PRO-2022 Cahier des charges

V3.3 - 20.03.2022

Team members & roles

Équipe 01 :

- Ackermann Simon (Développeur Backend simon.ackermann@heig-vd.ch)
- Boegli Noah (Développeur Backend, remplaçant chef de projet noah.boegli@heig-vd.ch)
- Comte Emmanuelle (Tests techniques, gestion des données et responsable qualité emmanuelle.comte@heig-vd.ch)
- Gachet Jean (Product Owner, Développeur Backend jean.gachet@heig-vd.ch)
- Gallay David (Chef de projet david.gallay@heig-vd.ch)
- Mendes Hugo (Développeur Frontend hugo.mendes@heig-vd.ch)
- Terrani Fabien (Développeur Frontend et responsable UI/UX fabien.terrani@heig-vd.ch)

(ordre alphabétique)

Project introduction & context

Have you ever found yourself endlessly Googling for meals ideas ? Or trying to choose what to cook for your next meal with your laser-game squad? Or been in a city you don't know, trying to find a restaurant that would match your tedious food preferences?

This is where our project positions itself.

Centered around a user's preferences: allergies and intolerences, meal type and origin, preparation time wished, we will solve the daily challenges that one may face when it comes to eating.

A few words about allergies

Since a few of our team members have allergies, we know that it's a complicated subject to manage and there is no correct way of doing it. A specific focus on this matter has been put and we have developed multiple ways to maximize the accuracy of the information.
However, we are not a medical platform and do not wish to be one. The information is and will be as accurate as our data provider can be (whether it is an external company or sample data found on the internet).
More on this subject can be found in the Specification / Global vision section.

Objectives

The market for food&eating realted websites is already well taken by major actors in our initial geographical permiter (French-speaking part of Switzerland): Marmiton, Betty Bossy, a simple Google search, everlasting discussion on whatsapp groups. If we want to be competitive, we have two choices

- Take on a specific competitor and be better than them
- Gather features of each competitor to have the best of all worlds in our product.
- Provide something they don't

We are aiming at the second and third options. We want to replace endless Google searches for a meal idea, endless Whatsapp discussions on what to eat at the family gathering.
With a recipe engine based on one or multiple user's "food profile", a meal idea can be found easily, directly linked to a recipe on our platform.

In conclusion, we will not outperform our competitors by being better than them, but by having a bit of each feature of our different competitors and allowing a user to remain on our platform for every part of the eating-planning process, which other services don't provide.

Detailed uses cases

1. Allergy concerns in meal choices (required)

Context

A young adult, about 30, invites some friends to eat on a Friday night. She wants to cook a nice meal, with a starter, a main course, and a desert. The budget is not a problem, but some of the guests have allergies.

Solution

She creates a group in the application and adds all of her guests. The application then suggests some meals taking into account the allergies of her guests.

2. Meals containing specific ingredients (required)

Context

A student is about to meet the parents of her boyfriend for the first time. She wants to impress them with a nice homemade meal. She wants red meat to be the center piece of the main course.

Solution

She inputs her criteria and the application suggests meals with rib eye steaks or beef tenderloin.

3. Meal cooked in a specified amount of time (required)

Context

A student doesn't have much time to prepare a lunch. He knows that he has leeks in his fridge that he needs to eat soon. Moreover, he needs a meal that requires about 20 minutes of preparation.

Solution

The application will therefore suggest a list of recipes containing leeks and cooked quickly.

4. Restaurant suggestion (required)

Context

An employee is in charge of choosing the restaurant for the end-of-the-year office party. He doesn't know the individual preferences of all of his coworkers, but they are all signed up on the platform.

Solution

He inputs who will partake and the approximate location where the restaurant should be. The application then suggests a list of restaurants best matching the preferences of his coworkers.

5. Voting on group meals (wished)

Context

A group of friends is planning to cook together. They each have individual preferences and can't reach a decision over what to cook.

Solution

They all join a group and the application suggests a list of possible meals. They can vote on their preferred choice. Once everyone voted, the application shows the results.

6. Shopping list (wished)

Context

A recipe has been chosen for a meal between friends. The host must now go and buy all the ingredients ahead of the meal.

Solution

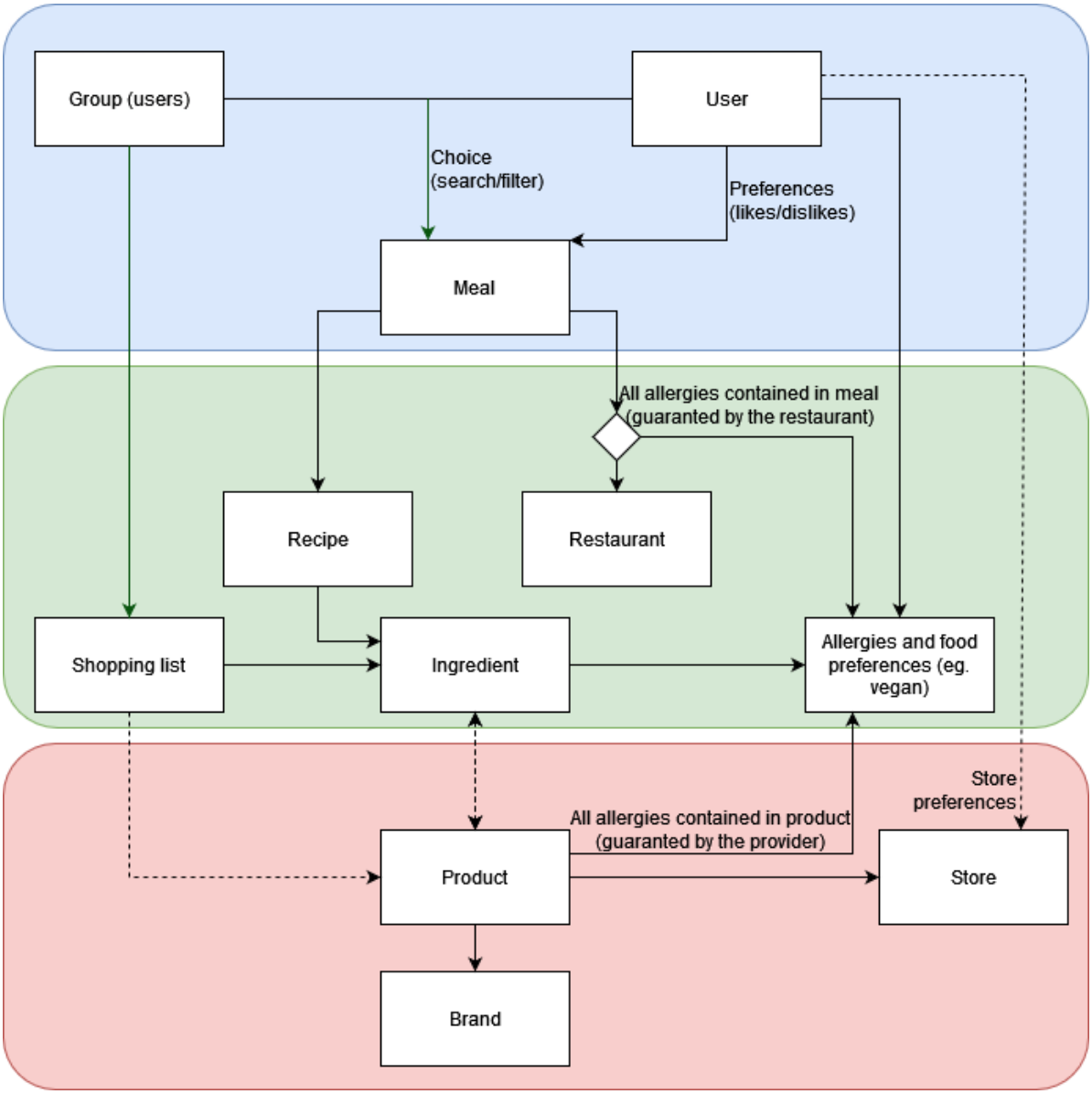
The application automatically generates a shopping list based on the recipe and the number of guests.

Specification

This section states the features that will be supported on each iteration.

Global vision

This diagram show global vision of the features that would be supported by the application.



Blue

The users can choose a meal for themselves or for a group. This is barely the core of the service and needs to be extended by other blocks.

Green

A meal is either cooked by the user using a recipe or provided by a restaurant.

- The meal provided by a restaurant contains a list of allergies, intolerences or food preferences (religion, veganism, etc.). Those are managed and (legally) guaranteed by the restaurant to be accurate.
- The meal cooked by the user with a recipe is made of ingredients. These have intrinsic allergies or food preferences incompatibilites, however this is not linked to provider-specific products that may contain more. This issue is addressed in the red block.

Red

An ingredient (whether it's a basic or composite ingredient) is only an abstract of a real product. Those products are sold by different providers and may have more allergies or food preferences incompatibilites based on how and with what they are made. Those are managed and (legally) guaranteed by the provider to be accurate.

Revenue stream

Active advertising have a negative impact on user experience which is bad for us as we want to create a community. Especially considering that the application will mainly be used on phones. This kind of service works better with:

- Premium tier adding extra features
- Product promotion: Companies can pay to have their products highlighted.

Considering that the benefits are proportional to the number of users, we need to gain popularity quickly to generate bigger revenues. If we prioritize premium features, we could generate our first incomes quicker, but the community would grow smaller as there would be fewer free features. The choice is made to prioritize importants freemium features. This is an investment in order to maximize the future benefits.

Iterations

Project start date: 9.3.2022
Project end date: 3.6.2022
Total: 12 weeks

- The project will be delivered at 3 stages of its development. The most important features are all integrated in the first two steps, but the importance alone won't define the priority. They were selected then planned in order to optimize the delivery process.
- If needed, features can be shifted to the next iteration. The third iteration is kept for this usage, otherwise it will be used to add extra features.
- An intermediate presentation will take place on the first iteration's end.

1. First working version (5 weeks -> Date: 13/4/2022)

User features

- User can sign up and sign in into the application using login/password
- User can sign up and sign in into the application using OAuth
- User can update their profiles
    - Add/remove their allergies/ethics (e.g. Vegan)
    - Add/remove preferences (likes and dislikes)
- Users can find a meal and a recipes based on their profiles and the following filters
    - Recipe contains specific ingredients
    - Recipe cooking-time under a specified value.
    - Type of meal (Asiatic, Italian, Mexican, ...)
- Users can shuffle the search result
- User can choose a meal provided by a restaurant

Additional features

- Definition of the entity model
- Research of source of data
- There is an easy way to import large amount of data (Meals, recipes, ingredients and products)
- There is a way to bind restaurant to meals and defined the allergens they contains
- Samples of real data will already be available
- There is a way to differentiate premium users from freemium ones

2. Groups Management & Restaurant (5 weeks -> Date: 18/5/2022)

User features

- User can add/remove other user as their friends.
- User can create and delete groups
- User can add/remove other users (even not their friends)
- User can select the meal and recipes for the group
- User can add its own recipes (available for anyone)
- User can mark recipes.

Additional features

- Data will be added in a larger scale.

Extra Weeks (2 weeks -> Date: 1/6/2022)

We keep some weeks to enhance the previous iterations and consolidate the deliverables. If we have enough time, we will add some premium features from the backlog.

Example of User features (paid features)

- User can plan meals for a period and generate a shopping list
- Agenda (google) integration: user can bind a meal date with its calendar

Additional features

- User can choose some meals and let the group vote
- Enhancement of existing features
- This iteration is also a buffer for previous iterations

Disclaimers

- This application will not provide integrated chats or forums, it does not aim to be a communication mean.
- This application won't allow to post
- This application is not a delivery service itself as UberEat or Smood. But it may integrate with those kind of service in future extensions.
- All the paid features won't be integrated right away. Most of them will be added after the production release along with the community's growth.
- The application won't handle multiple languages.

Technical Specification

Stack

"LAMP" Web application, divided into backend/frontend:

- Apache2-Php with Symfony framework for the backend.
- Javascript with Angular for the frontend
- MariaDB as DBMS
- OS Linux

Remarks

- The choices were made according to the skills and wish of the team's members as long as the technology suited the project well. This ensure that the technologies are mastered and reduce the risks.
- MariaDB was choosen over Postgresql because it is well integrated with php and it is easier to find DBaaS providers for MariaDB.

Resources needs

- There is no strong need for resources, even on connection peak during meal time
    - There will not be any feature that requires a lot of resources
    - There will not be any feature that will be solicited a lot on a small period of time (i.e. thousands of requests each secondes)
- The application is monolithic and will be launched on a single platform. Therefore:
    - it does not need to be cross-platform.
    - Docker is not a real advantage: Docker is very good to handle micro-services architectures
- The application's success lay on its popularity. The UX needs to be really well made to please the user and the overall project needs to be maintainable in order to be extended.

Deployment (Production and Dev)

Since our application is a monolith, docker is not a good choice.

- Docker is very good when it comes to micro-services architectures and broad-scaling, but stateful applications becomes harder to deal with. Docker support for storage is not as developed as the rest. There are plugins but most of docker's container plugins are not maintained anymore.
- Docker is also good to isolate services from others, but we don't plan to deploy other services than this one.
- We would need an environment where we would install docker when we could have used this environment to deploy the application directly.
- Self managed docker are pretty expensive for the needs we have.

In opposition,

- Most cloud providers provide snapshots solutions
- The application is very simple to deploy with a bootstrap script.

Also, we already have an hosting solution available and the choice was made according to possibilities we had.
We choose to use vagrant for developpement. It provides a way to develop quickly on any plateform in a similar way that the service will be deployed in production