Universidad Pontificia Comillas - ICAI
Degree in Telecommunications Engineering

Bachelor's final project

OPTIMIZING THE RESOURCES OF AN NGO THROUGH THE DIGITALIZATION OF ITS OPERATIONAL PROCESS

Author:
Victor Ovejero de la Cerda

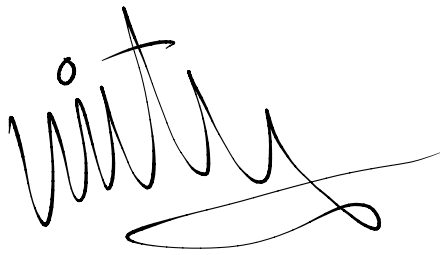
Supervised by:
Miguel Cabrera del Moral

Madrid
June 2023

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Optimización de los recursos de una ONG mediante la digitalización de su proceso operativo

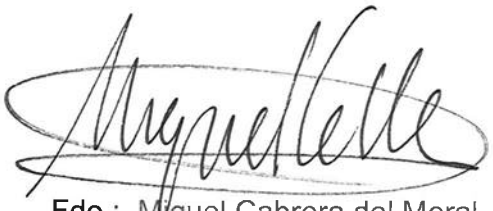
en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico **2022/2023** es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.



Fdo.: Victor Ovejero de la Cerda Fecha: 04/07/2023

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO



Fdo.: Miguel Cabrera del Moral Fecha: 04/07/2023

Universidad Pontificia Comillas - ICAI
Degree in Telecommunications Engineering

Bachelor's final project

OPTIMIZING THE RESOURCES OF AN NGO THROUGH THE DIGITALIZATION OF ITS OPERATIONAL PROCESS

Author:
Victor Ovejero de la Cerda

Supervised by:
Miguel Cabrera del Moral

Madrid
June 2023

Contents

List of Figures

List of Tables

Code Listings

Chapter 1

Project Abstract

1.1 Introduction

This project entails the design, development, deployment and implementation of a database, web application and data visualization system for the internal use of Refood España, an NGO in Madrid, Spain. Refood España is responsible for collecting and distributing spare food from restaurants and organizations within the food supply chain, later donating it to people in socio-economic vulnerable situations.

Due to the growing demand for food waste prevention and the proposal of a bill against food waste in Spain, Refood needs to digitalize its operational process. The objectives include maintaining a traceable donation system and increasing the robustness, simplicity and efficiency in its daily operational process, while maintaining data veracity and integrity. Additionally, a data visualization system is implemented for the efficient generation of dashboards and reports for collaborating entities and advertising purposes.

1.2 Project Structure

The project is structured in five parts:

- Phase 1: Optimizing the operational process of Refood.
- Phase 2: Designing a database and web application.
- Phase 3: Developing the web application.
- Phase 4: Deploying and implementing the web application in Refood.
- Phase 5: Designing and implementing a data visualization system.

The structure of the project is categorized by the order in which the parts are developed. First of all, digitalizing the operational process of Refood requires its redesign and optimization, enhancing the efficiency and reducing bottlenecks. Phase 1 is focused on the optimization of the operations of Refood, designed through the analysis of bottlenecks, time constraints and uncontrolled environments and variables that can lead to errors or accidents.

Second of all, phase 2 consists in the design of a relational database schema and a web application. The design of a new database provides the freedom to design the web application and functionalities optimally, ensuring the necessary entities, relationships and controlling redundancies for a robust solution. Additionally, the design of the web application calls for a User Interface, flow of views and functionality design, visualizing the web application before it is developed. Therefore, reducing the decision making needed within phase 3, having defined guidelines and design principles beforehand. Furthermore, the design of the web application requires a technology architecture map, defining what technologies will be used, as well as the connection between them.

Within the third phase, the code for the web application is developed and tested. The back-end, front-end are developed and boilerplate code is configured for the correct functioning of the web application. Additionally, the database is implemented within a MySQL server and connected to the web application for development. The web application is based on the CRUD (Create, Read, Update, Delete) model used to make web applications highly interactive with the database. The architecture of a CRUD application is seen in Figure 1.1.

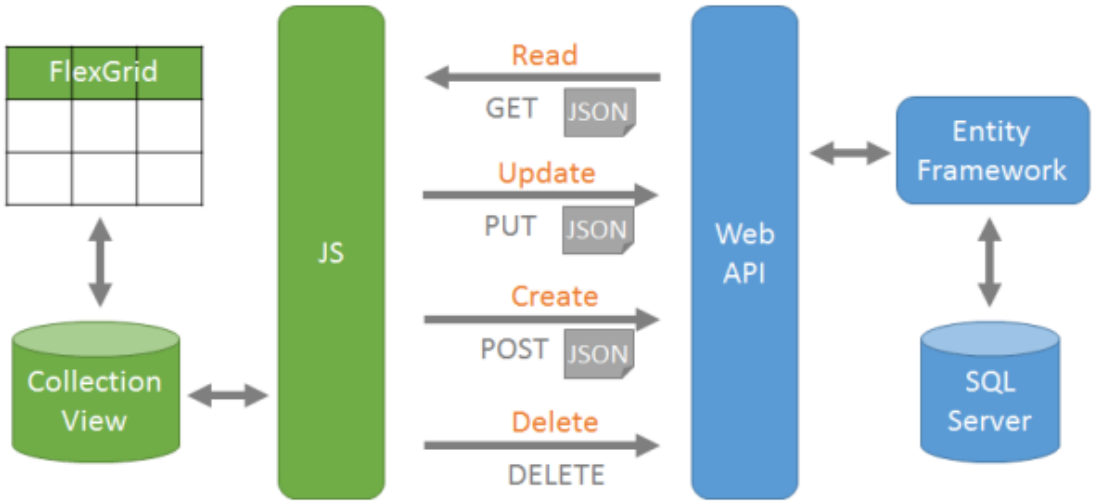


Figure 1.1: CRUD Application

Additionally, the web application is a monolithic application. It is built as one unit which encompasses the user interface, business layer and data access layer which directly

interacts with the database.

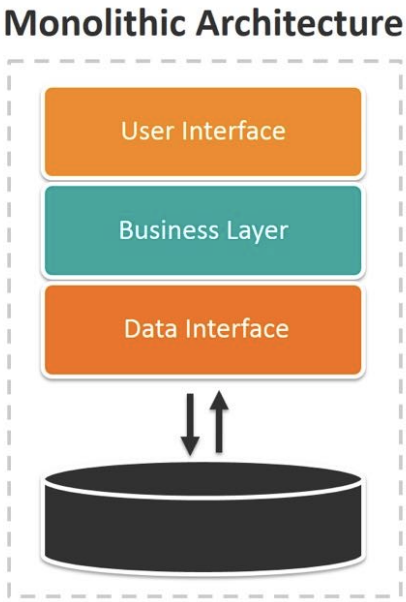


Figure 1.2: Monolithic Application Architecture

Phase 4, consists of the packaging and deployment of the web application and database in the cloud. Within the deployment, the automated scheduling of the production database is set up to optimize the allocation of resources and security of the application, only starting the database when Refood is operative. Additionally, it consists of the implementation of the web application within Refood. Creating a User Manual for volunteers and adapting the operational process to the new functionalities of the web application.

Finally, phase 5 entails the design and implementation of a reliable data visualization system for the generation of quarterly and annual reports. Collaborating entities of Refood demand an annual report of donations and data intake, aiding to visualize patterns and trends to reduce food waste behaviors within the organization. Additionally, allowing Refood and the collaborating entities to generate engaging visualizations to raise awareness about the impact of food waste, as well as the efforts being done.

1.3 Results

The objectives of the project have been achieved in their entirety. A working software to maintain the traceability of incoming and outgoing donations has been developed and implemented in Refood. For the past three months, the software has been used and no

errors or bugs have been notified. All donations can be found in the database and traced back through the user interface of the application.

The efficiency of the daily operational processes of Refood has been notably improved, reducing the time of operation by an estimated 20%. Additionally, the complexity of the data intake system has been reduced, enabling a wider range of volunteers with less technical knowledge to keep records of donations and make use of the web application. Therefore, granting Refood with an increased flexibility over work allocation for volunteers on a daily basis, not needing an experienced volunteer to keep track of traceability every day.

The solution has proven robust, since its implementation, there has been over six thousand entries recorded. The application is responsive and fast due to the front-end optimization of API response storing in the React state.

Finally, the data visualization system is being used for the design and generation of reports, ingesting data from the production data base. The report generation process takes less than one minute if the design has previously been developed, creating interactive dashboards that can be integrated within a website, the web application or directly embedded through HTML code.

1.4 Conclusions

The project was initiated due to the increasing need for the digitalization of Refood, pairing its increased difficulty to maintain an efficient operational process, to a dynamic and robust software that maintains traceability and enhances efficiency and data integrity.

The project has been successfully developed, with substantial possibilities for growth in scalability, security enhancement, functionalities and data visualization automation and integration within the software. The future of Refood and its digitalization entails being a multi-location organization with a centralized database and web application. Operating independently from each other, while using the same technologies and quality standards.

A deep insight is gained into applied web development in a professional and real-world environment, posing additional challenges to research projects. A concrete implementation plan and a testing phase are key components for successful deployments in developed organizations.

This project has had an impact on hundreds of people, beneficiaries, Refood personnel and collaborating entities, increasing the need for a fault-tolerant and robust solution. Enough resources have been provisioned for the scale of the application and the the performance of the system has been maintained throughout the implementation phase.

Chapter 2

Resumen del Proyecto

2.1 Introducción

Este proyecto consiste en el diseño, desarrollo, despliegue e implementación de una base de datos, una aplicación web y un sistema de visualización de datos para uso interno de Refood España, una ONG en Madrid, España. Refood España se encarga de recoger y distribuir comida sobrante de restaurantes y organizaciones dentro de la cadena de suministro alimentario, donándola posteriormente a personas en situación de vulnerabilidad socioeconómica.

Debido a la creciente demanda de prevención del desperdicio alimentario y a la propuesta de un proyecto de ley contra el desperdicio alimentario en España, Refood necesita digitalizar su proceso operativo. Los objetivos incluyen mantener un sistema de donación trazable y aumentar la robustez, simplicidad y eficiencia en su proceso operativo diario, manteniendo al mismo tiempo la veracidad e integridad de los datos. Además, se implementa un sistema de visualización de datos para la generación eficiente de cuadros de mando e informes para entidades colaboradoras y fines publicitarios.

2.2 Estructura del Proyecto

El proyecto se divide en cinco partes:

- Fase 1: Optimización del proceso operativo de Refood.
- Fase 2: Diseño de una base de datos y una aplicación web.
- Fase 3: Desarrollo de la aplicación web.
- Fase 4: Despliegue e implementación de la aplicación web en Refood.
- Fase 5: Diseño e implementación de un sistema de visualización de datos.

La estructura del proyecto se clasifica por el orden en que se desarrollan las partes. En primer lugar, la digitalización del proceso operativo de Refood requiere el rediseño y optimización del mismo, mejorando su eficiencia y reduciendo los cuellos de botella. la fase 1 se centra en la optimización de las operaciones de Refood, diseñada a través del análisis de los cuellos de botella, las limitaciones de tiempo y los entornos y variables no controlados que pueden dar lugar a errores o accidentes.

La fase 2 consiste en el diseño de un esquema de base de datos relacional y una aplicación web. El diseño de una nueva base de datos proporciona libertad para diseñar la aplicación web y las funcionalidades de forma óptima, garantizando las entidades y relaciones necesarias y controlando las redundancias para una solución robusta. Además, el diseño de la aplicación web requiere un diseño de interfaz de usuario, flujo de vistas y funcionalidades, visualizando la aplicación web antes de su desarrollo. Por lo tanto, se reduce la toma de decisiones necesaria dentro de la Fase 3, habiendo definido las directrices y principios de diseño de antemano. Además, el diseño de la aplicación web requiere un mapa de arquitectura tecnológica, definiendo qué tecnologías se utilizarán, así como la conexión entre ellas.

En la tercera fase se desarrolla y prueba el código de la aplicación web. Se desarrolla el back-end, el front-end y se configura el código ”boilerplate” para el correcto funcionamiento de la aplicación web. Además, se implementa la base de datos en un servidor MySQL y se conecta a la aplicación web para su desarrollo. La aplicación web se basa en el modelo CRUD (Create, Read, Update, Delete), utilizado para que las aplicaciones web sean altamente interactivas con la base de datos. La arquitectura de una aplicación CRUD se muestra en la Figure 2.1.

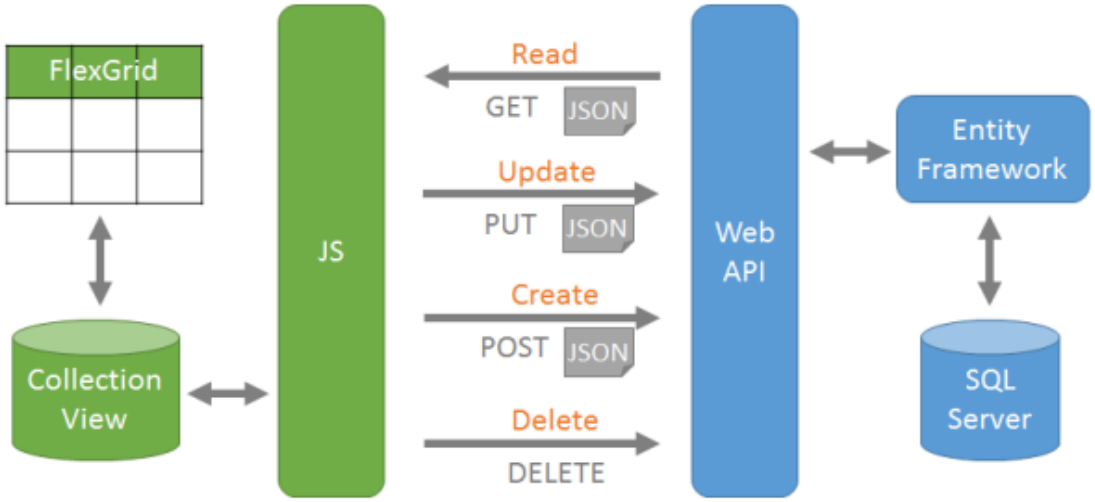


Figure 2.1: Aplicación CRUD

Además, la aplicación web es una aplicación monolítica. Se construye como una

unidad que engloba la interfaz de usuario, la capa de negocio y la capa de acceso a datos que interactúa directamente con la base de datos.

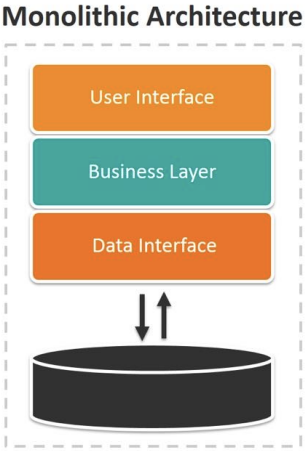


Figure 2.2: Arquitecture de Aplicación Monlítica

La fase 4, consiste en el empaquetado y despliegue de la aplicación web y la base de datos en la nube. Dentro del despliegue, se configura la programación automatizada de la base de datos de producción para optimizar la asignación de recursos y la seguridad de la aplicación, arrancando la base de datos únicamente cuando Refood está operativo. Adicionalmente, consiste en la implementación de la aplicación web dentro de Refood. Se crea un Manual de Usuario para los voluntarios y se adapta el proceso operativo a las nuevas funcionalidades de la aplicación web.

Por último, la fase 5 supone el diseño e implementación de un sistema de visualización de datos para la generación de informes trimestrales y anuales. Las entidades colaboradoras de Refood demandan un informe anual de donaciones e ingesta de datos, que ayude a visualizar patrones y tendencias para reducir los comportamientos de desperdicio alimentario dentro de la organización. Además, permite a Refood y a las entidades colaboradoras generar visualizaciones atractivas para concienciar sobre el impacto del desperdicio alimentario, así como de los esfuerzos que se están realizando.

2.3 Resultados

Los objetivos del proyecto se han alcanzado en su totalidad. Se ha desarrollado e implementado en Refood un software operativo para mantener la trazabilidad de las donaciones entrantes y salientes. Durante los últimos tres meses se ha utilizado el software y no se han notificado errores. Todas las donaciones pueden encontrarse en la base de datos y trazarlas hasta su origen a través de la interfaz de usuario de la aplicación.

La eficiencia de los procesos operativos diarios de Refood ha mejorado notablemente, reduciendo el tiempo de funcionamiento en un 20%. Además, se ha reducido la complejidad del sistema de entrada de datos, lo que permite a un mayor número de voluntarios con menos conocimientos técnicos a llevar un registro de las donaciones y hacer uso de la aplicación web. De este modo, Refood dispone de una mayor flexibilidad a la hora de asignar el trabajo diario a los voluntarios, ya que no es necesario que un voluntario experimentado controle la trazabilidad todos los días.

La solución ha demostrado su robustez: desde su implementación, se han registrado más de seis mil entradas. La aplicación es rápida gracias a la optimización en el front-end de la respuesta de la API almacenada en el estado de React, así evitando múltiples llamadas a la API en pequeños intervalos de tiempo.

Por último, el sistema de visualización de datos se está utilizando para el diseño y la generación de informes, tomando datos de la base de datos de producción. El proceso de generación de informes dura menos de un minuto si previamente se ha desarrollado el diseño, creando cuadros de mando interactivos que pueden integrarse en una página web, en la aplicación web o directamente incrustarse mediante código HTML.

2.4 Conclusiones

El proyecto se inició debido a la creciente necesidad de digitalización de Refood, emparejando su creciente dificultad para mantener un proceso operativo eficiente, a un software dinámico y robusto que mantiene la trazabilidad y mejora la eficiencia y la integridad de los datos.

El proyecto se ha desarrollado con éxito, con amplias posibilidades de crecimiento en cuanto a escalabilidad, mejora de la seguridad, funcionalidades y automatización e integración de la visualización de datos dentro del software. El futuro de Refood y su digitalización implica ser una organización con múltiples núcleos, una base de datos y una aplicación web centralizadas, operando de forma independiente entre sí, pero utilizando las mismas tecnologías y estándares de calidad.

Se adquiere una visión profunda del desarrollo web aplicado en un entorno profesional y real, lo que plantea retos adicionales a un proyectos de investigación. Un plan de implementación concreto y una fase de pruebas son componentes clave para el éxito de un despliegue en organizaciones desarrolladas, al igual que la colaboración con las diversas partes interesadas.

Este proyecto ha tenido un impacto en cientos de personas, beneficiarios, personal de Refood y entidades colaboradoras, lo que aumenta la necesidad de una solución tolerante a fallos y robusta. Se ha conseguido dotar de recursos suficientes en base a la escala de la aplicación y se ha mantenido el rendimiento del sistema durante toda la fase de implementación.

Chapter 3

Introduction

Systemic food waste is an increasing problem in the modern world. According to the United Nations Environmental Programme, food waste was estimated to be ”around 931 million tonnes or 17% of food available in world”, in 2021 [1]. According to Eurostat, ”nearly 57 (127 kg/inhabitant) are generated annually with an associated market value estimated at 130 billion euros” [2]. Additionally, The UNEP Food Waste Index Report also states that ”food waste alone generates around 8%-10% of global greenhouse gas emissions”. Inger Andersen, the Executive Director of the UNEP states that ”If food loss and waste were a country, it would be the third biggest source of greenhouse gas emissions”. The growing concern and importance of preventing food waste has led to the development of this project, aiding an NGO with its digitalization and scalability.

This project is elaborated for the collaborating entity, Refood España. Refood España is a non-governmental organization centered around solving two world-wide problems: Systemic Food Waste and World Hunger. Both problems are identified in the United Nations (UN) Sustainable Development Goals (SDGs) by 2030, as seen in Appendix A.

Refood is a growing organization, founded in Portugal in 2012, it has grown to various nuclei around Europe, forming itself in Madrid in 2018, in the area of ”Tetuán”. Presently, Refood Spain has saved a total of 25.000 rations of food, helping around 75 people every day. Refood Spain is an organization under a different management to the branch in Portugal and other parts of the world. Refood Spain is the organization for which the project is being completed, and it will be implemented in the Tetuán office in Madrid. From now on, Refood refers to the organization Refood Spain.

Up until now, Refood has saved around twenty tonnes of food from being wasted, and has donated it to families in need. However, a new bill was approved in Spain named ”Proyecto de Ley de prevención de las pérdidas y el desperdicio alimentario”, translating to a bill to prevent the loss and waste of food [3]. This bill has prompted stricter regulations regarding data security and traceability of the food-related donations. Refood has set to become a digitally driven organization, taking its daily operations from paper logs and excel, to a digital-only system, a web application. With this, Refood will

ensure higher efficiency in their daily operations, increased security, data integrity, data validation and finally, will allow them to improve their business intelligence.

3.1 Refood's Initial Operational Process

This project centers around digitalizing the operational process of Refood, the operations prior to the implementation of the project are highlighted. First of all, Refood is active for approximately four and hours a day, two hours for pick-up and two and a half for distribution. This is due to the nature of perishable goods being handled; the collection and latter distribution of food needs to be efficient and fast. To accomplish their goals, Refood works with restaurants, catering services and student residences to collect spare food on a daily basis. Then, the food is taken back to the shop, where volunteers ration it and distribute it into family-sized bundles, which are then given to the beneficiaries, all under 2 hours. The beneficiaries are families in economically disadvantaged situations.

Pick-Up

Refood volunteers initially arrive at the shop at six-thirty in the afternoon, they are provisioned with coolers, food containers and bags. Next, they travel to partnered businesses who have previously packaged their spare food for the day. Volunteers pick up all the food that is given to them, and give back the same amount of empty containers to the donor, for the next day pick-up. This way, the businesses always has containers in which to put their extra food.

Distribution

When volunteers arrive with the donated food, the distribution process starts. This is the step where the scope of the project begins. Initially, all food gets placed into separate, smaller containers of two to six rations, these containers are then separated between the beneficiaries in a way that is equitable depending on how many adults and kids are in the family. The containers are weighed individually and the information is written down on a piece of paper which will later be transcribed into an excel sheet. As the beneficiaries arrive, the food is given out and once everybody has been served and all food has been given out, one volunteer transcribes all the data from the piece of paper into the excel sheet.

The volunteers are not always the same, being on pick-up or distribution duty interchangeably due to the increasing need of available volunteers at the different times. Additionally, the volunteers are sporadic and work based on availability, forcing different teams every day, making the distribution process more complicated, and cumbersome.

Finally, after two and a half hours, Refood has distributed around one-hundred-sixty rations of food and has equitably given it to its beneficiaries. In the Figure 3.1, the timeline for the daily operations can be seen.

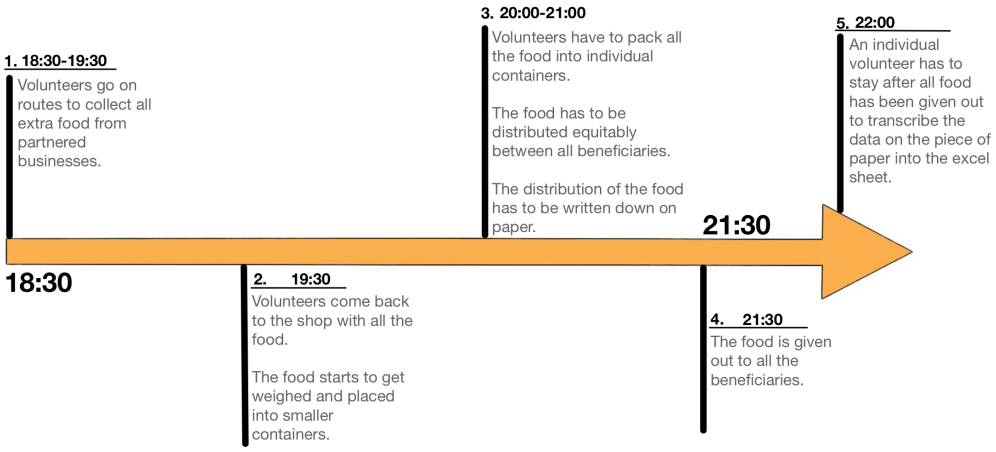


Figure 3.1: Timeline of Refood's Daily Operations

3.2 Motivation

Due to the current growth, Refood has observed a growing necessity for a system to maintain traceability of the incoming and outgoing food. Before this project, the state of the data management system consisted of a group of Excel spreadsheets in which all the data was kept, from incoming and outgoing food to the information regarding volunteers and beneficiaries. This led to various concerns regarding the security and integrity of the data.

Firstly, the excel consisted of various questionnaires that allowed for the intake of data. Nevertheless, the questionnaires lacked security, data validation and were prone to errors. Therefore, decreasing the efficiency of the operations and breaking the homogeneity of the data. For example, the type of food collected was not validated, any input would be accepted, leading to unorganized data that could not be analyzed. Moreover, the weighing standard varied, as initially the donations were measured in rations and later in kilograms. However, there was no direct translation between the two.

Secondly, security wise, the data was susceptible to a single point of failure. It was kept in an old laptop in the office of Refood, a small shop in the area of Tetuán, leading to the possibility of a stolen, broken or impaired laptop which would result in the partial or complete loss of data.

Thirdly, new European and national laws regarding food waste and organizations responsible for its management and distribution are being formalized. Because of this,

more scrutiny will be put into the veracity of the data collected by organisations such as Refood.

Additionally, potential for growth and ambition to help more people and reduce more waste will require a scalable digital database and system to make sure that all data collected is centralised in one secure database.

Finally, for Refood to operate as a single organization with different nuclei around Madrid and other cities in Spain without misinformation, malpractices and mishaps, a virtual system where all nuclei converge will allow Refood to seamlessly align them.

These factors are some of the main reasons for why Refood needs to plan, design, implement and deploy a new digital system in which data can be kept safely, and its integrity can be maintained.

Continuing with the line of reasoning, there are certain requirements and reasons for which Refood is embarking in the process of Digitalizing their current system. These reasons will be separated into four categories: data governance, efficiency, legal requirements and growth potential.

3.2.1 Data Governance

Data governance is a critical aspect of all modern, digital applications. For the use case in this project, the following definition will be followed, ”Data governance (DG) refers to the overall management of the availability, usability, integrity, and security of the data employed in an enterprise” [4].

One of the core objectives of the project is to align Refood’s data with the principles of data governance. To accomplish this, it is obligatory that the solution provides the organization with constant availability to the data, makes the data usable for all future needs, maintains its integrity and finally, is secure from breakdowns of the system or loss of physical devices.

Due to the following, a few conclusions can be reached for the design of the solution. First of all, from a security point of view, data should be kept in a safe environment, either in a private server or in the cloud, making sure that it is not possible to lose a device and therefore, lose the data. Additionally, a backup of the data needs to be kept at all times, independent from the server in which the original data is saved. Secondly, with the goal of maintaining the integrity of the data, the solution must limit the number of times in which users can enter hand-written inputs and, if it is necessary for users to do so, it should be validated before accepting it into the database [5]. Thirdly, for the data to be accessible at all times, the server or the cloud architecture should be constantly operating, or should be simple to turn on and off. Finally, for the usability of the data, a normalized relational database should be designed and implemented in a common and compatible database management system, simplifying future applications.

3.2.2 Efficiency, Improving Daily Operations

Before diving into the design of the solution, it is important to understand Refood's daily processes and needs to improve not only the efficiency but reliability of its daily operations. In this section, the daily operations of Refood will be discussed in detail, to obtain a general grasp of the problem and challenges.

Analyzing Figure 3.1, the steps one and two of the process are fairly simple and resources such as time and money cannot be optimized further. However, step three is the most time and effort consuming out of all of these steps and the most error prone as well, as pressure and time constraints for volunteers lead to mistakes in the data collection, food packaging and distribution; this is the step that needs to be optimized the most with the proposed solution.

The major efficiency-related challenges in step three come from portioning, redistributing and tracking the donations. There are various volunteers working simultaneously on portioning and noting where the food is going on a piece of paper. Other volunteers then may change the distribution of the food to make it more fair, without changing it in the physical log. This leads to mistakes and inefficiencies when saving the data in the excel. Additionally, with this system the traceability is not maintained, as the information about the intake does not carry over to the redistribution, losing track of what donor donated what portion.

This is the central challenge of the solution, improving the efficiency and reliability of the operations while maintaining a completely traceable system.

Additionally, step four is simple and fast, once the food is packaged and distributed, it is given to the beneficiaries as they show up to the shop, this step also does not need to be optimized.

Finally, step five, as seen in the diagram, goes over the timeline, this is because the process becomes unnecessarily extended in time in order to save all data into the excel. However this last step forces a volunteer to stay for longer to make sure all collected data for the day is transcribed. Step number five should be erased altogether, and merged into step three. This should be accomplished with the proposed solution, as the data will be saved in the database during step 3, without making the process any less efficient or complicated.

In conclusion, the proposed solution should help merge step five into step three. Additionally, optimize the time taken to distribute and package food as well as reduce the error rate in the data.

3.2.3 Legal Requirements

Refood is a Spanish organization working out of Madrid, falling under Spanish legislation. On the seventeenth of June, 2022, a new bill was passed called "Proyecto de ley de prevención de las pérdidas y el desperdicio alimentario"[3] which translates to bill for

the prevention of food loss and food waste.

This bill declares that all businesses within the food supply chain that allow for systemic food waste within their processes are mandated to follow the fifth article. The fifth article specifies the hierarchy of priorities for these businesses; the first one being the donation of unconsumed food for human consumption [3, p. 18]. Refood, being an organization which collects leftover food from businesses and gives it to families in need, is the first resource for any business involved when it comes to the donation of its excess food. With this, a higher demand for Refood's services are anticipated as well as higher regulations for the traceability of donated food. The first obligation is denoted in the article 6.3, which mandates all organizations within the food supply chain to quantify their unconsumed food and comply with public administrations [3, p. 19].

On top of the foreseen increase in donations, Refood should maintain high quality standards for its data regarding where the food is coming from as well as to whom it is going. This is necessary to trace back any possible mishandling of food in a case where any damages have been caused. For this, a highly traceable system is needed in which data integrity is respected and maintained.

3.2.4 Growth Potential

As mentioned, the growth potential of Refood is large, in part due to the laws being put in place due to the SDGs, however, large part of it is also due to the increased visibility and concern for food waste and food pollution.

It is important to note Refood's scalability model, which is that of creating new nuclei around cities and countries with each attending to its own area; each area entails its own restaurants and beneficiaries. For this, it is important that the proposed digital solution can give access to different data depending on the nuclei from which the user is connecting. Ideally, each nuclei will have access to the solution in order to track its daily activities and only have access to its own data, and not that of other nuclei. Additionally, there should be different roles, that of a regular user who is in charge of tracking incoming and outgoing food, an administrator, someone who can add users, modify beneficiaries, donors and more. The goal is, (within these roles) to separate the users in terms of their nuclei, even if the solution does not allow for this at the end of the project, it should be simple to modify it in order to add new roles and nuclei as separate users.

3.2.5 Business Intelligence

Business intelligence is extremely important for any organization, helping achieve its objectives through a better understanding of its data. It allows users, volunteers, the public, as well as the people in charge to access information on business operations, metrics and results. According to IBM, business intelligence is ”software that ingests

business data and presents it in user-friendly views such as reports, dashboards, charts and graphs" [6]. In order to understand how an organization is operating, it is necessary to understand its data. Incorporating business analytics and automatic visualizations and dashboards into the solution would help Refood and the public understand the contribution of the organization to the world.

The most important aspect of the project in order to implement a business intelligence solution is a clear relational database schema. With this, existing tools can be used to create interactive dashboards and live visualizations utilizing low amounts of resources, directly from the database.

To understand the complexity of the incorporation of business intelligence with the old system, a python script was developed to extract information about total donations made by food donors, showing the magnitude of their efforts by the number of rations donated. The following figure is the graph obtained from the python script.

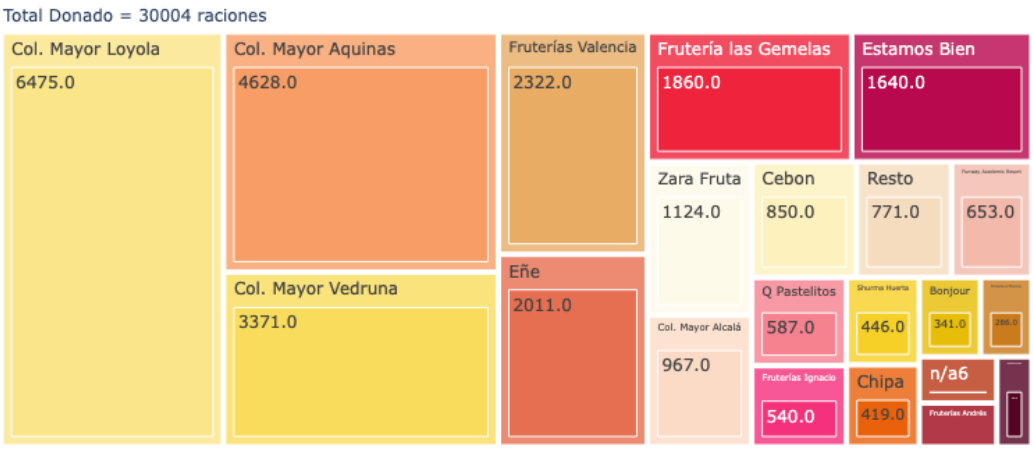


Figure 3.2: Visualization of donated rations by businesses

This visualization took a lot of resources due to the lack of organization in the data, around five hours were spent in the processing of the data. Additionally, some donations were not processed and had to be dismissed because they were wrongly saved into the database. Furthermore, in order to update the visualization with new data, the excel spreadsheets need to be processed again and merged with the already processed excel data.

However, this will not be necessary after the implementation of the solution, as the new data will be organized and its integrity and structure will be maintained. This will make dynamic visualizations possible and will grant Refood with more knowledge of its impact, as well as to advertise its efforts to other businesses and people.

In conclusion, these five aspects all contribute to the need of the digitalization of Refood's daily processes and are the initial steps toward identifying the problems and designing a fit solution. Brainstorming possible software solutions, implementations and designs, have all been part of the project and have led to the final solution currently in use. With these five reasons and requirements identified, it is possible to begin forming an idea of what the solution should look like and the objectives that need to be achieved with it.

Chapter 4

The Solution - Initial Stage

4.1 Project Scope

4.1.1 Objectives

The project has clear core objectives which have been defined with by Refood and the author; they need to be prioritized throughout the project. The core objectives are as follows:

1. Make a working software to maintain a traceable system of incoming and outgoing donations, complying with regulations.
2. Improve the efficiency and decrease the complexity of the data intake process.
3. Make a robust solution with enhanced data validation and data integrity.
4. Make a data visualization solution to generate advanced quarterly and annual reports.

It is important to visualize how these objectives differ from each other, and how they encompass many aspects of the organization. Therefore, the objectives of the project are divided into four categories. These categories include: solution oriented objectives, optimization oriented objectives, obligation oriented objectives and financial oriented objectives.

First of all, solution oriented objectives aim at solving current challenges within the organization's daily process, data structure and data analytics. The solution should provide a more efficient way to ingest the data of incoming and outgoing food, as well as the traceability of the system. Additionally providing error checking and a protective layer for the database, helping maintain the integrity of the data. Furthermore, it needs to rely on a more structured database which will aid the organization to analyze and visualize

the data more efficiently. A better data structure helps Refood gain thorough insights into their achievements and trends, through a cohesive data visualization system implemented with fewer resources than it is currently needed. These objectives are directed at solving the challenges mentioned in Sections 3.2.1 and 3.2.5.The core objective that falls into the solution oriented objectives category is number four.

Second of all, optimization oriented objectives focus on improving the ongoing daily process. Primarily, decreasing the time to complete the operational cycle, and decreasing the rate of mistakes in data collection per day. This category of objectives is the least prioritized in the decision making pyramid, however, they are in the interest of all stakeholders and are commonly taken into account throughout the project and more specifically, throughout the front-end design and programming stage. These objectives are directed at solving the challenges mentioned in Section 3.2.2. The core objectives that fall within the optimization oriented objectives are number two and number three.

Third of all, the obligation oriented objectives are a direct result of the new bill voted in congress which forces Refood to collaborate with public administrations in quantifying its impact, as mentioned in Section 3.2.3. These objectives are the first priority for Refood and include the complete traceability of the system, being able to trace back all the donations to the original donor. Additionally, it also includes keeping a correct measure of all incoming food in order to understand the total impact of Refood, as well as the current trends in systemic food waste. The objective that falls within the obligation oriented objectives is number one.

Finally, the financial oriented objectives are complimentary to the previous three categories and aim at achieving the previously discussed objectives with the least amount of economic resources possible. Refood is a non-profit organization with small financial capabilities and cannot afford high prices for hosting or developing an application.

Correspondingly, in order to achieve the aforementioned objectives, the expected deliverables need to be defined and agreed upon by stakeholders in order to efficiently arrive at the desired objectives.

4.1.2 Design Principles

Defining a set of design principles simplifies the decision making process, following the design principles as closely as possible throughout the design and development of the project. The following design principles are established:

- Prioritize the use of cost efficient or free technologies.
- Prioritize maintainable software for the long term usage of the application. Use strongly backed and financed technologies.
- Prioritize the future scalability of the solution to other Refood locations.

- Prioritize an intuitive User Interface.
- Prioritize early implementation and deployment.

First of all, Refood needs an affordable solution due to the nature of the organization, being a non-profit, it lacks the financial resources of a larger for-profit organization. Therefore, open-source technologies are prioritized in the technology architecture.

Second of all, the vision of the project is long-term, the solution is going to be used in the future and needs to be maintainable from the technical perspective. The technologies used should be strongly backed by large organizations or be extremely popular open-source technologies to be updated and maintained in the future.

Third of all, a future plan for the software is to be scaled up to other Refood locations around Madrid and Spain. The software is designed around scaling the application designing a database around a "Nuclei" entity to filter information to its respective Refood locations. Additionally, the application has the configuration to create new authorities to separate locations based on accounts with different authorities.

In addition, Refood has volatile volunteers which vary from week to week. Because of this, experienced volunteers cannot be present every day in the shop which complicates the decision of who will use the traceability system and if it will be used correctly. Because of this, the software should be designed around an intuitive interface that allows a user to flow seamlessly throughout the application. The process should easy to follow for volunteers with no previous experience.

Finally, Refood wants the application to be deployed as soon as possible to begin testing and collecting data. Because of this, a design principle has been defined to prioritize technologies which are faster to implement and deploy.

4.1.3 Description of the Solution & Deliverables

In order to fulfill all requirements mentioned in Chapter 3, and achieve the objectives in Section 4.1.1, the best fitted solution for a system of this kind is a web application and a newly designed database schema. Additionally, for the solution oriented objectives regarding data analytics and visualization, a data visualization solution is also defined and expected within the deliverables.

For one thing, the web application with the database schema will be the core of the project, as it will be the means to achieve the obligation and optimization oriented objectives, as well as part of the solution oriented objectives. Within the web application, all the data intake will be managed, including the input error checking and secure access to the database. Furthermore, it will force a structured database schema and will allow for higher physical security of the database, as it will be hosted in a server in another location, in contrast to being hosted within the computer in Refood. Additionally, it will aim at improving the efficiency of the daily processes of Refood and will maintain a

completely traceable system, as well as accurately quantify the total amount of food that is processed by Refood.

For another thing, the data visualization system will allow for improved insights into trends and behaviors of the donors and beneficiaries of Refood. Moreover, helping marketize the organization to potential new volunteers, donors and financial supporters sharing their impact within the community.

In conclusion, the deliverables for this project will be a web application, which includes the source code of the application in an accessible version control system and a executable file deployed in an accessible server with connection to the internet and the database. Secondly, a newly designed database schema, a database with the aforementioned schema connected to the web application. Thirdly, a data visualization system that allows for automatic report generation as well as the flexibility to design reports and dashboards in the future.

4.1.4 Assumptions

Finally, in order to correctly explain the scope of the project, it is important to understand under which assumptions it is being completed.

First of all, it is understood that Refood is responsible and capable of undertaking financial responsibilities derived from the deployment and use of the application, such as hosting services, electricity and internet bills, devices needed to use the application and more. This assumption is made under the notion that the financial oriented objectives need to be achieved and undertaking the lowest costs possible is of high priority for the organization.

Secondly, another assumption is that there will always be a volunteer responsible for the use of the application when it is being used. Even though the planned application is simple to use so that it has a flat-like learning curve, for its correct use, a volunteer with knowledge of how to access the application and input the data correctly will be needed to make sure the application is being used as intended. Nevertheless, an instruction manual has been made in order to explain the main features of the application and have an even faster learning curve.

Thirdly, due to the fast paced environment during the active operational hours within Refood, and due to the nature of web applications, an error, overload or glitch can cause the application to break at a certain point. It is assumed that volunteers always have an alternative data intake system, such as the previously used excel files, in order to save the data so it can later be added to the database. With this, the application will be fixed as soon as possible so it can be implemented again in the shortest time possible.

4.2 Stakeholder Analysis

The stakeholder analysis is vital to understand different perspectives, interests and obligations of the project. In this section, the roles and expectations of the identified stakeholders will be further understood. This comprehensive analysis will aid in organizing and prioritizing certain objectives and responsibilities based on the needs and wants of the actors involved.

4.2.1 Refood

First of all, one of the most involved stakeholders is Refood, the organization for whom the project has been developed. Their role is to help the author understand the operations of the organization as well as their goals and objectives with the software. Additionally, it is Refood who will be using the software, so it is their role to teach volunteers how to use it and adapt the shop to accommodate a computer in which the application can be used.

Additionally, it is of great importance to the project to fully understand the interests of Refood, as it is the beneficiary of the software. Firstly, improved efficiency and shorter operating time is crucial to the organization. This is due to the time constraints that Refood is working under, because of manipulation of perishing foods; which need to be maintained within the cold chain.

Secondly, Refood is a growing organization that needs to gain traction in order to gain more financial aid and subsidies. This is why the professional image of the organization is also within the main interests of Refood, modernizing and utilizing technology for the traceability of donated foods will aid the visibility of the organization to bigger companies and governmental organizations.

Thirdly, Refood has placed a thorough interest in data analysis, in order to understand how they are operating, as well as to find trends in the donated food and spread awareness to the donors. On top of this, Refood has promised quarterly reports of donated foods from their key donors, which need to be automated by the software so they can be scaled up or down and generated in a short time-span.

Lastly, as seen in Section 3.2.3, the government is pushing a new waste prevention law which forces Refood to maintain a quantitative record of all the incoming and outgoing food [3]. Therefore, it is vital for the data of the application to be saved in a safe environment where the data can be accessed and indexed easily, complying with government agents when needed.

4.2.2 The Volunteers

Secondly, another important stakeholder is the volunteers who run the operations each day. The volunteers are going to be using the web application to keep the traceability of

the food they are collecting, repackaging and giving away to the beneficiaries. Volunteers change often, with new ones coming in and older ones leaving the organization, this is why the application needs to be easy and intuitive, so the learning curve is flat and the usage of the traceability software can be taught in a short time.

Volunteers have two main interests in the project, one being the improved efficiency of the operations and the simplicity of the process being the other. They have to go on routes to collect all the food, and then another group of volunteers is waiting to repackage and distribute the food between all the beneficiaries. This process can be stressful and time consuming, while they have to measure and repackage everything into smaller containers, they also need to distribute it fairly between beneficiaries, taking into account the number of people and kids associated with each beneficiary. Because of this, keeping track and weighing each container can be a very frustrating process if it is not designed efficiently. Because of these requirements, a big focus is put on the simplicity and clarity of the design of the web application.

4.2.3 Financial Supporters

Thirdly, the financial supporters of Refood are important to take into account, as they provide the organization with the economic support needed to keep the operations running. Financial supporters do not have strict responsibilities, however, it is important to keep their interest in mind, as they are helping Refood grow through their financial aid.

The main interest of the financial donors is to obtain information on what impact Refood is having, how much food is being saved and how many people are being served each day. This can help donors visualize the impact of their donations and can motivate them to keep donating, showing the results that Refood is achieving every day. Because of this, a big emphasis is put into the data analytics and visualization of the project.

4.2.4 Beneficiaries

The beneficiaries are a big part of the organization and the project. The obligation oriented objectives are related to the health and well-being of the beneficiaries, and maintaining the traceability is a big part of the safety regulations within Refood. This is the primary reason for the development of the application and the core of the software functionality.

4.2.5 Donors

Another stakeholder is the donors of food, from restaurants and bars to large student residences and healthcare organizations. The donors have the responsibility of packaging their leftover food in big containers and have them ready at the time the volunteers go by

their shop. It is also their responsibility to not give rotten or uneatable food, this is why the traceability needs to be kept intact, as it is important to know where the food came from in case of a health violation.

The main interests of the donors are to understand food waste related trends, in order to prevent them in the future, and to comply with new government regulations. This project will offer the donors more advanced knowledge on key trends, visualizing their donations and generating quarterly reports to understand their contribution to the organization, as well as how many people they have helped.

4.2.6 The Government & Public Administrations

Finally, one of the key stakeholders is the Government and the Public Administrations responsible for food waste prevention and management. Their responsibility is to keep track of how much food waste is being produced in Spain, and how much of that can be and is being prevented. Additionally, it is their responsibility to identify health and safety violations within organizations in order to prevent malpractices and impose stricter regulations related to food handling and donations. They will look at the traceability records of the organization, for which it is important to keep a backup, in case of any technical problem. It is also important to maintain good data governance and design practices in order to maintain the integrity and security of the data, as well as its availability and easy indexing. This is the most important aspect of the project, and it should always be prioritized when designing and implementing the web application.

4.3 Project Planning and Scheduling

Due to the many stakeholders involved in the project, of which the two most relevant are Refood and the Government, the planning of the project was not unanimous and required the interests of both parties to be accounted for at all times. An initial plan was decided upon, in which the general tasks were highlighted and put into time slots. Furthermore, important milestones were identified by Refood as a priority, and fixed in time. The author had freedom to work on general tasks, but the milestones should be reached at the agreed dates.

4.3.1 Collaboration and Communication

Because the project is being done for an external organization, the collaboration has been a key part of the success and correct completion of the software. Additionally, in order to adapt the project to changing objectives, the communication between Refood and the author was vital to the project. In order to keep dynamicity within the project, the author used the Agile methodology, working in two-week-long sprints and reported

the improvements to Refood at the end of the sprints [7]. Please see Appendix B for more about the Agile Manifesto and Agile methodology.

4.4 Technology Architecture

First, the solution is to be based around reliable, commonly used technologies that have been tested at large. Second, it needs to have a centralized database to have an efficient and labour-saving data visualization system. Third, the cost of the solution should be minimal; the use of open source technologies is prioritized. Lastly, the solution will be implemented by the author only, so the technology stack should not require a large team of developers nor timely coding requirements. This way, more time can be invested in designing and implementing the solution. With this in mind, the following technology architecture is designed and proposed to Refood.

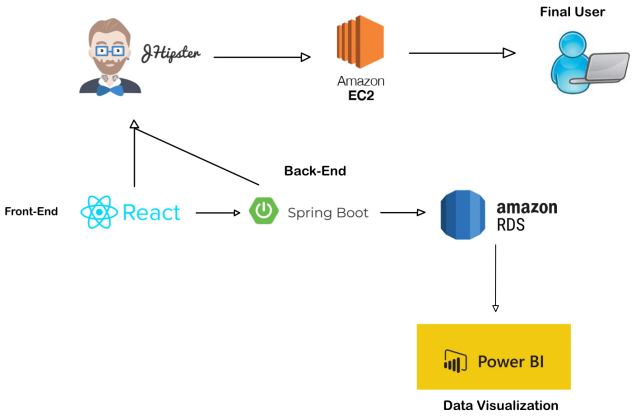


Figure 4.1: Technology Architecture

In Figure 4.1, the technologies planned for the development of the web application can be seen. Firstly, an SQL server will be deployed with a designed database schema which will allow for a centralized database. Secondly, the back-end of the web application will be developed with Spring Boot, a java based framework [8]. Additionally, the front-end will be programmed using Reactjs [9] and typescript [10]. All of these technologies will be explained thoroughly in Chapter 5 and contrasted with other available technologies which have been contemplated for the project, as well as the reasoning behind the selection.

Furthermore, in order to develop a structured web application folder structure and package it efficiently, an open-source web application generator called JHipster [11] will be used. JHipster is used in technology consulting companies in order to increase

efficiency and improve the code structure of large projects. Additionally, with the goal of implementing a dynamic data visualization system, PowerBI will be used in order to connect directly to the centralized database and create dynamic dashboards [12].

Finally, the cloud hosting service used is Amazon Web Services (AWS), from which the solution will primarily use two services: Amazon Relational Database Service (Amazon RDS) and Amazon Elastic Compute Cloud (Amazon EC2). Amazon RDS is be used to host and access the production database, and Amazon EC2 is used to host the web application [13].

However, the decision between hosting the web application in the cloud or on-site led to discussions about the maintenance, cost, efficiency of the solution and availability of the application to end users. First of all, the access to the application is intended to be open to future Refood locations around the city and even the country, so it cannot be kept in a local machine without a public IP. Additionally, a comparison between on-site servers and the cloud was carried out using a weighted decision matrix.

	On-site Servers	The Cloud
Costs (0.25)	10	100
Maintenance (0.2)	10	90
Security (0.15)	20	90
Scalability (0.05)	70	100
Reliability (0.1)	80	100
Data Backup (0.15)	20	100
Availability (0.1)	5	90
Total Score	**25**	**95,5**

Table 4.1: Weighted Decision Matrix - On-Site Servers vs The Cloud

The conclusion reached from Table 4.1 is that cloud computing services are a better fit for hosting the web application and database, and therefore AWS will be used in the solution. Some points worth clarifying from the decision matrix are that in terms of costs, Refood could claim an economic aid package from Amazon Web Services, AWS, as Refood is a Non-Profit Organization [14]. This aid would reduce the cost of a cloud solution considerably. Additionally, cloud solutions, such as AWS, offer extended security aspects such as access control management (ACM) systems and physical data center security [15].

Furthermore, regarding the maintenance of the application, using JHipster helps with the structure of the web application, making it easier to maintain over time. On top of that, Spring Boot and Reactjs are some of the most used frameworks for web application development in the world [16], [17]. Reactjs, JHipster and Spring Boot are all well-documented frameworks which are used and tested by many developers world-wide; making for a reliable and consistent solution.

Moreover, it is of utmost important to take into account the CI/CD of any digital solution. Due to updates in software, dependencies and libraries, web applications commonly return bugs and glitches, thus it is important to have access to the source code in order to change and adapt the solution to the constant change of the application development industry. The strategy for continuous use and continuous development of the application is explained in Chapter 7.

4.5 Design Requirements

The next step in the initial design of the web application is to identify and understand the objective and requirements for the application. Understanding what the organization expects and what needs to be achieved by the application is vital to a correct design of a digital solution. The requirements that Refood had in mind for the solution varied significantly with time, but working with an Agile methodology led to understanding these changes in advanced, and adapting the software to satisfy Refood's needs.

4.5.1 Application Design

Wireframe - First Version

Initially, a wireframe is made to understand the logic of the application in mind, it does not need to show the design of the application, but rather the flow of screens and decisions that the user can take within the application. This gives the developers an idea of how many screens need to be designed and programmed, what the application requirements are and where to start.

Once Refood's requirements and objectives for the application are understood, the wireframe in Figure 4.2 is created in order to convey a simple visualization of the solution before the development process starts.

As seen in Figure 4.2, the app consists of a Login step that once is completed leads to a verification of the role type of the profile. This step is shown as a future plan, meaning that it will not be implemented in early versions of the application but rather will be a future improvement. Secondly, the application will begin with a Daily Volunteer registration, in which the volunteer responsible for traceability for a specific day needs to input his details. Following this, the main menu is reached, from here two options are available, registering an outgoing package or finalizing the daily activity. If an outgoing package needs to be registered, the user will go through the steps necessary to register all outgoing packages, on the bottom-right of the diagram, and finally go back to the main menu when no more packages need to be accounted for. Once no more packages need to be registered, the user can end the session, prompting them with a dashboard showing the daily results, this also is part of the future plan. Finally, leading the user to

the log out step in which the process ends.

This wireframe makes the groundwork for the initial stages of the application design. With this, a second wireframe is created in which a visual design of the screens is completed.

Wireframe - Second Version

In the second version of the wireframe, a design has been made for the look and feel of the application.

As seen in Figure 4.3 and Figure 4.4, the wireframe has a new dimension, a visual characterization for each view of the application. Due to the visual nature of the second version wireframe, some steps were able to be identified as potentially unnecessary, and some small changes were introduced.

First of all, as seen in Figure 4.3 the assignment of the volunteers to their respective routes was deemed as unnecessary for the goal maintaining traceable inputs and outputs. Additionally, the process to register new donations is separated into all the important steps. Initially, the donor has to be declared, then the type of food that was donated, if this is correct, the next step would be adding the respective weight of the package and the container in which it was given out, maintaining a tally of the incoming and outgoing containers. Finally, the correct date has to be added in order to have a timeline of all the incoming packages.

From the latter version of the wireframe it can be seen that the objective is for the application to be fault tolerant, meaning that it allows as little hand-written user input as possible. In an ideal software, there should not be any manual input, as this disrupts future data processing and visualization. Different ways of writing inputs, spaces and different names given for the same thing will cause problems with the data. This is why as much of the input as possible is predefined by the application, only allowing users to select from an already existing list. In some cases, if information or options are missing from the list, these can be added by the user so it can be selected in the future. The final design of the application is shown in Section 7.

4.6 Application Lifecycle Management

Application Lifecycle Management, ALM, encompasses all aspects of the lifecycle management of the application, such as governance, development and maintenance [18]. The most important for this project being maintenance. Refood's main concern is regarding the future maintenance of the application; once the solution is designed and implemented, how it will be maintained.

There are different categories when it comes to software maintenance, these being Corrective, Preventative, Perfective and Adaptive [19]. The focus of this project being

on Corrective and Adaptive software maintenance.

4.6.1 Corrective Software Maintenance

Corrective software maintenance is required when an application does not work as expected. This can occur due to a bug or an unexpected error, which can disrupt the workflow of users and potentially block data from going into the database.

In order to have an effective software running, it is of utmost importance to have imposed strategies for error correction and bug fixing. However, as Refood is a non-profit organization without any in house IT specialists, a ticket system is hard to design and implement. A ticket system is used in most corporations as a way to communicate user generated errors in the software, to IT technicians. As ticket systems services are not necessary at this scale, each Refood location is given a dedicated organization email, so any error can be communicated and and looked into as fast as possible. With this, the error can be fixed and the new version is deployed as soon as possible. However, in order to make this process as efficient as possible, the system needs to be optimized for fast deployment speeds and easy access to the source code of the application, for which AWS and a GitHub repository will be used respectively.

4.6.2 Adaptive Software Maintenance

Adaptive software maintenance is when a software is modified in efforts to improve and adapt to current needs. The main difference between Corrective and Adaptive software maintenance is that the latter is not a reactive but rather a proactive maintenance, meaning that there are no strict time constraints in deploying the modifications. Adaptive software maintenance is not about having quick response times to user requests and troubleshooting bugs, but rather a proactive approach to improving a software for its long term benefits. For this, a feedback system is in place to understand the current needs of the organization and design adequate modifications to fit those requirements.

The feedback system is made up of surveys sent out to volunteers who use the software in order to understand the positive and negative experiences they might be having. These surveys are sent out every four to six months in order to have a sufficiently long feedback loop in which a new design can be made and implemented based on any necessary improvements. On top of this, Refood staff is constantly be overlooking the software and taking feedback from the users, also gaining a perspective into the use cases in which the software is beneficial, and other scenarios in which the software may not be as efficient or useful as expected. The results of the first survey are discussed in Section 7.3.

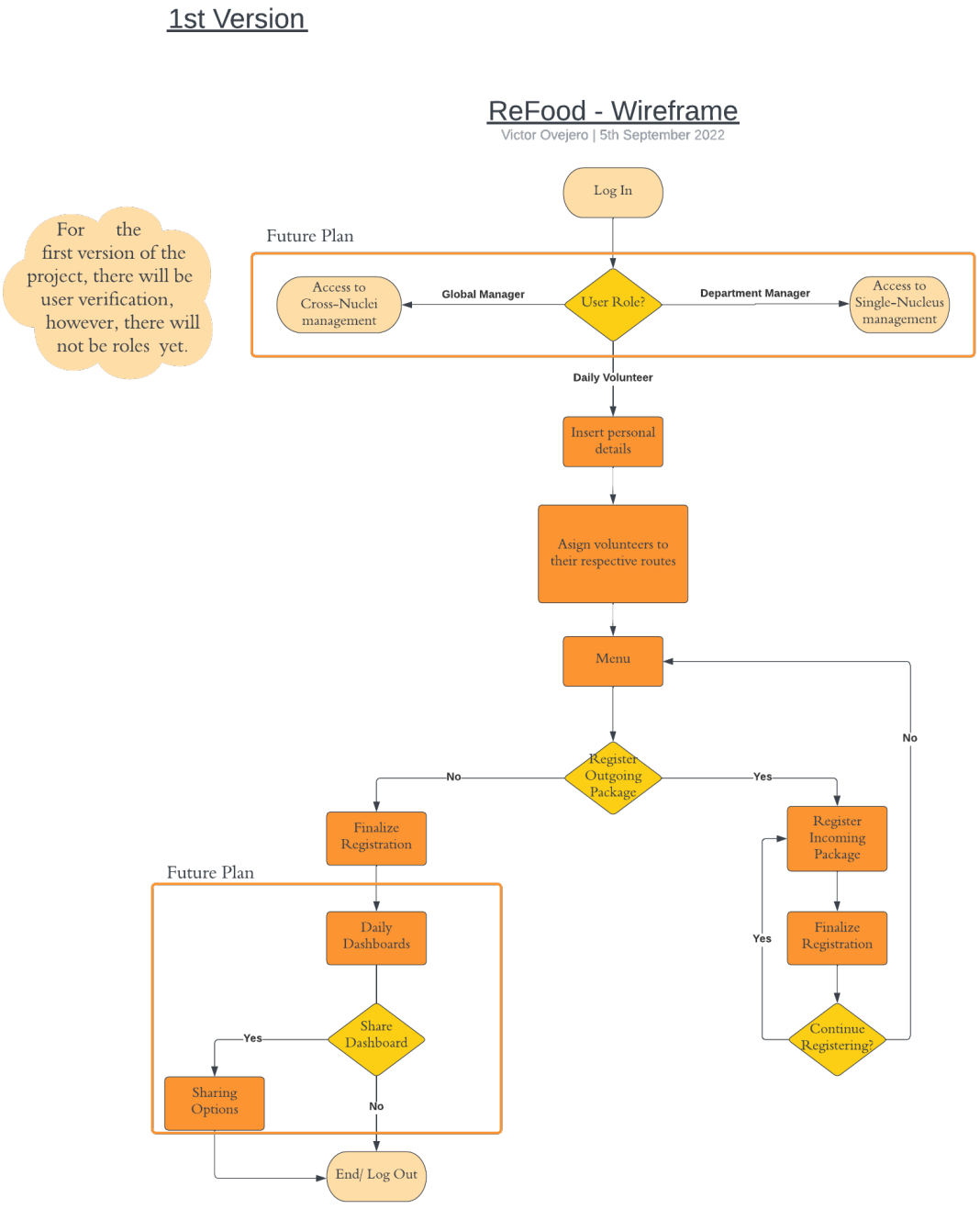


Figure 4.2: First Version Wireframe

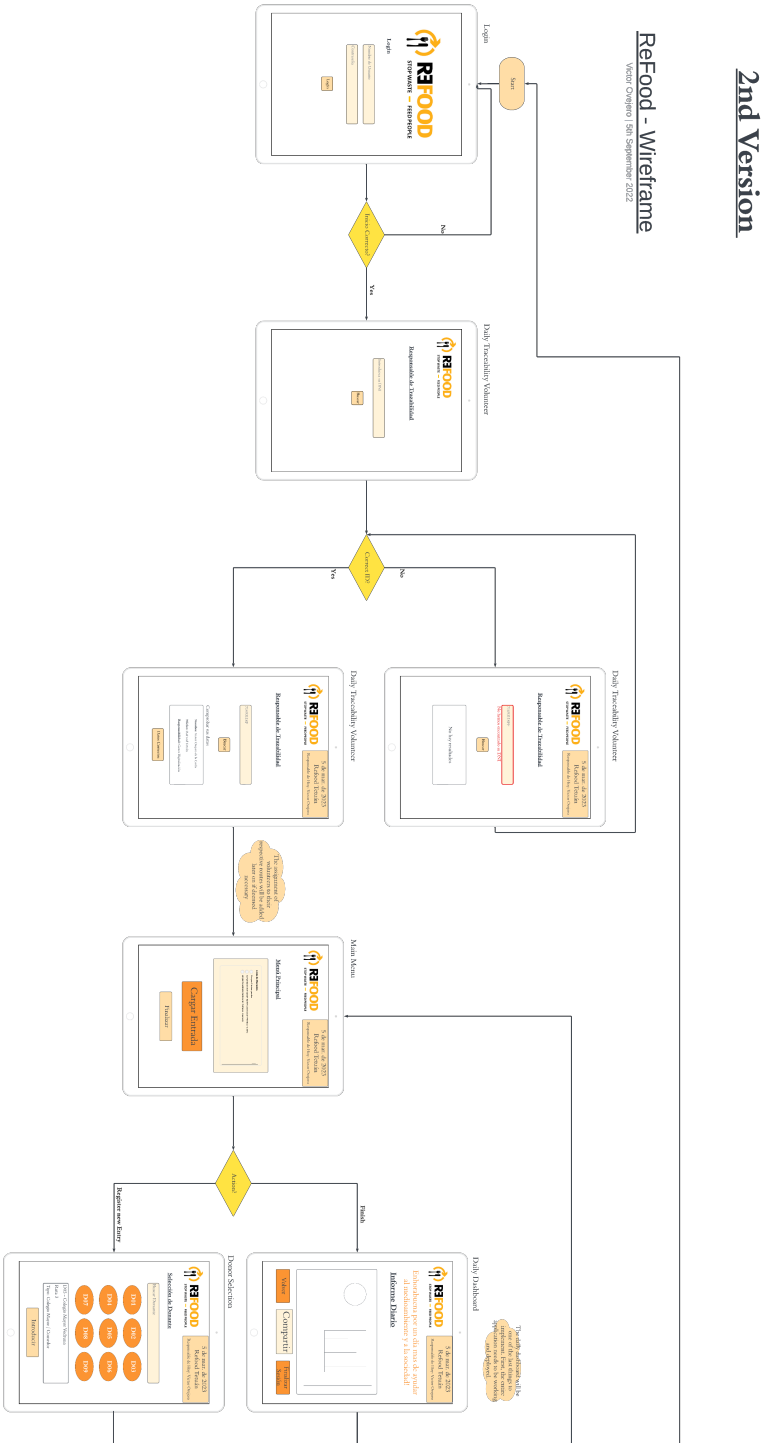


Figure 4.3: Wireframe First Half

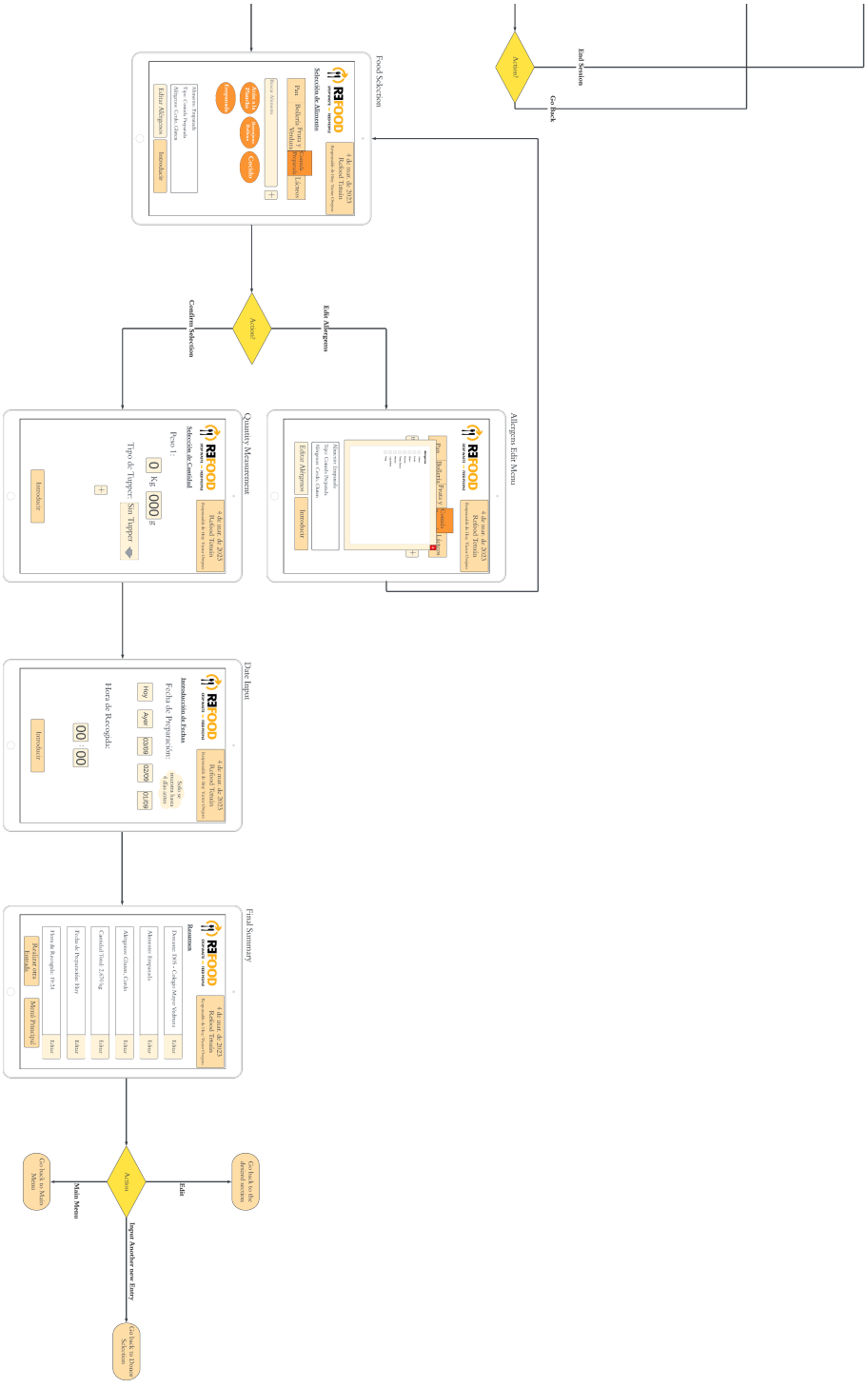


Figure 4.4: Wireframe Second Half

Chapter 5

Development of the Web Application

In this section, the selected technologies, mentioned in Section 4 are analysed in depth. Additionally, they are contrasted with other possible alternatives in order to understand the reasoning for the final architecture. Finally, the use of each technology will be explained thoroughly.

5.1 Web Application Architecture

There are two commonly used application architectures in software development: monolithic architecture and microservices architecture. The web application designed follows a monolithic architecture.

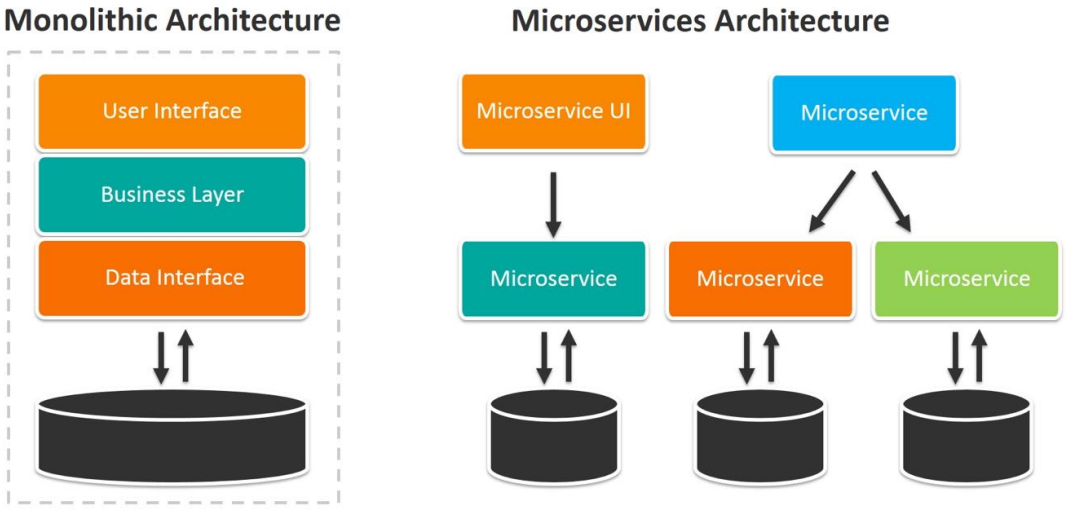


Figure 5.1: Monolithic vs Microservices

According to MuleSoft, a monolithic application complies with the following requirements [20]:

- A database consisting of many tables/entities, usually in a relational database management system.
- A client-side user interface consisting of HTML and usually JavaScript.
- A server-side application which handles HTTP requests and fetches and modifies data from the database.

On the other hand, a microservices architecture consists on building an application on a modular basis, rather than building one single monolithic unit, loosely coupled and independently deployed services are developed. Each microservice runs its own HTTP requests.

Monolithic architectures are commonly used for medium-sized CRUD applications. A monolithic architecture is notably simpler and faster to implement and deploy as all the code is in one place. As defined in Section 4.1.2, a design principle is to prioritize technologies that are faster to implement due to the urgent need of Refood to enforce a traceable system and collect data. Additionally, maintaining consistent data is enhanced in monolithic architectures, as all data operations are in one place. This allows for a centralized relational database which can be scaled to more Refood locations, also following the design principle of scalability.

Nevertheless, monolithic architectures can turn into complex projects in which no single developer can understand the whole of the application. However, this is the case for projects of larger scale, not for medium sized CRUD applications such as the case of this project.

Microservices architecture offer increased flexibility, as different microservices can be used for different purposes and created for a new functionality or scalable option. Additionally, enhances fault isolation, as an error caused by one microservice will not translate into another, increasing the fault tolerance of microservices applications.

However, microservices architecture has many challenges such as data consistency, having a different data access layer for each microservice. In addition, the operational complexity and inter-service communication can become increasingly complex as the number of microservices increase.

In the case of this project, entities have similar data access layers and functionalities, not needing to decouple them into modular microservices. Additionally, the increased worsened data consistency from microservice applications is not ideal for Refood, as one of the core objectives of the project is to enhance data validation and integrity, which can be managed more efficiently with a monolithic architecture. Finally, the monolithic architecture is more fitted to accomplish the core objectives and design principles of Refood, making it the chosen architecture for the web application.

5.2 Database Selection and Design

One of the pillar of web application development is the database. This is because the database provides a web application with many necessary tools. First of all, the database provides a structure for persistent data, allowing for the application to store data permanently. Secondly, it maintains the integrity and consistency of the data so that it can be trusted by external organizations. If properly implemented, a database can provide extra layers of security and safety to the information, being able to externalize it to a secure location and create backups in case of technical failure.

5.2.1 Selection of Database Technologies

Relational Database

First of all, if the data is structured and predictable and does not have the need for horizontal-scaling, relational databases are the most optimal choice. Relational databases allow for normalization, reducing the amount of data stored on disk by limiting duplicate data and centering around relationships between entities, while also supporting ACID (Atomicity, Consistency, Isolation and Durability). On the other hand, non-relational databases are dynamic, they can handle horizontal-scaling and do not have such a strict structure. However, non-relational databases do not support ACID, making its data less reliable and hindering its integrity. [21].

For the case of Refood, data is structured in tables and it relies heavily on relationships between entities in order to enable the traceability of entries. For example, an incoming donation will first be saved as a donation, then it will be distributed as many portions and finally distributed between different beneficiaries. These relationships need to be visible and easy to understand when looking at the database. With this, the first decision can be made: a relational database is the optimal choice for Refood.

SQL & Database Management System

The query language selected is SQL, for its wide-adoption and popularity [22]. Additional advantages from using SQL in the database implementation are the following:

- ISO standard and is supported by nearly all relational database management systems.
- Ensures ACID properties, making it highly reliable.
- Databases can implement security features.
- Large community and extensive support.

- Highly compatible with frameworks and languages. Often having libraries to interact with SQL databases.

Secondly, the two most popular management systems are MySQL and Oracle Database. The main point of comparison is that Oracle database is not open source, and has a full paid version on top of the enterprise edition which is free. On the other hand, MySQL is completely free and open-source, being ideal for Refood. Additionally, Oracle database is meant for larger enterprise databases, while MySQL can be used for less complex and smaller databases allowing for faster design and implementation [23]. For these reasons, MySQL is selected as the database management system for the project.

In conclusion, a relational database will be designed using SQL and implemented using MySQL. This is a free, scalable and widely adopted database architecture.

5.2.2 Design of the Database Schema

The design of the database schema changes with the development of the project, initially designing a schema and upgrading it further into the development of the web application.

First Schema: Initial Design and Normalization

The first schema of the database serves two purposes. First, it allows the author and Refood to visualize the functionality of the web application, defining a scope for the data that will be manipulated and stored within the application. Second, it allows for initial implementations within the application to begin testing.

As presented in Section 3.2.2, the operational process consists of an initial intake of foods from various donors, then the food is partitioned in smaller containers and weighed, and finally, this is given to the beneficiaries. The web application is able to manage these functionalities, while also storing information about the location of the operations, and what volunteers managed the operational flow each day. With this, the first schema design is presented.

In figure 5.2, it can be observed that all entities are directly or indirectly related to the entity ”Nucleo”. This is for the future scale up of the web application to new Refood locations, allowing to differ data between them. Next, the entity ”Donante” has a one-to-many relationship to the entity ”AlimentoEntrada”, as one donor can donate many times, but each individual intake only comes from one donor. At the same time, the entity ”AlimentoSalida” also has a one-to-many relationship to the entity ”Beneficiario”. Therefore, the traceability of the system is being kept, as we can trace an outgoing food that has been donated to a specific beneficiary, all the way back to the original donor, through the ”AlimentoEntrada” entity.

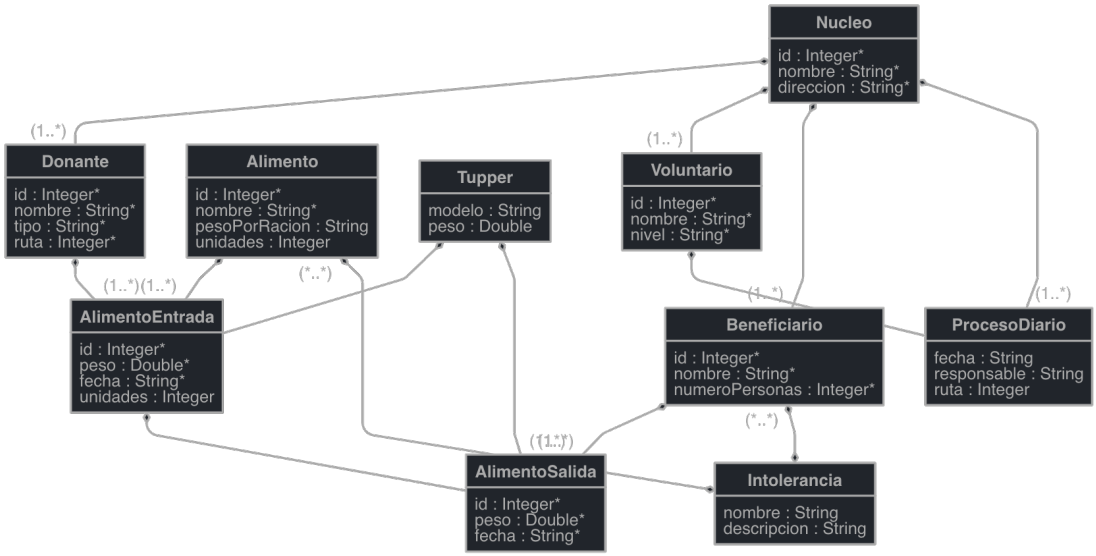


Figure 5.2: Database Schema: Second Version

Additionally, it is important to note that the schema presented in figure 5.2 is normalized up to the Third Normal Form. This means that the design satisfies the criteria of the First Normal Form, Second Normal Form (2nf) and Third Normal Form. First of all, in order to be normalized to the First Normal Form, it should have no repeating groups, it decomposes repeated groups into different relational entities and every table has a primary key. Secondly, it is in the Second Normal Form because there are no partial key functional dependencies. This means that every non-key attribute is functionally dependent on the entire primary key. Finally, the schema is in the Third Normal Form because it has no transitive functional dependencies. This means that there are no indirect relationship between two attributes through a third attribute [24].

Other entities are defined from an operational standpoint, such as the entity ”Voluntario”, ProcesoDiario and Tupper. These entities are used to keep track of volunteers, containers used and information on the process of the daily operations of Refood.

The schema seen in figure 5.2 is an initial schema, where the fields are only the strictly necessary to create relationships between the entities and have a general understanding of the flow of data. Additionally, the database is normalized in order to maintain the integrity and accuracy of the data by reducing redundancies. The second version will focus on its optimization, extension of functionalities and modifications to overcome technical difficulties with the deployment of the application.

Second Version: New Entities and Technical Constraints

The second version of the database schema has shortened the names of the entities due to technical constraints, changed the traceability process and added new field.

Firstly, the names of the entities were shortened to allow for a correct deployment of the application to AWS. The deployment of the application is done with CloudCaptain, a tool used to automatically deploy applications to AWS [25]. When implementing the database schema into MySQL, relationship tables are created and named appropriately after the entities which they connect, as seen in Section 5.2.3. These names would become longer than 63 characters and would exceed the name length limit allowed by AWS, not allowing the application to be deployed.

Secondly, the functionality of registering outgoing food was changed. Instead of having one entity for the incoming food and another for the outgoing food, it was decided that a third entity was needed, ”Checkout”. The entity ”Checkout” is used to map all the outgoing foods to one specific beneficiary as one package. The objective of this change was to increase the efficiency of the operation, allowing volunteers to focus on portioning the food safely rather than on the traceability of the system. The problem with the first schema is that each outgoing food needed to be weighed in order to know how much food is given to each beneficiary. This is repetitive and time consuming, as weighing and noting every donated portion would notably increase the time and complexity of the operations. With the new logic, each portion could be packaged and distributed without having to weigh it, only needing to weigh the packages at the end, once per beneficiary. As seen in figure 5.3, the new entity ”Checkout” has a one-to-many relationship with the entity ”Beneficiary” and a many-to-many relationship with the entity ”AlSal”. The traceability of the system is still maintained, and the operational process has been notably simplified to increase the efficiency.

Additionally, a new entity ”Socio” is generated. The entity ”Socio” represents a monthly financial donor, this can be anyone who donates a fixed monthly amount. The addition of financial donors to the database was important to the financial department in order to keep track of who is donating. Nevertheless, aside from the storage of the information, there are no functionalities within the final web application that are meant to control or keep track of donations nor handle the donations.

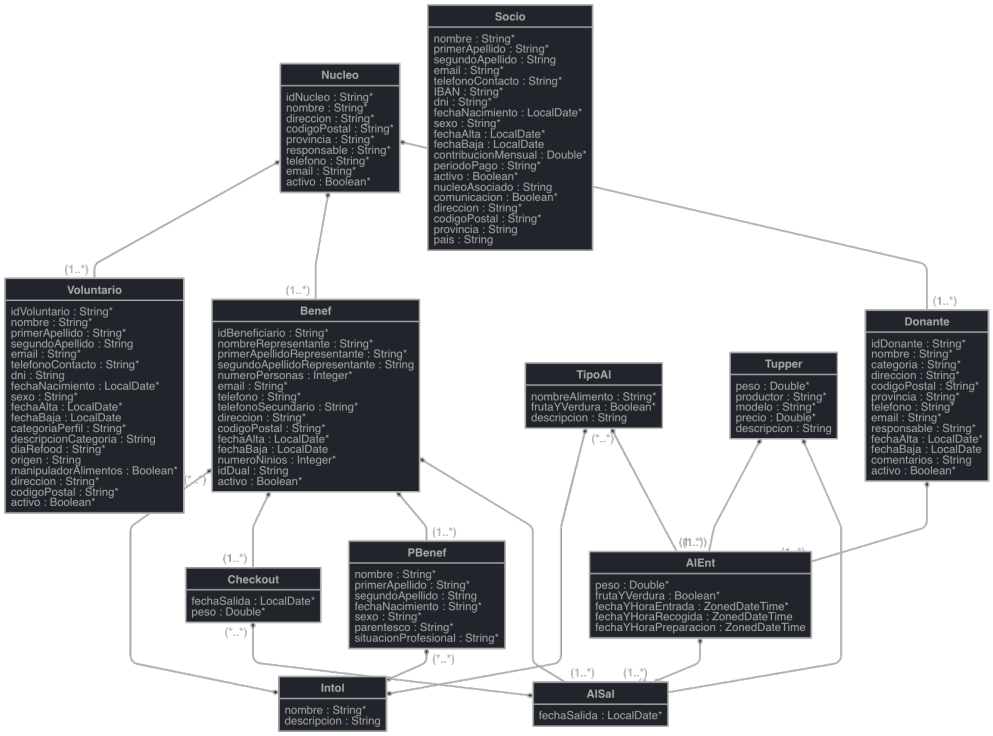


Figure 5.3: Database Schema: Second Iteration

This is the final version, the database schema shown in figure 5.3 is the current database in production and has proven error-free and efficient, as seen in Chapter 7.

Finally, the entities are described in the following table. The first column is the initial name of the Entity, the second column is the final name of the Entity in the second version, and the third column is the description of the entity. Some entities are deleted and some are created after the first iteration, that is represented by ”N/A” when the entity does not exist in the specific version.

Entity v1	Entity v3	Description
Nucleo	Nucleo	The Refood shop location. Each Refood location operates individually.
Donante	Donante	The donor. Each donor has to have an individual location from which the daily donations are picked up.
Beneficiario	Benef	The beneficiary. Each beneficiary has one person of contact which represents a group of people, or family who live under the same address.
AlimentoEntrada	Al-ent	Incoming food. An incoming food is characterised by the type of food, donor of the food and date of donation. If one type of food is received in many containers, it will be accounted for as one, bigger donation.
AlimentoSalida	Al-sal	Outgoing food. An outgoing food is a portion from an incoming food which is packaged and donated to an individual beneficiary.
Alimento	Tipo-al	Type of Food. The entity Type of Food describes possible foods that can be donated by any donor, such as chicken or pizza.
Tupper	Tupper	Tupperware. This entity describes the container in which food is donated, helping Refood keep track of their containers.
Voluntario	Voluntario	Volunteers. A volunteer is someone who helps Refood by collecting food from the donors or by helping in the transformation of incoming food into outgoing food.
ProcesoDiario	N/A	Daily Process. This entity connects shows what volunteers did what routes and each day.
N/A	Checkout	All portions going toward one beneficiary are weighed and packed together, the package becomes a Checkout row.
N/A	Socio	Financial Supporters who donate every month.

Table 5.1: Description of Database Entities

5.2.3 Development Database

The implementation of the database will be done in MySQL, as mentioned in Section 5.2.1. In order to create a database in MySQL, it is necessary to write the SQL queries to

generate the desired tables and attributes, as well as the relationships. Nevertheless, by using Liquibase, JDL Studio and JHipster, the SQL queries can be dynamically generated and executed, creating a database in MySQL from the original JDL description. A more detailed view of the JDL syntax and the database schema in JDL is given in Appendix C.

The database is correctly generated in MySQL, by explicitly configuring the endpoint, username and password in the configuration file of the application, within the "resources/config" folder. The application connects to MySQL through the Java Database Connectivity (JDBC) interface. Nevertheless, this is only used at the low level connection, as JHipster uses Java Persistance API (JPA) for the majority of data access operations through Spring Data JPA, providing more functionalities and a layer of abstraction from JDBC.

It is important to note that JHipster also generates and manages the user authorities and the login functionalities. It keeps the user information encrypted in the Database.

In conclusion, the configuration for the implementation of the database is mostly handled by JHipster. Nevertheless, this is only for development purposes and testing. In production, the database is kept in the cloud, the configuration is slightly different, but the database schema can be reproduced from the local MySQL server.

5.3 Web Application Generator

As mentioned in Section 4.4, a code generator is used to generate the skeleton of the application. The reasons for using a code generator are the following:

- Enhance the efficiency of the initial web application development.
- Adapt to the best practices and standards of web application development.
- Automated creation of CRUD operations for the generated entities.
- Database integration.
- Generates a responsive web interface.
- Quality assurance with test implementation using Cypress.
- Support of authentication mechanisms such as jwt and OAuth2.

The generator selected is Jhipster, a modern and open-source full stack application generator. Modern development has evolved around code generators, which create a local project with a standard folder structure, writing all the boilerplate that would take an experienced team of developers weeks of continuous work. Nevertheless, using a

code generator does not allow for a personalized web application, as the developers need to adapt and personalize it to their organization or project goals.

The use case for JHipster within this project is to generate the initial folder structure and boiler plate code as to shorten the time to an MVP. The priority for Refood is to have a viable product as early as possible in order to begin producing quarterly reports for the primary donors. Additionally, migrating to the new database as soon as possible with the goal of maintaining a clean track of the traceability.

The following folder structure is generated by JHipster when the selected options are front-end in Reactjs and Maven as the build automation tool and compiler.

```
application/
├── .jhipster/
├── .mvn/
├── node_modules/
├── src/
│   ├── main/
│   │   ├── docker/
│   │   ├── java/
│   │   ├── resources/
│   │   └── webapp/
│   └── test/
├── target/
└── webpack/
```

First of all, in the folder structure, all entries ending with ”/” represent a folder, while other entries represent a file.

The folder structure follows standard practices to facilitate the understanding of how the web application is designed. Therefore, aiding external organizations or developers looking at the code, as an organized structure will make it easier for a developer to recognise how the app is built. The general structure is unpacked in this section. Additionally, deeper layers of the structure will be shown and explained in the following sections.

.jhipster

First of all, the ”.jhipster” directory is a hidden folder that generates the entities and relationships in json when importing the database from JDL Studio. This allows jhipster to map the database from JDL Studio to the back-end and fron-end.Additionally, it allows Liquibase to dynamically generate the necessary SQL queries to generate the database in MySQL, this will be expanded in Section 5.2.3. JDL Studio is a tool to draw UML

(Unified Modelling Language) diagrams using the JDL syntax. JDL Studio is part of the JHipster project [26].

.mvn

Secondly, the ".mvn" directory is another hidden folder that has the necessary dependencies and basic configuration for the Maven compiler to work correctly. Additionally, the "node_modules" folder contains all external dependencies for the project, downloaded from the node environment.

src

Thirdly, the "src" directory contains most of the application functional code, it contains the front-end and the back-end as well as the necessary configurations to connect to the database. Additionally, it contains the general configuration of the web application. This is the main folder that the developer works on, adding functionalities, changing configurations, connecting the database and implementing the design of the application. Within the "src" folder there is the "main" and "test" folders.

main

Next, the "main" directory holds the front-end and back-end code in the "webapp" and "java" folders respectively. It also contains the "resources" folder which has all the configuration of the app, most importantly, the database configuration, access ports, username and password, cache options, CORS configuration and more. This is also where Liquibase keeps logs and configuration files of the database management and changes. Finally, JHipster supports containerization with docker, allowing for the application to be developed and run on docker containers. This is not used in the project, but it is also not removed for possible future expansion or need for containerization.

test

The "test" directory contains all test-related code and files, the back-end tests, front-end e2e tests and unit tests. Additionally, it contains the configuration files for the testing tools. In the case of this project, the front-end e2e testing tool used is Cypress, an open-source project under the MIT License [27].

target

The "target" directory is where the executable file is generated when the web application is compiled and packaged. Additionally, other files which are generated from the compilation of the application or any other available jhipster command.

webpack

Finally, the ”webpack” directory is a Reactjs-specific folder to bundle and manage dependencies. It also manages plugins and contains the necessary support for a development server used to see changes to the front-end of the application live, without compiling the code.

5.4 Back-end Development

5.4.1 CRUD Operations

CRUD (Create, Read, Update, Delete) applications can perform the following actions on a database:

Action	Description
Create	Input new rows of information in existing entities by explicitly specifying the contents of each attribute.
Read	Access existing data in a database. This can be specific rows, full tables or individual attributes through the instance id.
Update	Update existing records in a database. The new values need to be specified, overriding the existing content.
Delete	Erase existing records from a database. The contents will be unrecoverable unless cached or saved in a backup.

Table 5.2: CRUD Actions

A CRUD application is used when the main objective is for the users to seamlessly interact with a database through a user-friendly interface. Essentially, CRUD is the basis of any application that operates with persistent data, allowing the back-end to interact with the database.

5.4.2 Repository-Service Pattern

The interaction and logic for the data transfer between the web application and the database is handled by the back-end.

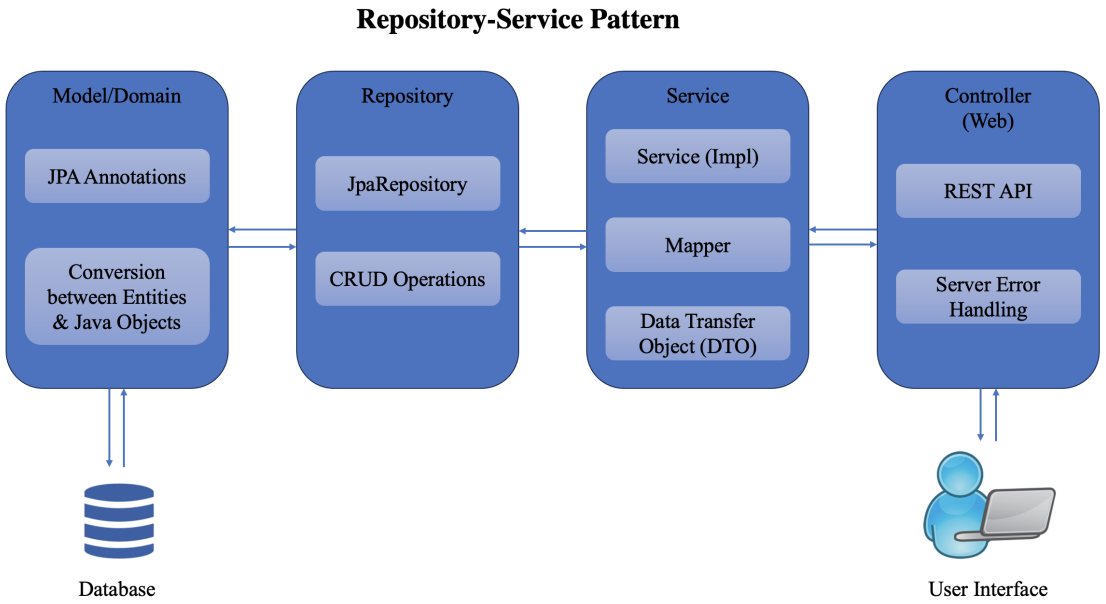


Figure 5.4: Repository-Service Pattern Diagram

The Repository-Service Pattern diagram seen in Figure 5.4 allows the web application to transform, query and apply intelligence to the data in order to generate an interface for the front-end.

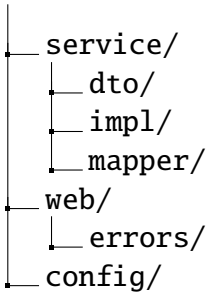
Firstly, the entities described in the database schema are modeled to Java objects through the Model/Domain layer and Spring JPA Annotations. Secondly, the Repository layer maps CRUD operations and queries to java methods, allowing the application to use methods to query data from the database. Thirdly, the Service layer contains the business logic of the back-end, coordinating calls to the repository and external APIs. Additionally, it maps the data to DTOs, making it more compact for optimal transmission over the network. Finally, the Controller or Web layer is responsible for processing incoming requests from the client, handling user input and responses.

The individual components are explained thoroughly in the next Section.

5.4.3 Code Structure

The folder structure of the back-end code is the following:

```
java/com/refood/trazabilidad
├── domain/
├── repository/
└── security/
```



domain

The ”domain” directory contains the model intelligence, mapping java-based objects to the database. Within the domain directory, all the entities and attributes in the database are directly mapped through JPA annotations to java objects. These java objects can be used to code the application intelligence, serving as the core representation of the data and forming the backbone of the application.

repository

The ”repository” directory bridges the application intelligence to database querying and CRUD operations. There is a repository file for every domain file, representing an entity from the database. From Spring Data JPA, the repository classes extend JpaRepository, making configured out of the box methods for CRUD operations available [28].

One option to query the data directly from the domain directory is to use the ”@Query” annotation from the Spring Data JPA, allowing to use SQL-like syntax to perform queries on the data. These queries are mapped to java methods and can be used throughout the back-end [29].

Another option is to use predefined methods within the JpaRepository class such as ”findAll()” or ”getById(ID id)”. The ”findAll()” method returns all records from the entity connected to the repository. Additionally, the ”getById(ID id)” method expects an ID as an input, and will return the record with the specified id.

security

The ”security” directory contains the security and authentication intelligence of the web application. The web application uses JWT authentication, providing each user with a Base64 encoded secret key. Java has a built-in Base64 coder and encoder, allowing to code and decode the token with a single method.

service

The ”service” directory contains the business-logic of the application. The intelligence that cannot be implemented in RESTfull endpoints or SQL queries is implemented

within the service layer. Additionally, the service bridges the gap between the database access layer and the web layer.

Within the service layer, the business logic is defined in a class for each entity. Nevertheless, it is also used to encapsulate the data to be transferred between the back-end layers and over the network. In order to achieve this efficiently, the following steps are taken.

1. Data Transfer Object (DTO) Directory
    - Efficiently encapsulate data that will be sent over the network, reducing the total amount of data that needs to be sent, optimizing the performance of the application.
2. Mapper Directory
    - Manages two methods, one for converting entities to DTOs and the other for converting DTOs back to entities.
3. Implement (Impl) Directory
    - Contains the implementation classes of the service layer, where the intelligence of the back-end is implemented.
    - The implementation is separated from the definition of the classes to facilitate the readability of the code.

web

Finally, the ”web” directory is the highest level of abstraction in the back-end, it also known as the controller. The web layer handles the incoming user requests and inputs, bridging them to the necessary classes and methods from the service layer. The interface for the front-end is designed through endpoints for each entity. The aggregation of all endpoints defines the RESTful API of the web application.

Red Hat states that, ”REST is a set of architectural constraints, not a protocol or a standard” [30]. In order for an API to be considered RESTful, it needs to adhere to the following criteria:

- An architectural framework consisting of clients, servers, and various resources, where HTTP is utilized to manage and control request transactions.
- Communication between the client and server is stateless, implying that no client data is retained between GET requests, and every request operates independently without any connection to others.

- Cacheable data that streamlines client-server interactions.
- A hierarchical system that arranges different types of servers—such as those handling security, load balancing and more—into layers for the efficient retrieval of requested data. This structured system is transparent to the client.

The Spring Controller structure adheres to the REST criteria. Additionally, the error handling classes are also defined within the web layer, sending the errors over the network and notifying the user in case of any unwanted behavior from the back-end of the web application. These errors are also known as server-side errors.

config

The ”config” directory contains java configuration files for the back-end described in Table 5.3.

File	Description
AsyncConfiguration.java	Configuration of asynchronous tasks such as threads.
CacheConfiguration.java	Configuration of caching within the application. Allows for storage of frequently used resources to optimize the performance of the application.
LiquibaseConfiguration.java	Configuration of destination of Liquibase logging and schema files. Changeset files are generated at the specified directory when a new JHipster entity is created.
LoggingConfiguration.java	Configuration of the Logging behavior of the application. When the application is running, logs are generated to study the behavior of API calls, ports being used, the date and time of actions taken by the application as well as unexpected behaviours and errors.
SecurityConfiguration.java	Configuration of authentication and endpoint filter chain. The password encoder, token provider and JWT (JSON Web Token) are configured.
WebConfiguration.java	Configuration of web-related settings, which handle HTTP requests and responses. These include CORS (Cross-Origin Resource Sharing) and HTTP message converters.

Table 5.3: Back-end Configuration Files

5.4.4 API Documentation

The API is documented through Swagger, a tool for API development under the Apache License 2.0 [31]. The API is the interface for the front-end of the application, the documentation is used to design front-end functionalities, mapping views and functionalities to certain endpoints.

Endpoints for Daily Operations

The main purpose of the API in the project is to access the data from the centralized database and input the daily donations and portioning of food to maintain the traceability. The most commonly used endpoints are explained in the following tables.

Endpoint	Method	Parameters	Response	Description
/api/al-ents/{id}	GET	id:Integer	Object	Returns row with specified id. Used to get a detailed view of a specific row.
/api/al-ents/{id}	PUT	id:Integer	HTTP Code	Modify all data of row with specified id. Used to override a row.
/api/al-ents/{id}	DELETE	id:Integer	HTTP Code	Deletes row with specified id. Used to delete a specific row by its id.
/api/al-ents/{id}	PATCH	id:Integer	HTTP code	Modify partial data of row with specified id. Used to partially update a specific row.
/api/al-ents	GET	None	Page (20)	Returns paginated list of rows. Each page is 20 rows long. Used to list entities in the web application.
/api/al-ents/	POST	None	HTTP code	Add a new row to the database. Used for saving data.
/api/al-ents-all/	GET	None	List	Used to list all rows at once. Used for select dropdowns.

Table 5.4: Endpoints: ”Al-sal” Entity

Endpoint	Method	Parameters	Response	Description
/api/authenticate	POST	Req. Body	NONE	Initiate authenticacion process. Takes username and password as Request Body. Returns 200 if authentication is correct.
/api/account/	POST	Req. Body	HTTP Code	A set of endpoints to reset and change the password. Returns 200 if the password is altered.
/api/register	POST	Req. Body	HTTP Code	Register a new user. Takes username and password as Request Body. Returns 201 if the user is created.
/api/admin/users	GET	None	Page (20)	Get a page of 20 rows of user information. The password is not part of the response.
/api/admin/users/	GET	None	Page (20)	Returns paginated list of rows. Each page is 20 rows long. Used to list rows in application.

Table 5.5: Endpoints: Other Resources

The shown endpoints in Table 5.4 are used to create, read, update and delete rows from the ”Alimento de Entrada” entity, which represents the received donations. The endpoints are repeated for all entities.

Additionally, the endpoints in Table 5.5 are used for user authentication purposes and account options such as change of username and password. In total, the API has 94 different endpoints.

5.5 Front-end Development

5.5.1 Available Technologies

The technologies used for the front-end are React, an open source JavaScript library that uses components to create reactive user interfaces. Additionally Typescript, a high level

language used to enhance the type safety in web application development. Even though React is a JavaScript library, it will be referred to as a framework to be compared to AngularJS and Vue.js, which are two distinct JavaScript frameworks.

JHipster supports three front-end frameworks: AngularJS, Vue.js and React, a comparison was drawn between them to choose the most optimal for the development of the web application. Additionally, the advantages and disadvantages of implementing the project with Typescript are also studied.

Typescript

Advantages	Disadvantages
Type Safety, offering static types that can catch errors during compiling time rather than runtime.	Steep learning curve with verbose annotation.
Enhances development experience providing advanced auto completion, navigation and refactoring.	Less flexibility due to its strictness with types and code structure.

Table 5.6: Typescript Comparison

React

Advantages	Disadvantages
Makes use of the Virtual DOM, increasing its performance and making it lightweight.	It is only a library, extra libraries and tools are needed for a complete development system if necessary.
Developed and backed by Facebook, constant updates and new developments and improvements.	It has unidirectional data flow, meaning data can only travel from parent components down to children components, not the other way around.
Flexibility to choose between many libraries and tools.	Typescript-compatible, but not integrated.
Component-based, allowing for reuse of components across views and functionalities enhancing flexibility and code structure. Increasing the robustness of the code due to less unpredictability stemming from large projects	
React is highly scalable, being currently used by popular companies such as Meta, Netflix and Airbnb.	
One of most popular JavaScript frameworks in the world with 210.000 stars on github	

Table 5.7: React Comparison

AngularJS

Advantages	Disadvantages
Typescript integrated and native to Angular.	Steep learning curve, being the most complex framework out of the three.
Developed and backed by Google, constant updates and new developments and improvements.	Use of Real DOM, reducing the performance and being slower than React and Vue.js
Two-way data binding is available, allowing components to share data be- tween them from child to parent and parent to child.	Less documentation, having a smaller community than React.
Component-based, allowing for reuse of components and modular growth of project.	
Highly scalable, being used by companies such as Google, Microsoft, IBM and Forbes.	
A complete framework, allowing for a full development environment with tools for routing, HTTP requests and more.	

Table 5.8: AngularJS Comparison

Vue.js

Advantages	Disadvantages
Small learning curve, considered the easiest framework of the three.	Steep learning curve, being the most complex framework out of the three.
Component-based, allowing for modular development of applications, en- hancing the robustness of larger projects.	Smallest community, with less documentation and support.
Use of Virtual DOM, enhancing the performance.	Over-flexibility, allowing for inefficient development and bottlenecks.
Like Angular, it is two-way data binding, allowing for bidirectional communication between components.	Not backed by a large enterprise, could lead to less updates and poor man- agement of the framework.
	Typescript-compatible but not integrated.

Table 5.9: Vue.js Comparison

React and Typescript were chosen for the development of the project due to the flexibility of React and the type safety of Typescript. Additionally, React is highly scalable and component-based, used by large enterprises and is the most popular and well-documented JavaScript framework.

5.5.2 React for Front-End Development

React is a component-based JavaScript framework, building applications by designing individual components and modules allowing the user interface to be controlled more efficiently and dynamically. React is used to build single-page applications (SPAs), dynamically loading the content from the web server and showing it to the user. On the other hand, multiple-page applications rely on the web browser to loading new web pages when the user navigates through the application. The advantage of SPAs is that they are more dynamic, the performance is increased and the responsiveness and feel of the application are improved over multiple-page applications.

In React, components have a lifecycle, from the moment they are rendered on screen to the moment they are dismounted, this is controlled by the state of the application. The lifecycle of the component enables a user to interact with the application, using the interactive components to use all functionalities. An example of how a component-based SPA is build can be seen in Figure 5.5.2.

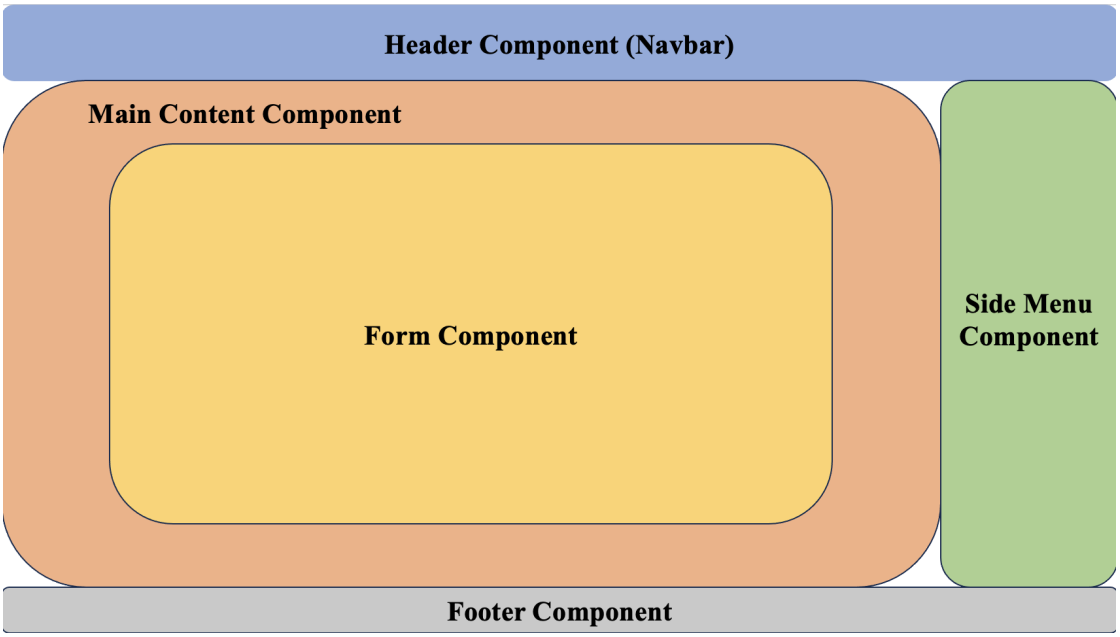


Figure 5.5: React Components

As seen in Figure 5.5.2, the example SPA has five components, displaying a common layout of a web application. Some components are common throughout all views of the application, while others are mounted and dismounted more often to enhance the experience of the user.

1. Header & Footer Components

   The header and footer components are seen in most web applications, used for travelling through the web application, showing menus, contact information and specific sections of interest. These components stay throughout the experience of the user and are not dismounted. However, they may be modified and adapted to the state of the application or different user authentication levels.

2. Side Menu Component

   The side menu component can be dismounted and mounted depending on the desired functionality. If the web application is showing a login form in the main content, the side menu will not be mounted. Nevertheless, if the user is in the main menu of the application, the side menu component may be shown in order to fit more information into the page.

3. Main Content Component

   The main content component is used to divide the web application. Components within the main content component are functionalities of the web application,

these may be form components or information about the organization. These components are meant to be interactive for the user and used to intake data and show relevant content. On the other hand, content outside of the main content component is used to move within the different sections of the application, they are navigation components.

4. Form Component

   The form component shows an example of a component within the main content component. A form component is to allow a user to input data into the web application, an example of this is a contact form. Components within the main content component are the focus of the web application, where the attention of the used should be placed.

In conclusion, component-based SPA front-end frameworks allow for a dynamic experience to the user, enhancing the latency and performance of the web application.

Components can be implemented in two different ways. Firstly, imported components from external libraries which have a known behavior, allowing developers to reuse these components throughout the application and reduce time in developing a personalized component. Secondly, components programmed by the developer to fit the desired use of the application.

The web application designed and implemented for Refood is a component-based SPA. The application implements both, personalized components and open-source external components in order to enhance the user experience and offer numerous functionalities.

State of the Application

In React, the state of the application represents the exact, unique combination of states from each component, the current state determines the view of the web application. The state is represented as a built-in React object and can be accessed and indexed to know the current state of the application or certain components. Whenever the state of a component changes, the component is re-rendered, allowing the component to change its form or content, depending on its programmed behavior.

The state is commonly managed component-wise, meaning that each component has its own state, and the combination of the states of all components make up an application state. Nevertheless, there are tools and libraries used to centralize the state, such as Redux [32].

Keeping the state in a centralized object offers numerous benefits. First, it makes the application predictable. When a web application grows in size, the state also grows, and the possible number of states within the application grows exponentially, the behaviour of the application can become unpredictable, leading to bugs and inefficiencies. Secondly,

Redux allows for enhanced debugging capabilities, keeping track of changes to the centralized state allowing to observe when the state change unexpectedly.

The web application implemented uses Redux to manage the state. The state contains most information needed within the application, such as the entities from the database and configuration state attributes to mount or unmount components. The states is persisted throughout the application, and allows for a reduced number of API calls, enhancing performance. Additionally, the state can be accessed from any component, bridging components more efficiently, being able to share information instantly, without needing to pass it through the component props every time. The centralized state helps overcome the problem of the unidirectional data flow, as a child component can now update the centralized state and the parent component can adjust.

Connection to the Back-end

To connect the front-end and the back-end, HTTP requests are made by the front-end to the back-end, allowing for the intelligent transfer of data from the server to the client. For this, Axios, a promise-based HTTP client for node.js is used [33].

Axios has a wide browser support, it is promise-based, leading to more more readable asynchronous code that is easier to maintain. It has automatic transformation of received data, dynamically transforming JSON responses into a JavaScript object. Additionally, Axios is provisioned with an error handler, when any response outside of 2XX is received, Axios handles it as an error and throws and exception, which can be handled. Finally, Axios has configurable global settings for handling API requests.

5.5.3 Folder Structure

The folder structure of the front-end code is the following:

```
webapp/
├── index.html
├── app/
│   ├── config/
│   ├── entities/
│   ├── modules/
│   ├── shared/
│   ├── app.tsx
│   ├── index.tsx
│   └── routes.tsx
└── content/
    └── images/
```

webapp

The ”webapp” directory is where all the front-end logic is located. As seen in the folder structure, the root index.html file is located within the webapp, this is the only HTML file in the web application, where all the components are mounted, maintaining the principle of an SPA.

app

The ”app” directory holds all configuration and Typescript files.

config

The ”config” directory contains configuration files for setting up the state with Redux, setting up the API calls with Axios, defining constants used throughout the application and configuring the middleware of the application, allowing for side effects to be run without blocking state updates.

entities

The ”entities” directory contains most content of the application. It contains all the front-end logic of the entities, the entities menu and the routing between the entity-related pages.

The structure within the entities directory is shown and the structure within one entity is explained in depth.

```
entities/
├── al-ent/
├── al-sal/
├── benef/
├── checkout/
├── donante/
├── intol/
├── nucleo/
├── p-benef/
├── socio/
├── tipo-al/
├── tupper/
├── voluntario/
├── menu.tsx
└── routes.tsx
```

First of all, the ”menu” file specifies the entities shown in the header component, and the ”routes” file specifies the routing of the web application.

Secondly, the directory of the entity ”al-ent” is used to explain the logic of the entity-related functionalities of the application, being the most commonly used in the daily operations of Refood.

There are four views for each entity:

1. List View

   The ”list view” of the entities allow Refood volunteers to view the data, for example, past donations. This allows volunteers to correct possible mistakes that were done during the input of the data. Nevertheless, the list view of certain entities, such as the Beneficiary or Volunteer List View is protected, and can only be seen by users with higher authority; protecting the information of individuals.

   The following image shows the list view of al-ent.

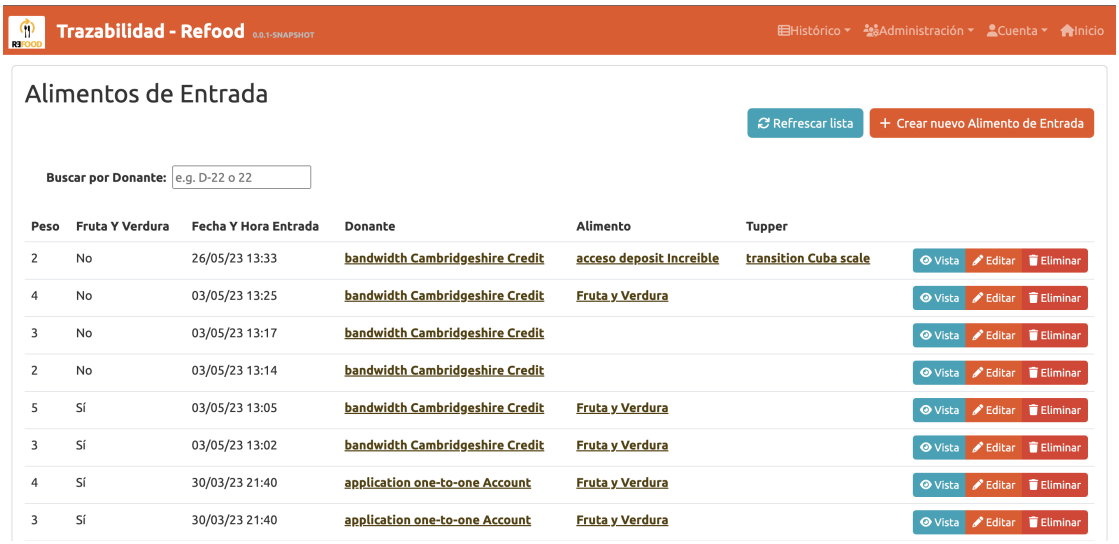


Figure 5.6: List View: Al-ent

2. Detailed View

   The ”detailed view” of a specific entity instance is used to expand the information of an entity and make it easy to read and identify.

   The following image shows the detailed view of an al-ent instance.

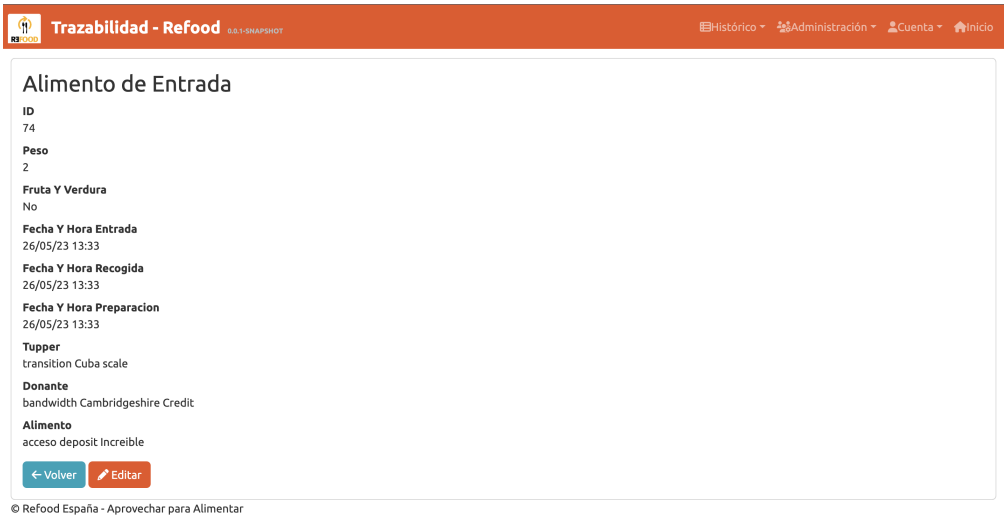


Figure 5.7: Detailed View: Al-ent

3. Create New Entity Instance View

   The ”create new entity instance” view contains an input form for the desired entity. Forms are personalized for each entity, facilitating the data input process and enhancing the efficiency of volunteers.

   The following image shows the ”create new entity instance” view for al-ent.

Figure 5.8: Create New Entity Instance View: Al-ent

4. Update Existing Entity Instance View

   The ”update existing entity instance” view contains an input form for an existing entity instance. The instance can be edited and saved accordingly, erasing the previous data and overriding it with the new input.

   The following image shows the ”update existing entity instance” view for al-ent.

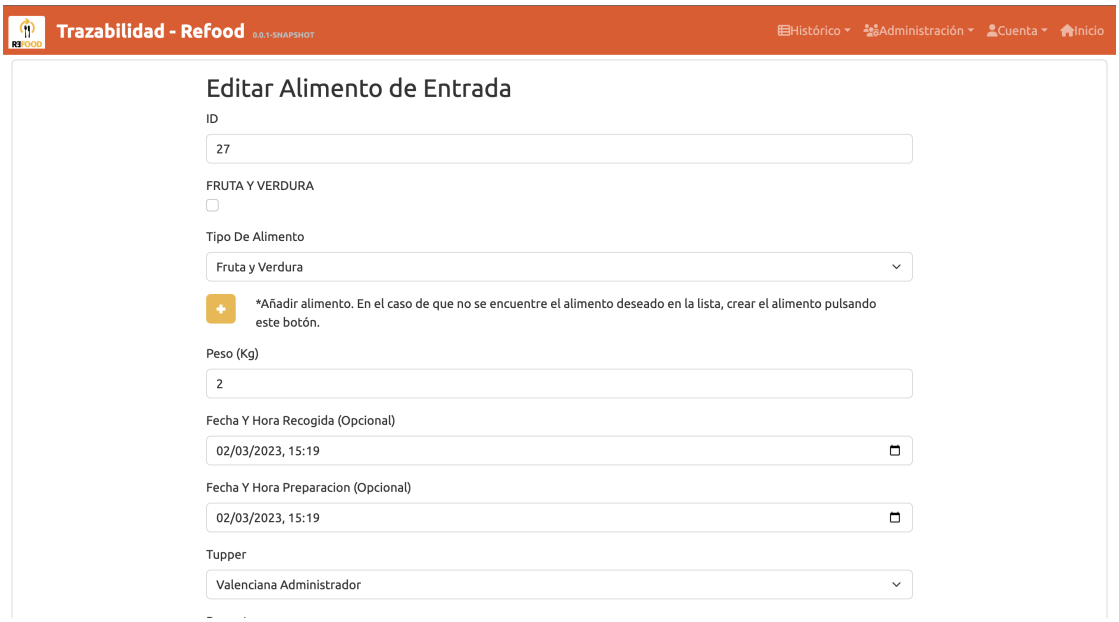


Figure 5.9: Update Existing Entity Instance View: Al-ent

The directory structure withing the ”al-ent” folder is the following:

```
al-ent/
├── al-ent-delete-dialog.tsx
├── al-ent-detail.tsx
├── al-ent-reducer.spec.ts
├── al-ent-update.tsx
├── al-ent.css
├── al-ent.reducer.ts
├── al-ent.tsx
├── index.tsx
├── SearchBar.tsx
└── tipo-al-modal.tsx
```

al-ent-delete-dialog.tsx

The file ”al-ent-delete-dialog.tsx” is a modal component, a modal is a pop up that shows over the other components. It is mounted when the delete button is pressed, an shows two options, to cancel or to confirm the action. It is dismounted when the action has been decided.

al-ent-detail.tsx

The file ”al-ent-detail.tsx” is the component for the Detailed View previously described. It is mounted when the ”detail” button is pressed, and dismounted with any other action from the user, as it is a static view with detailed information about an object.

al-ent-reducer.spec.ts

The file ”al-ent-reducer.spec.ts” is used to test the reducers of the Redux store. The store in Redux is where the central, complete state is stored. Additionally, reducers take the current state and an action, and return a new state. This is used to program the intelligence to the application, changing the state as a new action is performed.

al-ent-update.tsx

The file ”al-ent-update.tsx” holds the ”Update Existing Entity Instance” view. It is mounted when the button ”edit” assigned to a certain entity instance is pressed. It is unmounted when the instance is updated and saved, or when the operation is cancelled.

al-ent.css

The file ”al-ent.css” is used to style all ”al-ent”-specific components which are differentiated from other entities. There is a centralized CSS file for the generalized look and feel of the application, which applies to all shared components.

al-ent-reducer.tsx

The file ”al-ent-reducer.tsx” is where the Redux reducers are defined. Within this file, the back-end API is called using Axios, as it will be explained in Section 5.5.2. Then, the reducers are fed with the response from the API and the current state, so an action can be taken based on the two.

al-ent.tsx

The file ”al-ent.tsx” holds the ”Create new Entity Instance” view. It is mounted when a user needs to create a new al-ent instance, and there is a direct access from the main

menu, as this is one of the three views which are used during the daily operational process of Refood.

index.tsx

The file ”index.tsx” explicitly configures the routing intelligence within the al-ent entity, mounting each component depending on the current path. The general routing for all entities is the following.

Path	Mounted Component
/	Component ”AlEnt” is mounted, showing the List View.
/new	Component ”AlEntUpdate” is mounted, showing the ”Create New Entity Instance” View.
/:id	Component ”AlEntDetail” is mounted, showing the Detailed View.
/:id/edit	Component ”AlEntUpdate” is mounted, showing the ”Update Existing Entity Instance” View. This differs from the ”/new” path as in this case, the existing information for the specific instance is fed to the form, already showing the options filled in.
/:id/delete	Component ”AlEntDeleteDialog” is mounted, showing the modal component defined in the file ”al-ent-delete-dialog.tsx”

Table 5.10: Routing Entity Functionalities

SearchBar.tsx

The file ”SearchBar.tsx” is the search bar component used in the List View. It filters all the entity instances and returns the ones that match with the input in the search bar. The search bar compares the ”assigned ID” attribute for all entity instances, as these are the IDs assigned and known by volunteers. Additionally, they are primary keys in the entities with a List View used by volunteers, not the private List View for higher authority users.

tipo-al-modal.tsx

The file ”tipo-al-modal.tsx” is another modal component used to add new type of foods to the database. It has the ”Create New Entity Instance” view for the ”tipo-al” entity, being able to access it directly from the ”al-ent” entity when it is needed.

Each entity has a similar structure, with some entities having components for specific functionalities only available for them. Nevertheless, the views, the reducers and routing are implemented identically for all entities.

5.5.4 Client-Side Routing

Client-side routing allows the user to switch between different views or components when the user interacts with the application. For the project, React Router is used, allowing for a declarative and flexible style of programming [34]. React Router allows for authentication differentiation, using the component ”PrivateRoute”. This React Router component only allows specified user authorities to use specific links and routes, the defined authorities are Admin and User.

Client-side routing has the following benefits:

- Performance: Navigating between views and components once the initial page has been loaded is a lot faster, as the content does not need to be loaded from the server.
- Experience: Provides a more efficient and smooth routing throughout the application.
- Optimized Server Load: Because views do not need to be loaded from the server, the server load is reduced significantly. Consequently, server is liberated for API requests and other server needs.

The following table describes the general routing of the application as well as the authentication level needed to access the route.

Path	Authority	Mounted Component
/	Unregistered	The root path. Mounts the ”Home” Component, showing the main menu for an unregistered user.
/login	Unregistered	Login View. Mounts the ”Login” Component, allowing a user to login.
/logout	All	Logout View. Mounts the ”Logout” Component, allowing a registered user to logout.
/account/*	User, Admin	Account Management Functionalities. Two further routing options: /account/settings & /account/password. Both endpoints mount the ”Settings” and ”Password” Components respectively, which show the account management and configuration views.
/register	Unregistered	Register View. Mounts the ”Register” Component, showing the register form.
/activate	All	Confirmation of Activation. If a user has just registered an account and the result is successful, the application gives a confirmation and links to /login. If the result is unsuccessful, the application gives an error message and gives instructs to register again.
/reset	All	Password Reset Functionalities. Two further routing options: /reset/request & /reset/finish. Both endpoints mount components to request a password reset, ”PasswordResetInit” Component and the result of resetting the password, either an error or a success, ”PasswordResetFinish” Component.
/admin/*	Admin	Administration Functionalities. The administration routing is more extensive, utilizing six more routes: /usermanagement/*, /health, /metrix, /configuration, /logs and /docs.
/*	Admin,User	Entity Functionalities. All other routes which do not fall into the previous categories are entity routes. They have been described in Table 5.11
/*	All	Page Not Found. Any other routes that do not fall within the previous categories lead to ”Page Not Found” Component.

Table 5.11: General Routing

5.6 Software Testing

Web applications are complex systems with numerous possible actions taken by the user, of which all of them need to be handled so there are no errors or bugs. Testing is a process of evaluating the behavior of a software, checking that it does what it is supposed to do [35]. The process is optimized through the design of individual tests for each functionality. An individual test is a process in which an action is taken, comparing its outcome to the expected outcome, if both are the same, the test is successful, if not it is considered failed.

The International Software Testing Qualifications Board (ISTQB), the leading global certification scheme for software testing, defines the following types of testing [36]:

- Functional Testing

  Functional testing checks specific functionalities of the software, making sure they work as expected. Functional tests imitate user behaviors, covering scenarios that can be reached through the user interface. An example of a type of functional tests are End-to-End tests (e2e tests). In e2e tests the entire behavior of the application is tested, from the user interface to the connection to the database.

- Non-Functional Testing

  Non-functional testing encompasses aspects that are not related to user-specific actions, like performance testing. Performance testing is a testing method to determine the speed, and responsiveness of a software, independent to the user action.

- User Acceptance Testing

  User Acceptance testing is performed to determine if intended users accept the system. It is a formal testing method focusing on user needs, requirements and business processes. The results are evaluated and contrasted with the used acceptance criteria.

- Regression Testing

  Regression testing is done when changes to previously tested components and functionalities have been implemented. Therefore, making sure that the changes do not incur in new bugs, performance issues or untouched parts of the software have become affected, uncovering new failure points.

5.6.1 Testing the Web Application

For the scope of the project, the web application has been thoroughly tested from a functional aspect, focusing on designing e2e tests for all entities, settings, administration

and account management purposes. This is because, the primary focus is that no user generated errors and bugs should be allowed, interrupting the operational flow of the organization. Henceforth, leading to partial loss of data and volunteer dissatisfaction with the application.

The testing tool used is Cypress, improving the testing experience by testing the application visually in the browser. Additionally, allowing to build tests more efficiently through a extensive set of tools.

In total, one-hundred-and-fifteen e2e tests have been designed and implemented within the web application. The tests are divided into files, each file representing either a specific functionality of the application, or an entity.

In the following table, the result of the tests is shown. The first column represents the functionality or entity being tested, in the second shows the time to complete all tests within that file, the third is the total number of tests which ran from the file, the fourth column shows the number of tests passed and the fifth column shows the number of tests failed.

File	**Time**	**Nº of Tests**	**Passed**	**Failed**
Login	00:06	5	5	0
Password	00:09	6	6	0
Register	00:10	9	9	0
Settings	00:08	5	5	0
Administration	00:06	6	6	0
Al-ent	00:08	7	7	0
Al-sal	00:08	7	7	0
Benef	00:12	7	7	0
Checkout	00:08	7	7	0
Donante	00:11	7	7	0
Intol	00:07	7	7	0
Nucleo	00:10	7	7	0
P-benef	00:09	7	7	0
Socio	00:13	7	7	0
Tipo-al	00:07	7	7	0
Tupper	00:07	7	7	0
Voluntario	00:13	7	7	0
Total	02:41	115	115	0

Table 5.12: Test Results

First of all, the administration and account management menus have been tested, as well as the Login, Password Reset and Login functionalities. Tests for account

management and administration purposes are distinct, an example being that only an email in the correct format is accepted when registering a new account.

Nevertheless, as seen in table 5.12, each entity has seven tests, each one testing a different entity-related functionality. These functionalities are shared across all entities and are described in the following list:

1. Entity List View should be accessed through the Menu.

   Each Entity List View should be accessed through the navigation bar. Nevertheless, different authorities have different accesses, these authorized access are also tested for. Making sure that User authorities only have access to Al-ent, Al-sal and Checkout List View.

2. Create New Entity Instance View should be accessed from Entity List View.

   The Create New Entity Instance View needs to be accessed from the Entity List View. Additionally, the back button within the Create new Entity Instance View needs to lead back to the Entity List View. This is tested across all entities.

3. Details button should load Detailed View and be able to return to List View from there.

   The details button assigned to each entity instance within the Entity List View has to lead to the Detailed View. Additionally, the back button has to bring the user back to the Entity List View.

4. Edit button should load Update Existing Entity Instance View and be able to return to List View from there.

   The edit button assigned to each entity instance within the Entity List View has to lead to the Update Existing Entity Instance. Additionally, the back button has to bring the user back to the Entity List View.

5. Edit button should load Update Existing Entity Instance View and be able to save modified instances from there.

   The edit button assigned to each entity instance within the Entity List View needs to lead to the Update Existing Entity Instance. Additionally, the entity should be saved with the original data for the untouched attributes, and the newly updated attributes.

6. Delete button should delete entity instance.

   The delete button assigned to each entity instance within the Entity List View needs to delete the entity instance.

7. User should be able to create and save entity instance.

   The user needs to be able to create and save a new entity instance successfully.

Folder Structure

The folder structure within the ”test” directory is the following.

```
test/
└── javascript/
    └── cypress/
        ├── e2e/
        │   ├── account/
        │   ├── administration/
        │   └── entity/
        ├── support/
        └── tsconfig.json
```

The ”e2e” directory holds all cypress e2e tests for account management, administration menus and entities. Additionally, the ”support” directory holds constants and custom methods used throughout all tests. Finally, the tsconfig.json configures the cypress library, specifying the included files.

In conclusion, testing an application is considered ”good practice” before launching for production, identifying most bugs and errors before launching, improving the user experience and using less resources in the long-run for bug fixing.

5.7 Packaging the Web Application

Deploying an application to production mode requires a compiled and packaged web application, uploading this file or package to the production server. JHipster is compatible with two build automation tools, Maven and Gradle. In this project, Apache Maven is used to manage, run, compile and package the application [37].

Maven is based on the concept of project object model (POM), defined in Extensible Markup Language (XML) to manage the configuration and dependencies of the web application. For the deployment of the application to Amazon Web Services, the application needs to be packaged into a Web Application Resource file (WAR file), this is configured within the POM of the application.

The Maven command, ”mvn -Pprod package” is used to package and compile the application. The WAR file is generated within the ”target” folder in the general folder structure as seen in Section 5.3. Once the file WAR file has been correctly generated, the application can be deployed to AWS.

5.8 Deployment of the Application

The deployment of the web application has been done with CloudCaptain, an open source application deployment and management project [38]. The application is deployed in Amazon Web Services, using Amazon RDS and Amazon EC2 to host and manage the database and the web application respectively.

5.8.1 Initial Deployment

First of all, CloudCaptain needs to be given access to the deployment AWS account through the following steps:

1. Create a new IAM policy named ”boxfuse-policy”.

   A new Identity Access Management (IAM) policy is created to define the resources which will be accessible by the CloudCaptain role. For deployment of this project, the following JSON is defined:

```
{"Version": "2012-10-17","Statement":[
{"Sid":"allow","Effect":"Allow","Resource":["*"],"Action":[
    "ec2:*","ebs:*","kms:*","elasticloadbalancing:*","
   autoscaling:*","rds:*","cloudwatch:*","logs:*","route53:*","
   acm:*",
    "iam:*InstanceProfile*","iam:*Role*"]},
{"Sid":"s3AllowBucket","Effect":"Allow",
    "Action":["s3:CreateBucket","s3:ListAllMyBuckets","s3:
   GetBucketLocation"],"Resource":["arn:aws:s3:::*"]},
{"Sid":"s3AllowObject","Effect":"Allow","Action":["s3:*"],"
   Resource":["arn:aws:s3:::boxfuse-*"]},
{"Sid":"ec2Deny","Effect":"Deny",
    "Action":["ec2:*"],"Resource":["*"],"Condition":{"
   StringEquals":{"ec2:ResourceTag/boxfuse:ignore":"true"}}},
{"Sid":"rdsDeny","Effect":"Deny",
    "Action":["rds:*"],"Resource":["*"],"Condition":{"
   StringEquals":{"rds:db-tag/boxfuse:ignore":"true"}}}]}
```

   Code Listing 5.1: CloudCaptain IAM Policy

   The IAM policy is granting access to the following AWS services:

   - Amazon Elastic Compute Cloud (EC2).

     A service that allows users to rent virtual machines for their own purpose, such as hosting web applications.

   - Amazon Elastic Block Store (EBS).

     A service that offers block-level storage that can be attached to EC2 instances and is used by RDS for functionalities such as database snapshot creation.

- Amazon Encryption Cryptography Signing (KBS).

  A service that offers a centralized key management system for encryption or digital signatures.

- Amazon Relational Database Service (RDS).

  A service that simplifies the setup, operation and scaling of relational databases in the cloud.

- Amazon Elastic Load Balancer (ELB).

  A service that allows for the distribution of network traffic load to enhance the scalability of applications.

- Amazon Application Auto Scaling.

  A service for automating scaling of scalable individual AWS resources.

- Amazon Cloud Object Storage (S3).

  A service that provides object storage in the cloud through a web interface.

- Amazon CloudWatch.

  A service to monitor applications running on AWS, an in-house server, or even other cloud providers.

- Amazon Route53.

  A scalable Domain Name System (DNS) service.

- Amazon Certificate Manager (ACM).

  A service that provisions and manages SSL/TLS certificates within AWS resources.

Additionally, the IAM policy is also controlling access to the RDS and EC2 servies within the AWS account, to prevent the misconfiguration of CloudCaptain predefined settings and parameters.

2. Create a new IAM role named ”boxfuse-role”.

   The newly created role is used by CloudCaptain to deploy and manage the EC2 instance and the application. The role is attached to the previously created IAM policy, giving CloudCaptain access to the described services.

3. Select the AWS Region where CloudCaptain will deploy the application.

   The final step is to select the AWS region where the application is going to be deployed. The selected region is eu-west-1 (Ireland). This is because it is the closest to Madrid at the time of the deployment.

CloudCaptain is now configured to deploy the web application, once the account has been attached to the local CloudCaptain account, the following command is executed in the root directory of the application: ”boxfuse run -env=prod -db.type=none”.

The command is divided into three parts:

1. boxfuse run

   ”boxfuse” is the local CLI command of CloudCaptain, while ”run” is the subcommand that initiates a deployment in CloudCaptain.

2. -env=prod

   This is the first flag, telling CloudCaptain the environment that is being used for the deployment. Within the application there are two environments: ”dev” and ”prod”. Where, ”dev” is for development and and ”prod” is for production.

3. -db.type = none

   This is the second flag, specifying that the deployment should only be for the application, only a EC2 instance should be created where the application is hosted. The database is not deployed with CloudCaptain, as it restricts the flexibility of the deployment deleting the database every time the application needs to be updated.

   The RDS service is created manually and specified within the application properties of the application, so it connects when it is executed within the EC2.

At this stage, the deployment of the application is complete. The web application is hosted in an EC2 instance and connected to an database managed by the RDS. The architecture of the deployment in the cloud is shown in Figure 5.10.

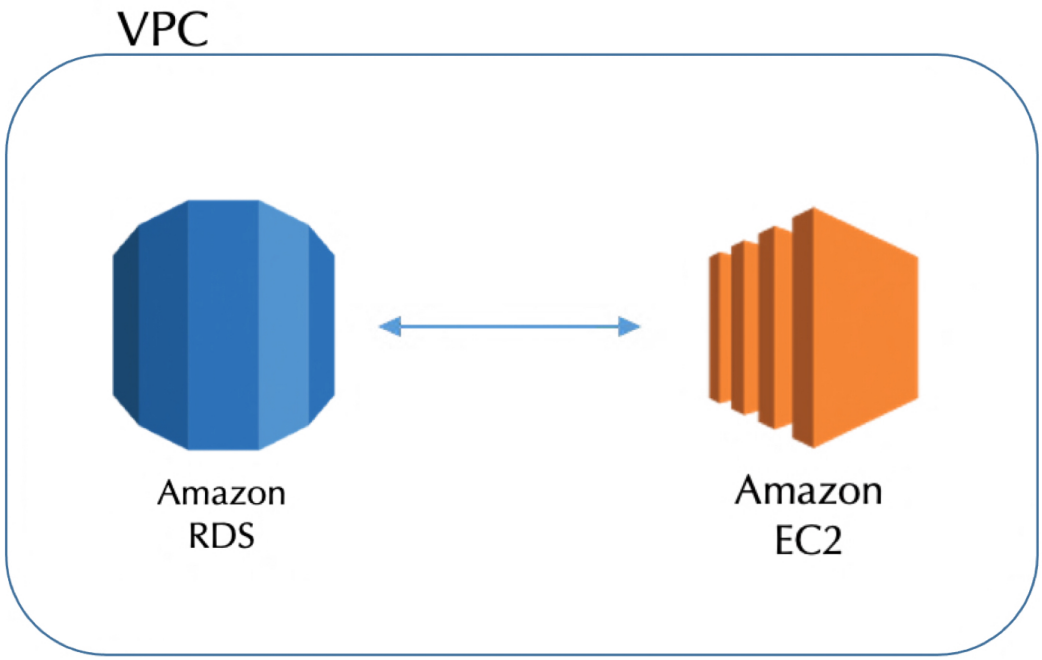


Figure 5.10: Initial Deployment Architecture in The Cloud

5.8.2 Scheduling the Database

Optimizing resources and being environmentally friendly are a big priority of the project. The application is used four days a week, from Monday to Thursday, for around three hours each day. While the application is not being used, it should not be available, reducing costs of deployment and reducing the carbon footprint of the web application. For this, the Relational Database Service is scheduled, only being available during operational hours, liberating space from the AWS servers, thus saving electricity, and reducing the incurred costs.

To achieve this, The Cloud deployment architecture is further enhanced by designing an automatic scheduler for the RDS database. This is done by using Amazon EventBridge Scheduler, and utilizing two lambda functions to switch the database on and off when necessary.

Amazon EventBridge Scheduler allows to manage, create and run scheduled tasks. Additionally, AWS Lambda is a compute service that is serverless, enabling the execution of code without the necessity for server provisioning or management.

The enhanced architecture is shown in Figure 5.11.

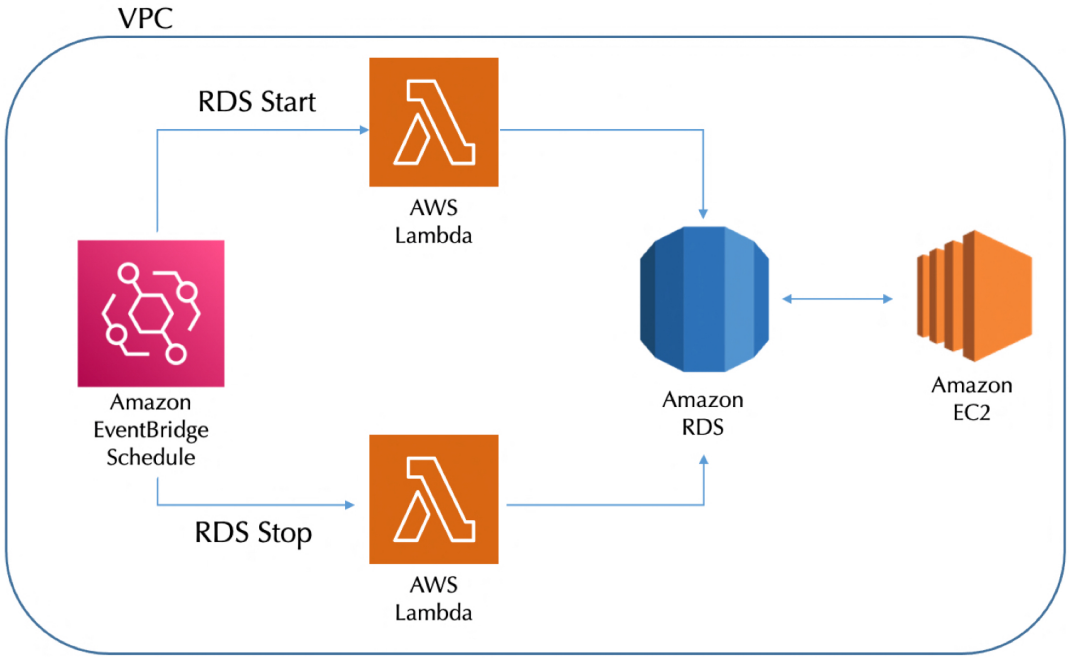


Figure 5.11: Enhanced Deployment Architecture in The Cloud

First of all, two lambda functions are created, one to start and another to stop the database. This does not restructure the database configuration nor deletes its data, it only makes it unavailable, reducing the computing power needed for AWS to maintain the RDS.

In order to connect the lambda functions to the RDS, environment variable are created and attached to the RDS. Next, a python script is written to allow the Lambda functions to connect to the RDS and either stop or start it [39].

```
1
2  import sys
3  import botocore
4  import boto3
5  from botocore.exceptions import ClientError
6  def lambda_handler(event, context):
7      rds = boto3.client('rds')
8      lambdaFunc = boto3.client('lambda')
9      print ('Trying to get Environment variable')
10     try:
11         funcResponse = lambdaFunc.get_function_configuration(
       FunctionName='RDSStartFunction')
12         DBinstance = funcResponse['Environment']['Variables']['
       DBInstanceName']
13         print ('Starting RDS service for DBInstance : ')
```

```
    except ClientError as e:
        print(e)
    try:
        response = rds.start_db_instance(DBInstanceIdentifier=
    DBinstance)
        print ('Success')
        return response
    except ClientError as e:
        print(e)
    response = client.list_dashboard_versions(AwsAccountId='
    926747041849', DashboardId='2222',MaxResults=10)
    return json.dumps(response, default=str)
```

Code Listing 5.2: Python Script for Lambda Start Function

```

import sys
import botocore
import boto3
from botocore.exceptions import ClientError
def lambda_handler(event, context):
    rds = boto3.client('rds')
    lambdaFunc = boto3.client('lambda')
    print ('Trying to get Environment variable')
    try:
        funcResponse = lambdaFunc.get_function_configuration(
    FunctionName='RDSStopFunction')
        DBinstance = funcResponse['Environment']['Variables']['
    DBInstanceName']
        print ('Stopping RDS service for DBInstance : ')
    except ClientError as e:
        print(e)
    try:
        response = rds.stop_db_instance(DBInstanceIdentifier=
    DBinstance)
        print ('Success')
        return response
    except ClientError as e:
        print(e)
    response = client.list_dashboard_versions(AwsAccountId='
    926747041849', DashboardId='2222',MaxResults=10)
    return json.dumps(response, default=str)
```

Code Listing 5.3: Python Script for Lambda Stop Function

Second of all, the EventBridge scheduler is configured to execute the Lambda functions with a certain cyclical schedule. This is defined through CRON expressions, used to define periodical events in software environments.

Lambda Function	CRON Expression
Start	30 17 ? 1-7,9-12 MON-THU *
Stop	30 23 ? 1-7,9-12 MON-THU *

Table 5.13: CRON Expressions for EventBridge Scheduler

CRON expressions follow a specific structure, separating each parameter by a space. Firstly, the first two parameters represent the minute and the hour at which the event is executed. Secondly, the third and fourth parameters are the days of the month and the months for which the event is scheduled. Thirdly, the fifth and sixth parameters represent the days of the week and the year(s).

As seen in Table 5.13, the Start and Stop functions only differ at the time at which they are executed. The Start Function is executed at 17:30 and the Stop Function at 23:30. The other fields are identical, specifying that the functions should be executed at any day of the month, every month except August, as Refood is not operational in August. Additionally, the cron expressions states that the functions should only be executed from Monday to Thursday, as they are the four days of the week for which Refood is operational. Finally, both expressions allow for all years.

With this, the RDS is scheduled to start and stop at the specified date and time, saving energy and monetary costs. This is the current architecture used in the production web application implemented in Refood.

Chapter 6

Data Visualization

6.1 Advantages of Data Visualization

- Simplifies Complex Data.

  Complex datasets can be simplified through visual aids, making them easier to understand. Complex behaviors, trends and patterns are difficult to identify in a numerical datasets.

- Facilitates Quicker Interpretation of Information.

  The human brain can understand information more rapidly through visual aids than in raw numerical form. Thus, data visualization allows organizations and individuals to interpret information more efficiently.

- Aids in Trend Identification.

  Information can be visually separated and categorized, making numerical data more comprehensible. Through clustering and color differentiation, trends can be identified and recognized more rapidly.

- Engages Audience & Facilitates Storytelling.

  To help attract more donors and volunteers to the organization, the visualization of landmarks and achievements are engaging and visually stimulating. Therefore, facilitating storytelling and attracting a higher number of people toward Refood.

Data visualization allows Refood to understand trends, and measure the total amount of food donated through visually engaging dashboards. Additionally, it enables Refood to quantify its contribution to reducing food waste and fighting environmental challenges, such as the reduction of carbon dioxide generation.

6.2 Technology Selection

Two technologies were analysed as candidate solutions for the data visualization system: Kibana and Logstash, or PowerBI. Firstly, enabling a dynamic data visualization system, Logstash [40] and Kibana [41] can be used to parse and transform data from various sources, achieved with Logstash and visualize the data with Kibana. Both of these technologies belong to The Apache Software Foundation [42]. On the other hand, PowerBI offers a free and powerful data visualization solution that can be connected to the RDS, generating interactive visualizations [12]. This allows for generations of scheduled reports that can be shared with volunteers and financial supporters, as well as with the donors.

In the approach taken within this project, the database is centralized, leaving only one source of data. Additionally, because Refood only operates two hours a day, and no insights need to be drawn within that time, the pipeline does not need to be analyzed in real time, being the strength of Logstash and Kibana. The objective regarding business intelligence is to generate reports on a fixed time interval, such as monthly reports, which can be done efficiently with PowerBI. The results of the data visualization solution are discussed in Section 6.3.

6.3 Creation of Dashboards

The creation of dashboards[1] is not automated, nevertheless, dashboards can be designed ahead of time. Consequently, they can be fed with the necessary data in order to generate them in minutes, directly from PowerBI. From these graphs and dashboards, reports can be generated in order to share the knowledge obtained with stakeholders, as well as creating engaging visualizations to transmit the power and influence of an organization to the public.

The initial stage of the creation of dashboards is their design and focus. First of all, dashboards have been designed to fulfil the general purpose of Refood, quantifying and visualizing the total data of donations, as well as generating donor-specific dashboards to create personalized reports. The design in data visualization is just as important as the data being shown, the information needs to be processed by the viewer in a short amount of time, making it simpler to comprehend.

6.3.1 General Dashboard

The first dashboard that is created is the General Dashboard. This is a collection of graphs that represent the work done by Refood as a whole, quantifying the daily intake,

[1] A dashboard is a collection of data visualizations, graphs and charts that together convey information about a project or organization.

the total amount of food donated, the donations done by each donor and the most common type of food donated. Finally, it also visualizes the amount of CO_2 emissions that have been avoided as a result of the prevented food waste. The conversion of kilogram of food waste to CO_2 emissions is shown in Appendix D.

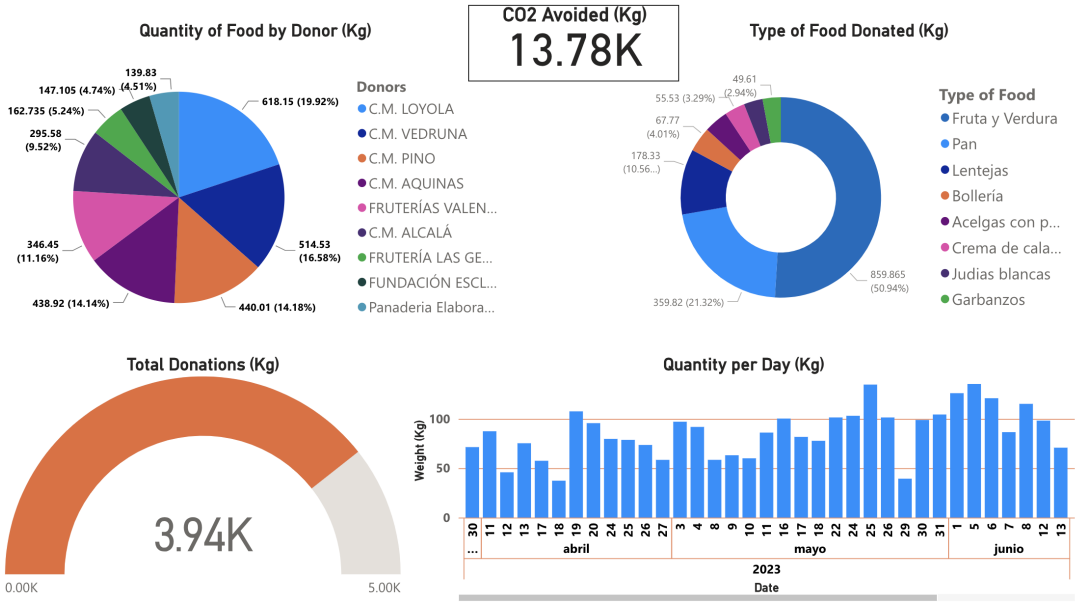


Figure 6.1: General Dashboard

The General Dashboard can be seen in Figure 6.1. Firstly, on the top left, a pie chart is used to show the quantity donated by each donor, the figure shows the top nine donors or donors above one-hundred-and-thirty kilograms of food donated. Secondly, on the top right, a ring graph is used to show the types of food donated by quantity, showing the top eight. Thirdly, on the bottom left a gauge graph is used to represent the total amount of food donated since the implementation of the web application. Fourthly, on the bottom right, a bar graph is used to visualize the evolution over time of the total donations made to Refood, each bar representing a day. Finally, at the top, the total prevented CO_2 emissions is shown, helping understand the environmental impact of Refood.

6.3.2 Top Donors Dashboard

The second dashboard that is created is the Top Donors Dashboard, seen in Figure 6.2. The objective of this dashboard is to visualize the donations done by the five collaborating student residences, which are the top donors since the start of Refood. This dashboard helps understand trends, as well as gaining insights into possible relationships

and similarities between them. A donor can be selected and the donor-specific data is highlighted within the two graphs on the right, as shown in Figure 6.3.

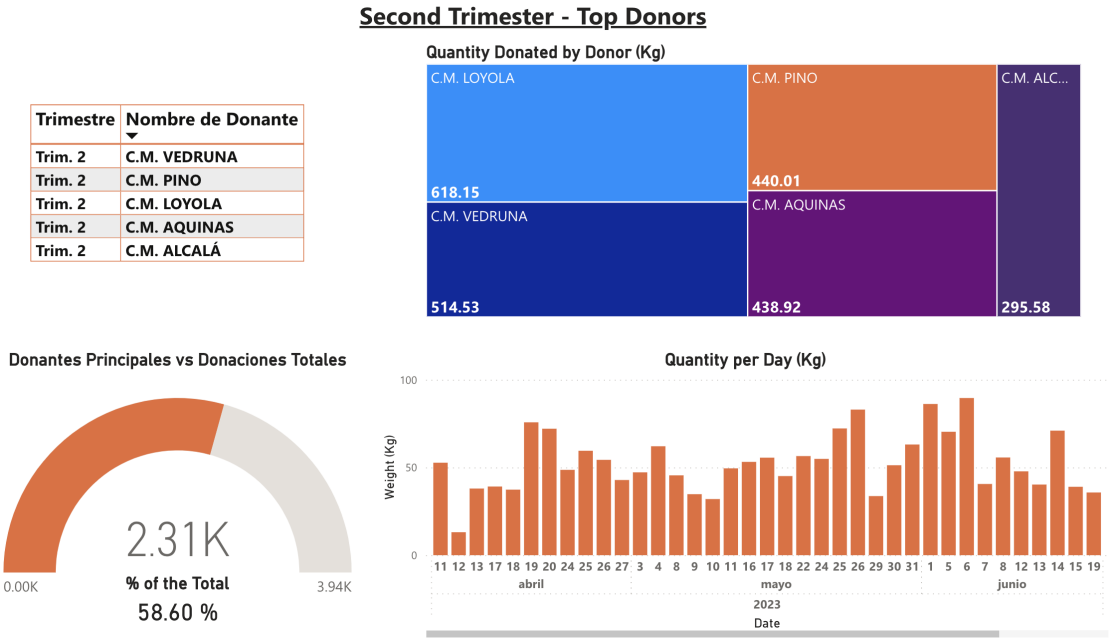


Figure 6.2: Top Donors Dashboard

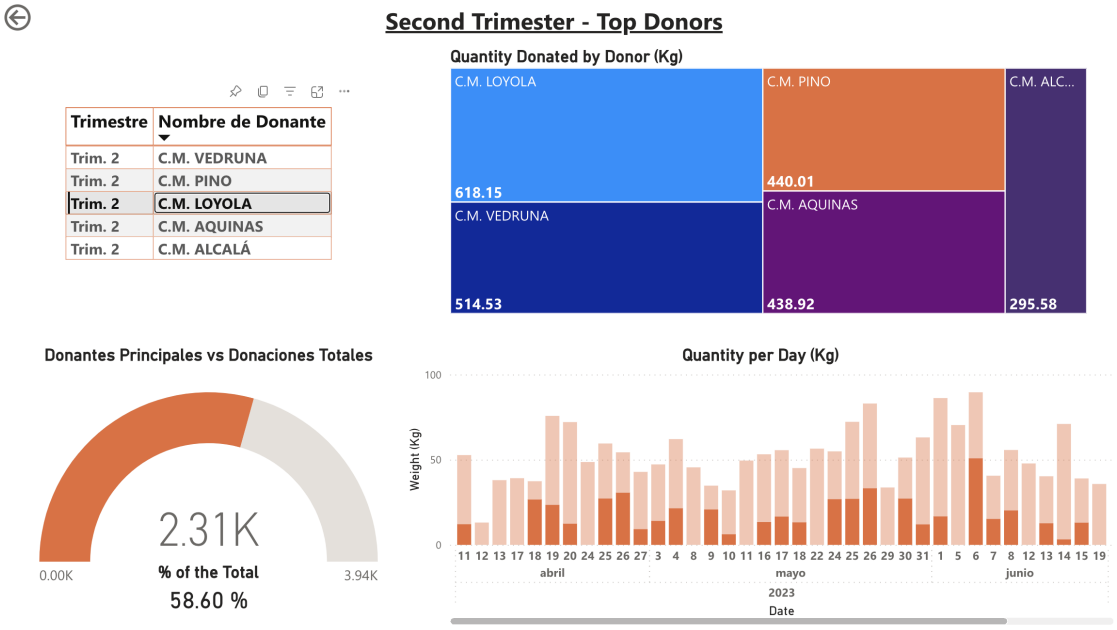


Figure 6.3: Interaction with Top Donors Dashboard

Firstly, on the top right, an interactive table shows the donors for each trimester. Secondly, the gauge graph on the bottom left shows the comparison of donations by the primary donors to the total donations made to Refood, calculating the percentage of the total. Thirdly, the top right the graph presents the quantity donated by each donor in a treemap. A treemap graph, represent the data hierarchically, decreasing the size of the rectangles based on the quantity. Finally, the bar graph on the bottom right shows the daily donations by the top donors.

6.3.3 Individual Donor Dashboard

Lastly, at the end of each year, individual reports are generated for the largest donors. The creation of reports for individual donors strengthens the partnership and allows donors to understand how their food production can be optimized to reduce waste. In order to achieve this, a Individual Donor Dashboard has been designed and generated.

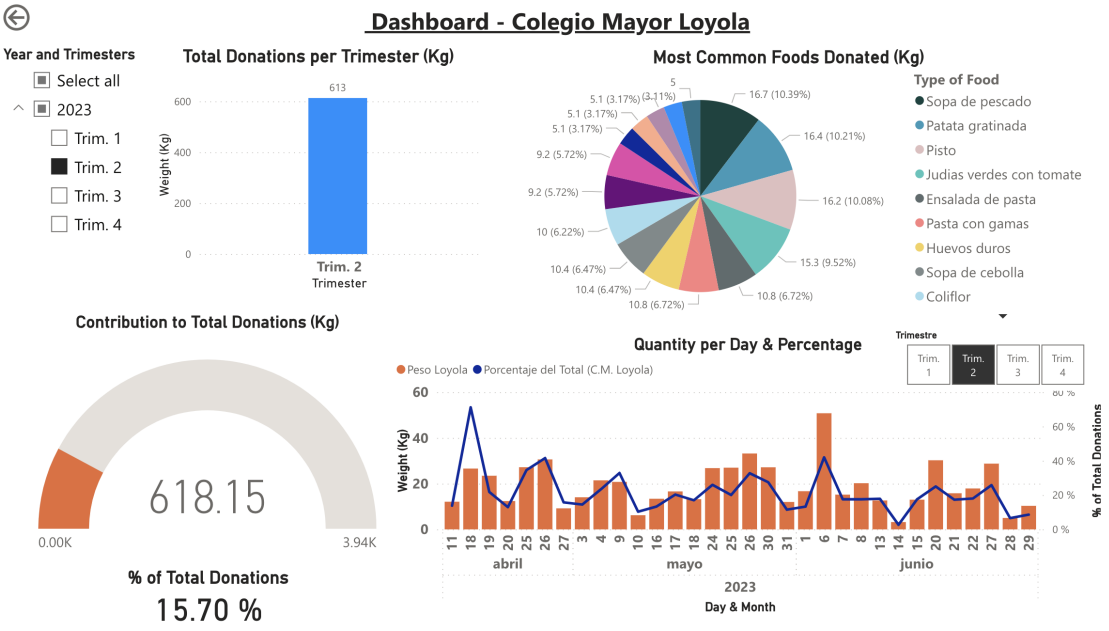


Figure 6.4: Individual Donor Dashboard

The Individual Donor Dashboard is the initial step to the creation of personalized reports for donors. Firstly, on the top left, a filter for the year and trimester is used to simplify the visualization, focusing on the desired period of time. On the right of the filter, a bar graph quantifies the total donations made for that period, per trimester. Secondly, on the top right of the dashboard, a pie chart is used to present the most commonly donated foods. Thirdly, on the bottom left, a gauge chart to sum the donated quantities by the donor throughout the selected trimesters. Additionally, offering a visual comparison to the total amount of donations received by Refood. Finally, on the bottom right, a bar and line graph where the bars represent the total weight of donations each day. In addition, the line denotes the percentage of total donations of which are a contributed by the donor, for the specific day.

Chapter 7

Results

7.1 Introduction

The result of the project constitutes the developed web application, the feedback obtained after the implementation of the application and the data visualization dashboards generated from the solution. After the results have been analyzed, the impact of the project and achievements related to the primary objectives are assessed. Finally, a summary of the results of the project is presented, focusing on the challenges, achievements and highlights.

7.2 Web Application Demonstration

In order to present the developed web application, a demonstration of the use of the web application during the operational daily process of Refood is presented. The following steps is what the volunteer in charge of the traceability does during a day at Refood.

7.2.1 Landing Page.

The landing page is the first thing a user sees when entering the public link of the web application.

Figure 7.1: Landing Page and Login

A user can log in directly from the landing page in order to access the resources needed to begin the traceability process.

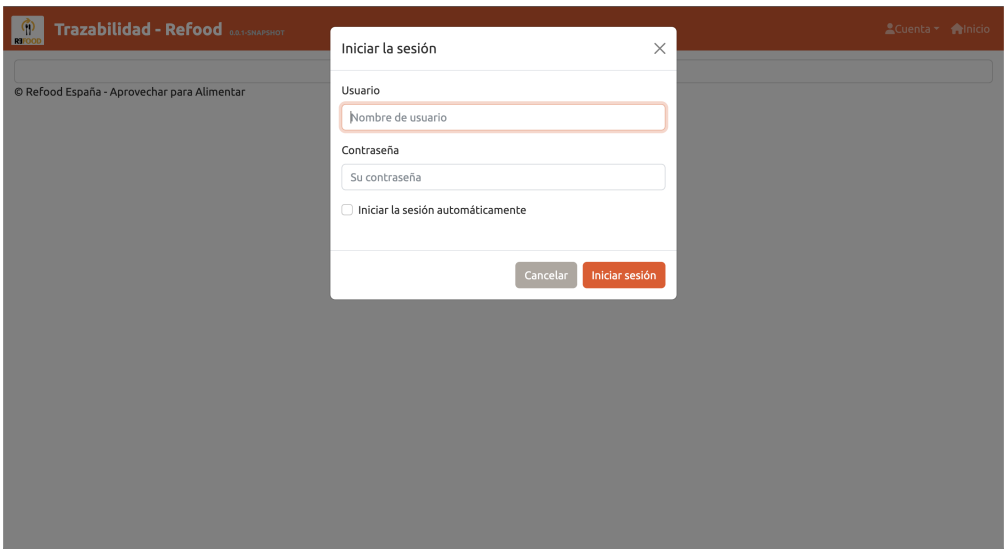


Figure 7.2: Login Questionnaire

The login form seen in Figure 7.2 demands a username and a password. Once they are entered correctly, the user has access to the main menu and all other accessible resources depending on the authority level of the account. In this example, a user authority is

used, restricting access to personal and private data, as well as the configuration and administration menus.

7.2.2 Main Menu.

The main menu offers routing to all available resources, either from the main content or the menus within the navigation bar.

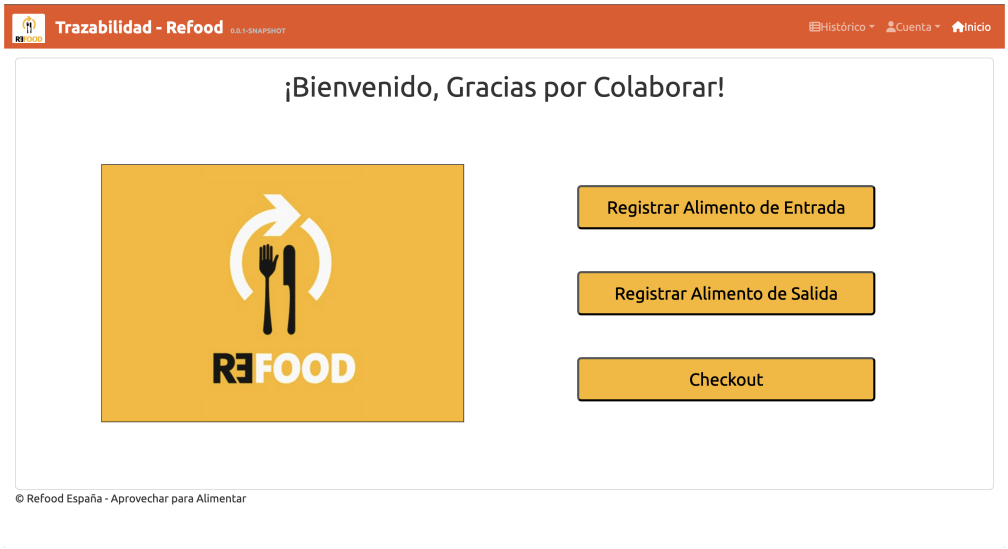


Figure 7.3: Main Menu

As seen in Figure 7.3, the three core functionalities of the application are accessible from the main content, allowing more efficient access to the forms to save the data. The first step is to save an incoming donation within the ”Registrar Alimento de Entrada” form.

7.2.3 Save Incoming Donation.

First of all, when an incoming donation arrives at Refood, it is weighed and saved into the database as an ”Alimento de Entrada”, translating to incoming food. This is done through the ”Registrar Alimento de Entrada” form. In this example, a fictional donation of five kilograms of Caesar Salad is received from the donor ”C.M. Loyola” and inputted into the form.

Figure 7.4: Received Donation Form

7.2.4 Outgoing Donations

Continuing the received donation, the volunteers partition the salad into two ration, which are given to two fictional beneficiaries, Julian and Juan.

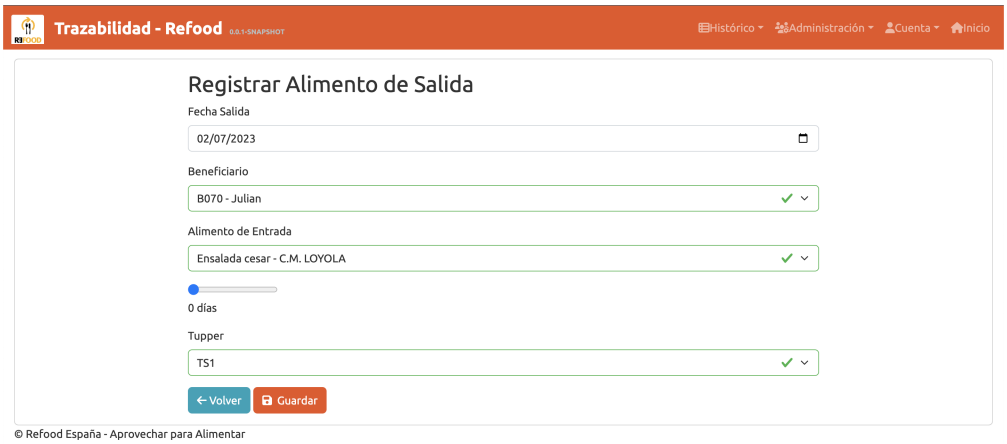


Figure 7.5: Outgoing Food Form

As seen in Figure 7.5, the beneficiary and the previously registered donation are specified in the Outgoing Food Form or ”Registrar Alimento de Salida” form, this way,

the outgoing food can be traced back to the initial donation. The same form is filled in for the second beneficiary, Juan.

7.2.5 Checkout

Finally, in order to finalize the process, when all outgoing foods for the beneficiary have been registered, a checkout is created. The checkout form specifies the total weight of the donated package to the beneficiary, as well as the outgoing foods which are registered to the beneficiary for that day.

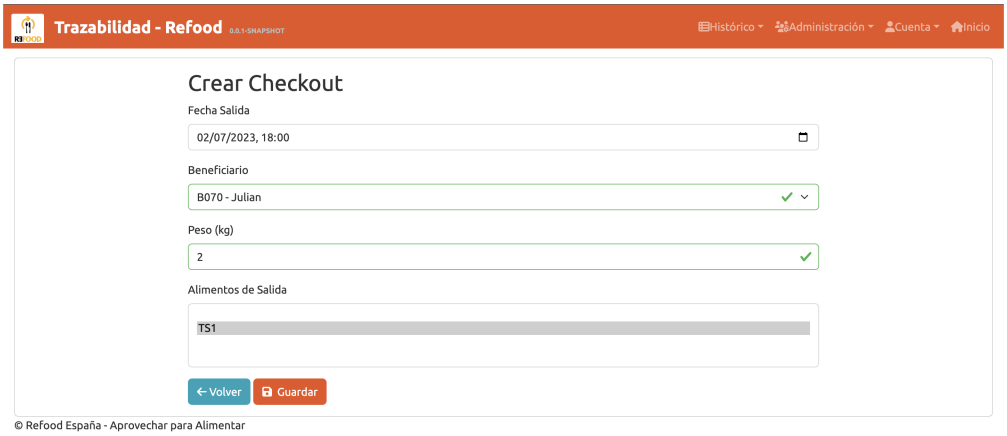


Figure 7.6: Checkout Form: First Beneficiary

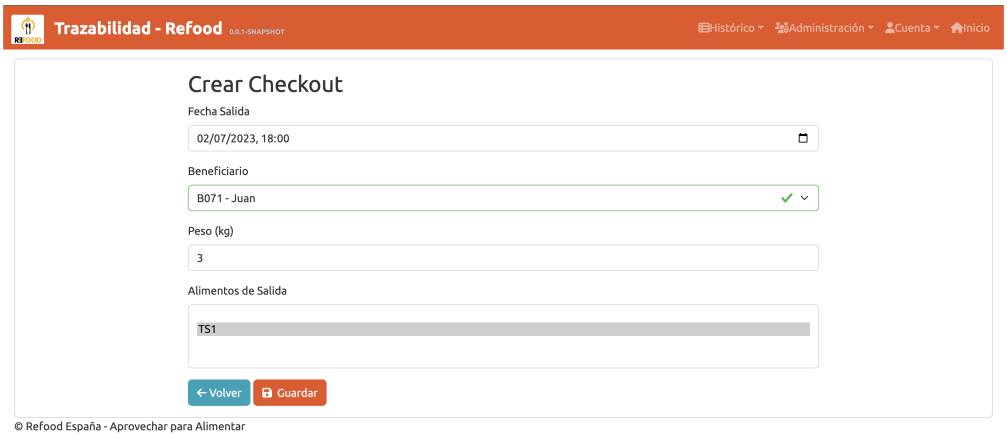


Figure 7.7: Checkout Form: Second Beneficiary

In Figures 7.6 and 7.7, the checkout form can be observed. As anticipated, the partitions are not even (3kg and 2kg), but the total weight of both donations is equal

to the weight of the incoming donation (5kg). Once the checkout forms are saved, the process for both beneficiaries is completed, and the traceability of the donations is maintained throughout.

7.3 Survey and Feedback

Currently, the web application has been implemented in Refood for the past three months, being used on the daily by volunteers, having registered over one-thousand incoming donations and over four-thousand portions given out. Consequently, the users have tested the application and have been able to provide feedback, remarking points of improvement, as well as additional ideas for enhanced user experience. The survey is presented in Appendix E and the responses are presented in Appendix F.

As mentioned in Section 4.6, a survey is sent to the volunteers responsible of record-keeping through in order to understand the performance and impact of the application, as well as the general impression. There are nine respondents, some of the main questions and responses in the survey are discussed.

- In comparison with the previous system, how efficient is the new one?

  An objective mentioned in Section 4.1.1, is to improve the efficiency of the daily operations of Refood. Shortening the time it takes volunteers to maintain the traceability of the donation, and distributing the food as rapidly as possible to maintain the cold chain of perishable goods. Understanding if the efficiency has been improved based on user experience aids to understand if the objective is being achieved.

  Out of nine respondents, three of them were not a part of Refood before the implementation of the new system, five of them said it was ”very efficient” and one said that it is ”a little bit more efficient”. None of them responded with ”less efficient” or ”equally efficient”, showing that the application has had a positive impact on efficiency.

- How easy to use is the new web application? (1 being very complicated, 5 being very simple?

  Another primary objective of the project is to decrease the complexity of the data intake process. Refood is dependent on voluntary workers, because of this, the volunteers operate based on availability, leading to a high fluctuation of volunteers on a day-to-day basis. Therefore, incoming volunteers need to be trained on how to use the traceability system and experienced volunteers may not always be available. Maintaining a simple and intuitive process will lessen the necessity for extensive training of incoming volunteers, as the web application will be simple to use.

The responses to the survey question make an average 3.89 out of 5. This is not ideal, nevertheless is a strong starting point as most users believe that the software is intuitive and simple, with only one users rating it a two out of five.

- What would you improve about the new traceability software?

  First of all, the user that rated the simplicity of the application with a two out of five is taken into consideration, understanding the reasons for the perceived complexity of the system. The user suggested the following:

  - The step of weighing the food can be simplified avoiding the extra step of the checkout form.
  - Data can be presented in a more visual way. An example of this being the name of the containers could be more intuitive.
  - Some functionalities could be optimized and automatized. Such as when selecting the type of food, rather than looking through a list, the user suggests having a text input that auto fills to the closest match.

  These suggestions are all taken into account and if possible will be implemented to improve the user experience and intuitiveness of the web application.

  Second of all, other suggestions are also noted, with some general discontent over the state of keeping record of container information.

  The final question asks how the users would rate the web application on a scale of one to five.

- In general, how do you like the traceability software? (1 being the least and five being the most)

  The result is very positive, with seven users rating it with a five, and another two users rating it with a four. The web application has been very well received within the organization, providing a dedicated and intuitive way to keep record of donations that most volunteers can understand and use. It is visually pleasing, reactive and very fast, making the user experience very enjoyable and modernizing the organization, improving the quality of the data.

7.4 Impact Assessment

The impact of the project on Refood is analyzed through its accomplishment of the four primary objectives.

7.4.1 Objective 1

First of all, maintaining the safety of the beneficiaries is a priority to Refood and this project. Designing a traceable system enables Refood to make certain where food is coming from, in case of any health issues stemming from the donations. Therefore, ensuring the health of beneficiaries and the trust in the organization.

The software has been used for three months, not producing any errors, allowing volunteers to freely keep record of donations and not lose any data. The data is centralized in a secure database on the cloud, which allows for easy access, as well as privacy and security of private information.

Finally, indexing and visualizing the data has been simplified due to the efficient and scalable database design, allowing Refood personnel to fetch any information necessary. This may be for a government institution or for the public media, granting Refood with the ability to ingest, present and visualize its data and highlight its achievements.

7.4.2 Objective 2

Second of all, ensuring an efficient and less complex system to keep record of donations is a constituent of the primary objectives of the project.

On one hand, the preceding software did not allow for efficient, real-time data acquisition, slowing down the operations. The daily operations would start around forty minutes before the first input in the database, this is because the data would be first written down on a physical piece of paper, and then copied into the system. On the other hand, With the current software, the first input happens the first minute of operation, as incoming donations are weighed and saved into the database before they are distributed, ensuring the veracity and integrity of the data. Additionally, with the web application, a volunteer does not need to stay an extra thirty minutes to input the daily information into the software, as this can be done dynamically during the distribution and donation of food to beneficiaries.

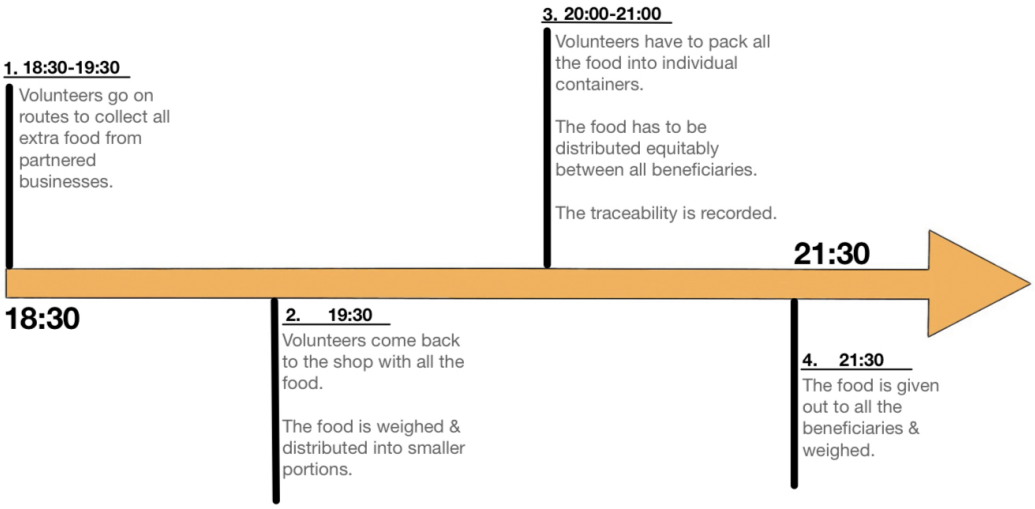


Figure 7.8: Optimized Operational Process Timeline

As seen in Figure 7.8, the process is significantly more dynamic and allows volunteers to keep records of the data in real time. The web application has proven efficient, being rated with a 4.22 out of 5 in efficiency, as recorded in the feedback survey sent out to users. This has also impacted the time that volunteers have to stay in the shop, reducing it by an average of twenty minutes, as no extra labour needs to be put into copying the information from the physical paper to the system, unlike with the preceding software solution.

Nevertheless, the system ought to be less complex, allowing new volunteers to learn it without an extended recap or lesson. Making simple-to-use software is increasingly difficult with a large variation in the user pool, with users ranging in age, education and background, as well as technological fluency. Nevertheless, according to the survey, the software is intuitive, and can be used by new volunteers after being shown an example of how to record an incoming donation. However, there are still improvements to be made on the complexity of the software, making it more accessible to older volunteers with less digital knowledge and worse eyesight, making the compact, extensive lists and fields hard to visualize for them.

7.4.3 Objective 3

Third of all, designing and implementing a robust software solution that enhances data validation is part of the primary objectives.

The software is robust when it enables volunteers to efficiently save the necessary data to the database, without incurring in errors or bugs within the common functionalities or abnormal uses of the application. Therefore, enhancing the user experience and not

losing any data during the daily operational process.

Additionally, strengthened data validation increases the veracity and validity of the data. Enabling users to input only correct information, not allowing for typing mistakes or errors in the data input process. The database becomes more accessible and no information is lost when processing and visualizing the data, as fields and inputs are homogeneous and follow a predictable structure predefined by the software.

7.4.4 Objective 4

Finally, implementing a data visualization solution is the final component of the primary objectives. The solution is used to generate reports to advertise the organization and provide donors with trends and behaviours to better understand their food waste patterns.

Data visualization enables Refood to understand trends in quantity donated, as well as donor-specific trends. Therefore, Refood can anticipate to days in which a regular donor does not make a donation, or days in which a large quantity of food is expected and extra volunteers are needed.

Optimizing the resources used by volunteers to travel to donors which do not have food on a specific day is highly efficient, decreasing the time it takes a volunteer to complete a route. Additionally, adapting to busy and not so busy days enables the organization to only have the necessary amount of volunteers, not lacking nor exceeding the optimal number, which would result in dissatisfaction from the volunteers.

Additionally, empowers Refood to enhance its social media presence, attracting more volunteers and donors. Thus, growing its impact by increasing the number of donors and beneficiaries. Therefore, leading to a larger contributions to CO_2 reduction and food waste prevention.

7.5 Future Developments

7.5.1 Reduce Complexity in the User Interface

Through months of testing and the feedback received, the short-run future development is the optimization of the User Interface. It is a key element of the core objectives to reduce the complexity of the system and reduce the stress from needing an experienced volunteer on site.

Various design elements will be modified, Tupper instance names, larger fonts and clear forms, are part of the feedback received in the survey. Additionally the automation of data such as Checkouts completion will be implemented, allowing volunteers to complete all checkouts with the click of a button.

Additionally, drop-down elements are getting larger every day, as new donors and types of food are added, increasing the time to find the desired elements. Therefore,

to enhance the efficiency, drop-downs will be changed to auto-fill text input elements, where the user can write down the initial characters of the item they are looking for, getting the closest match as a result.

Continuous developments and improvements will be done to the User Interface in order to adapt the application to the needs of the users.

7.5.2 Scale Up to Multiple Refood Locations

Refood is growing, a new location has opened up in Madrid and there are more to come. In order to cope with the increasing size, the application has been designed with the capability of scaling up to various users at once, giving service to various Refood locations at the same time, while only giving access to their respective data, their donors and their donations.

In order to do this, the API requests need to be optimized, to reduce the traffic load in the server. Right now around four-thousand entries are fetched at once, in the future, the fetch should only be done for the required instances every time. This will cause the front end to make many more HTTP requests, nevertheless, they will be multiplexed in time, reducing peak traffic loads and bottlenecks.

Additionally, new user authorities need to be created for each location, differentiating them by the ”Nuclei” entity and blocking inaccessible data in the front-end and back-end validation. The front-end validation comprehends limiting the HTTP requests to only specific data. The back-end validation makes sure that the authority within specified within the HTTP request has access to the requested data, this is specified within the business intelligence of the back-end, in the service layer.

7.5.3 Integration of Data Visualization

The integration of real-time dashboards within the web application will enhance the user experience and motivate volunteers, visualizing their impact every day. The automatic dashboards should be accessible through the web application and embedded directly, showing updates of the registered foods. Therefore, ensuring the proper functioning of the application as well as serving as a validation for the data, to check if the input has been done correctly.

Additionally, at the end of each day, the embedded real-time dashboards will be connected to the social media of Refood, being one click away from publishing engaging posts about the results for the day, encouraging more volunteers and donors to contribute and work with Refood.

7.5.4 Enhanced Security in the Cloud

The cloud deployment will be thoroughly studied to optimize the resources even more, not only scheduling the database but also the web application hosted in the EC2 instance.

Additionally, the Amazon Virtual Private Cloud will be divided into subnets to control access to the individual resources, such as the database and the host server port.

In order to increase the security of the database the following measures will be taken:

- Security Groups Security groups are used within AWS VPCs through subnets. Security groups are assigned to existing subnets, serving as a virtual firewall for incoming and outgoing traffic.
- Network Access Control Lists (NACLs) NACLs are defined ranges of IP adresses as well as specific IP address for which inbound and/or outbound traffic is allowed, acting as firewall for all IPs not specified in the Network Access Control Lists.
- VPC Endpoints VPC endpoints allow for private connectivity between AWS resources, such as the EC2 instance and the RDS instance. The control is done within the VPC without using a VPN.

7.5.5 Machine Learning Algorithms for Prediction of Donations

With the new database schema, data can be exploited for deeper insights into the behaviors of collaborating entities, as well as trends in the quantities donates, being able to categorize donors daily, to predict whether they will donate food that day or not. The machine learning categorization algorithm will serve the purpose of notifying those entities with less chance of donating on a specific day to make sure that they will be donating. If not, the route will be modified to reduce the time it takes volunteers to collect the food.

Additionally, machine learning algorithms can be used to predict the type of foods that are donated by each donor, suggesting at the top of the drop-down the predicted type of foods that will be donated, decreasing the time it takes to collect the data.

Finally, with the collected data, the expected total quantity of donations can be predicted, allowing Refood to adapt the number of volunteers that are recruited for each day to the amount of food expected, also optimizing the work of volunteers and enhancing their experience with the organization, not over committing volunteers on days where few donations are received, wasting their time.

7.5.6 Creation of Mobile App for Volunteers

A creation of a mobile app for volunteers is planned to be developed, this mobile app will replace the current system for signing up to work on a specific day, which is done through excel forms sent in Whatsapp groups.

This mobile application will be used to sign up and keep track of who will be showing up what day, keeping track of the current number of volunteers signed up. Therefore, limiting overbooking and controlling the usual responsibilities of volunteers to have a diversity of knowledge within the group, aiming at having one volunteer with knowledge about the traceability software, other volunteers which commonly portion the food, and more volunteers which commonly go on routes to pick up the food.

Additionally, events, news and important notifications will be sent through the app, informing volunteers of any special days, days where Refood is closed, the general Refood schedule, as well as abnormal news and notifications.

Finally, the app will be used to manage the process of becoming a volunteer such as registering their information and handing in the certificate of food handler or signing up to do it, which is a necessary step for any volunteer in Refood due to strict regulations within Madrid [43].

7.6 Summary of Results

To conclude, the project entails the design and implementation of a database and the design, code, deployment and implementation of a web application for Refood. Additionally, a data visualization solution has been implemented to utilize the new data structure enhance the business intelligence of Refood.

The four primary objectives of the project have been clearly defined and achieved:

- A software for tracking the traceability throughout the life cycle of incoming and outgoing donations, has been designed, programmed and implemented, complying with regulations and ensuring the safety of beneficiaries.
- The efficiency of the daily operations of Refood is enhanced, ensuring the veracity and validity of the data as volunteers can record data in real time. Additionally, the complexity of the process has been reduced, allowing volunteers to adapt and learn the software in a short period of time.
- The implemented solution is robust, having been implemented for three months and not producing errors. Additionally, the data in the database is checked and validated, ensuring its integrity and validity.
- A data visualization solution has been designed and implemented, enabling Refood to generate visual dashboards and reports. Therefore, enhancing the experience and insights of donors and launching its social media present and advertising campaigns. Additionally, allowing Refood to detect trends and predict behaviors of donors more efficiently and accurately.

Nevertheless, there is room for improvement regarding the complexity of the solution, a more simplistic design can be accomplished and is planned for future updates of the software. Therefore, allowing a wider range of volunteers to use the software, enhancing the flexibility of Refood for days without available experienced volunteers.

Additionally, the technology stack used is open-source and/or free, except the deployment in the cloud. This allows for the continuous use of the solution and future improvements, as the technologies are financed and supported by large organizations, commonly updating and improving them.

Finally, the feedback of volunteers and the board of Refood is utmost positive. The software has been adopted seamlessly and the dashboards and reports are being designed to be sent out to donors and to be published in the social media and website of the organization. The increased efficiency and ease of volunteering in the shop is observed. Leading to a reduced time from not having to stay in the shop to finish saving the data and the increased user satisfaction seen from the survey responses. The full measure of achievements demonstrates an extremely positive outcome from the project.

The following figures are a selection of images from Refood, where the volunteers and application can be seen.

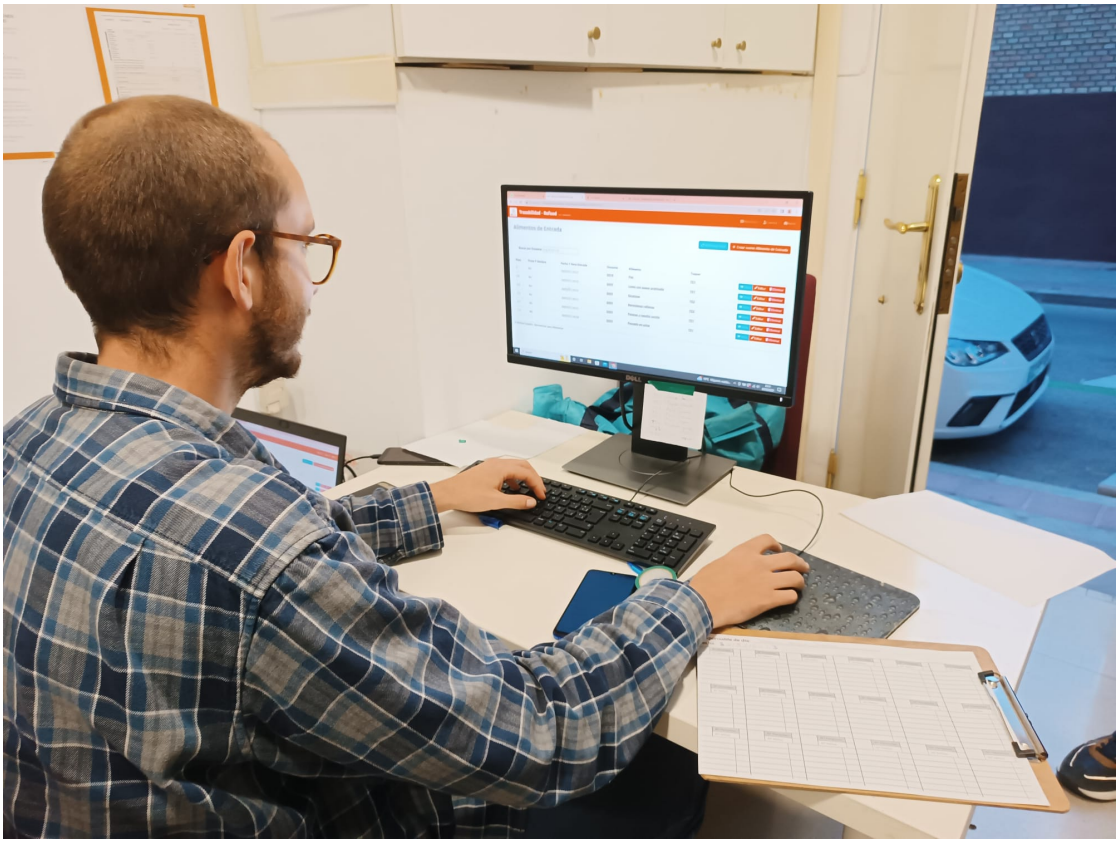


Figure 7.9: Application in Refood 1

Figure 7.10: Application in Refood 2

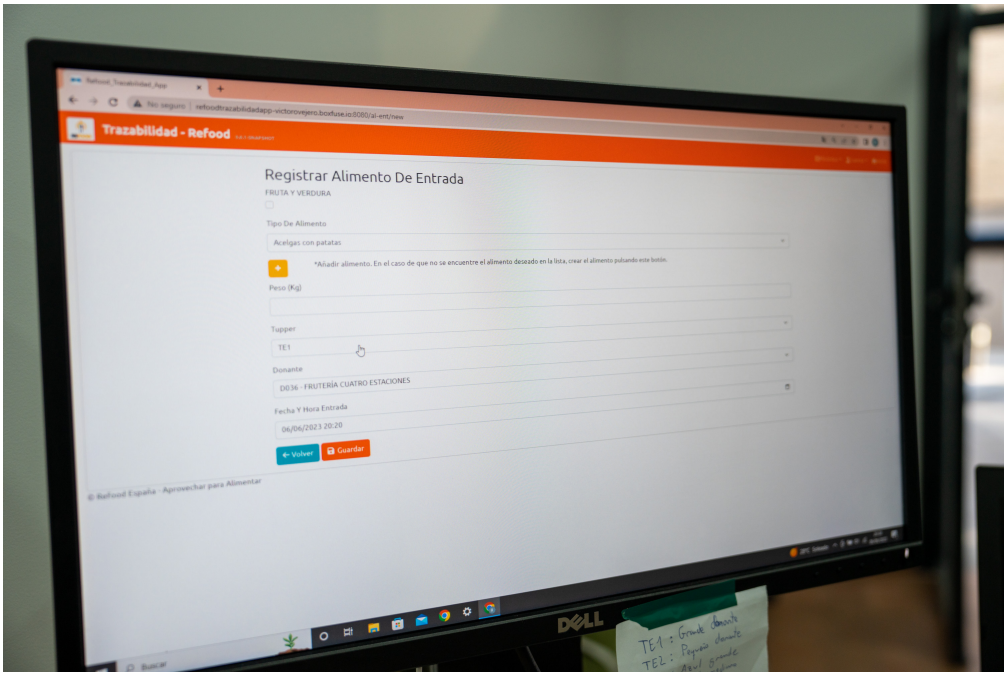


Figure 7.11: Application in Refood 3

Appendix A

Sustainable Development Goals

The Sustainable Development Goals are a set of seventeen objectives defined by the United Nations that aim to ”end poverty, protect the planet and ensure that by 2030, all people enjoy peace and prosperity” [44].

The main SDGs that are in line within this project are:

- Goal 2: Zero Hunger
- Goal 12: Responsible Consumption and Production
- Goal 13: Climate Action

Zero Hunger

The Goal 2 of the SDGs aims at ending all forms of malnutrition throughout the world. The target 2.1 states, ”ensure access by all people, in particular the poor and people in vulnerable situations, including infants, to safe, nutritious and sufficient food all year round”.

Refood is an organization that donates food to people in need, currently serving around sixty people each day. The project developed allows Refood to maintain health and safety standards by tracking the traceability of donations. Additionally, enabling the organization to be more efficient in its donation intake process, allowing it to increase the number of donors and therefore, beneficiaries. Finally, the visualization of the main achievements and impact of Refood attracts new donors and volunteers, increasing the amount of donations and benefits of Refood.

To quantify the impact, for the past three months since the web application has been implemented in the organization, a total of 3.94 tonnes of food has been donated to families in a vulnerable situation. The average European consumes more than the global average, around 780.6 kilograms of food [45]. Therefore, in three months, Refood has saved and donated the total amount of food eaten by five people in Europe in one year.

Extrapolating to a full year, and Refood will donate enough food to feed 20 people, because there are around 60 beneficiaries; Refood is providing around 33% of the total food consumed by all the beneficiaries. This number is expected to increase in the upcoming years, in part, due to the digitalization of its operational process.

Responsible Consumption and Production

The goal 12 of the SDGs aims at increasing awareness around, responsible consumption and food waste. Additionally, focusing on reducing the inefficient processes around production and decreasing the amount of food wasted. This project is focused on targets 12.3 and 12.5. Firstly, the target 12.3 is to halve the per capita global food waste by 2023. Secondly, the target 12.5 is focused on the reduction of food waste through prevention, reduction and recycling.

Refood works directly with businesses within the food supply chain in order to take the excess production of food and donate it to people in need. This project serves two main purposes, firstly, finding trends and patterns in overproduction of food, alerting donors of how to increase the efficiency and decrease food waste. Secondly, scaling the organization so that more donors can be part of the program, reducing increasingly more food waste around cities and helping increase awareness around responsible consumption and production.

To quantify the impact, the top donor of Refood has donated 618.15 kilograms of food in three months. In the past, this food would be thrown away and disposed of, contributing to the increasing problem of food waste. Additionally, 859.87 kilograms of fruits and vegetables in good state have been donated by Refood, allowing even the most perishable foods to be donated in time to people in need.

Climate Action

The goal 13 of the SDGs is focused on global warming, aiming at the reduction of CO_2 emissions, being the lead cause for the rise in temperature. Additionally, having a large focus on preventing and reducing the impact of natural disasters. This project is focused on targets 13.1 and 13.3. Firstly, the target 13.1 aims at increasing the resilience to climate-related hazards and natural disaster, nevertheless, natural disasters are not part of the scope. Secondly, the target 13.3 states, ”Improve education, awareness-raising and human and institutional capacity on climate change mitigation, adaptation, impact reduction and early warning”, being a core part of the primary goals of this project, to increase awareness and enable Refood to reach more people through engaging visualizations.

The data visualization system designed and implemented opens a door for Refood to reach larger crowds and enable the visualization of its impact on communities and climate change.

To quantify this, Refood has, in the last three months, prevented around 13.78 thousand kilograms of CO_2 emissions that would have been generated due to the food waste. To visualize this, according to the U.S. department of Agriculture, one tree can absorb around 21.7 kilograms of CO_2 in one year [46]. In order to prevent 13.78 thousand kilograms in three months, 2540 new trees would need to be planted.

Appendix B

Agile Methodology

On February, 2001 a group of IT professionals and leaders from the software industry set out to create the Agile Manifesto, in which twelve principles were agreed upon [7]. These principles would later constitute the idea for Agile Methodology, and it has shifted the way software has been developed since then. Agile methodology puts a strong emphasis in the communication between the client and the software developers as well as in the flexibility and reactiveness to changes in client needs and requirements.

Nowadays, big organizations use modern softwares to maintain an agile methodology within the company such as Jira and Kanban. However, for smaller organizations and individual projects, this is not needed to adopt an agile way of working. By following some of the principles from the Agile Manifesto, the goal is to provide continuous adaptation to the needs of the organization, Refood. To achieve this, every two weeks the author of this project conducts meetings with Refood, showing progress and understanding new requirements and design improvements provided by Refood. In order to show the usefulness of an Agile Methodology, there will be cases in which adopting agile methods has changed the outcome of the project and has led the author to make necessary changes prior to the deployment of the application, these cases will be highlighted.

Appendix C

Database Schema in JDL

```
entity Nucleo {
    idNucleo String required,
    nombre String required,
    direccion String required,
    codigoPostal String required,
    provincia String required,
    responsable String required,
    telefono String required,
    email String required,
    activo Boolean required
}

entity Donante {
    idDonante String required,
    nombre String required,
    categoria String required,
    direccion String required,
    codigoPostal String required,
    provincia String required,
    telefono String required,
    email String required,
    responsable String required,
    fechaAlta LocalDate required,
    fechaBaja LocalDate,
    comentarios String,
    activo Boolean required
}

entity Benef {
    idBeneficiario String required,
    nombreRepresentante String required,
    primerApellidoRepresentante String required,
    segundoApellidoRepresentante String
```

```
34      numeroPersonas Integer required,
35      email String required,
36      telefono String required,
37      telefonoSecundario String required,
38      direccion String required,
39      codigoPostal String required,
40      fechaAlta LocalDate required,
41      fechaBaja LocalDate,
42      numeroNinios Integer required,
43      activo Boolean required
44  }
45
46  entity PBenef{
47      nombre String required,
48      primerApellido String required,
49      segundoApellido String,
50      fechaNacimiento String required,
51      sexo String required,
52      parentesco String required,
53      situacionProfesional String required
54  }
55
56  entity AlEnt {
57      peso Double required,
58      frutaYVerdura Boolean required,
59      fechaYHoraEntrada ZonedDateTime required,
60      fechaYHoraRecogida ZonedDateTime,
61      fechaYHoraPreparacion ZonedDateTime
62  }
63
64  entity AlSal {
65      fechaSalida LocalDate required
66  }
67
68  entity Checkout{
69      fechaSalida LocalDate required,
70      peso Double required,
71  }
72
73  entity TipoAl{
74      nombreAlimento String required,
75      frutaYVerdura Boolean required,
76      descripcion String
77  }
78
79  entity Tupper{
80      peso Double required,
81      productor String required,
82      modelo String required,
```

```
    precio Double required,
    descripcion String
}

entity Intol {
    nombre String required,
    descripcion String
}

entity Voluntario {
    idVoluntario String required,
    nombre String required,
    primerApellido String required,
    segundoApellido String,
    email String required,
    telefonoContacto String required,
    dni String,
    fechaNacimiento LocalDate required,
    sexo String required,
    fechaAlta LocalDate required,
    fechaBaja LocalDate,
    categoriaPerfil String required,
    descripcionCategoria String,
    diaRefood String required,
    origen String,
    manipuladorAlimentos Boolean required,
    direccion String required,
    codigoPostal String required,
    activo Boolean required
}

entity Socio{
    nombre String required,
    primerApellido String required,
    segundoApellido String,
    email String required,
    telefonoContacto String required,
    IBAN String required,
    dni String required,
    fechaNacimiento LocalDate required,
    sexo String required,
    fechaAlta LocalDate required,
    fechaBaja LocalDate,
    contribucionMensual Double required,
    periodoPago String required,
    activo Boolean required,
    nucleoAsociado String,
    comunicacion Boolean required,
    direccion String required,
```

```
132     codigoPostal String required,
133     provincia String,
134     pais String
135 }
136 
137 relationship OneToMany {
138     Nucleo to Donante{nucleo}
139     Nucleo to Benef{nucleo}
140     Nucleo to Voluntario{nucleo}
141     Tupper to AlSal{tupper}
142     Tupper to AlEnt{tupper}
143     Benef to AlSal{benef}
144     Donante to AlEnt{donante}
145     AlEnt to AlSal{alEnt}
146     TipoAl to AlEnt{tipoAl}
147     Benef to PBenef{benef}
148     Benef to Checkout{benef}
149 }
150 
151 relationship ManyToMany {
152     Benef{intol} to Intol{benef}
153     PBenef{intol} to Intol{pBenef}
154     TipoAl{intol} to Intol{tipoAl}
155     Checkout{alSal} to AlSal{checkout}
156 }
157 
158 paginate Intol, Nucleo, Tupper, Socio,
159 TipoAl, Donante, Benef, AlEnt, AlSal, Voluntario,
160 Checkout, PBenef with pagination
161 service all with serviceImpl
162 // Use Data Transfer Objects (DTO)
163 dto * with mapstruct
```

Code Listing C.1: Database Schema in JDL

Appendix D

Conversion: Kilogram of Food Waste to CO_2 Emissions

The conversion of avoided CO_2 emissions is done by calculating the average emissions per kilogram of individual foods [47]. In Table D.1, the CO_2 emissions per kilogram of food produced can be observed for each type of food. The emissions per type of food are weighted from a set of two multipliers: 1 and 0.2. On one hand, the factor of 1 is given to the types of food that are commonly donated to Refood, such as fruits, vegetables, bread and pastries. On the other hand, the factor of 0.2 is given to the foods that are not commonly donated to Refood such as beef, chocolate and prawns.

Therefore, the foods commonly received by Refood contribute more to the calculation of the average CO_2 emissions than those which are less common. With this, the impact of the less donated foods are assumed to contribute five times less than those commonly donated to Refood.

In conclusion, the conversion used by Refood, differentiating the common foods that it prevents from going to waste from the less common foods donated, is calculated. As seen in the last row of Table D.1, the calculated conversion of a kilogram of food waste to a kilogram of CO_2 emissions is 3.49; commonly rounded up to 3.5.

Food	CO_2/kg	**Multiplier**	**Weighted**
Apples	0.43	1	0.43
Bananas	0.86	1	0.86
Barley	1.18	0.2	0.236
Beef (beef herd)	99.48	0.2	19.896
Beef (dairy herd)	33.30	0.2	6.66
Beet Sugar	1.81	1	1.81
Berries & Grapes	1.53	1	1.53
Brassicas	0.51	1	0.51
Cane Sugar	3.20	1	3.2
Cassava	1.32	0.2	0.264
Cheese	23.88	0.2	4.776
Citrus Fruit	0.39	1	0.39
Coffee	28.53	0.2	5.706
Dark Chocolate	46.65	0.2	9.33
Eggs	4.67	1	4.67
Fish (farmed)	13.63	1	13.63
Groundnuts	3.23	1	3.23
Lamb & Mutton	39.72	0.2	7.944
Maize	1.70	1	1.7
Milk	3.15	0.2	0.63
Nuts	0.43	1	0.43
Oatmeal	2.48	1	2.48
Onions & Leeks	0.50	1	0.5
Other Fruit	1.05	1	1.05
Other Pulses	1.79	1	1.79
Other Vegetables	0.53	1	0.53
Peas	0.98	1	0.98
Pig Meat	12.31	1	12.31
Potatoes	0.46	1	0.46
Poultry Meat	9.87	1	9.87
Prawns (farmed)	26.87	0.2	5.374
Rice	4.45	1	4.45
Root Vegetables	0.43	1	0.43
Soy milk	0.98	0.2	0.196
Tofu	3.16	0.2	0.632
Tomatoes	2.09	1	2.09
Wheat & Rye	1.57	1	1.57
Average Conversion	N/A	N/A	3.49

Table D.1: Greenhouse Gas Emissions per Kilogram of Food

Appendix E

Feedback Survey

Traceability Application

Refood

* Required

1. How long have you been part of Refood? *

2. What days do you volunteer? *

- [ ] Monday
- [ ] Tuesday
- [ ] Wednesday
- [ ] Thursday

3. What responsibilities do you have? *

- [ ] Going on routes
- [ ] In charge of traceability
- [ ] Portion distribution
- [ ] Other

4. In comparison with the previous system, What do you think of the traceability application? *

- ◯ It is much more efficient
- ◯ It is a little bit more efficient
- ◯ More or less the same
- ◯ It is a little bit less efficient
- ◯ It is a lot less efficient
- ◯ I do not know the previous system

5. How easy to use is the new system? (1 being the most difficult and 5 being the easiest) *

1	2	3	4	5

6. How would you rate the User Interface of the application? *

☆ ☆ ☆ ☆ ☆

7. How would you rate the process of data intake with the new web application? *

☆ ☆ ☆ ☆ ☆

8. What would you improve about the new application? *

9. What do you like the most about the new system? *

10. What are you missing from the new system? *

11. In general, are you liking the new system? *

☆ ☆ ☆ ☆ ☆

12. Is there anything else you would like to add?

Appendix F

Survey Responses

F.0.1 Question 1

Respondant	Response
1	Dos meses
2	3 meses
3	2 meses
4	Una año y medio
5	4 meses
6	Algo más de un año
7	Inicio
8	4 años
9	1 año

Table F.1: Responses: Question 1

F.0.2 Question 2

Respondant	Response
1	Martes
2	Martes
3	Miércoles
4	Martes; Miércoles
5	Jueves
6	Jueves
7	Jueves
8	Martes
9	Lunes

Table F.2: Responses: Question 2

F.0.3 Question 3

Respondant	Response
1	Voluntario de Ruta
2	Voluntario de Ruta; Encargado de la Trazabilidad; Encargado de Distribuir la Comida
3	Voluntario de Ruta; Encargado de Distribuir la Comida
4	Encargado de Distribuir la Comida; Encargado de la Trazabilidad
5	Encargado de Distribuir la Comida; Voluntario de Ruta; Encargado de la Trazabilidad
6	Voluntario de Ruta; Encargado de Distribuir la Comida
7	Encargado de la Trazabilidad; Encargado de Distribuir la Comida
8	Otras; Encargado de la Trazabilidad; Encargado de Distribuir la Comida
9	Encargado de la Trazabilidad

Table F.3: Responses: Question 3

F.0.4 Question 4

Respondant	Response
1	No conozco el sistema anterior
2	No conozco el sistema anterior
3	No conozco el sistema anterior
4	Muy eficiente
5	Un poco más eficiente
6	Muy eficiente
7	Muy eficiente
8	Muy eficiente
9	Muy eficiente

Table F.4: Responses: Question 4

F.0.5 Question 5

Respondant	Response
1	4
2	3
3	5
4	4
5	2
6	3
7	4
8	5
9	5

Table F.5: Responses: Question 5

F.0.6 Question 6

Respondant	Response
1	4
2	4
3	5
4	4
5	4
6	5
7	5
8	5
9	5

Table F.6: Responses: Question 6

F.0.7 Question 7

Respondant	Response
1	4
2	4
3	5
4	4
5	3
6	5
7	4
8	4
9	5

Table F.7: Responses: Question 7

F.0.8 Question 8

Respondant	Response
1	Pues no puedo opinar mucho porque no estoy en el local. Si se puede mejorar y agilizar la trazabilidad en origen, pero eso depende de la disposición de los colegios mayores
2	Es muy tedioso introducir los tuppers que se destinan a cada beneficiario. Estaría bien que una vez seleccionas un beneficiario puedas meter todos los tuppers que quieras sin volver a seleccionarlo. Y también que no sea necesario guardar cada tupper que se añade al beneficiario. Poder añadir los 4 o 5 que sean y darle a guardar
3	”Llamar a los beneficiarios por su nombre. Cambiar el nombre de los tuppers”
4	”En ocasiones aparecen mensajes de error y no sabemos la causa. Que alguna información se quede automatizada allí”
5	”-El nombre del tamaño de los tuppers que fuese más intuitivo (P, M, G). -Mejorar los avisos cuando alguna acción no está contemplada y no sabes por qué no te deja meter la entrada o da error -Empezar a escribir y que te salga por defecto la entrada que contenga esa palabra - Poder poner los KG totales de las salidas de cada beneficiario justo después de meter las comidas, en vez de tener que hacer un check de cada al final”
6	No era la encargada de trazabilidad.
7	Enumerar alfabéticamente todas las opciones, tanto entradas como salidas. También añadiría la casilla de pan al igual que esta actualmente la de fruta. El resto está genial!!
8	”- Mejoras de búsqueda en los drop down - Automatización de insercion masiva de entradas - Campo de texto que filtre las búsquedas”
9	nada.

Table F.8: Responses: Question 8

F.0.9 Question 9

Respondant	Response
1	No tengo mucha opinión
2	Es muy útil
3	La facilidad de usarla
4	Que es mucho más amigable que Excel y que se guarda mucha información que ya hemos utilizado antes
5	Los campos desplegables
6	No era la encargada de trazabilidad.
7	Interfaz y sencillez
8	Es extremadamente rapida e intuitiva
9	todo.

Table F.9: Responses: Question 9

F.0.10 Question 10

Respondant	Response
1	Orden alfabetico de los platos/alimentos /productos
2	Intégrate con ChatGPT y que reparta los tuppers entre los beneficiarios en función de la cantidad de tuppers que haya
3	Nada
4	Sería bueno poder visualizar con grafica ciertas cosas, como que beneficiarnos reciben más comida en KG, que donantes donan más o menos comida, comparar en kilos las donaciones por día, para identificar tendencias
5	Lo descrito en el punto 8. Quizá podría hacerse una base de datos para normalizar las entradas. Es decir, que ya haya unas preparaciones predeterminadas, ya que casi siempre es la misma comida (sopa/ tortilla/ guiso verduras / guiso carne/ guiso pescado/pollo guisado/pollo tomate/pollo salsa, etc.).
6	No era la encargada de trazabilidad.
7	Enumerar alfabéticamente todas las opciones, tanto entradas como salidas. También añadiría la casilla de pan al igual que esta actualmente la de fruta. El resto está genial!!
8	Insercion masiva de datos
9	nada.

Table F.10: Responses: Question 10

F.0.11 Question 11

Respondant	Response
1	4
2	5
3	5
4	5
5	4
6	5
7	5
8	5
9	5

Table F.11: Responses: Question 11

F.0.12 Question 12

Respondant	Response
1	Poco puedo opinar
2	Nada más
3	no
4	No se me ocurre por ahora, pero me gusta mucho y es una gran mejora comprado con la anterior
5	Creo que ya lo he comentado todo
6	No era la encargada de trazabilidad.
7	Muchas gracias por hacerla!!! La verdad que nos ahorra bastante tiempo
8	Es un punto de inflexión para la organización y sus voluntarios
9	no.

Table F.12: Responses: Question 12

Bibliography

[1] Hamish Forbes and Tom Quested Clementine O'Connor. *Food Waste Index Report 2021*. United Nations Environment Programme, 2021.

[2] *Food waste and food waste prevention - estimates*. en. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Food_waste_and_food_waste_prevention_-_estimates (visited on 07/02/2023).

[3] Gobierno de España. *Proyecto de ley de prevención de las pérdidas y el desperdicio alimentario*. https://www.mapa.gob.es/es/prensa/anteproyectoleydesperdicio_tcm30-620834.pdf. 2022.

[4] *What Is Data Governance and Why Does It Matter?* en. URL: https://www.techtarget.com/searchdatamanagement/definition/data-governance.

[5] *What is Data Integrity?* URL: https://database.guide/what-is-data-integrity/.

[6] *What is Business Intelligence and How Does it Work? — IBM*. en-us. URL: https://www.ibm.com/topics/business-intelligence.

[7] *Manifesto for Agile Software Development*. URL: https://agilemanifesto.org/.

[8] *Web Applications*. en. URL: https://spring.io.

[9] *React*. en. URL: https://react.dev/.

[10] *JavaScript With Syntax For Types*. en. URL: https://www.typescriptlang.org/.

[11] *JHipster - Full Stack Platform for the Modern Developer!* URL: https://www.jhipster.tech/.

[12] *What is PowerBI?* en-au. URL: https://www.microsoft.com/en-au/videoplayer/embed/RWQAMw?jsapi=true&postJsllMsg=true&maskLevel=0.

[13] *Cloud Computing Services - Amazon Web Services (AWS)*. en-US. URL: https://aws.amazon.com/ (visited on 06/26/2023).

[14] *AWS for Nonprofits — AWS*. en-US. URL: https://aws.amazon.com/government-education/nonprofits/.

[15] *Cloud storage vs. on-premises servers: 9 things to keep in mind*. en-us. URL: https://www.microsoft.com/en-us/microsoft-365/business-insights-ideas/resources/cloud-storage-vs-on-premises-servers.

[16] *2021 JavaScript Rising Stars*. en. URL: https://risingstars.js.org/2021/en.

[17] *2020 Java Technology Report*. en. URL: https://www.jrebel.com/blog/2020-java-technology-report.

[18] shmcarth. *Application lifecycle management (ALM) with Microsoft Power Platform - Power Platform*. en-us. Oct. 2022. URL: https://learn.microsoft.com/en-us/power-platform/alm/overview-alm.

[19] shmcarth. *Application lifecycle management (ALM) with Microsoft Power Platform - Power Platform*. en-us. Oct. 2022. URL: https://learn.microsoft.com/en-us/power-platform/alm/overview-alm.

[20] *Microservices vs Monolithic architecture*. en. URL: https://www.mulesoft.com/resources/api/microservices-vs-monolithic (visited on 07/04/2023).

[21] *Relational Vs. Non-Relational Databases*. en-us. URL: https://www.mongodb.com/compare/relational-vs-non-relational-databases (visited on 06/24/2023).

[22] *11 Most In-Demand Programming Languages in 2023*. en. Dec. 2020. URL: https://bootcamp.berkeley.edu/blog/most-in-demand-programming-languages/ (visited on 06/24/2023).

[23] Mike Wolfe. *MySQL vs Oracle SQL*. en. Aug. 2021. URL: https://towardsdatascience.com/mysql-vs-oracle-sql-a97a7659f992 (visited on 06/24/2023).

[24] *Normalisation*. URL: https://db.grussell.org/section008.html (visited on 06/25/2023).

[25] *CloudCaptain • Immutable Infrastructure Made Easy*. URL: https://cloudcaptain.sh/ (visited on 06/25/2023).

[26] *JDL-Studio*. URL: https://www.jhipster.tech/jdl-studio/ (visited on 06/25/2023).

[27] *JavaScript Component Testing and E2E Testing Framework — Cypress*. en. URL: https://www.cypress.io/ (visited on 06/25/2023).

[28] *JpaRepository (Spring Data JPA Parent 3.1.1 API)*. en. URL: https://docs.spring.io/spring-data/data-jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html (visited on 06/27/2023).

[29] *Spring Data JPA - Reference Documentation*. URL: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repositories (visited on 06/27/2023).

[30] *What is a REST API?* en. URL: https://www.redhat.com/en/topics/api/what-is-a-rest-api (visited on 06/27/2023).

[31] *SwaggerHub — API Design & Documentation with OAS and AsyncAPI*. URL: https://swagger.io/tools/swaggerhub/?utm_source=aw&utm_medium=ppcg&utm_campaign=SEM_SwaggerHub_PR_EMEA_ENG_EXT_Prospecting&utm_term=swagger&utm_content=511173019635&gclid=CjwKCAjwkeqkBhAnEiwA5U-uM4x4U6FmonhNjAifMZNzYlB1NkGgz7wv0ySb3qUKpmVIfLt4IM8bixoBwE&gclsrc=aw.ds (visited on 06/27/2023).

[32] *Redux - A predictable state container for JavaScript apps. — Redux*. en. URL: https://redux.js.org/ (visited on 06/28/2023).

[33] *Getting Started — Axios Docs*. URL: https://axios-http.com/docs/intro (visited on 06/28/2023).

[34] *Feature Overview v6.14.0*. en. URL: https://reactrouter.com/en/main/start/overview (visited on 06/28/2023).

[35] *What is Software Testing and How Does it Work? — IBM*. en-us. URL: https://www.ibm.com/topics/software-testing (visited on 06/29/2023).

[36] *Welcome to ISTQB®*. en. URL: http://istqb-frontend-staging.s3-website.us-east-2.amazonaws.com/ (visited on 06/29/2023).

[37] *Maven – Welcome to Apache Maven*. URL: https://maven.apache.org/ (visited on 06/29/2023).

[38] *CloudCaptain •Immutable Infrastructure Made Easy*. URL: https://cloudcaptain.sh/ (visited on 06/30/2023).

[39] Rajendra Gupta. *Automatically Start/Stop an AWS RDS SQL Server using AWS Lambda functions*. en-US. June 2020. URL: https://www.sqlshack.com/automatically-start-stop-an-aws-rds-sql-server-using-aws-lambda-functions/ (visited on 07/01/2023).

[40] *Logstash: Collect, Parse, Transform Logs — Elastic*. URL: https://www.elastic.co/logstash/.

[41] *Kibana: Explore, Visualize, Discover Data*. URL: https://www.elastic.co/kibana.

[42] *Welcome to The Apache Software Foundation!* URL: https://apache.org/.

[43] *food handlers*. en. May 2017. URL: https://www.comunidad.madrid/en/servicios/salud/manipuladores-alimentos (visited on 07/04/2023).

[44] *Sustainable Development Goals — United Nations Development Programme*. en. URL: https://www.undp.org/sustainable-development-goals (visited on 07/03/2023).

[45] GOODSEEDVENTURES. *Worldwide Food Consumption Per Capita*. en-US. Dec. 2021. URL: https://goodseedventures.com/worldwide-food-consumption-per-capita-2/ (visited on 07/03/2023).

[46] *The Power of One Tree - The Very Air We Breathe*. en. URL: https://www.usda.gov/media/blog/2015/03/17/power-one-tree-very-air-we-breathe (visited on 07/03/2023).

[47] J. Poore and T. Nemecek. "Reducing food's environmental impacts through producers and consumers". In: *Science* 360.6392 (2018), pp. 987–992. DOI: 10.1126/science.aaq0216. URL: https://www.science.org/doi/abs/10.1126/science.aaq0216.