

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título DEVELOPMENT OF A PLATFORM TO CONNECT AND COMMUNICATE FOOD DONORS AND FOOD BANKS en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2019/2020 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Carlos Ripoll Ramzi

Fecha: 26/08/2020

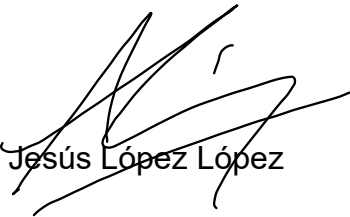


Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Álvaro Jesús López López

Fecha: 26/08/2020



# DEVELOPMENT OF A PLATFORM TO CONNECT AND COMMUNICATE FOOD DONORS AND FOOD BANKS

Author: Carlos Ripoll Ramzi: 201409243@alu.comillas.edu

Supervisor: Alvaro Jesús López López

Collaborating entity: ICAI - Comillas Pontifical University

**Resumen— El proyecto pretende agilizar la comunicación entre donantes de alimentos y entidades distribuidoras de alimentos. Consiste en la creación de dos aplicaciones en la nube y una plataforma de BackOffice, empleando la infraestructura Microsoft Power Apps.**

**Abstract-- The project aims to streamline communication between food donors and food distribution entities. It consists of the creation of two applications in the cloud and a BackOffice platform, using the Microsoft Power Apps infrastructure.**

Keywords: **Food bank, Microsoft Power Platform, Microsoft Power Apps**

## I. INTRODUCTION

Currently, more and more companies are aware and predisposed to help those most in need. The Spanish Federation of Food Banks (FESBAL), which consists of 54 Spanish associated food banks, distributes food to more than 7000 charities, helping to combat hunger for more than one million beneficiaries. [1]

To throw some numbers in, only in the Community of Madrid, almost 200 companies donate food regularly to the 553 associations linked to the Madrid Food Bank [2]. The donating procedure involves phone calling every single association that could be interested in the food. This method has several drawbacks:

- It is a slow procedure that requires a great effort on the part of the donor.
- It takes a long time to make calls.
- Not all receiving entities are called: only those that have had contact with the company, so new entities may not be called.
- Any change or update to a donation or request must be communicated to all interested parties independently.

As the reader might predict, it would not be surprising to believe that some companies might be refusing to follow this procedure to donate their products. This means that some food that could feed families in risk might be being thrown away.

The objective of this Project is to develop a platform which allows donation management and provides fast and reliable communication food-donating companies and the receiving and distributing entities.

## II. STATE OF THE ART

The project will consist in three modules:

1. Database (DB) + DB Management System (DBMS): The database will store the information, and the DBMS will grant access to the DB and ensure its cohesion and integrity.
2. Online Application: This application will be the user interface. It will be in charge of communicating with the user (via forms) and show the requested data (tables, etc.).
3. Linker: This part will connect the first module to the second one, receiving instructions from the web application and transforming them into DBMS language (usually SQL) to access the information stored in the DB.

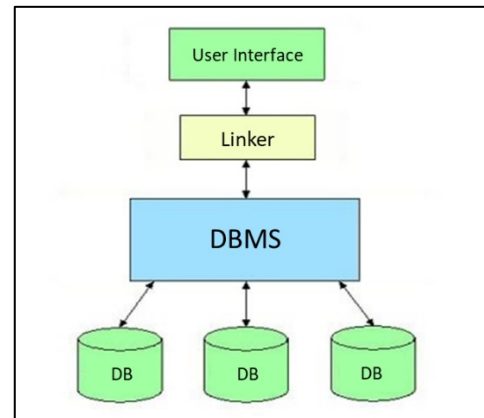


Figure 1. Physical schematics

While the two first modules are pretty standard (Oracle MySQL, MS SQL Server are examples of DB + DBMS widely used, while Nintex or Hyperbase provide the creation of Business Management Apps based on views and forms), the linker between them is not that easy to implement. Before Cloud Computing became popular (and SaaS, PaaS solutions started to show up), companies would program their own linker in a low-level programming language such as C# or C++. However, nowadays integral solutions are more common to see, due to their extended functionality and their ease to use.

Integral Solutions consist of PaaS and SaaS solutions. Both include DB + DBMS and the linker to their web applications. The main difference between them in this context is that, while SaaS solutions include pre-made applications for the user, PaaS only provide tools to create them (they need to be programmed).

A good example of PaaS are Oracle Cloud Platform and Salesforce CRM.

Salesforce CRM provides DB + DBMS and the creation of web applications in open-source languages such as Ruby, Java or PHP. Furthermore, it provides out-of-the-box tools which allow, for example, downloading apps made by other users, selling your own applications and data analytics. Its license price starts from \$25 [3]. As an example, Mediaset’s CRM uses SaleForce CRM. Its main advantage is its fast speeds, while its main drawback for the sake of this Project is the necessity of programming the web application, which would require the Food Bank to have a dedicated team for dealing with it.

Oracle Cloud Platform provides:

- Information managing, deep learning and AI tools.
- Online Desktop and Smartphone app developing: These apps are Java-based. Oracle’s app designer helps reduce the needed coding.
- Third-party cloud services integration.
- Business Analytics
- Security layer: Authentication, security applications.

Its main drawback is its cost: In order to use this service, the client needs to own a Oracle Server, starting its price in \$2000. This makes it unsuitable for small projects.

The most relevant SaaS infrastructure is Microsoft Power Apps. MSPA is the improved version of MS Access and belongs to the group of Microsoft cloud applications commercially named Microsoft Power Platform:

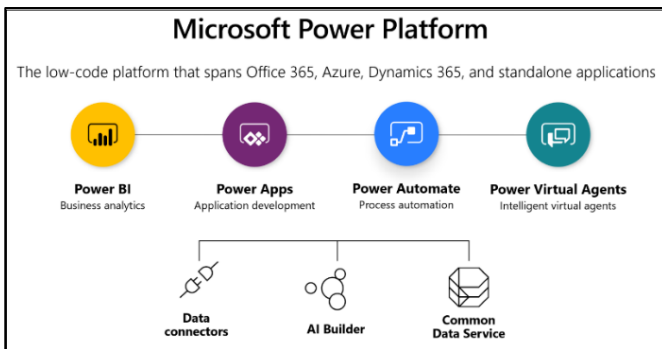


Figure 2. MS Power Platform components

MS Power Apps consists of a Cloud Platform and a list of tools, connectors and services which allow the client to generate online apps without a single line of code. [4] These apps can run in either local or online DB (Excel, Sharepoint, etc.) or on top of a MS SQL Server DB, and Microsoft’s Common Data Service will provide extended features to the DBMS such as a security layer.

Power Apps can be:

- **"Canvas"** applications: These are those that, as their name suggests, start with a blank canvas, on which the developer includes modules, components, etc., to design the user interface. In addition, it allows the redistribution and change of shape and orientation of the elements. Once the application is designed, it must be communicated with all the data sources by means of not very complex (high-level) formulas. They put

the full burden of customization on the designer, who will have to make all relevant changes to the application.

- **"Portal"** applications: They allow creating responsive websites that can be shared with users outside the organization
- **Model-driven applications:** Specially designed for working with relational databases, whether stored on the Microsoft platform (Common Data Service, CDS) or in a local or online relational database ( Excel, Sharepoint, SQL Server, etc.). The display screens are predefined to fit the data model (hence the name), and while the user interface is not as customizable as in canvas apps, the user can quickly make changes to the app by altering the your "application map". In a matter of minutes, the application can be updated without the need for technical knowledge.

MS Apps license costs 8.40€ per month (the cheapest solution so far) and has the advantage of belonging to a reliable company such as Microsoft.

The tool election will be performed based on four aspects:

- Usable, manageable, and updateable by a nonskilled user: Any kind of coding needed (link programming, app programing) is undesired.
- Price: Scalable prices will be desired over fixed-price solutions, especially for the Project. Monthly subscription fees are preferred over single payments.
- Ease to use by a nonskilled user (food bank volunteers).
- Suitability to implement BI tools (for further developments).

The performance of the tools can be seen in the following table:

Tool	Cost	Integral	BI	Non skill
MS Power Apps	Monthly (8.40€)	Yes	Yes, but charged	Yes. Codeless
Salesforce Platform	Monthly (\$25)	Yes	Yes, in some subscriptions	No
Oracle CP	Fixed (\$2000)	Connects to Outlook	BI included in some subscriptions	Yes
Self-made solution	N/A	No cloud No email	Would need to be programmed	No

The tool which will be used for this project is MS Power Apps because of two key factors:

1. Oracle Cloud Platform is too expensive to use and requires coding
2. Salesforce requires coding knowledge and therefore does not meet the requirement of being updated by a non-skilled user.

### III. PROJECT DEFINITION

The Project consists of the creation of two applications (accessible both from a web browser and a mobile application) that allow food donor companies to communicate with food banks, so that the former can publish donations on the server to which the latter can respond with requests for food. All this will be carried out using the Microsoft Power Platform.

### IV. PROJECT OBJECTIVES

The project will be considered accomplished if the following requisites are met:

1. Both donors and beneficiaries can log in to their user accounts and fill in their personal information.
2. Receivers are able to subscribe to donors, so that they receive a notification whenever the donor publishes a new donation.
3. Donors can create new donations, specifying both the product being given out and the total quantity offered.
4. Receivers can submit donation requests linked to those donations, specifying the amount of food they would like to be given.
5. Donors can assign the offered resources among all the active requests, closing the donation whenever they are done. Receivers will then receive a notification which will inform of the final status of their request: accepted or denied.
6. Both the model and the apps can be exported from the Testing Tenant and stores, and re-installed into the Operations Tenant

### V. MODEL DESCRIPTION

Simplicity has been the key factor when developing the model, in order to reduce the possibility of malfunctioning.

The assumption of each donation consisting of only one product has been made, so the product name will be a text field belonging to the donation, rather than the product being a whole entity. This way the total number of entities can be reduced to 4: Donors, Receivers, donations, and requests. This

is the simplest yet most efficient model that has been thought of to meet the requirements of the Project.

The Entity-Relationship diagram of the proposed model is as follows:

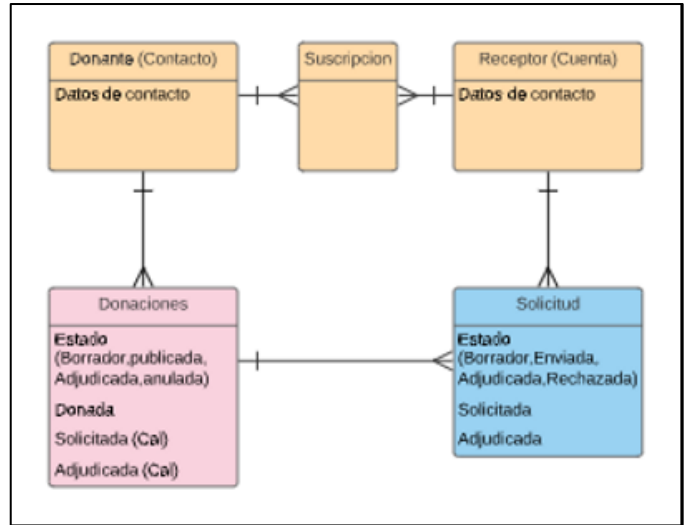


Illustration 3. Entity-Relationship model diagram

As can be seen, the relationship of donors with subscribers is allowed through the M: N Subscription relationship. Donors will have a 1: N relationship with donations, and receivers will be related to requests in the same way. In addition, there will be a 1: N relationship between donations and requests, since several requests may be submitted to a donation, which are only directed to one donation.

Donations may have any of the following states:

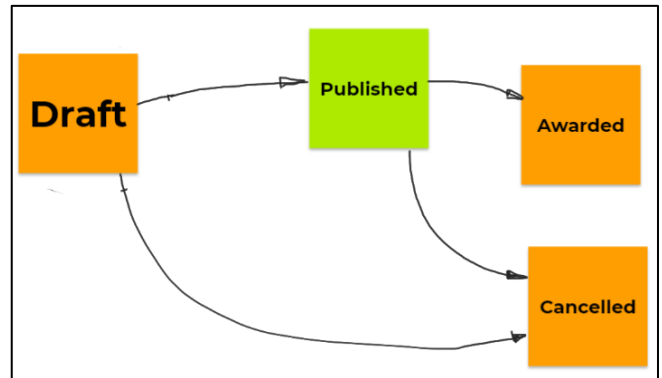


Illustration 4. State machine diagram for donations.

### VI. IMPLEMENTATION

To develop the Project, these last Model-driven apps have been chosen. These apps consist roughly of three parts:

1. Views of one or more entities: The values of the desired fields of each record (instance) of an entity are shown in table form. These views will allow you to sort the records based on any of their fields and perform searches within the table. It is the equivalent of executing a “Select [...] from [...]” command. Below is an example of the view used in the project, called “Active Donations”:

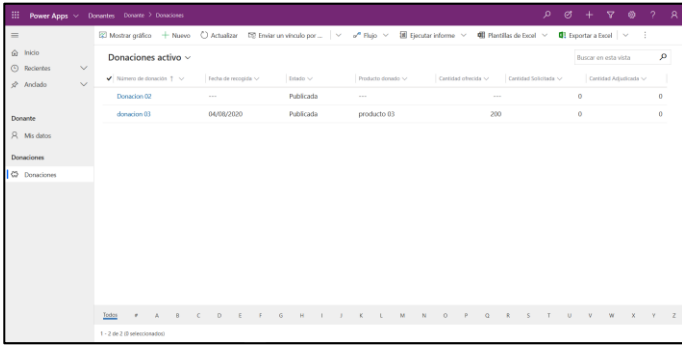


Illustration 5. MPA view example.

- Entity forms: They allow the creation of a record or the updating of the values of its fields (attributes). Every form must contain at least all the fields “required by the company” (that is, those fields that cannot have a null value). An advantage of using Power Apps in the project is the possibility of presenting different forms to users depending on their security role, and in this way creating forms that restrict access to certain fields, defining them as read-only (for example: In the project, the donors, in their Request entity form (property of the receiver) can only modify the field "awarded amount", and the other fields (requested amount, application number ...) are not modifiable. This can be seen in the following illustration:

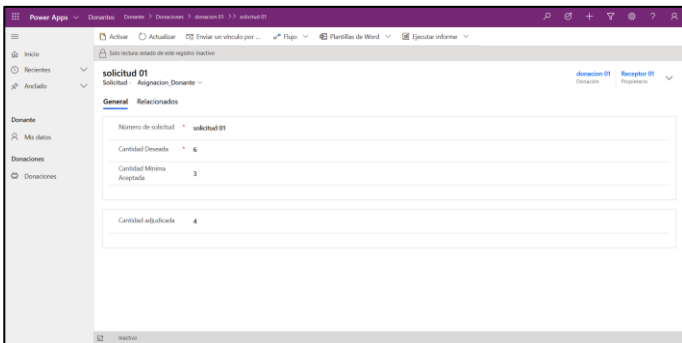


Illustration 6. MPA form example.

- Entity workflows: Workflows are customizable routines performed by the application itself. These routines can run in the background and perform actions such as updating values of the fields of a record, sending e-mails, or executing other workflows belonging to records that are directly related (that is, that are “one jump” in the ERD) with the record in question. This last functionality will be used to create “domino effects” and thus be able to execute entity workflows at more than one jump. This will be explained later. The following illustration shows an example workflow, with the actions that you will perform sequentially:

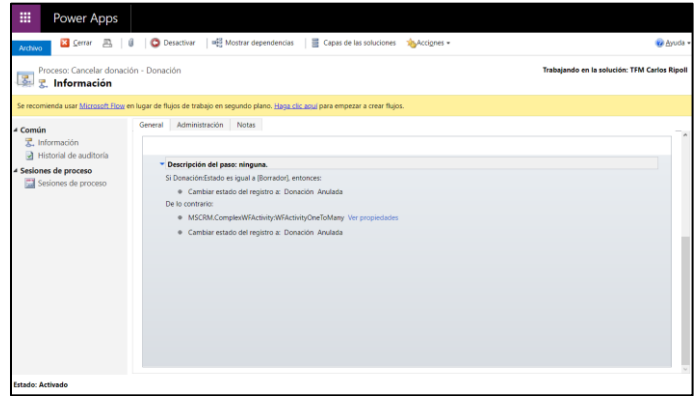


Illustration 7. MPA workflow example.

VII. RESULTS

The results are these two model-driven power apps:

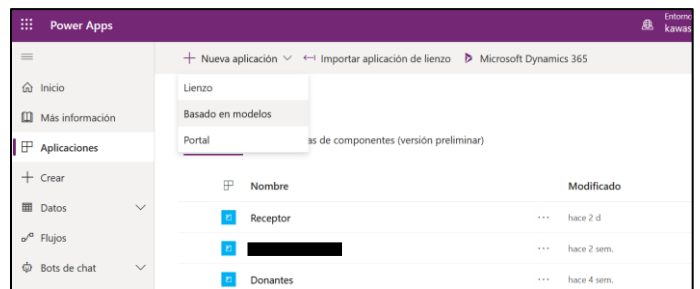


Illustration 8. Final Power Apps.

These two apps and the BackOffice software both work as intended and meet all of the requirements. The solution in which everything is contained can be exported from the testing Environment (sandbox) into the Use Environment (the one from the food bank). Both apps have been successfully tested. The following section of the paper summarizes a use case, done from both a PC browser (Donor) and a Smartphone with Dynamics 365 App.

The test performed to the applications have been:

Functionality	Test
User sign-up	Login. User information providing
Subscriptions	Nearby donor searching. Subscription creation, management, and access
Donations	Donation creation (demo)
Requests	Request creation (demo)

<p><b>Email notifications</b></p>	<p>Email-related workflows run</p>
<p><b>Donation awarding and closure</b></p>	<p>Closure of a donation with both accepted and rejected requests</p>

will be accepted. Every receiver will receive an email informing about their request status.

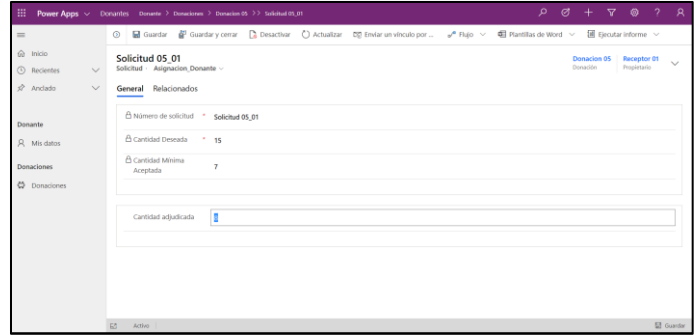


Figure 6. Resource assignment

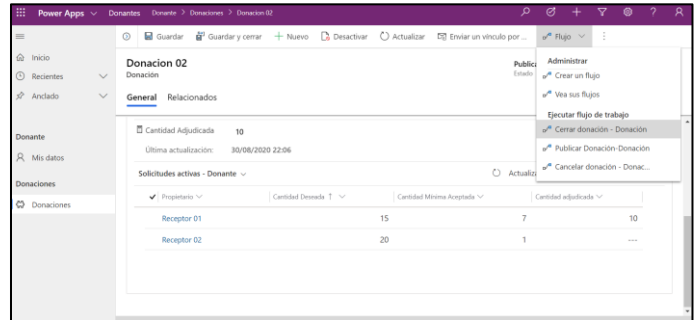


Figure 7. Donation Closure

The results of these tests have been successful. The following sub sections show the key parts:

**A. Donor app**

The donor app allows donors to view their active donations and to create one:

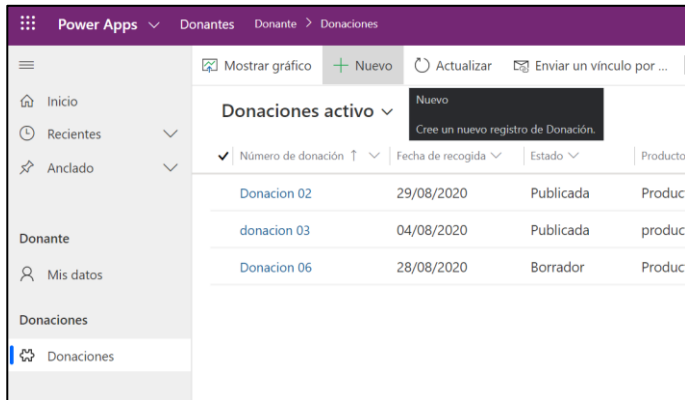


Figure 3. Donations view (donor app).

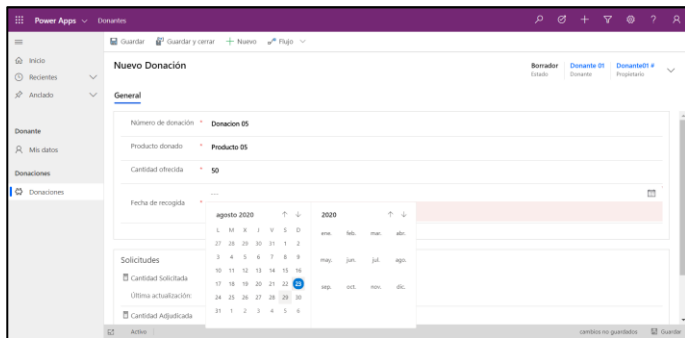


Figure 4. Donation creation form (donor app)

Once a donation is created, it can be published or cancelled:

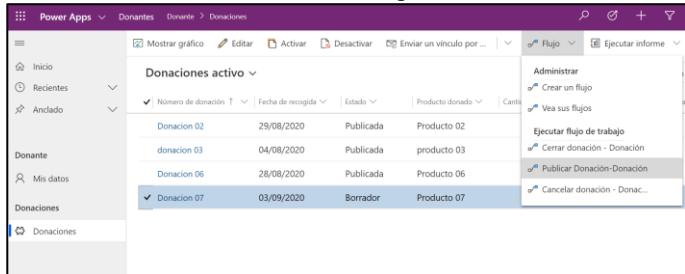


Figure 5. Donation cancellation or publishment

After a donation is published, receivers can request the products that are offered. Donors can enter those requests and assign them resources. Once they are done with all the requests, they can close the donation. All the donations with zero or no assigned resources will be cancelled, and the rest

**B. Receiver App**

Receiver app allows searching for active donors and subscribing to them:

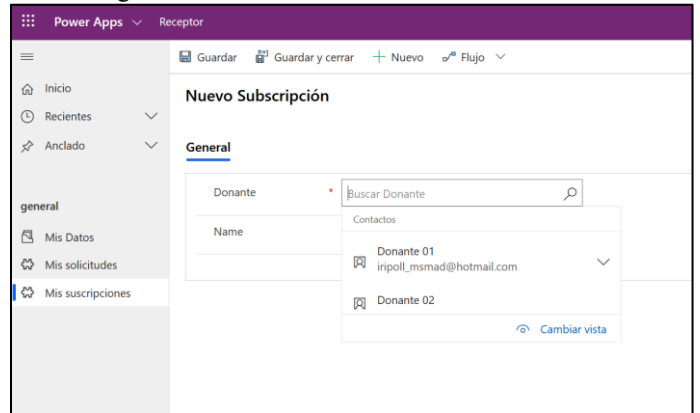


Figure 8. Subscription creation

Once the subscription has been made, the recipient will receive notifications when the donor posts a donation.

Once a donor publishes a new donation, any recipient can create a request to that donation:



Figure 9. Request creation

Figure 12. Automatic email creation

Recipients need to include the quantity they desire to receive and the minimum quantity they are willing to accept.

Figure 10. Request form (recipient app)

Any Request can be deleted whenever by selecting it from the requests menu and hitting the 'delete' button.

Figure 11. Request deletion.

### C. Email generation

E-mails are automatically generated from the request entity, so that one email is sent to the owner of each request. The structure of these emails has been edited so that they generate automatically with the relevant information:

As it can (hardly) be seen, yellow-shaded text corresponds to any entity field that is going to be looked up (owner of the register, awarded resources, etc.).

### D. Smartphone app layout

The Smartphone app is generated automatically at the same time as the desktop one when the data model is finished. Therefore, both of them offer the same functionality. However, the smartphone app offers a different layout: Forms and views are presented in a different format so that user experience and usability are maximized. The following illustration shows the donation form from the Recipient's smartphone app:

Illustration 11. Request creation form (smartphone)

Smartphone apps hide all the menus that are visible in the desktop app in order to save space. On the other hand, it offers a drop-down tree-like menu that is very intuitive to

use. The following illustration shows the donations drop-down menu from the donor's app:

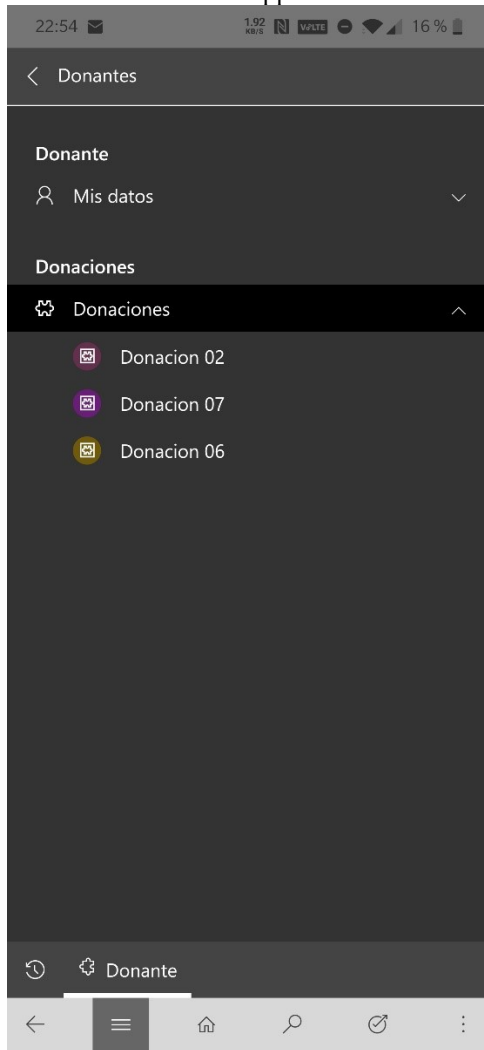


Figure 13. Quick drop-down view of donations

### VIII. PROJECT COST AND VIABILITY STUDY

In order to carry out this Project, the basic **requirements** are:

1. A computer with Internet access
2. (Optional) A smartphone with internet access and with the “Dynamics 365” app installed (available at the regular app store).
3. A trial Office 365 environment (Tenant), where the solution will be nested. Once the Project is finished, it will be available for exporting and installation into the definitive environment (the Food Bank environment).
4. Five accounts with Power Apps Licensing. One of them will be the used to develop the Project (administrator role) while the other four will be used for testing (two donators plus two receivers).

The Project's total cost will be the cost of the five accounts plus the Tenant renting. The cost of a monthly subscription to Microsoft Power Apps is 8,40€ as of August 25, 2020 [2]. The Tenant rental price is unknown.

Timewise, the project started in June although conversations with Food Bank volunteers were held in early March. The

week of completion was August 17-23th, after ~320 hours of work (there was a lot of learning involved).

The **viability study** is a bit tricky. Since FESBAL is a nonprofit organization, a project does not pay off if it generates greater revenue in form of money. Instead, the viability study needs to find out if this is the optimal way that money can be invested to help feed people at risk of poverty, or if there are better ways to spend that money (for example, buying food with that money).

The viability study will only consider the Community of Madrid and the Community of Madrid Food Bank. Starting data are:

- The average person eats from 700g to 1200g of food daily depending on their gender and condition. [5]
- The average cost to feed a person for the food bank is 68.83€ [5]
- The Community of Madrid's Food Bank receives 321500kg/month of food donations via their 553 associated entities [2]
- It is assumed that the license cost will be 8.40€ (worst case scenario). All 553 associations will purchase a license, and 100 donor licenses are also purchased.

From the first two bullet points, it is assumed that the average person eats 30kg of food per month (one kilogram per day). Dividing the average monthly cost by the average monthly food, the marginal cost of a kilogram of food is 2.30€/kg. Therefore, if the cost of the project provides more than 2.30€/kg, the project will be viable.

The total cost for the 653 licenses is 5485.2€ per month. With this money, 2384kg of food could be bought at the 2.30€/kg rate. An increase in 2384kg from the 321500kg monthly food that is donated counts for a 0.74% increase.

Therefore, we can conclude that the Project is viable since a 0.74% increase (at least) in donations seems reasonable to expect.

### IX. CONCLUSIONS

The conclusions of this Project are positive since:

- **Functionality-wise**, the two apps and the backoffice software meet all the functionality. What is more, the apps are easy to use and keep updated. The whole solution can be exported and stored, for further implementations.
- **Viability-wise**, the required donations growth (0.74%) to make the project profitable looks very realistic to achieve.
- **Ethic-wise**, this project aligns itself with six of the **Sustainable Development Goals**:
  - 1: End of poverty
  - 2: Zero hunger
  - 3: Health and well-being.
  - 10: Reduced inequalities.
  - 12: Responsible production and consumption.
  - 13: Climate action.



## X. REFERENCES

- [1] FESBAL, «¿Quiénes somos?,» [En línea]. Available: <https://www.fesbal.org.es>. [Último acceso: 28 08 2020].
- [2] M. F. Bank, «Memoria Anual 2018,» Madrid, 2018.
- [3] «Salesforce Platform,» Salesforce, [En línea]. Available: <https://developer.salesforce.com/platform#:~:text=The%20Salesforce%20Platform%20empowers%20developers,hardware%20provisioning%20or%20application%20stacks..> [Último acceso: 28 09 2020].
- [4] M. España, «¿Qué es Power Apps?,» 02 08 2020. [En línea]. Available: <https://docs.microsoft.com/es-es/powerapps/powerapps-overview#:~:text=Power%20Apps%20es%20un%20conjunto,las%20necesidades%20de%20su%20empresa..> [Último acceso: 21 08 2020].
- [5] R. C. Naseiro, *OPTIMIZACIÓN DE RECURSOS EN EL BANCO DE*, Madrid, 2020.
- [6] N. Unidas, «Objetivos de Desarrollo Sostenible,» UN, [En línea]. Available: <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/>. [Último acceso: 23 08 2020].