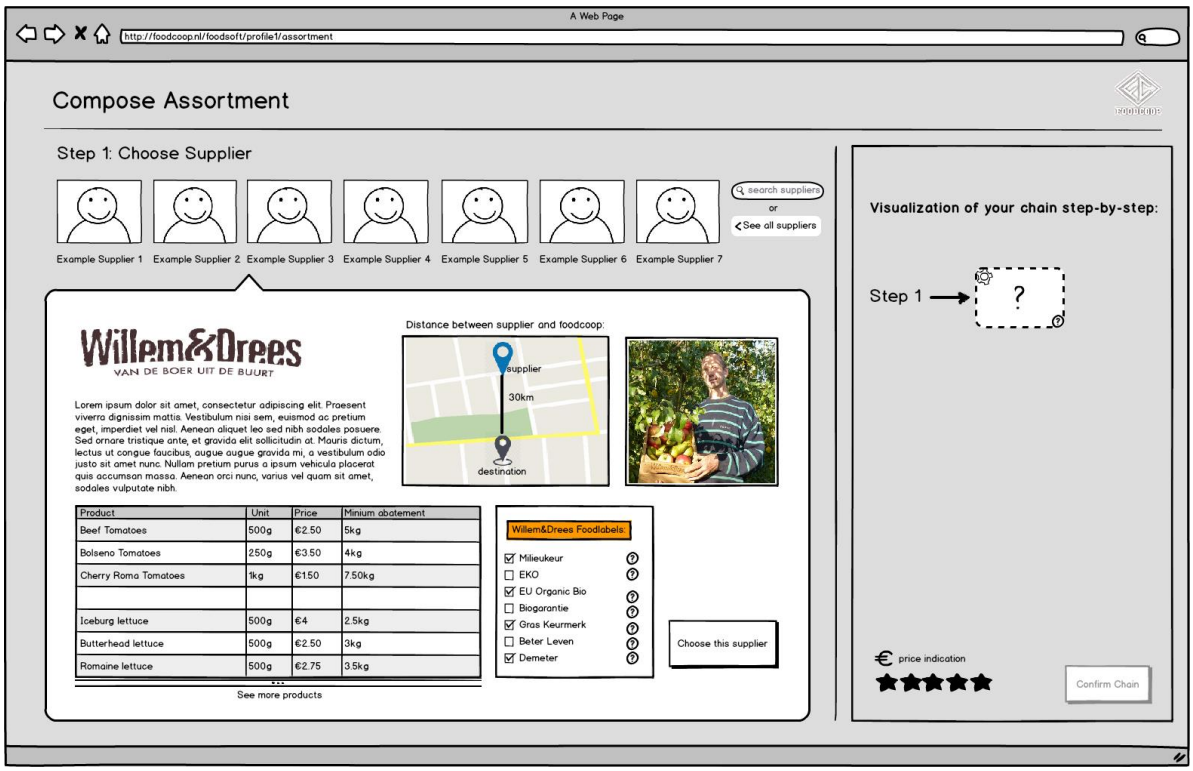


Figure 5: Another available supplier in the first step of choosing a supplier

The selection of product category ‘strawberries’ leads to a selection of example suppliers who all deliver strawberries. Sophie is able to click on other supplier profiles like (for example) ‘Willem &

Drees' to decide which supplier fits best to your values. She decides that BioRomeo is the better choice (more relevant to her values). She selects BioRomeo and clicks the "Choose this supplier" button.

5. Choose transporter

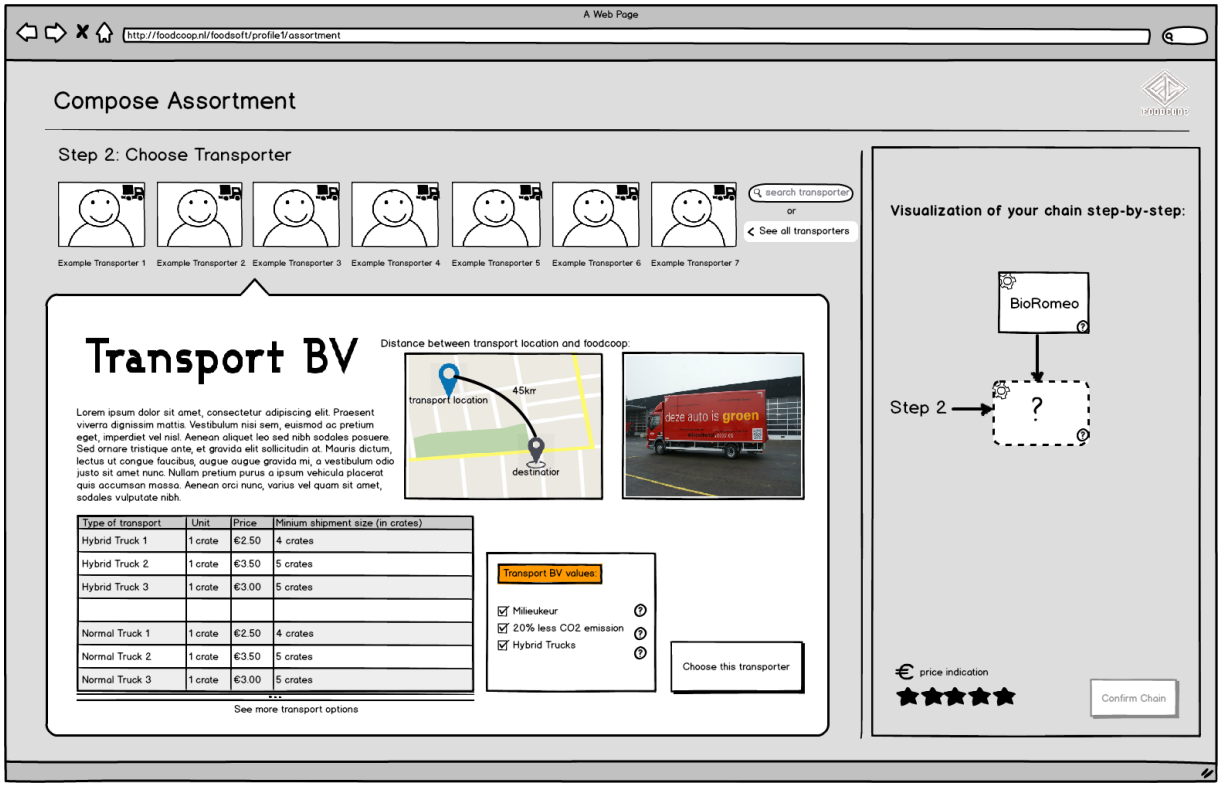


Figure 6: Transporter selection after having chosen a supplier to start with (step 2 in the process)

The first step of composing the chain is done, Sophie wants her strawberries to come from 'BioRomeo'. This is visualized in the chain on the right.
For making the jam, the strawberries have to be processed. To realize this, the strawberries have to be transported to a processor. Therefore Sophie makes a choice of transporter. While browsing through the available transporters, Transporter BV stands out in her opinion because of their use of hybrid trucks and guarantee of 20% less CO2 emission. She clicks the button "Choose this transporter".

6. Chosing a processor

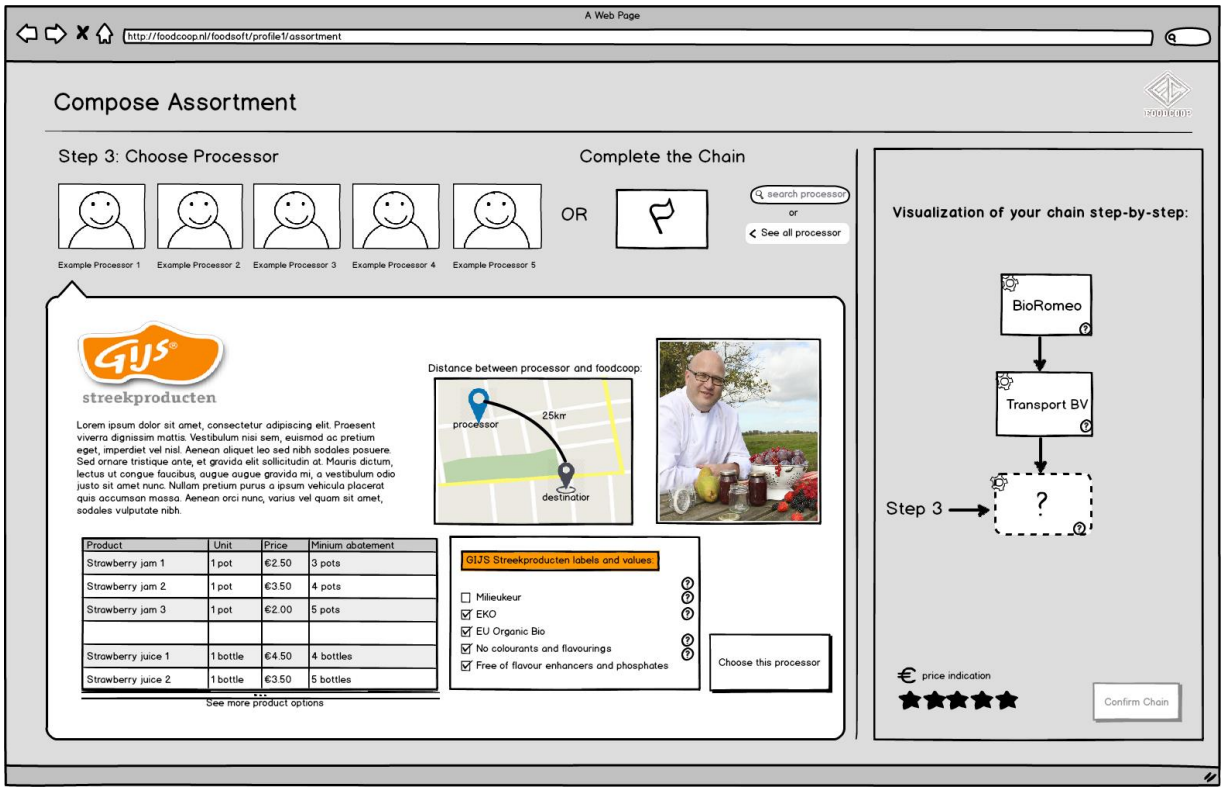


Figure 7: Selection of a second processor to process the strawberries (step 3 of the process)

At this point it is actually possible for Sophie to select her destination and complete the chain. This would be the case if Sophie wanted to have just strawberries. Because she wants to make jam out of it, she needs to choose a processor.
GIJS Streekproducten seems like an excellent choice because they don't use flavour enhancers. This is a big thing for Sophie. She clicks the “Choose this processor” button.

7. Chosing a transporter

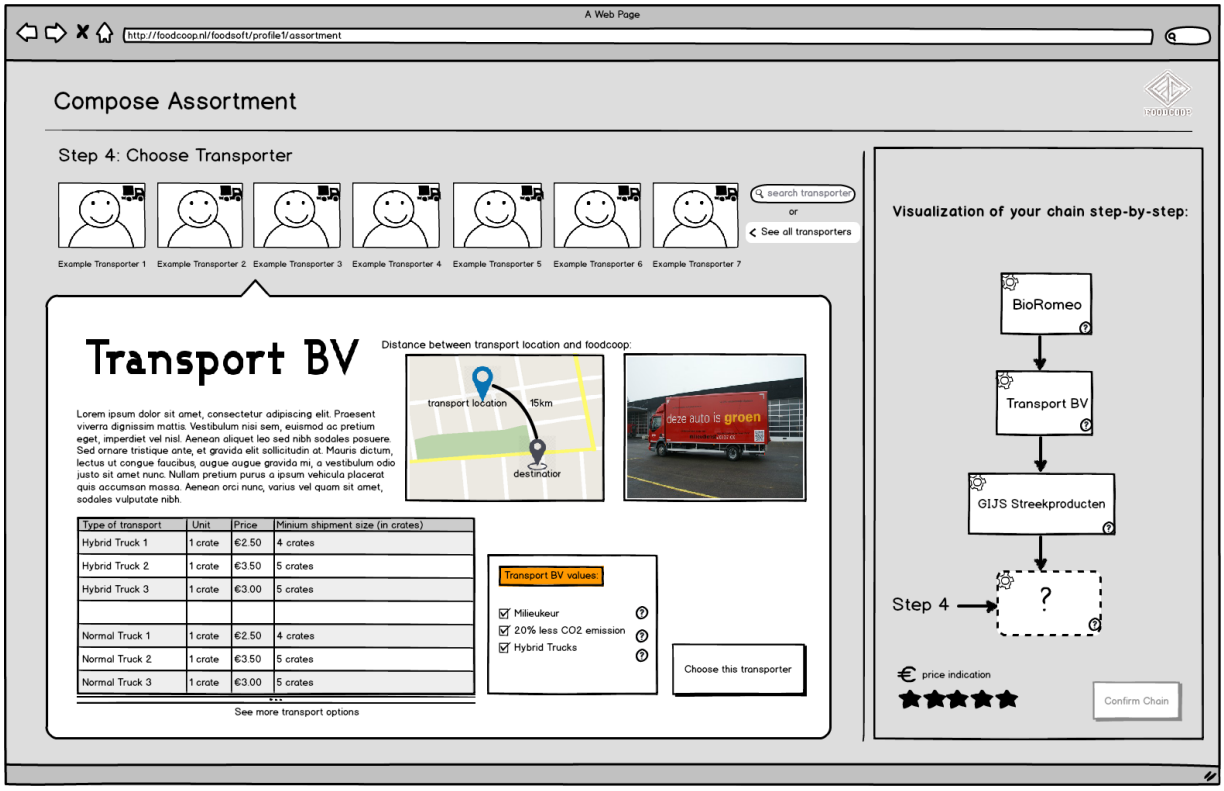


Figure 8: Selection of a second transporter to deliver the strawberries to the second processor

At this point the chain is almost done and Sophie needs to choose another transporter to deliver the jam to its final destination.
Fortunately, Transport BV can also transport between the processor and her destination. She clicks “Choose this transporter”.

8. Chosing another processor

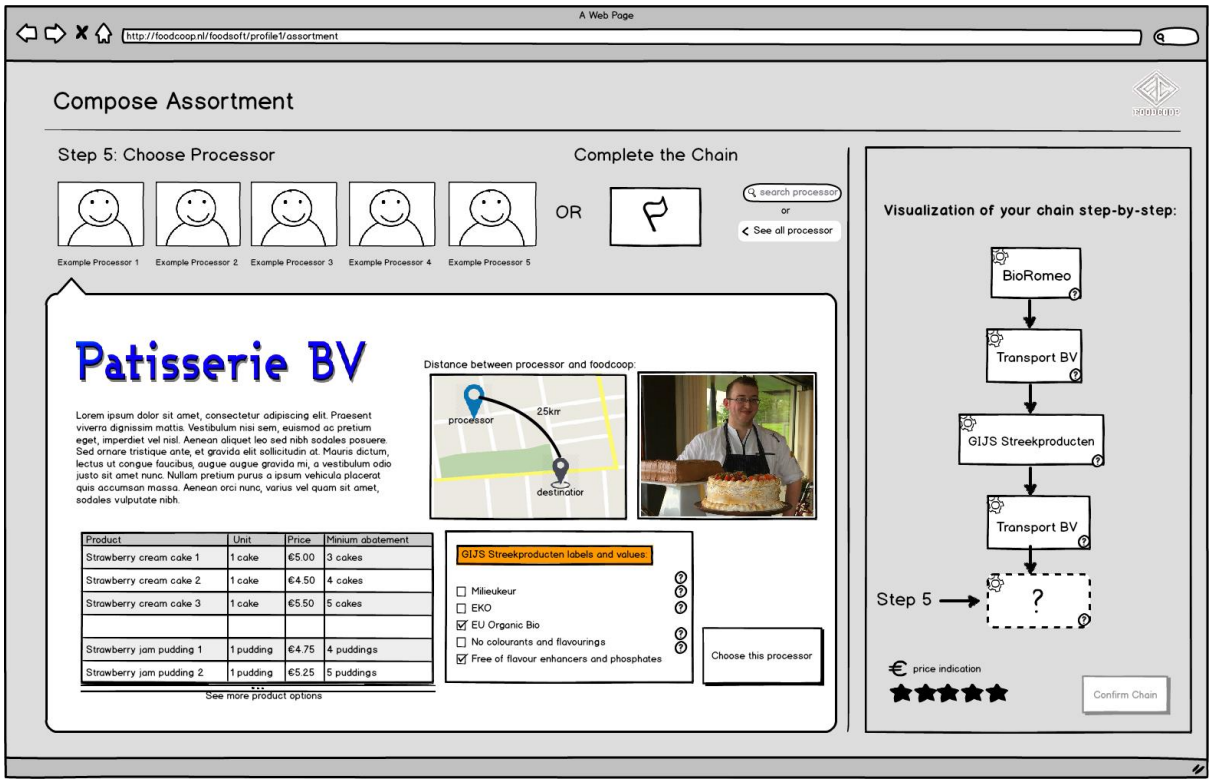


Figure 9: Selection of a second processor

Optionally, Sophie could use the strawberry jam as input for another processor to make pie out of it. She wants jam only so she completes the chain by clicking on the flag icon (destination).

9. Chosing a destination

Figure 10: Selection of the product destination on the map

After having completed the process by clicking the button “Confirm Chain”. Sophie receives the following message.

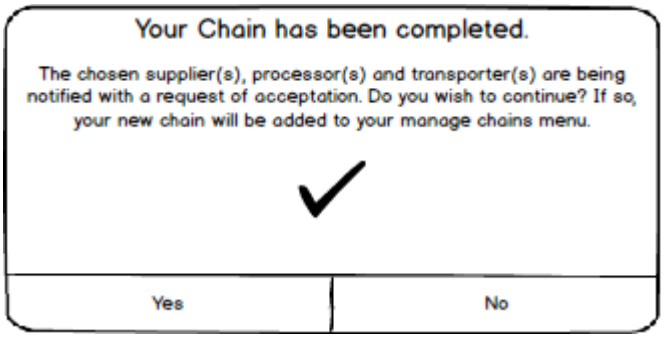


Figure 11: Chain confirmation dialog

10. Pending chain

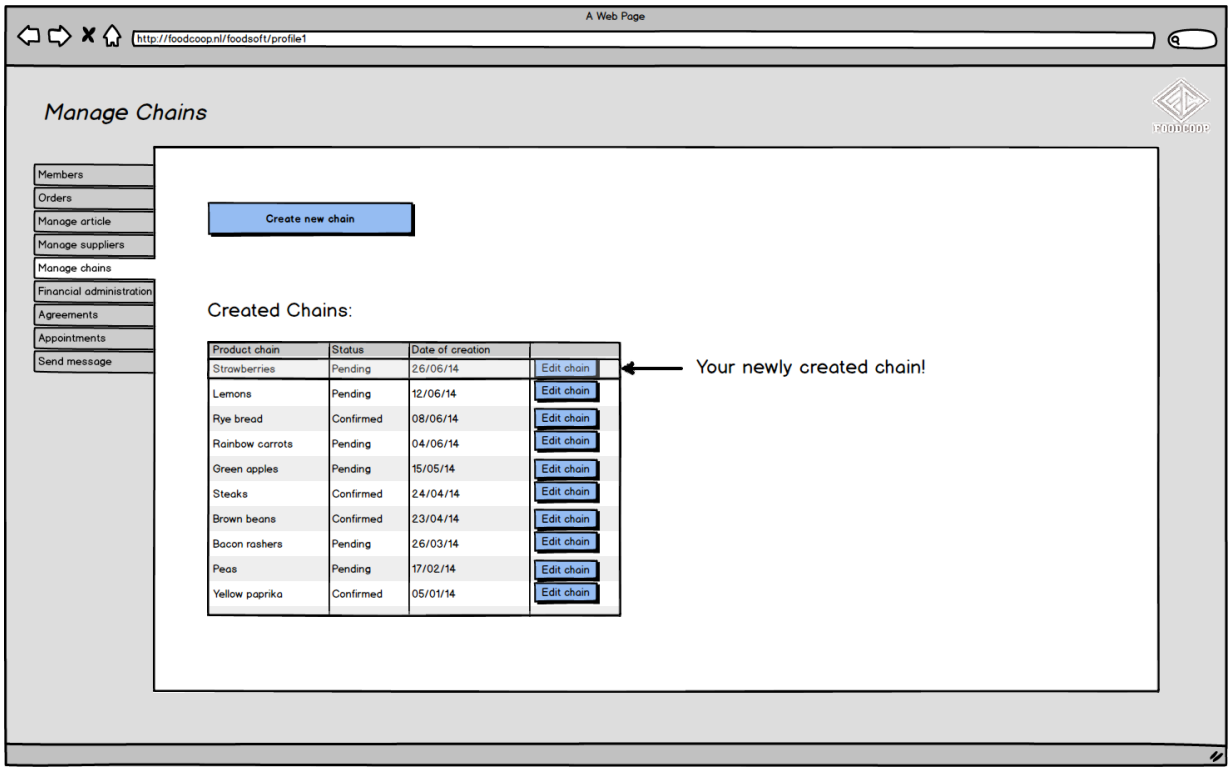


Figure 12: "Manage chains" screen with a newly created chain.

After the chain is confirmed it will show up in the "Manage Chains" screen as a pending chain. When all the stakeholders (suppliers, processors) have confirmed, the chain will be available and the product can be ordered.

3 Discussion

We want to make clear that the above mock up is just our interpretation of the problem translated to the screen. There are possibly many other ways you can visualize it. We are dealing with complex problems and we assumed many things during the process. We will dive deeper into some of these assumptions.

Consequences

The user interface needs to show the user what the consequences of his or her choices are. We believe that the most important questions here are:

- How do you define a consequence?
- How do you express a consequence so that it is apparent to the user?

A consequence could be *less CO2 emission*, but also the *price indication* of a complete chain. We have shown this in the mock ups, but we won't elaborate on how to exactly calculate or measure these consequences. That is a (difficult) challenge on itself and it lies outside the scope of this project.

Values

Similarly to consequences; it is a challenge to identify a foodcoop's (or a person's) values. This too lies outside the scope of our project. We will try to provide some ideas for the purpose of inspiration.

One idea is to capture these via a wizard-like interface where the user answers a set of questions that will determine his or her values. However, it would be challenging to come up with these questions because, for example, everyone wants their food to not be genetically manipulated if they would have the option and you would get similar answers (and values).

Another approach is to weigh values against each other; "which is more important? value A or value B?".

It is important to come up with an approach that can capture the user's values so that it facilitates the process of systematically determining which suppliers and processors best match his or her values.

Collaboration between foodcoops

Another important and valuable aspect for foodcoops using Foodsoft would be the ability to cooperate with each other. This can, for example, establish a situation where foodcoops can group up and request orders from suppliers that only deliver at a certain number of orders. When they collaborate, more people can join in on a product order and the minimum amounts can be reached. A second useful component of the system would be the ability to discuss with each other as foodcoops. This can be about various kinds of topics like the promoting of a supplier that is really corresponding to someones values in their way of operating or exchanging tips and ideas about chains or organizational aspects.
Sharing chains can also be a great addition. This way accepted chains don't always have to be made from scratch and existing chains can be chosen of which another would not have thought of in the first place.

All the ideas we explained above are additions to the software we think would be really valuable when implemented. However this falls out of the boundaries of our study. In the mockup screen below we present a representation of how we think these ideas could be

implemented in a user interface but we want to emphasize that this is only one example of a possible visualization. These are our ideas that we hope can be used as inspiration for further development in the future.

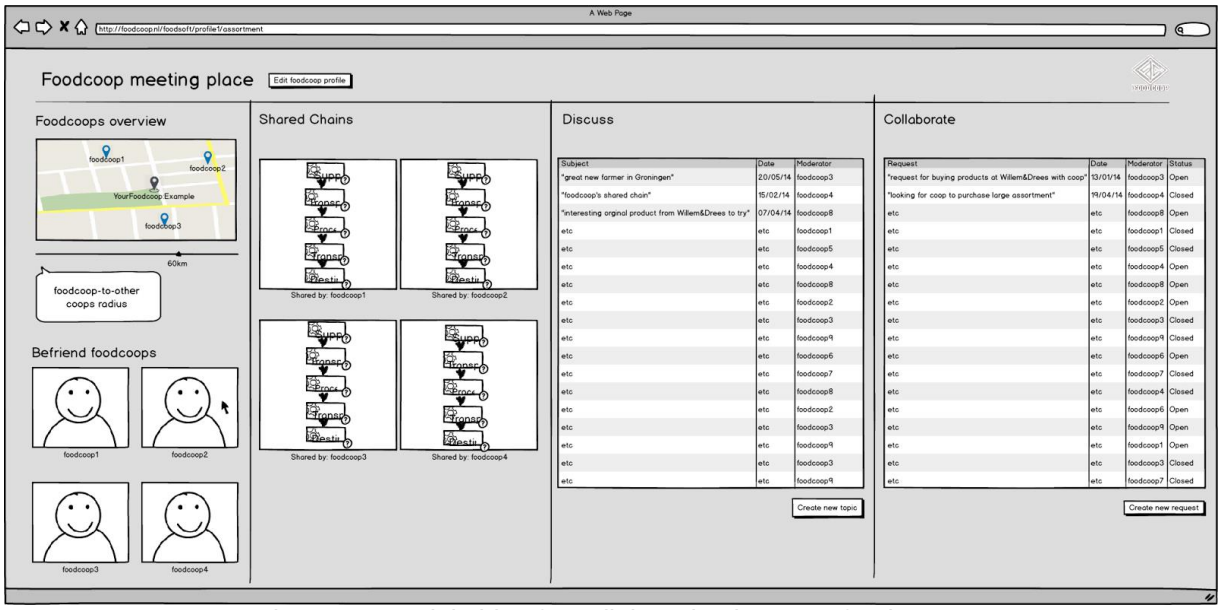


Figure 13: Module idea for collaboration between foodcoops

For the ideas above, we created a separate module containing four different sections. The first section displays an overview of the foodcoops you have cooperated or discussed with and are in that way ‘befriend’ with each other. Here anyone can easily click on other foodcoop profiles and see information about the values they live up to. This way it is possible to see whether another foodcoop has corresponding values and for example decide to request a collaboration.
Shared chains provides an overview of different examples of chains made by other foodcoops. These can be used by anyone if the chain is accepted by the concerning suppliers, processors and transporters. In the mockup we have visualized the chains by showing little screenshots of the example chain.
For the discuss and collaboration sections we came up with the idea of implementing a forum system where topics can be posted and replied to by different foodcoops. In the collaboration section a foodcoop can place requests on a message board to a single or all foodcoops.