Requirements

Clara Gros
Babacar Toure
Yoann Roussel

November 2019

Contents

1 Identification of needs

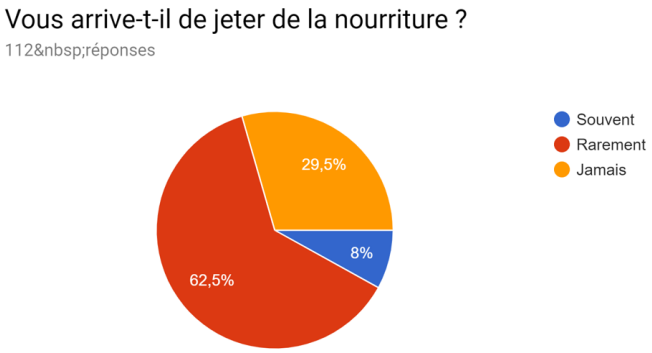
1.1 Description of the app

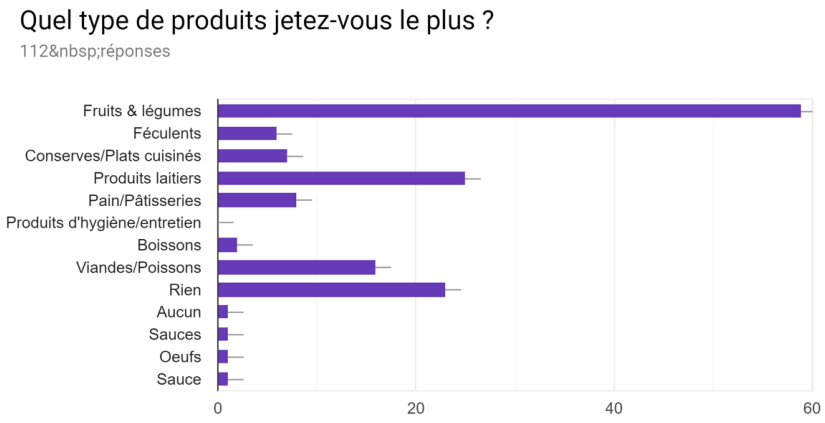
Our goal is to develop an app that allows users who are residents on a university campus to be able to give away their excess food and collect others' for free. This project was inspired by the other similar already existing app called Too Good To Go: aware of the extent of food wastage among university students, we decided to develop an app that would allow for redistribution of unused food on the scale of a campus.

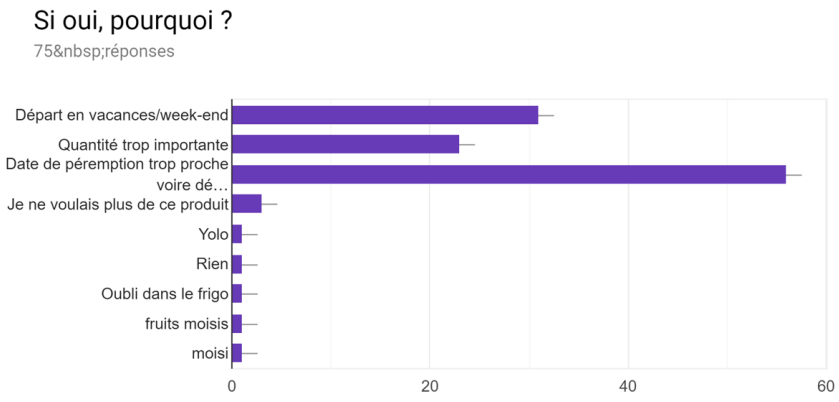
1.2 Survey

We first wanted to estimate how receptive university students would be: for that we carried out a survey and successfully reached out to over a hundred students from several campuses: the goal of this survey was to evaluate and refine the need for such an application, as well as quantify the potential user base we could expect to have.

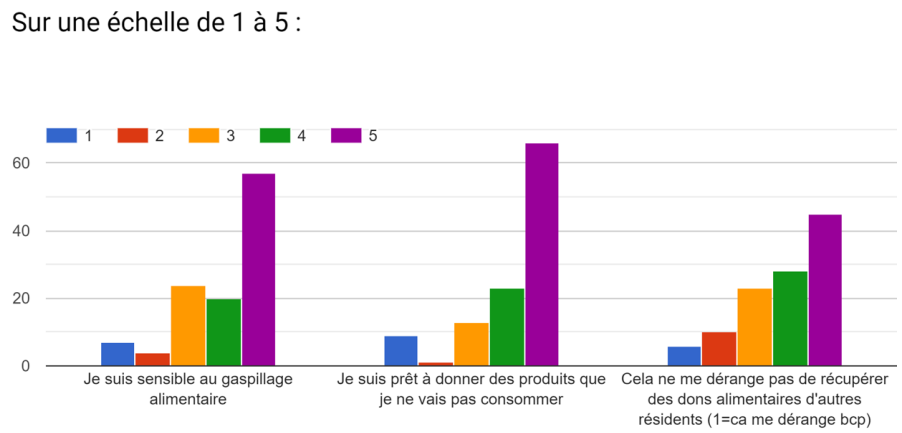
- Need
  One of the objectives of this survey was to evaluate how much food is wasted and the reasons for that wastage: what we found out reported in the charts below is that over two thirds of interrogated students said to waste food products, the main ones being fruits, vegetables and dairy products: the reason cited by three quarters of respondents was products were closing in on or had already passed their expiration date, with another major cause being leaving campus on holidays.





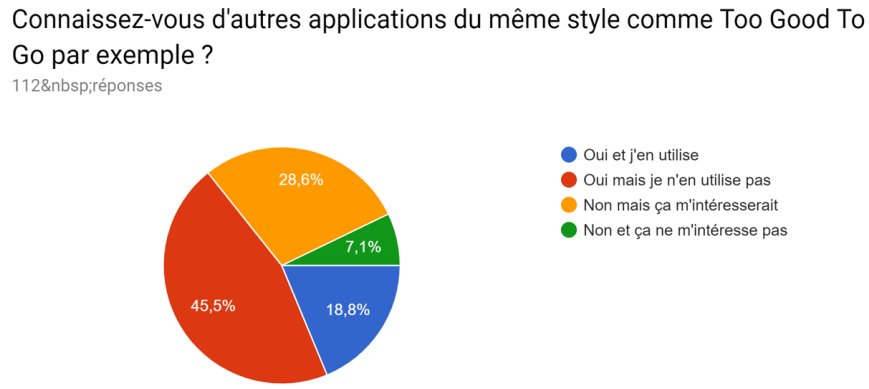


- Limitations
  We were also curious of the possible limitations and drawbacks this project would have, and according to respondents one limitation is users' willingness to accept food products from other people, which as seen in the chart below is less on average than their willingness to give away unwanted products, although not by as much as we would have thought.



Some also explicitely mentioned the safety risk of food coming from an unverified source:
"Ce qui me dérangerait, c'est de ne pas savoir comment a été stockée la nourriture et de ne pas savoir par exemple si la chaîne du froid a bien été respectée"

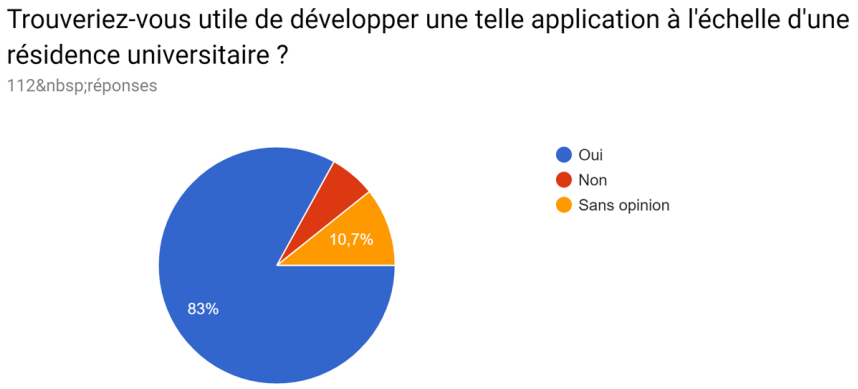
- Already existing apps
  There already exist similar applications that do not however target a geographically localized customer base like we plan to do. We asked the students if they knew of one and in that case if they actively used one, or if they didn't know of one and in that case were interested in using one. We were pleasantly surprised do discover the amount of respondents (over a quarter) who didn't know of any such app but were willing to start using one:



From this chart we can see that less than a third of people who knew of

such apps use one, but close to 80% of respondents who didn't were interested in using one, from which we can deduce currently available apps do not fulfill the demand criteria well enough and/or do not appeal enough, an issue we hope to overcome with our app.

- Support
  We have received very positive reviews on our project, and many verbally expressed their support:
  "Une excellente initiative !"
  "Très bonne idée d'appli !"



2 Perimeter

2.1 Objective

Our main objective is to code an application along with a webapp for admin and a database. The application is expected to allow users to create an account and specify their personal information such as name and room number, and be able to log in and out: they then have access publications corresponding to food product offers and are able to accept them and create their own: they should then be put in contact with whichever user published the offer so that they can come to an agreement over the transfer of the product. Users may withdraw their own offers at any time and have the option to report offers and messages that do not fit Terms of Services (see section Safety constraints below).

2.2 Resources and constraints

- Time constraint:
  We are limited on time as the project is expected to only last until the end of the school year: further to that, we have a mid project presentation in January which correspond to the early departure of one team member.

- Financial constraint:
  We have no budget as in the currently expected state of the project by the end of the year we have no planned expanses (see section development).

- Human constraint:
  We have little experience in application development, which is why the beginning of the project is reserved for learning using provided tutorials before actually starting the programming phase.

2.3 Already existing solutions

As mentioned earlier other applications exist that fulfill a similar role, the most widely used one being Too Good To Go. However this application functions differently, as it targets grocery stores willing to sell their excess food at a lower price to individuals, thus having to separate customer bases. We found others like HopHopFood that offer a product similar to ours, but they are deployed on a much larger scale with no real geographical limitation which makes it unpractical. We have decided to instead aim at a more restricted but potentially more loyal user base.

2.4 Safety constraints

As we plan on maintaining databases that can contain personal information regarding users, it is necessary to tackle information security. As our application will offer open communication among users it is also necessary to impose adequate Terms of Services as well as allow active moderation of publications.

3 Technical environment

In this section, based on our perimeter and on the state of the art we wrote before, we are going to choose the best suited technologies to build our app. We will use two different technologies on the admin side and on the client side.

3.1 Client side

The customer will have a mobile application and must be connected to the Internet to be able to make donations or collect food products. Given our time constraints, it will be a native mobile application because we don't have time to learn a new language neither to develop a hybrid application, nor to develop separately on Android and iOS. The choice therefore remains to be made between these two operating systems.

With a combined market share of Android and iOS of about 99%, these systems have become unstoppable. However, Android can count on a market share four to five times higher than that of Apple. In Germany, for example, Android's market share is 81.5%, compared to 17% for Apple. The situation is similar on the Spanish and French markets.

So we choose Android not only because the underlying programming language is Java (which we already know a little bit about) but also because on the scale of a university residence, we would like to reach as many people as possible and therefore develop on the most widespread OS.

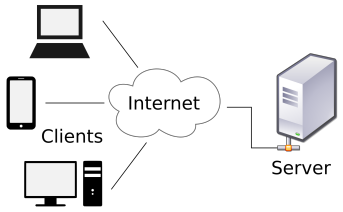
3.2 Admin side

The role of the administrator would be more of a moderator role with regard to both users and food products circulating on the application (functionality for reporting inappropriate content). That's why we immediately thought of Django, a framework delivered by Python that almost instantly created an administrator interface to keep control over the database with the basic CRUD operations. Thus we would provide the administrator with a very simplistic web app to moderate the mobile app.

4 Technical environment

4.1 Client-server model

Client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs, which share their resources with clients. A client does not share any of its resources, but it requests content or service from a server. Clients therefore initiate communication sessions with servers, which await incoming requests. Servers are classified by the services they provide. For example, a web server serves web pages and a file server serves computer files. A shared resource may be any of the server computer's software and electronic components, from programs and data to processors and storage devices. The sharing of resources of a server constitutes a service.



For this project we will use an undetermined server (see with the Viarézo association). In the development phase we will use our computer as a server. A database, connected to this server will be indirectly linked to the mobile client application via the rest framework **Django**. We chose Django because it is a Python framework that we already partially master and it is very easy to use thanks to the tutorials provided by the official documentation. In addition, it

will provide an API to communicate with the db without writing SQL queries but with a model system. Finally, the big advantage of Django is the almost instant creation of an administrator interface which will save us a lot of time.

4.2 Database

4.2.1 Server-side

For the choice of the server-side database, we chose a relational database. Indeed, SQL Databases define and manipulate data based on structured query language (SQL). These are most popularly used and useful for handling structured data that organizes elements of data and standardizes how they relate to one another and to different properties. This is exactly what we need because we will have several tables users / products / scan) and we will need to establish links between them all the time.

We have chosen to use the MySQL database manager. First of all, economic. We conduct a project using OpenSource products or products under free license from manufacturers (such as MicroStrategy Reporting Suite). Certainly, if we were in a large project, choose a proprietary database option like Oracle (which is for many the best choice for study, practice or consensus). In addition, MySQL is quite intuitive and easy to use, even if these features are limited enough, it will be more than enough to meet the needs of our application.

There are dozens of RDBMS, each with its own advantages and disadvantages. I briefly present here four of them, among the best known. I apologize immediately to the fans (and even simple users) of the many RDBMS that I failed to present.

- **Oracle Database**, published by Oracle Corporation (which, I remind you, also publishes MySQL) is a paid RDBMS. Its high cost means that it is mainly used by companies. Oracle manages large volumes of data very well. It is not necessary to buy an oracle license for a small project, as the performance will not be much different from that of MySQL or another RDBMS. On the other hand, for large projects (several hundred GB of data), Oracle will be much more efficient. In addition, Oracle has a very powerful procedural language (at least more powerful than the procedural language of MySQL): PL/SQL.

- **PostgreSQL** Like MySQL, PostgreSQL is an open source software. However, it is less used, especially by beginners, because it is less known. The reason for this lack of knowledge is probably partly due to the fact that PostgreSQL has long been available only under Unix. The first Windows version only appeared when version 8.0 of the software was released in 2005. PostgreSQL has long been more efficient than MySQL, but these differences tend to decrease. MySQL now seems to be equivalent to PostgreSQL in terms of performance, except for a few operations such as data insertion and index creation. The procedural language used by PostgreSQL is called PL/pgSQL.

4.2.2 Client-side

On the client-side, among the large choice of SQL databases, we chose to use SQLite mainly it is a small, highly reliable, embedded, fully-featured, compact and self-contained relational DBMS available as a public domain software package. As compared to other database management systems, SQLite is not a client-server database engine. It is regarded as a popular database, especially for application software like Web browsers and even for mobile app developers to store data from frontend mobile apps. We will use it to store local information such as application settings chosen by the user. SQLite stores data to a text file on a device. Android comes in with built in SQLite database implementation.

4.3 Graphic design

The graphic charter will be provided by us. Once it has been designed and written, it will be submitted to a panel of users to judge its relevance. It will group and codify:

- The logo that must be able to be adapted and declined on all the company's communication supports. It will be found on letterhead, quotes and invoices, posters, sales brochures, flyers, website and company vehicles, for example.
- The fonts and typographical attributes to be used, generally a title font and a content font.
- The choice of colours through colour schemes adapted to the requirements of the different communication media and the different backgrounds available (coloured or white background). It must include the Pantone, CMYK, RGB and hexadecimal values of each colour used.
- The rules for inserting each element (logo, title, baseline,...) and for each medium: margins and positioning in the document. Editorial rules (tone and style) to help any writer to make his documents consistent.

5 Functional requirements

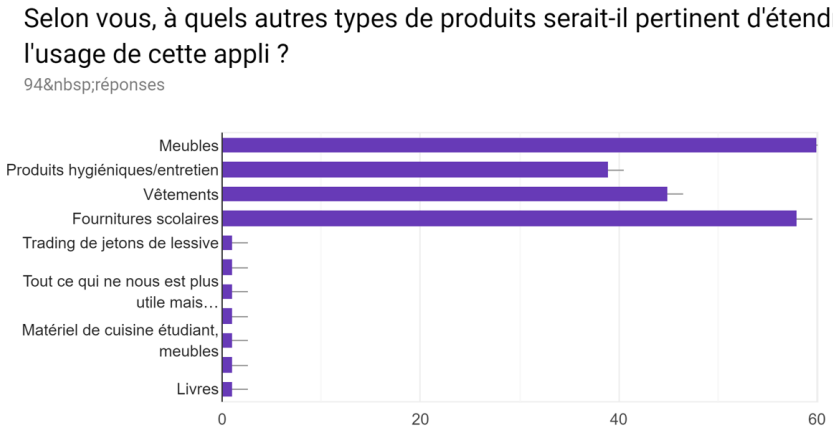
See document wireframes_v1

6 Development prospects

We are limited by time constraints so for the moment the scope is well defined and is limited to an Android application in French for students at the Gif residence (CESAL) and limited to food waste only. We can therefore easily see areas for improvement that have also been suggested to us by our panel of potential customers.

From a technological point of view, we could develop this application on iOS to reach more people. We could also develop the app in several languages such as Portuguese and English (many foreigners in the residence).

Regarding our concept, we could consider developing it on other campuses and extending the concept of "anti-food wastage" to other products as indicated in this response to our survey.



The improvements made to our project will be defined later with regard to the progress of our project compared to the planned schedule.

7 Timeframe, planning and organization of the team

see results in doc planning