



UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Safety and quality based traceability system for food supply chains

João Paulo Gonçalves da Silva Curto

Dissertação para obtenção do Grau de Mestre em
Engenharia Eletromecânica
(2º ciclo de estudos)

Orientador: Prof. Doutor Pedro Miguel de Figueiredo Dinis Oliveira Gaspar

Covilhã, dezembro de 2019

Resumo

O surgimento de novas tecnologias e novos processos tem reduzido a familiaridade e capacidade dos consumidores em discernir a qualidade dos produtos adquiridos bem como a capacidade de verificar se todas as regras de higiene e segurança foram cumpridas.

A rastreabilidade pode ser uma ferramenta capaz de assegurar a segurança e a qualidade de alimentos perecíveis bem como otimização de processos e ganho económico. Ainda assim, é regularmente considerada uma mera burocracia e um fardo económico, principalmente pelas micro, pequenas e médias empresas (PME). No entanto, estas constituem a maior parte do sector alimentar e uma vez que a rastreabilidade não é desejada para além do cumprimento de requisitos legais, a adopção de sistemas de rastreabilidade tem sido bastante lenta.

De modo a determinar as maiores vantagens e desvantagens dos modelos de rastreabilidade, implementação e tecnologia, foi efectuada uma revisão de literature com crítica focada nas PME. Ganho económico, mais qualidade e segurança, maior eficiência e um contacto mais próximo com os consumidores são algumas das principais vantagens. Elevados custos de implementação, vantagens nem sempre tangíveis e fracamente definidas, falta de compatibilidade, perspectiva focada no consumidor e exposição de informação sensível são algumas das principais desvantagens.

Com o resultado desta análise, foi desenvolvida uma estrutura open-source para sistemas de rastreabilidade. Esta estrutura inclui um modelo de comunicação externa e é acompanhada de software capaz de cumprir todas as tarefas requeridas para a garantia de qualidade tanto interna como externamente. Abrange cadeias de abastecimento alimentar inteiras e mantém registos de qualidade e segurança sem necessidade de elevada capacidade em tecnologias de informação, característica incomum das PME. É imperativo que os sistemas de rastreabilidade tenham uma estrutura de troca de informação comum. Sem a mesma, estes sistemas perdem muita da sua utilidade pois apenas seriam usados para propósitos internos e isso significa que teriam uma maior dificuldade em auxiliar recolhas de produto impróprio e em adicionar valor aos produtos. É também imperativo que este tipo de estrutura seja desenvolvida com foco em recursos existentes tais como os fluxogramas do sistema HACCP como meio de segmentação de processos e avaliação de qualidade. Com isso em mente, o volume de informação é dividido por todas partes de acordo com as suas necessidades e capacidades de financiamento. A maior parte da informação é mantida pelos reguladores devido à sua maior facilidade de acesso a fundos. Isto aumenta a facilidade e flexibilidade de implementação de sistemas de rastreabilidade por parte das empresas.

Utilizando todas as ferramentas desenvolvidas foi efectuada uma simulação duma cadeia alimentar hipotética com objetivo de observar todas as variações de qualidade e a capacidade de transmissão de informação. Verificou-se que o sistema é capaz de avaliar a qualidade e transmitir eficazmente toda a informação necessária.

Palavras-chave

Vantagens, desvantagens, estrutura de comunicação, ferramentas de rastreabilidade, simulação.

Abstract

The onset of new technologies and new procedures for food production has reduced the familiarity and capability of consumers to discern the quality of products bought, as well as being unable to verify if all health and safety regulations were complied with.

Traceability can be a tool for safety and quality assurance for perishable food as well as for process optimization and economic gain. However, it is often considered mere bureaucracy and an economic burden. Such is prevalent in micro and small-sized (MSE) enterprises. As they constitute most of food sector, the adoption of traceability systems is quite slow and mostly to satisfy legal requirements. In order to determine the main advantages and disadvantages of traceability models, implementation and technologies, a literature review and MSE focused analysis was performed. Economic gains, more quality and safety, better efficiency and a more direct contact with consumers are some of the main advantages. High implementation costs, poorly defined benefits, lack of compatibility, consumer focused perspective and exposure of sensitive information are some of the main issues. With the result of that analysis, an open-source traceability framework was developed. The framework includes an external communication model and is accompanied by software capable of fulfilling all tasks required for quality assurance both internally and externally. It encompasses whole food supply chains and maintains records of quality and safety while not necessitating mature IT capabilities, uncommon characteristic of MSE's. It is imperative for traceability systems to have a common structure for information exchange. Without it, these systems lose much of their utility as they will only be usable internally and will have reduced capacity to add value to products and manage recalls. It is also imperative to develop this type of framework by focusing it on existing resources such as HACCP flowcharts to define gateways for quality evaluation. With that in mind, the volume of information is divided between all stakeholders according to their necessities and funding capacities. Most of the information is stored by regulators as they have access to more funding. This improves the ease and flexibility of implementation of traceability systems by the companies.

Using all developed tools an hypothetical food supply chain was simulated. The objective of this simulation was to observe all quality variations and capacity to transmit information. It was verified that the developed system was able to evaluate quality throughout an entire chain and to effectively communicate all necessary information.

Keywords

Advantages, disadvantages, communication framework, traceability tools, simulation.

Contents

1	Introduction	1
1.1	Definition and importance of traceability	1
1.2	The problem under study and its relevance	2
1.3	Objectives and contribution of the dissertation	2
1.4	Dissertation overview and organization	3
2	State of the art	5
2.1	Application of traceability systems and granularity	5
2.2	Benefits, necessities, requirements, obstacles and components of traceability systems	10
2.3	Models, methods, algorithms and supply chain management	17
3	Materials and methods	25
3.1	Traceability model overview	25
3.2	Model constituents	26
3.2.1	Modules	26
3.2.2	Layers	27
3.2.3	Segments	27
3.3	Archetype for information sharing	27
3.3.1	First segment	28
3.3.2	Second segment	30
3.3.3	Third segment	32
3.3.4	Fourth segment	34
3.4	Archetypes for shared files	34
3.4.1	Information for final consumers	34
3.4.2	Information for direct consumers	36
3.4.3	Quality validation and unique code	37
3.4.4	Quality after transport and quality verification	37
3.5	Choice of language and development environment	37
3.6	Operation of the constituents	38
3.6.1	Production layer	38
3.6.2	Regulator layer	55
3.6.3	Unique code layer	62
4	Simulation	63
4.1	Algorithms and formulas	63
4.1.1	Quality and keeping quality decay formulas	63
4.1.2	Routing algorithm	64
4.2	Entry data	64
4.3	First segment results	65
4.3.1	Inventory management module results	65
4.3.2	Processing stage management module results	65
4.3.3	Quality validation module results	66
4.3.4	Order management module results	66

Safety and quality based traceability system for food supply chains

4.3.5	First segment quality and keeping quality variation	66
4.4	Second segment results	67
4.4.1	Quality after transport module results	67
4.4.2	Inventory management module results	68
4.4.3	Processing stage management module results	68
4.4.4	Quality validation module results	68
4.4.5	Order management module results	68
4.4.6	Second segment quality and keeping quality variation	68
4.5	Third segment results	70
4.5.1	Quality after transport module results	70
4.5.2	Inventory management module results	70
4.5.3	Processing stage management module results	70
4.5.4	Quality validation module results	71
4.5.5	Order management module results	71
4.5.6	Third segment quality and keeping quality variation	71
4.6	Supply chain quality degradation	72
4.6.1	Quality	72
4.6.2	Keeping quality	73
5	Limitations and performance analysis	75
5.1	Limitations	75
5.2	Performance analysis	76
6	Conclusion	79
6.1	General conclusions	79
6.2	Suggestions for future work	82
	Bibliography	85
A	First segment modules code snippets	93
A.1	Inventory Management Module	93
A.1.1	MainApp.py	93
A.1.2	Tab1Layout.py	94
A.1.3	Tab1Functions.py	95
A.1.4	Tab2Layout.py	99
A.1.5	Tab2Functions.py	100
A.1.6	Tab3Layout.py	102
A.1.7	Tab3Functions.py	103
A.2	Processing stage management module	105
A.2.1	MainApp.py	105
A.2.2	Tab1Layout.py	105
A.2.3	Tab1Functions.py	107
A.2.4	Tab2Layout.py	112
A.2.5	Tab2Functions.py	112
A.2.6	Tab3Layout.py	113
A.2.7	Tab3Functions.py	113
A.3	Order management module	113
A.3.1	Run.py	113

Safety and quality based traceability system for food supply chains

A.3.2	OMMServer	113
B	Regulator layer code snippets	149
B.1	Run.py	150
B.2	Regulator server	150
B.2.1	__init__.py	150
B.2.2	Config.py	150
B.2.3	Models.py	150
B.2.4	NonUsers	150
B.2.5	Users	155
B.3	Docs	170
C	Unique code layer code snippets	171

Safety and quality based traceability system for food supply chains

List of Figures

3.1	Diagram of the traceability model.	26
3.2	Flowchart of the first segment.	29
3.3	Flowchart of the second segment.	31
3.4	Flowchart of the third segment.	33
3.5	Inventory management module new entry tab.	39
3.6	Inventory management module entry confirmation.	40
3.7	Inventory management module inventory tab.	40
3.8	Inventory management module inventory table.	41
3.9	Inventory management module inventory table (cont).	41
3.10	Inventory management module QMP plot	42
3.11	Processing stage management module entry tab.	42
3.12	Processing stage management module new operation confirmation.	43
3.13	Processing stage management module information input confirmation windows.	43
3.14	Processing stage management module operational history.	43
3.15	Processing stage management module operational history(cont).	44
3.16	Processing stage management module operational history(cont).	44
3.17	Processing stage management module operational history(cont).	45
3.18	Processing stage management module QMP plot.	45
3.19	Order management module homepage.	46
3.20	Order management module availability form.	47
3.21	Order management module availability result.	47
3.22	Order management module login page.	48
3.23	Order management module personal page.	48
3.24	Order management module inventory page.	49
3.25	Order management module stage N page.	50
3.26	Order management module first step of the order form.	51
3.27	Order management module second step of the order form.	52
3.28	Order management module third step of the order form.	53
3.29	Regulator home page.	55
3.30	Regulator search page.	55
3.31	Regulator search result page.	56
3.32	Regulator search result page.	57
3.33	Regulator search result page.	58
3.34	Regulator login page.	59
3.35	Regulator layer personal page.	59
3.36	Quality after transport step 1.	60
3.37	Quality after transport step 2.	60
3.38	Quality after transport step 3.	61
3.39	Quality validation form.	61
3.40	Quality validation result.	61
4.1	Quality degradation throughout the first segment.	66
4.2	Keeping quality variation throughout the first segment.	67

Safety and quality based traceability system for food supply chains

- 4.3 Quality degradation throughout the second segment. 69
- 4.4 Keeping quality variation throughout the second segment. 69
- 4.5 Quality degradation throughout the third segment. 71
- 4.6 Keeping quality variation throughout the third segment. 72
- 4.7 Quality degradation throughout the supply chain. 73
- 4.8 Keeping quality variation throughout the supply chain. 74

- A.1 Order management module file hierarchy. 147

- B.1 Regulator layer file hierarchy. 149

List of Tables

2.1	Review of application of traceability systems and granularity.	9
2.1	Review of application of traceability systems and granularity (continued).	10
2.2	Scientific research covering the benefits, necessity, requirements, obstacles and components of traceability systems.	16
2.2	Scientific research covering the benefits, necessity, requirements, obstacles and components of traceability systems (continued).	17
2.3	Important concepts concerning models, methods, algorithms, and supply chain management.	22
2.3	Important concepts concerning models, methods, algorithms, and supply chain management (continued).	23
3.1	Information for Final Consumers example.	36
3.2	Quality After Transport example.	37
3.3	Summary file structure.	54
3.4	Conditions file structure.	54
3.5	Graph file structure.	54
3.6	Quality file structure.	54
3.7	Route file structure.	55
4.1	Tijskens and Polderdijk [99] Table 4.	64
4.2	Entry data for simulation.	64
4.3	First segment inventory entry.	65
4.4	First segment inventory updated values.	65
4.5	First segment processing stage management module output.	65
4.6	First segment order management module prediction.	66
4.7	Quality After Transport module entry data.	68
4.8	Quality After Transport module results.	68
4.9	Second segment processing stage management module output.	68
4.10	Second segment order management module prediction.	68
4.11	Quality After Transport module entry data.	70
4.12	Quality After Transport module results.	70
4.13	Third segment processing stage management module output.	70
4.14	Third segment order management module prediction.	71
5.1	Bendaoud et al. [102] Table 1 functions and achievements.	78

Acronyms List

UBI	Universidade da Beira Interior
IMM	Inventory Management Module
PSMM	Processing Stage Management Module
OMM	Order Management Module
UCGM	Unique Code Generator Module
QVM	Quality Validation Module
QATM	Quality After Transport Module
RFIC	Repository of Information for Final Consumers
IDC	Information for Direct Consumers
IFC	Information for Final Consumers
AC	Auxiliary Comparator
MSE	Micro and Small Enterprises
SME	Small and Medium Enterprises
GS1	Global Standards One
FAO	Food and Agriculture Organization of the United Nations
ICT	Information and Communication Technologies
RFID	Radio Frequency Identification
EPC	Electronic Product Code
HACCP	Hazard Analysis and Critical Control Points
IoT	Internet of Things
FIFO	First In First Out
FEFO	First Expired First Out
LSFO	Least Shelf-life First Out
IDE	Integrated Development Environment
OS	Operating System
TRU	Traceable Resource Unit
CIP	Critical Information Point
T&T	Track & Trace
EDI	Electronic Data Interchange
CSF	Critical Success Factor
FLS	Food Logistic System

Chapter 1

Introduction

1.1 Definition and importance of traceability

Traceability does not possess a single and clear definition. There are several definitions, joined with different needs and regulations, that causes the development of traceability systems to thread in many, and remarkably different directions. Thus, there is not a common understanding about the definition and structure for food traceability systems [1].

Food traceability can be defined as the ability to track a product batch and its history through the whole, or part, of a production chain from harvest through transport, storage, processing, distribution and sales or internally in one of the steps in the chain. Traceability is a generic issue and independent of the type of product, production and the control system it serves. Traceability is a quality control tool and should be implemented systemically, leading to the development of a traceability system that records information about a product and its movements [2].

Other definitions of traceability exist. ISO 8402 defines traceability as the ability to trace the history, application or location of an entity by means of recorded identifications [3]. This standard was then superseded by the ISO 9000 that defines traceability as the ability to trace the history, application or location of that which is under consideration [4]. The standard ISO 22005 is identical to the ISO 9000 but is specific to food supply chains [5]. All these standards have an additional clause that indicates that when relating to products, the information about the origin of materials and parts, the processing history, and the distribution and location of the product after delivery should be kept. The Codex Alimentarius defines traceability as the ability to follow the movement of a food through specified stage(s) of production, processing and distribution [6]. The General Food Law, defines traceability the ability to trace and follow a food, feed, food-producing animal or substance intended to be, or expected to be incorporated into a food or feed, through all stages of production, processing and distribution [7]. It can be seen that different food related or normalization organizations focus on the same objective using different subjects. Although international regulations exist for traceability, each country can implement additional regulations to increase the comprehensiveness and effectiveness of traceability. In Europe, regulations are already comprehensive and establish Europe as the leader and example to follow [8]. Thus, a robust traceability platform exists that can benefit from the automation of information exchange as a way to facilitate the enforcement of regulations, speed up sanctions and limit the dissemination of improper commodities, as, according to Lupien [9], the possibility of occurrence of alimentary quality and safety flaws still exists. Traceability is used in several activities, as information technologies, statistical analysis, electronics, biology, logistics, supply chain management and food industry. Traceability systems have the potential to increase security, quality, reliability and precision. These advantages have an impact in consumer protection and company productivity [10].

Traceability allows the consumers to trust consumed products and pay the correct price. It also prevents the occurrence of health issues, increases environmental sustainability and prevents the invasion of non-indigenous species [11]. Traceability is one solution to accomplish the sustainable development goals as described by the United Nations. Particularly, can be used to

help reducing Poverty and Hunger and improving Health and Climate.

1.2 The problem under study and its relevance

According to Opara [12], the demand for traceability systems has been growing significantly in the last years due to faults in alimentary security and the emergence of genetically modified organisms. These situations contribute to the reduction of consumers' trust and increase the concern with the potentially negative consequences of existing practices. In order to prevent those consequences, new regulations were introduced and aim to hold responsible entities that neglect alimentary security. To correctly determine responsibility, enterprises must demonstrate that all diligence were complied with. Consequently, rises the necessity of a system that can assure product quality and allows regulators and companies to know the time and space evolution of quality and safety of food products to better deal with eventual food crisis.

Thus, a traceability system based on quality and safety functions as a preventive tool, helps producers to effectively manage the quality, locate and isolate threats to alimentary security. The evolution of information and communication technologies (ICT) has increased the speed and precision of data collection and thus, can present reliable and transparent information to the consumers.

Traceability can be part of a system that provides information about all production stages. To achieve chain-wide traceability, it is necessary that all stakeholders can monitor and store information internally and be able of transmitting it externally in order to connect outputs (selling) and inputs (buying).

In addition to the existence of considerably different traceability systems and their lack of compatibility, traceability suffers from issues that need to be solved, as the exposure of sensitive information, emergence of copies, necessary investment and loss of autonomy.

The review presented in this dissertation aims to clarify the main issues associated with traceability models and systems. Although some issues arise on a general level, the analysis and solutions proposed in this paper give special consideration to the implementation of these systems in MSE's, due to the general specificities of this kind of companies, such as the reduced number of collaborators, lack of expert technicians in traceability, the reduced available amount for research and development activities and for the acquisition of traceability systems, either hardware (data loggers, communication systems, network devices, among others) or software (alert systems, support decision-making applications, among others).

1.3 Objectives and contribution of the dissertation

Using the principles obtained by reviewing scientific literature, a traceability framework was developed. This framework includes a communication model and software created to allow for MSE's to derive the generalizations made with the purpose of abstractions into their own needs. This framework focuses on quality and safety information monitoring and dissemination. Traceability can be considered an aspect of information monitoring, as by doing so, better control over materials and products can be achieved, even if merely on an internal level. However, information must be correctly and efficiently monitored, leading to the ability to optimize a process by

enhancing its strengths and minimizing its weaknesses. However, most of the enterprises that compose food supply chains are MSE's which often do not possess enough disposable income to invest in robust and automatic monitoring systems. And so, information is often recorded and kept manually. Such procedure increases the probability of error and drastically reduces the amount of data that can be recorded. These consequences of manual records can be severely reduced by automatic systems, but then several issues arise:

1. How to collect data automatically?
2. What investment to make?
3. How to profit from more information?

The answer to the first two questions is a well defined structure for collecting and sharing information. Having a structure with well defined rules and requirements, dictates a minimum threshold of information collection. By having that threshold, it becomes possible to calculate the minimum possible investment to achieve the ability of collecting the required information. Consequently, it then becomes possible to expand upon the minimum required and determine the cost of a more comprehensive system correctly aligned with corporate possibilities and objectives. The answer to the third question is communicating the information obtained previously. However, if no confirmation of the veracity of the information is made externally and scientifically, all trade becomes trust based, as already is, and so, traceability becomes no more than mere bureaucracy and resource sink for companies, although it will still be able to increase sustainability and therefore profit from fewer wasted materials. Thus, traceability has to be appealing for both companies and regulators and the framework developed aims to do so in a way that facilitates the communication between the two parties while being flexible enough not to hamper the development of either MSE's, regulators and the framework itself, as the necessity for changes may arise. Ultimately, the objective of this framework is to help companies profit from better information communicated to the consumers, help the consumers to impartially and scientifically verify information and transparency and help regulators to have a more proactive posture towards food safety.

1.4 Dissertation overview and organization

The remainder of this dissertation is organized in the following manner:

- Chapter 2 consists in a review and critique of scientific literature. The chapter is divided into sections each approaching different themes as to better organize the review and the critique.
- Chapter 3 describes the conditions under which all programming necessary for this dissertation was made and tested, the communication model, all file archetypes and the functioning of all tools created.
- Chapter 4 contains the simulation of a simple food supply chain. The objective of this chapter is to demonstrate the capabilities of the prototype traceability system.
- Chapter 5 describes the limitations of the system and presents a performance analysis based on work developed by other authors specific for the evaluation of a traceability system.

Safety and quality based traceability system for food supply chains

- Chapter 6 concludes the body of this dissertation and suggests future works.
- Annexes A, B and C contain part of the code that led to the presented tools. This division happens for organization purposes, Annex A corresponds to the first segment of the production layer, Annex B corresponds to the regulator layer and finally Annex C corresponds to the unique code layer.

Chapter 2

State of the art

The scientific review is divided into three sections. It starts gathering and analyzing scientific research works related to the application of traceability systems and its granularity. Granularity is the quantity of matter represented by a single identifier. Usually, a fine granularity implies large investment in a system capable to sustain it. However, it enables the collection of more information that adds value to the commodity or it is useful for logistics. For coarse granularity, implement cost is much lower but becomes difficult to obtain useful information. There is an optimum level of granularity that allows a company to obtain more gains with a lower investment. An internal evaluation is usually required to determine that level, since it depends on the product and processes.

The second section uses some research work to provide insights about the benefits, necessity, requirements, obstacles and components of traceability systems. Afterwards, the analysis and discussion focus on models, methods, algorithms, and supply chain management. Each section includes a table of reviewed content and the concepts used for review. This division should not be considered exclusive as each study has much more content that what can be exposed here. All content exposed in this review can be summararily seen in Traceability in food supply chains: Review and SME focused analysis - Part 1 [13].

2.1 Application of traceability systems and granularity

There has been a scientific effort to develop several traceability systems in this last decade to deal with the new requirements of governments, companies and consumers. This section provides the analysis and discussion of the application of traceability systems in food context and their granularity.

Beulens et al. [14] present a practical case of the implementation of a traceability system that assures quality and safety throughout a food supply and distribution chain. A structure was developed for the information exchange and quality standards that can be applied to all stakeholders. It was necessary a common syntax, as well as the functional integration of the traceability system in the administrative systems, quality control and automation systems. It also required the automation of attribution and reading of identification codes, automatic generation of reports and process optimization. An impartial entity was defined to control the integrity of the system, consumer and order records to allow more efficiency in recalls. This structure was conceived to be applied in the short term and be continuously developed in the long term. These concepts are fundamental to the introduction of a broad traceability system. Bollen et al. [15] analyzed the production of apples aiming to determine the steps where mixing of input units occurred and how to minimize them, as traceability is lost because of mixing of inputs without an obvious separation. The mixing of input units causes loss of information related to product origin and product series. The authors observed the whole process with cameras and used markers. It allowed to segment the continuous process without interference and

Safety and quality based traceability system for food supply chains

develop a probability model that allowed the prediction of the dimension of the mixing in each pack. Thus, the packing process became less random and kept the information relative to the processing series, but not to earlier processing stages. However, this study suffers from some issues as the development of a system only usable by the company analyzed and for its present situation. The system may present some difficulty in transmitting information to other entities further down the supply chain due to lack of compatibility between systems. There will also be some difficulty when the production process changes, because it will be necessary to implement a new system that is adequate to the new process. This kind of information, although useful for logistics, does not add value to the product. Thus, it is required a system that does not have to be reformulated when the processing changes, that is compatible with other systems and is able to value products.

In this context, Skoglund and Dejmek [16] used fuzzy traceability to limit the consequences of mixing in continuous production. In this situation, the only obvious separation between inputs are cleaning operations. The deterministic approach of traceability should be complemented by fuzzy traceability as is necessary to acknowledge the presence of elements whose origin cannot be easily determined or quantified. To solve this problem, a turbulent diffusion was used to quantify the presence of an element in another, since this study refers to milk production. By quantifying diffusion, it becomes possible to quantify the percentage of elements corresponding to each origin, as it is possible to identify and quantify the mixing zone. This type of traceability is for internal use and allows to avoid mixing of materials with different qualities, besides facilitating the implementation of management and control systems. Again, this is an application of traceability whose consequences have been discussed above. Still, this type of traceability is very important and useful. The collected information can be transmitted to others by using an external traceability system, but it requires access to company data and equipment to implement it.

Frosch et al. [17] deal with a practical case of an application of a traceability system. However, due to low granularity and low recording frequency, records are kept manually. Measures to improve the system are provided, since it is only applied fish products and not the ingredients added to it. A decade ago, it was underlined the reluctance of companies to the implementation of more capable traceability systems as they consider that the existing methods were enough and that they would not have enough benefits from the investment in more comprehensive systems.

Wang et al. [18] developed a traceability system that monitors current quality and predicts its future variation. It is based on wireless sensors and Radio frequency Identification (RFID) and is constituted by a graphical interface, quality decay prediction module and warning module. The system uses Global Positioning System (GPS) data relative to the vehicle position and information given by the sensors monitoring the commodities. The graphical interface provides real time data, warnings and how to deal with the situation. The warnings are given as soon as the prediction module finds an anomaly that interferes with the satisfaction of an order while in route. This system serves to aid transportation and does not collect traceability information along an entire food supply chain and is restricted to wireless sensors and RFID. Other example is provided by Li et al. (2010) that present a very simple traceability system applied to cucumber production. The objective is to determine the amount of fertilizer to use. The system is composed by a computer, acting as a server, where the information is stored and can be accessed by the decision-support module that will indicate the amount of fertilizer needed in a certain part of the plantation. The access is made via PDA. Despite being a simple and very specific traceability system, it demonstrates very clearly the necessity for traceability systems that do

not involve a large investment and shows the availability of technology to implement it.

In the other hand, Thakur et al. [19] analyzed a grain elevator. The grain was transported to silos where it was analyzed and stored without recording information about its origin. So, it became impossible to link inputs and outputs. To solve this problem, an entity-relationship (ER) model was used. In this model, all entities are characterized by whichever relevant characteristics and relationships are the interaction between entities. Then, a database was created in which entity is assigned a unique identifier and, based on the relationship with other entities, a new identifier is generated. A Relational DataBase Management System (RDBMS) is used to implement the model. By having a well-defined structure for the assignment of identifiers it becomes possible to keep traceability information. Still, some information loss will occur due to mixing within the silos. This study also suffers from the issues mentioned above, but by having the proposed structure it is possible to greatly reduce information loss and it becomes much easier to identify problems' causes and their responsibility.

Karlsen et al. [20] analyzed a traceability system used in the production of farmed salmon with the objective of determining Critical Traceability Points (CTPs) at different levels of granularity. To determine CTPs, the results from triangulating qualitative interviews, observation and documentation analysis were compiled in order to cover the flaws of each method. According to the characteristics of production, several batch sizes were determined and subsequently analyzed in order to determine which yielded the best results for process control and optimization. The analysis criteria are: the capability of maintenance of existing methods, the repetition of information, phenomenon associated with a too fine granularity, and the uselessness of information obtained, which will occur if granularity is too coarse. Granularity does not depend exclusively on the companies, it also depends on regulators, consumers expectations and the type and amount of information they wish to access. This type of analysis is of extreme importance for the correct application of a traceability system on a SME, as correctly determining granularity provides an equilibrium between usable data, investment to make, risk and responsibility.

Borit and Olsen [21] used a traceability system to help stopping illegal fishing by using globally unique identifiers. Uniqueness is guaranteed by GS1 coding (e.g. barcodes). Records of activity compliant with necessary regulations are associated to an identifier and products can be commercialized.

Huang et al. [22] present a very simple, non-isolated, traceability system in which there is cooperation between producers and retailers (supermarkets). It was developed to a client that wishes to access the product data use a machine that reads a code and reports the information. However, it requires the user to have a username and password registered in the system. Although that necessary personal account, it can serve future purposes like studying consumer trends. It is considered, for this study, that such would be cumbersome as it introduces unwanted complexity on a simpler traceability system with no added benefit. End consumers should access product history without need for registration in the traceability system. Also, Pizzuti et al. [23] developed a non-isolated traceability system applicable to agro-food products in which clients can access product history via browser. Such facilitates information sharing, however proprietary tools were used, and the developed structure is here deemed too complex to be applicable to small producers and, with some limitations, to medium sized producers.

Lavelli [24] studied a particular and isolated case of the implementation of a traceability system on a medium-sized corporation dealing with poultry meat. The system is presented as a tool that promotes quality, organization, risk management and product value. The possible requirement of a fuzzy algorithm to determine contamination is highlighted as its absence can translate into a wrong evaluation of contamination and consequently excessively increase the scale of recalls.

Safety and quality based traceability system for food supply chains

Hu et al. [25] developed a traceability system applicable to an entire vegetable production and distribution chain. The presented system is very comprehensive and possible to adapt to other supply chains due to fundamentals concepts upon which it was conceived. These fundamental concepts include monitoring of all processing stages, transport and distribution as well as using web-based services to effectively report information. This kind of comprehensiveness is of extreme importance to the correct application of traceability systems as it maximizes the utility of the collected information and facilitates its transmission.

Trebar et al. [26] developed a sensor-based traceability system. The sensor performance is well discussed comparing to the traceability system. This system is based in the implementation of gateways in predetermined points where monitoring is executed per opposition of continuous monitoring. Such allows the simplification of traceability systems without too great efficiency loss as well as allows the development of a platform for immediate use, due to simplicity and lower implementation costs, that can be further developed over time and company necessities and capacities change.

Parreño-Marchante et al. [27] used the EPCglobal Architecture Framework to map a process, collecting information through interviews. Based on that information, a traceability system adequate to the situation is devised, being capable of not losing information between entities. The system includes a production and distribution chain and, even though it is a single use case, implementation problems are reduced as they only affect entities outside the chain. Liu et al. [28] presents a traceability method for a specific case, a company that produces and commercializes eggs. The presented method is simple in theoretical terms but hard to apply as it needs a significant investment in technology capable of dealing with large volumes of data typical of a video-surveillance system. As technology evolves, Alfian et al. [29] applied a traceability system based on a wireless sensor network. The purpose of this traceability system to is aid distribution and allows to verify that the product did not come from an illegal source. This system was designed to be able to increase its scale but used proprietary tools and such compromises its applicability in other agro-food sectors.

Wang et al. [30] applied a traceability system based on a wireless sensor network that considers Hazard Analysis and Critical Control Points (HACCP) standards to assure quality and safety. The traceability system and the information flow along the whole process are modeled. Despite being a comprehensive work, as it monitors and reports information since the beginning of the process to the end until the final consumer, it is an isolated case and as such, suffers from already discussed issues although not as severely due to its comprehensiveness.

As demonstrated, traceability systems can have a profound impact in productivity, logistics and sustainability. To achieve those benefits, it is necessary to implement traceability correctly. Although the presented systems correctly fulfill the application they were destined to, they sometimes lack comprehensiveness, are applicable to a single company in its present situation or are too demanding to MSE and SME's. Thus, it rises the necessity of developing a traceability model that can be applied to an entire food supply chain, that is flexible enough to allow each company to adjust granularity, with reduced capital investment and that is able to be developed and adapted over time.

Table 2.1 gathers by topic and chronological the application of traceability systems in food context and their granularity.

Table 2.1: Review of application of traceability systems and granularity.

Authors	Products	Granularity	Important concepts
Beulens et al. [14]	Eggs	Undisclosed	Shared quality standards, shared information infrastructure, prototype traceability system integrated in a SME supply chain
Bollen et al. [15]	Apples	Output: packages linked to input bins	Mixing causes traceability information loss, model to determine and minimize mixing
Skoglund and Dejmek[16]	Milk	Output: packages linked to input silos	Fuzzy traceability, batch virtualization, mixing algorithm
Frosch et al. [17]	Fish	Output: lots linked to fishing vessels	Measures to improve traceability in a manual traceability system scenario
Wang et al.[18]	Undisclosed	Containers	Traceability as a tool to improve distribution and minimize capital loss and waste
Li et al. [31]	Cucumbers	Terrain lot	Traceability as tool to aid production and comply with regulations
Thakur et al. [19]	Grain	Shipment to customer	Implementation of traceability in a heavy mixing scenario, quality evaluation
Karlsen et al. [20]	Fish	Several tested	Impact of granularity in the usefulness and cost of a traceability system
Borit and Olsen [21]	Fish	Dependent on the records of activity	Traceability as a tool to enforce regulations and quality
Huang et al. [22]	Red jujubes	Final product batch	Consumer access to traceability information, traceability system composed by several different subsystems

Safety and quality based traceability system for food supply chains

Table 2.1: Review of application of traceability systems and granularity (continued).

Authors	Products	Granularity	Important concepts
Pizzuti et al. [23]	Frozen vegetables	Dependent of the company	Model for frozen vegetables, supply chain traceability
Lavelli [24]	Poultry meat	Final product lot	Traceability as a tool to comply with regulations, fuzzy traceability
Hu et al. [25]	Vegetables	Variable, dependent on the stakeholder	Supply chain traceability
Trebar et al. [26]	Fish	Box	Supply chain traceability based on wireless sensors
Parreño-Marchante et al. [27]	Fish	Box	Application of a traceability system in two SME's
Liu et al. [28]	Eggs	Dependent on the stakeholder	Real time traceability system in a Chinese egg supply chain
Alfian et al. [29]	Kimchi	Box	e-pedigree traceability system with temperature and humidity monitoring
Wang et al. [30]	Peach	Undisclosed	Sensor based supply chain traceability system

2.2 Benefits, necessities, requirements, obstacles and components of traceability systems

This section highlights the benefits of traceability systems, required concepts for their correct operation, demonstrates their necessity, exposes obstacles to implementation and concepts or technology that composes them.

Jansen-Vullers et al. [32] indicate that traceability is composed by four elements: (1) physical integrity, relative to batch dimension and separation; (2) data storage, relative to movements and processes; (3) product identification, linked to the previous element in order to identify how a commodity was produced; and (4) report of the information obtained. Traceability has some requirements, namely historical relationships between batches, operations over batches and the values associated to those operations and used means. Batch is defined as the quantity jointly produced, sharing the same characteristics and production cost. The necessity of recording used raw material batches in order to determine the constitution of product batches is highlighted, as well as to avoid aggregating information relative to production within the product batches. Otherwise, it becomes difficult to determine the origin of problems that arise. Such data should be associated with the production stages. A traceability system should be able to model the relationship between stage dependent and independent data, and to model the composition of products, model operations, properties of the products and productions assets. However, the scope of this study is restricted to non-cyclical and convergent processes, which is not always the case of the food industry.

Bogataj et al. [33] studied the drop in value of products in a food production and distribution chain. It is stated that these chains need highly visible and well monitored material flows. On that perspective, there is the necessity for traceability systems based on food quality and safety as they allow that a given product is in the correct location at the correct time. That characteristic is important as the purchase or sale at the best price.

Regattieri et al. [34] stated that traceability is based on four pillars: (1) product identification, (2) data to trace, (3) product routing and (4) traceability tools. Inadequate exposition of information can lead to the emergence of copies and the increase in price, which can negatively affect the food sector as it is susceptible to small price variations. Even so, the correct implementation of traceability systems translates in process optimization, increased security and quality, more adequate marketing and competitive edges. To be efficient, a traceability system must transmit information about commodities in a precise, timely, complete and consistent manner. This also leads to faster problem resolution as they enable the efficient determination of issues and affected products. Also, Gessner et al. [35] discussed the importance of maintaining correct records using information technologies and their advantages per opposition to manual records. Information technologies can record information with a level of detail and precision that would be unfeasible manually. The authors refer some alimentary crisis that occurred in United States of America. It then becomes important to add that food crisis can never be fully averted due to the occurrence of errors related to limitations of the traceability systems themselves or inadequate use, but they can be greatly reduced through a broader and more effective quality control. Additionally, Jedermann and Lang [36] monitored a product's quality during transport through a wireless sensor network. The authors demonstrate that the products are subjected to greater and potentially more dangerous variations during transport than an average value can show. Thus, it emerges the necessity of careful quality monitoring during transport.

Hsu et al. [37] analyzed a traceability system applied to very simple fish supply case. The authors highlight the necessity of automatic traceability systems in order to be more efficient.

Donnelly et al. [38] studied transformations in lamb meat and their impact in keeping traceability information. Transformations are critical points for traceability because it is easy to lose information about the constituents of a product and not always information is added about the transformations. This leads to loss of traceability because determining a product history becomes difficult due to gaps in history. Meanwhile, Chrysochou et al. [39] inquired 107 persons from 12 different countries to study the factors that influence the consumers' perspective about traceability information carriers. The authors concluded that the most important factors are: (1) trust in information, (2) convenience, (3) impact on quality and safety of the products, (4) consumer and environmental health impact and (5) potential ethical consequences in what concerns privacy. This last factor is of extreme importance and constitutes a big obstacle to the implementation of comprehensive traceability systems.

Karlsen et al. [40] indicate that the correct application of a traceability system implies four phases. The first consists in tracing the company's information and material flowchart. The second phase consists in the application of the flowchart determined before, result analysis, possible corrections and inclusion of a method to uniquely identify materials and companies. The third phase consists in the application of the previous phase and the fourth phase consists in the analysis of results of the previous phases and in the implementation of necessary corrections. Furthermore, van Der Vorst et al. [41] studied the influence of quality variation on logistic decisions. Monitoring of quality has the potential of positively affecting those decisions as result of analyzing production performance and reduction of waste. This concept can easily

Safety and quality based traceability system for food supply chains

be integrated in a quality and safety based traceability system, since the monitoring is already being executed by the system. Thus, it becomes clear that traceability and performance analysis are just two aspects, or two consequences, of information monitoring.

Bosona and Gebresenbet [42] described traceability systems as an essential tool for food supply chain logistics. Using traceability systems based on information technologies allows an increase in planning capacity and in the efficiency of logistic procedures. The main driving forces that push the development of this systems are also described. They are: (1) regulations, for example, the European General Food Law; (2) social, as consumers become aware of what they consume and want more information in order to improve their capability to select products and to be able to trust what is consumed; (3) safety and quality, as they prevent the transmission of diseases to the public and also enables the determination of infected products and halt their spread; (4) economic, this is the less influential driving force, as it is common for the acquisition traceability systems to be initially expensive; and (5) technological, as the onset of cheaper technologies promotes the development of systems by considerably reducing the initial cost. Some advantages and obstacles of traceability systems are referred: (1) increase of consumer satisfaction; (2) better management in a crisis scenario; (3) reduction of logistic costs; (4) closer contact with the consumer and the development of more adequate products; and (5) contribution to technological and scientific development and to sustainability, resource limitations, limited available information, lack of information structure, limits in the ability to operate the system and lack of information about the benefits of traceability systems. Moreover, Storoy et al. [43] indicated some general principles of traceability systems: (1) identification of units to trace, (2) documentation of transformations and (3) standardization of information exchange. Due to lack of standardization, information exchange is inefficient as it consumes much time and funds. On an individual level, large-dimension companies, have made significant progress in what concerns these systems. Still, those systems are often proprietary and for internal use. Such makes difficult to exchange information between entities. The food sector is very concerned with sharing data due to its sensitive nature, and refuse to share it without safety guarantees. The structure used by the authors recommends the usage of the GS1 and the use of a specific language for each activity sector. For the purpose of this study, using GS1 is recommended for its worldwide use, but a specific language for each sector may introduce unnecessary complexity when uniqueness is guaranteed.

Aung and Chang [44] stated that globalization led food commodities to travel longer distances between producers and consumers. Due to alimentary crisis, consumers demand quality, safety and transparency. Alimentary crisis has repercussions on different levels. Socially, there is the danger to public health, damage to commerce and tourism and litigation. Economically, there is a reduction in activity and high health costs. Environmentally, there is an increase in waste, where due to decomposition methane is created. In this sense, traceability systems have been gaining importance as they can have a profound impact in efficiency and sustainability. They allow better management, quality and safety tracing and monitoring and allow to differentiate products with attributes difficult to differentiate. There is necessity to monitor and trace quality throughout the entire chain in order to assure quality and safety. Thus, traceability systems should be characterized by their scope, the quantity of information obtained, depth, how high or low in the chain can a product's history be traced, and finally, precision, related to the degree of safety that a given product can be identified and its movements described. Also, Asioli et al. [45] stated that exist few empirical studies about the determinants of different levels of traceability capacity at company level, the incentive to the implementation of traceability technologies and their performance. It is easy to define costs but hard to define benefits since

they are often intangible. Sixty Italian companies belonging to the fishing sector were inquired and it is concluded that the implementation and maintenance costs were, usually, lower than expected, but so were the benefits. The authors also concluded that the precision of a traceability system is often more preponderant than its scope. Reduced benefits and the evaluation of traceability systems made from the point of view of the consumers has been making the adoption of these systems somewhat slow. It is stated that a traceability solution that is too comprehensive is unable to consider the specificities of each company and that a more specific solution is required. For the purpose of this study, it is considered that such specification would serve as an obstacle to information sharing between different entities, but that may be more efficient on an internal level. Additionally, Dabbene et al. [46] indicated the main causes of recalls to be incorrect labeling, faults during production, incorrect packing and lack of identification of conditions that compromise safety. This situation leads to loss of consumer trust and costs with withdrawals and destruction of affected products. Most companies are unable to correctly estimate the amount of product that needs to be removed nor have efficient methods to assure an effective recall. Traceability systems can help companies to execute better recalls and so, a good performance indicator is the capacity to reduce capital loss associated to recalls. Another performance indicator is the existence of a systemic information loss. Traceability by itself does not increase the value of a product but helps other systems to identify desirable characteristics, communicate them and their value. Using an automatic traceability system can increase the precision of collected data and the capacity to identify a unit. There has been an emergence of industrial standards for traceability that ease information exchange. Traceability is regulated on three levels, national and international laws and regulations and additional certification. Aiello et al. [47] studied traceability systems as instruments that potentiate economic gains of a company. With the emergence of non-invasive technologies that allow the evaluation of data automatically and in real time, there is a good opportunity to optimize the amount of materials to order and when to promote a sale. However, the implementation of these systems often requires costs with external consultants, software acquisition and staff training. As the granularity also has a large impact in the cost and practicability of traceability systems, a model to assess the ideal granularity is presented. However, this study is limited to systems supported by RFID, becoming limited to companies that deal with low cost food products. At the same time Germani et al. [48] explained the necessity of adding information about all elements that have an effect in the environmental sustainability as is the case of, among others, energy consumption and chemical debris. This addition is important to traceability systems. Flaws and opportunities to improve the degree of sustainability and efficiency of any given process can be determined by carefully monitoring these types of parameters. Furthermore, Thakur and Forås [49] explored a system that monitors temperature over time. This system uses EPC and the collected data can be viewed online. It is demonstrated that the products suffer greater temperature variations than expected. Also, Forås et al. [50] portrayed the evolution of traceability systems in Norway, a country that assisted to a considerable evolution of those systems in terms of comprehensiveness and robustness. Authors affirm that an efficient traceability system should be able to track and trace whichever identifiable unit and should possess the capacity to unequivocally identify it, its transformations and report that information effectively. Hsiao and Huang [51] studied how the information relative to temperature history is transmitted throughout food supply chains. It is concluded that this exchange is easier between producers and consumers than between suppliers and producers. Thus, there is lack of transparency in information exchange. It should be noted that this lack of transparency can easily occur with the history of other parameters. A traceability system can greatly limit this lack of transparency

Safety and quality based traceability system for food supply chains

when information exchange is imposed and its veracity verified by an impartial external authority. Meanwhile, Alonso-Rorís et al. [52] discussed about the necessity of a low-cost traceability system that can be used by several companies and that is also customizable in order to fulfill each company's individual necessities. Such solves one of the main issues associated with these systems, the high cost. This proposal leads to the implementation of flexibility and commonality in information exchange.

Dandage et al. [53] described the Indian food sector, the waste, the fraud, the insecurity. Those concepts are used as justification for the need of food traceability systems. Although the Indian situation is worse than other world regions, those circumstances are observed globally, albeit with different dimensions. In this sense, Raak et al. [54] presented the sources of waste of food products. Themes like overproduction, bad logistics, human errors, equipment faults, residues, power outages, among others were approached. Traceability systems can minimize the impact of such circumstances due to quality focused monitoring.

Olsen and Borit [55] present the necessary components of a traceability system: mechanism to identify resource units, mechanism to identify the transformations applied to those units and mechanism to store information relative to transformations. A traceability system should be able to answer the following questions: how is the identifier associated to the resource units? What is the structure of the identifier? In what context is the identifier unique? How is the information relative to transformations stored? How are the percentages of constituents stored? How is the data relative to each transformation stored? Facts do not exist in a traceability system, only affirmations and means of verification. Such happens because of errors in data collection or preservation. By other hand, Matzembacher et al. [56] characterized food supply chains by their inability to connect information, lack of precision and lack of capability in supplying timely fundamental data in case of crisis. Traceability systems are pointed as a solution as they facilitate the determination of deficiencies and speed up the response. The same technological evolution that allows the automation of traceability systems has been accompanied by the evolution in the food processing. This condition means that the technology that allows a closer proximity with the food products is accompanied by technology that reduces that same familiarity. Special attention is given to packaging to increase familiarization, as it is the part of the product containing more information about any characteristic that is not immediately obvious for the consumers who are willing to pay more for more information. Traceability systems can amplify and speed up the process of obtaining that information. By doing that, consumer trust is retained since the transparency between companies and is increased. There are several obstacles to the implementation of traceability systems: (1) lack of awareness about the benefits, being considered as bureaucracy, (2) fear or unwillingness on the part of business partners, (3) economic limitations, (4) limits in the available information and (5) lack of standard structure for information sharing. To achieve the biggest benefits, traceability systems should be implemented throughout entire food supply chains, although it is very difficult to achieve due to different companies' structures, rules and system along the chain. During that period, Ndraha et al. [57] studied temperature abuse in food supply chains. Such happens due to diverse factors as employed practices, equipment limitations and the position of the commodities during storage and distribution. Although these concepts are not a part of this study, the introduction of a traceability system capable of correct monitoring should be able to report information that allows the timely identification and resolution of these issues. Additionally, Stranieri et al. [58] analyzed the Italian wine sector, in which have been implemented traceability systems. Thirty-nine companies of the sector were inquired in order to determine the reasons that led to the adoption of these systems. Some adhered to comply with legal requirements, others to

avoid fraudulent activity, social pressure or voluntary certification. This study demonstrates that companies invest in more complex systems when they believe that traceability systems are valuable tools. When the adoption is made just to comply with legal requirements, enterprises prefer simpler and flexible systems. Voluntary adoption of more complex traceability systems usually means that a reorganization of whole chain must take place. This reorganization stimulates an effective distribution of responsibility between stakeholders and an increase in the transparency of transactions. Even so, companies that adopted more complex systems sometimes have difficulty to determine how to apply and how much to invest.

Óskarsdóttir and Oddsson [59] underline the difficulty to assure the quality of food commodities as their production often involves many stages and their transport can be on an international level. Traceability systems allows (1) reducing product recalls and alimentary crisis, (2) reinforcing reputation, (3) protecting brands, (4) increasing management efficiency, (5) gaining competitive edges, (6) improving access to markets, (7) assuring consumer trust, (8) reaching legislative or complimentary certification objectives, (9) reaching consumer expectations, (10) reducing risk and responsibility, (11) reducing operating costs and stocks, and (12) improving inventory management and provisioning. Several traceability technologies are compiled and recommend which to use based on a decision tree. Still, the biggest challenge to the implementation of traceability systems is the lack of a standard structure for sharing data.

The necessity of traceability systems is clearly demonstrated by the need for regulatory compliance, consumer demand for safety, quality and transparency, corporate necessity to avoid fraudulent activity and inability to efficiently execute a recall. Although recalls may have several causes, being unable to adequately remove unsafe products can have severe consequences as food crisis and their associated impact on society. Thus, traceability systems can increase security and quality, optimize logistics and production, potentiate capital gains and increase consumer satisfaction. To achieve these results, traceability systems have to be able to model the supply chain, the companies using them and all transformations associated. To have those abilities, it is necessary to identify all batches, whether inputs or outputs, document all operations over batches and communicate all information to an external impartial authority able to scientifically assess the validity of the information. Still, there are several obstacles that restraint the development and deployment of traceability systems. These include, high costs, reduced available information and capacity to operate the systems, reluctance to the implementation by business partners and lack of information sharing structure. In order to effectively develop and implement comprehensive chain-wide traceability systems, these issues need to be addressed, if not, traceability systems will lose most of their utility and benefits.

Table 2.2 summarizes the benefits, necessity, requirements, obstacles and components of traceability systems described in the relevant scientific literature.

Safety and quality based traceability system for food supply chains

Table 2.2: Scientific research covering the benefits, necessity, requirements, obstacles and components of traceability systems.

Authors	Important concepts
Jansen-Vullers et al. [32]	Elements and requirements of traceability systems
Bogataj et al. [33]	Benefits of traceability systems
Authors	Important concepts
Regattieri et al. [34]	Pillars of traceability systems, consequences of inadequate information exposition, benefits and requirements of efficient traceability systems
Gessner et al. [35]	Advantages of using IT technology to record information and necessity of traceability systems to avoid alimentary crisis
Jedermann and Lang [36]	Necessity of quality monitoring traceability systems due to the subjection of products to greater variations than expected
Hsu et al. [37]	Benefits of automatic traceability
Donnelly et al. [38]	Necessity of recording information about transformations applied to products
Chrysochou et al. [39]	Consumers' need for information and impact of that information
Karlsen et al. [40]	Phases of implementation of traceability systems
van Der Vorst et al. [41]	Benefits of monitoring in logistic decisions
Bosona and Gebresenbet [42]	Increase in supply chain logistic efficiency, main driving forces behind the development of traceability systems, advantages and obstacles
Storoy et al. [43]	General principles of traceability systems, review on the state of information exchange and necessity of safety guarantees for information exchange
Aung and Chang [44]	Necessity of traceability systems due to the impact of food crisis, characterization of traceability systems and benefits of application
Asioli et al. [45]	Review on the factors that led Italian wine companies to implement traceability systems and cost benefit description
Dabbene et al. [46]	Causes and consequences of recalls, traceability systems proposed as solution, performance indicators of those systems and levels of regulation
Aiello et al. [47]	Benefits and obstacles of RFID based traceability systems and model to assess optimal granularity
Germani et al. [48]	Monitoring elements that have an impact in sustainability can lead to opportunities to process optimization
Thakur and Forås [49]	Demonstration of variations subjected to products

Safety and quality based traceability system for food supply chains

Table 2.2: Scientific research covering the benefits, necessity, requirements, obstacles and components of traceability systems (continued).

Authors	Important concepts
Forås et al. [50]	Requirements of robust traceability systems
Hsiao and Huang [51]	Need for traceability systems to increase transparency in information exchange
Alonso-Rorís et al. [52]	Necessity for low-cost and adaptable traceability systems
Dandage et al. [53]	Traceability systems as tools to avoid waste, fraud and insecurity
Raak et al. [54]	Causes of waste and traceability systems as tools to avoid it
Olsen and Borit [55]	Essential components of traceability systems and useful questions for performance evaluation
Matzembacher et al. [56]	Corporate inability to provide useful and timely data for the resolution of food crisis, consumers' willingness to pay for information, advantages and obstacles of food traceability systems
Ndraha et al. [57]	Traceability systems as tools to prevent temperature abuse in food supply chains
Stranieri et al. [58]	Factors leading to the adoption of traceability systems and corporate necessities
Óskarsdóttir and Oddsson [59]	Necessity of traceability systems to monitor quality and technology capable of being incorporated on a traceability system

2.3 Models, methods, algorithms and supply chain management

After describing the application of traceability systems in food context and their granularity, as well as its benefits, necessity, requirements, obstacles and components, this last section presents traceability and quality models, their use methods, usable algorithms and concepts relative to supply chain management.

Sloof et al. [60] used two methods to model quality. One is through the variation of inherent properties of a perishable commodity that degrade over time. The other is based on the value that the consumers attribute to those properties. However, this value is appreciated differently according to the application and consumer experience. By monitoring and quantifying inherent properties and reporting them, a traceability system provides tangibility to those characteristics. Thus, it becomes possible to attribute a value to them. So, quality can be quantified by the additional value conferred by any given property. This value is highly subjective but, due to the self-regulatory nature of markets, it will stabilize over time. That makes the value more predictable, less subjective and easier to attribute.

Bechini et al. [61] present a simpler architecture for traceability systems than other authors. A simple architecture is more advantageous in the short term due to its easy implementation. In the long term presents a simple development as well as allows smoothing the transition to a fully automated traceability system with minimal loss of autonomy by the companies. Although other traceability systems can correctly operate as they were presented, they are, sometimes, too demanding in their requirements for implementation. In this sense, it is considered that there is a lack of a single traceability system that eases the transition between the absence

Safety and quality based traceability system for food supply chains

of a traceability system to a fully automated and efficient one. By other hand, Jedermann et al. [62] present a concept for a wireless sensor possessing autonomous configuration, real time access and capable of autonomous verification and decision. The objective is to analyze the conditions subjected to products during transport and alter the route as necessary. In this case the information is collected with the purpose of aiding transport logistics and management. This information could be used for traceability by detailing product history during transport to other stakeholders.

Hsu et al. [63] present an algorithm for distribution of food perishables. The study highlights the importance of a good distribution system in the maintenance of nutrients and that sometimes forces a less efficient distribution. The premise of the algorithm is the delivery of the goods in the best quality possible. Due to quality monitoring and consequent quantification it becomes possible to deliver food commodities in more desirable quality levels, depending on the application. In other context, but on the same topic, Kelepouris et al. [64] present a model for a traceability system based on Radio-Frequency Identification (RFID) and Electronic Product Code (EPC) to be used in the agricultural sector. The authors highlight that companies that deal with perishables food products are of small or medium dimension and usually do not possess the necessary means to implement traceability systems, especially if fine granularity is required. Besides, these companies do not want to risk spending considerable resources in these systems if their benefits are not tangible and well defined. Also in this sense, Heese [65] explored inventory management through the use of wireless sensors. Correctly managing inventory is considered essential to keep a record of the quality variation. Although not necessary in a traceability system, its inclusion can help reducing the probability of error by keeping a closer watch in the products that are not immediately processed or dispatched.

Xiaofeng et al. [66] developed a model that allows to determine the correct allocation of commodities in order to maximize profit. The authors expand on models for price variation in the sense that it also considers the entry of materials and not only the exit. Also, Kwok et al. [67] created a software supported by wireless sensors with the objective of ensuring authenticity of a given product. The model of the system and its constituents are presented in detail. Its constitution was limited to product authentication, but it can be considered as a function of a more comprehensive traceability system. With the same objective, Bechini et al. [68] try to model how different parts of a food supply chain can relate and exchange information through the introduction of a common communication paradigm. Extensible Markup Language (XML) was used to obtain information of stakeholders. The elaboration of a model that allows information sharing in a standardized manner is necessary.

Woo et al. [69] present an architecture for a wireless sensor and passive RFID based traceability system. Makes use of the ER model, but it adds a temporal dimension to incorporate quality degradation. This last characteristic is very important as the ER model does not consider the time consumed by each entity or by the relationships between them. Being restricted to the mentioned sensors, the concepts that can be extrapolated to other traceability systems are limited. Meanwhile, van der Vorst et al. [70] explored the concepts of Least Shelf-life First Out (LSFO) versus First In, First Out (FIFO) and simulates its utility in a sliced pineapple supply chain. Although this study focuses on the logistic analysis and the increase of the chain efficiency, some very important concepts were described. Those concepts are the utilization of maximum, minimum and average values for an effective monitoring of remaining shelf-life. It is advised to slightly alter to maximum, minimum and real current values in a real-time monitoring case. It is considered these values change eases the efficiency analysis of a given production stage by its closeness to the maximum or minimum values. It also allows a better price scaling

through more precision and easier visualization of monitored data. Covering the same subject, Zhou et al. [71] analyzed the information sharing between sellers and buyers. In the context of the present study, the information sharing should be left to the seller's consideration but restricting it, at its minimum, to legal requirements. This condition allows to preserve corporate autonomy and flexibility on how to deal with customers. Also, Hu et al. [72] present a model for a traceability system to use in cases where there is a relatively high probability of contamination. An algorithm was developed that considers the presence of a certain material in a product group. Additionally, Thakur and Hurburgh [73] define methods for bulk grain internal and supply chain traceability. A Unified Modeling Language (UML) Use Case diagram was elaborated for supply chain traceability. The UML Use Case diagram indicates the requirements of the traceability system and which stakeholders require it. The requirements were: (1) record of breeding practices, (2) farming practices, (3) handling and storage practices, (4) processing practices, (5) authenticate claims, (6) compliance with food safety regulations, (7) protect integrity of brand name and (8) document chain of custody. Another UML based model was developed in order to define information exchange. A function modeling methodology for describing manufacturing functions, namely "Integrated computer aided manufacturing DEFinition for Function Modeling" (IDEF0) was used for internal traceability. The model inputs were business needs, consumer preferences and regulatory needs, which also served as control for the model. Developing the internal traceability system implies several steps, namely: determination of the traceability plan, implementation of the traceability plan, evaluation of the system performance, system validation and system maintenance. Information exchange between stakeholders can be done using Electronic Data Interchange (EDI), Extensible Markup Language (XML), TraceCore XML (TCX) and a relational database management system. The presented methodology can encompass enterprises from small to large dimension, and although it was developed with the bulk grain supply chain in mind, the inherent concepts leading to its development can be applied to other supply chains.

Olsen and Aschan [74] discuss some benefits of traceability systems as the rationing of logistic information and competitive advantage through improvement of documentation. The application of these systems is dependent of the product, commercial obligations, size of the company and the supply chain it belongs to. To successfully apply a traceability system, it is necessary to develop a model of food supply chains. A methodology is defined to develop those models considering limitations of stakeholders, privacy and data access. It implies interviewing staff in order to develop the model. As effective as the developed traceability system may be, its application may be slow since all stakeholders must wait for the interviewing process, information compilation, treatment and application.

Thakur et al. [75] present a method to keep traceability information using the Electronic Product Code Information Services (EPCIS) framework, using RFID and unique identifiers generated electronically. This framework is used to avoid proprietary systems that lose much utility for their inability to transmit information to others due to lack of compatibility. It records and reports information about the transport of commodities. So, the information about the transformations or mixing is not kept and relies on the next event to describe the product. This is a common issue, and since it is an external traceability framework, the information relative to the processes subjected to commodities is often accumulated on the final identifiers. This condition happens due to lack of a traceability system capable of operating externally and internally.

Karlsen et al. [76] discuss the concept of granularity in a specific case as to better understand how traceability is affected by it. It is possible to use bigger units when the risk of contamination is low and quality control is less strict. The use of Internet based traceability systems

significantly facilitates information sharing without compromising the control of each company over its own information and the necessities of its users. The authors reinforce that the knowledge of all costs allows the implementation of a system whose granularity is adequate to the situation. Additionally, Bakker et al. [77] highlight the necessity for inventory management systems to monitor distinct occurrences in distinct locations and accommodate pricing variation induced by quality variation. Inventory management systems should be a part of traceability systems in order to monitor heterogeneous quality decay of materials or products that are not used or sold immediately. Also, Wang and Li [78] stated that expiration dates are unable to expose the variations subjected to products. The study underlines the importance of correct monitoring since the quality decay leads to a demand decay. To keep product demand, the use of quality stages and associated discount per stage is proposed. A method to determine the optimum discount of each stage is presented. This model is dependent of specific characteristics of each corporation. Still, it serves as a starting point to a local application capable of automatically applying the discounts. In this context, Tromp et al. [79] explored the advantages of dynamic expiration date reinforcing the interest and utility of dynamic expiration dates. Also, Grunow and Piramuthu [80] explored expiration dates associated to perishable products. The study states that, often, the products will have enough quality for consumption and that expiration dates are conservative estimates. Thus, there is a security margin for the consumption of perishables. Nevertheless, it may cause additional waste. This consequence can be avoided by better monitoring of the properties that influence security and quality. The use of RFID to monitor that properties is presented as solution. Additionally, models are described for the determination of the quality level and to assess the profit from the use of that technology, through the value of the information transmitted to the consumer. Despite being a comprehensive solution, it is limited to RFID, which limits this system's application range. Therefore, there is a necessity for more versatile models that allow the use of several different technologies. To overcome this limitation, Verdouw et al. [81] proposed a method to trace an operational map based in the Internet of Things (IoT) perspective on a floricultural supply chain. Although it focus on a specific application, the inherent concepts used can be applied to other cases with other perishables.

Piramuthu et al. [82] discuss about the utility of traceability systems in the loss reduction in products' recall whose levels of quality and safety are not acceptable for consumption. A model that can determine economic loss reduction is presented. The model does not consider transportation costs, monitoring system, marketing and considers that the quality decay starts after production. Although, it demonstrates the utility of correctly monitoring the quality of perishable products.

To be able to manage the previous assets, Pizzuti et al. [83] present the Food Track&Trace Ontology as an answer to the lack of standards for storing and sharing information. To contribute in this sense, Pahl and Voß [84] highlight the need to take into account the deterioration of food perishables in their distribution and presents some algorithms to determine that deterioration. Meanwhile, Hertog et al. [85] present a simple generic algorithm to determine product quality. Based on the evaluation of quality, an algorithm was developed that allows to optimize the product distribution. Although the algorithm is computational processing demanding, small scale application should not become excessively time consuming. The algorithm is based on the First Expired, First Out (FEFO) model, that authors consider an evolution of the FIFO model. This model implies that the transport to the consumer is considered. As other authors, it is underlined the lack of compatibility between existing systems and the enormous obstacle that such condition represents. In this context, Jedermann et al. [86] underlines the importance

of monitoring quality and remaining shelf-life in food supply chains as a method to increase efficiency and reduce waste. Compares the results of FIFO model, commonly used, to the FEFO and LSFO models. This last two models are more efficient logistics wise and ensure more gain and less waste, but need a more careful and demanding monitoring in order to be applied. The main difference between the last two models is that the first considers the time-window until the product becomes improper for consumption and the latter considers the time-window until it becomes no longer salable. This distinction is very important as the consumers tend to avoid products nearing their validity. In this sense, the LSFO model is more useful to portray cases where a company produces or distributes the commodities and another sells it to the end consumer.

Badia-Melis et al. [87] present trends and necessities of food traceability systems. The Critical Tracking Event that relates to all information about the events that manipulate the products is presented. The Food Track&Trace Ontology, already used in other research works, is characterized by agents, products, raw materials, used means and methods, is also presented. The TraceFood framework is presented. This framework defines a non-proprietary structure for information sharing on an international level. The theme of intelligent traceability focused on quality to reduce waste is discussed. On this topic, FEFO is compared to the common FIFO methodology. Fuzzy Cognitive Maps to determine mixing between inputs are also discussed. The IoT perspective is highlighted as a mean to present consumers with the cumulative history of products, although there is lacking a structure for storing and sharing information.

Saak [88] presents a simple hypothetical case to determine the increase of product value due to the information provided by a traceability system. Some applicability is lost due to the hypothetical basis, but is, nevertheless, a good starting point to real case analysis.

Qian et al. [89] discuss on the granularity of traceability systems. The finer the granularity more precise will be the information collected and easier it becomes to solve problems. For a traceability system to be efficient, granularity should allow a precise description of the process. A method to determine the optimal granularity of a traceability system based on queries made to entities in the sector is proposed. That value will have to be determined for each case through evaluation of objectives and capabilities of each company. In other context, Gaukler et al. [90] explored the benefits of dynamic expiration dates per opposition to fixed ones and uses a real case to demonstrate them. The benefits are the same as those exposed by other authors, such as, increase in efficiency, sustainability and profit and waste reduction. However, the determination of the uncertainty when measuring quality is highlighted as well as the necessity of always considering the quality wherewith a product is acquired and the quality when a product is manipulated. By other hand, Accorsi et al. [91] developed software applicable to entire food supply chains with the objective to help planning, analysis and control using a graphic interface and IoT. Traceability is not a part of this software but it would be a future expansion opportunity since the software is not restricted to specific food supply chains.

Óskarsdóttir and Oddsson [92] explore information technologies that can be used in traceability systems. A method was developed to aid the selection of the most appropriate technology considering a company's necessities and objectives. It is introduced the concept of gateways, which in this context consists in monitoring in well-defined points of a process. This concept is very important as it facilitates the implementation of traceability systems and reduces the costs of implementation.

Table 2.3 briefly describes traceability and quality models, their use methods, usable algorithms and concepts relative to supply chain management.

Safety and quality based traceability system for food supply chains

Table 2.3: Important concepts concerning models, methods, algorithms, and supply chain management.

Authors	Important concepts
Sloof et al. [60]	Quality modeling through variation of inherent properties and the value consumers attribute to them
Bechini et al. [61]	Simple architecture for traceability systems
Jedermann et al. [62]	Concept for an autonomous wireless sensor
Hsu et al. [63]	Algorithm for distribution of perishables based on their deliverance at the highest possible quality
Kelepouris et al. [64]	Model for agricultural traceability system based on RFID and EPC
Heese [65]	Inventory management using wireless sensors
Xiaofeng et al. [66]	Model for the correct allocation of commodities in order to maximize profit
Kwok et al. [67]	System model and software for wireless sensor-based product authentication
Bechini et al. [68]	Model for information sharing between stakeholders in a supply chain based on XML
Woo et al. [69]	Model for sensor-based traceability system using the ER model adding the temporal dimension
van Der Vorst et al. [70]	Comparison of LSFO vs FIFO in a supply chain; max., min. and avg. values to determine the quality uncertainty
Zhou et al. [71]	Review on information sharing between sellers and buyers
Hu et al. [72]	Model for a traceability systems usable in high probability of contamination scenarios and algorithm to determine contamination
Thakur and Hurburgh [73]	Methods to elaborate internal and external traceability models
Olsen and Aschan [74]	Benefits of traceability systems and methodology for modeling food supply chains
Thakur et al. citeThakur2011a	Framework for traceability using EPCIS and RFID
Karlsen et al. [76]	Discussion on granularity and its effect on traceability
Bakker et al. [77]	Necessity of inventory management as part of traceability systems to monitor heterogeneous quality decay
Wang and Li [78]	Fixed expiration dates are inefficient to expose variations subjected to products; Algorithm to apply discount according to quality variation to keep demand
Tromp et al. [79]	Utility and benefits of dynamic expiration dates

Safety and quality based traceability system for food supply chains

Table 2.3: Important concepts concerning models, methods, algorithms, and supply chain management (continued).

Authors	Important concepts
Grunow and Piramuthu [80]	Fixed expiration dates are inefficient product quality. Quality assessment using RFID and profit analysis
Verdouw et al. [81]	Method to virtualize operations based on the IoT perspective
Piramuthu et al. [82]	Traceability systems as tools to increase recall efficiency and model to determine economic loss reduction due to the use of those systems
Pizzuti et al. [83]	Model for the standardization of information storage and sharing
Pahl and Voß [84]	Algorithms to determine deterioration of food perishables
Hertog et al. [85]	Algorithm to evaluate quality decay and to distribute perishables based in FEFO
Jedermann et al. [86]	FIFO versus FEFO and LSFO
Badia-Melis et al. [87]	Review on trends and necessities of traceability systems, concept of CTE; FTTO and the TraceFood, FIFO vs FEFO and FCM to determine mixing
Saak [88]	Increase in product value due to information provided by traceability systems
Qian et al. [89]	Methodology to determine optimal granularity
Gaukler et al. [90]	Benefits of dynamic expiration dates per opposition to fixed expiration dates
Accorsi et al. [91]	Software to aid planning, analysis and control using IoT
Óskarsdóttir and Oddsson [92]	Methodology to select the most appropriate technology to incorporate in a traceability system

Chapter 3

Materials and methods

A traceability model was developed with the analysis performed in the previous chapter. This model encompasses all stages of production, from the very beginning in farms to the very end when the product reaches the final consumer, i.e. from farm to fork. This model aims to help reducing waste while improving sustainability and profit by optimizing production with minimal interference and investment. It is an enterprise focused model giving special attention to MSE's. It is considered that this kind of focus is more beneficial to companies, regulators and consumers due to the increased acceptance from the companies.

As this model was developed with modularity in mind, its constituents are, naturally, the modules. These modules include inventory management, processing stage management, order management, quality validations and unique codes. These modules can be improved or even removed as necessity and limitations dictate, if such changes do not interfere with the flow of information.

The flow of information is the most rigid part of this model as it depends on that flow to operate properly and not on the kind of information that is passed on between stakeholders. The flow of information is the most rigid part of this model as it depends on that flow to function properly and not on the kind of information that is passed on between stakeholders.

The development environment refers to the language, software and hardware used during the development of the module templates. It was purposefully kept restrictive to better emulate the conditions that MSE's and SME's are often subjected due to both budget limitations and investment reluctance. The hardware does not require special features and technical specifications for processing the traceability system. All software used for development was completely free for self-development, restricted however to its free distribution. Pricing the distribution of the software created will incur in legal and monetary penalties. As this is a prototype system and is intended, no matter the course of future development, to be freely available.

All information described here is also available in *SME And Quality Focused Traceability Architecture To Increase Sustainability And Profit* [93] and *Traceability in food supply chains: SME focused traceability framework for chain wide quality and safety - Part 2* [94].

3.1 Traceability model overview

As presented before, this model encompasses all stages between the first acquisition of raw materials to the final consumers. The model diagram can be seen in Figure 3.1. It is considered that focusing the model in enterprises that compose the agri-food sector is the easiest and most beneficial for all stakeholders. As can be seen, neither final consumers nor regulators were neglected in this model. As a matter of fact, their function is made less cumbersome and mostly automatic as companies submit for quality validation and, being verified its quality all information goes to a repository that can be easily accessed by the final consumers.

This model aims to increase profit by optimizing production through quality and safety monitoring and the consequent possibility to reduce waste.

Safety and quality based traceability system for food supply chains

By monitoring safety and quality any item in the food supply chain can be better prepared and better priced according to the possibilities and needs of production and the target audience. In summary, all tasks are divided into layers according to the entities that perform those tasks. The Unique Code Layer corresponds to entities responsible for attribution of unique codes that validate a transaction and is composed by a module that attributes the code. The Production Layer corresponds to all entities that produce and commercialize perishable food and is composed by modules that manage inventory, processing and customer orders. The Regulator Layer is composed by modules that verify quality when a company wishes to sell products, that verify quality when a product is received by another and a repository of information available to all that wish to verify the history of any given product. Figure 3.1 presents the complete traceability model and all information flows. A more detailed exposition of all components will be given in the next sections.

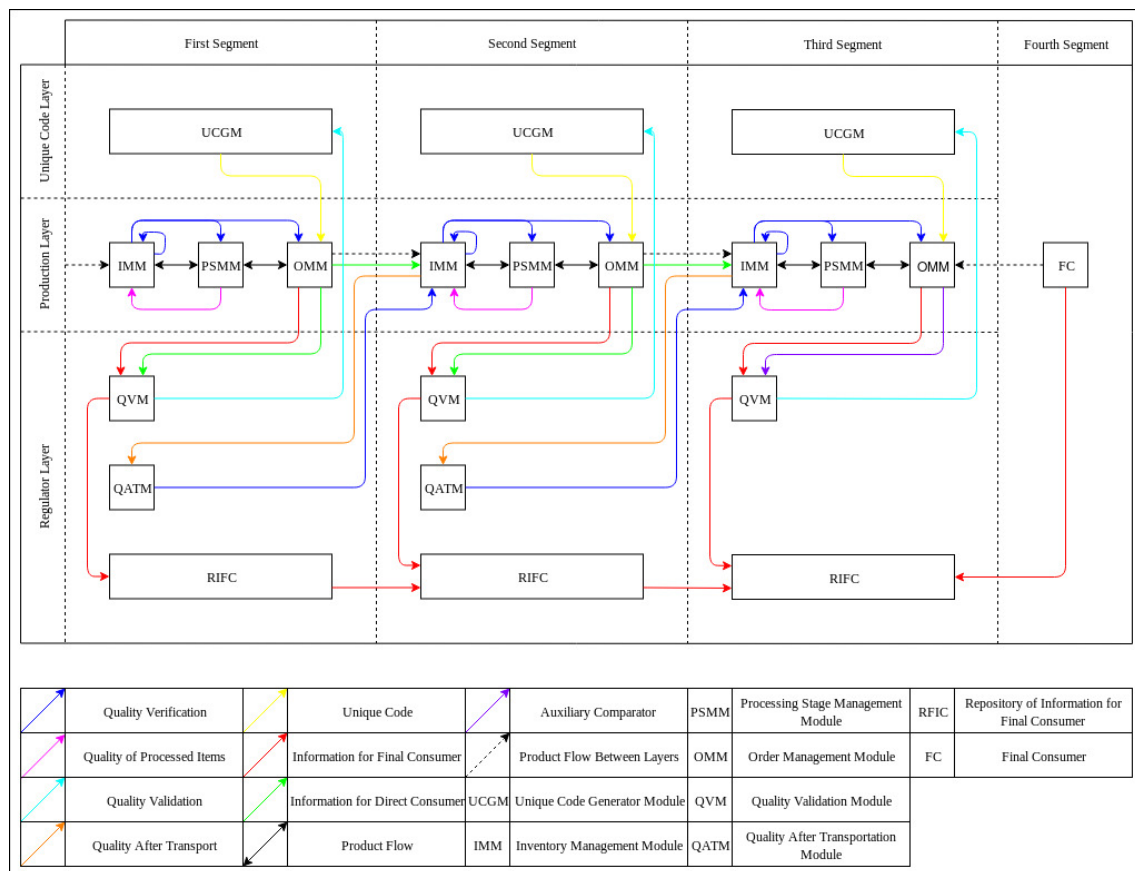


Figure 3.1: Diagram of the traceability model.

3.2 Model constituents

3.2.1 Modules

As all constituents of this model are equal by segment, apart from the fourth one, which is only meant to symbolize the interactions with the final consumer, it would make no sense to describe constituents by number of occurrences. Thus, any constituent will only be described once and from a top down approach.

The Unique Code Generator Module (UCGM) accepts products marked as valid for trade and

confers them with a unique identifier.

The Inventory Management Module (IMM) accepts input and output of materials and products as well as quality and keeping quality information relative to all items in inventory, i.e. goods not being used in any given point in time. The Processing Stage Management Module (PSMM) operates in the same manner as IMM, but is applied to products being transformed in any given point in time. The Order Management Module (OMM) manages all orders and information necessary to relay to other layers and segments.

The Quality Validation Module (QVM) is tasked with the verification of veracity of all information given by the OMM.

The Quality After Transport Module (QATM) is responsible for the comparison of the quality that the product was supposed to be delivered with the quality of the product actually delivered.

The Repository for Information for Final Consumers (RIFC) is a container of all the information related to validated products.

3.2.2 Layers

All activities are divided into three layers according to their functions and the entity of the stakeholders. The regulator layer encompasses all functions and constituents necessary for regulators to correctly perform their function. The production layer comprehends food supply chains and all companies participating in the chain. The stakeholders of the unique code layer would be entities equivalent to Global Standards One. By making this division it becomes easy to group entities by function and convey a simple, and more understandable structure to all stakeholders.

3.2.3 Segments

This segmentation has the same purpose of the previous ones, making a simple, more understandable, more organized architecture by separating all stakeholders in food supply chains by their most significant mean of acquisition of raw materials and most significant type of buyer. Companies in the first segment obtain most of their raw materials by gathering from nature and sell their products mostly to other companies. Second segment companies buy most of their raw materials from companies in the first segment and sell them to other companies further down the supply chain. In the third segment companies acquire their raw materials from other companies and sell them directly to the fourth segment, composed exclusively of final consumers. A simple, hypothetical, example would be a packed salad bought in a supermarket. The first segment collects all ingredients for the salad and sells them in bulk to the second segment which mixes and packs the ingredients and sells them, too in bulk, to the third segment, the supermarket, which in turn sells the individual packs to the fourth segment, the final consumers.

3.3 Archetype for information sharing

In order to better illustrate the interactions between all constituents of this model, all segments will be separated into different subsections. However, due to the similarities between all segments, only the first will be described in detail and the others will be described by their differences in comparison to the first.

3.3.1 First segment

The first segment includes obtaining raw materials, identifying them, evaluating their quality and keeping quality using scientific methods and using all that information as input to the IMM. For as long as any raw material or intermediate product remains in inventory, the IMM will always record the environmental conditions and their impact in quality and keeping quality. When items in inventory leave for processing the PSMM will get the data recorded by the IMM for each item. That data will serve as input in the PSMM and will be used to create an operational record for that stage. When processing finishes, an exit is created by the PSMM. That exit serves as an input for a new entry in the IMM. It is recommended that should be one PSMM per HACCP flowchart stage or equivalent. As orders are received, the OMM handles all information requests and fetches data from both IMM and PSMM in order to generate the Information for Direct Consumers (IDC) and the Information for Final Consumers (IFC) files. The first contains more technical information relative to constituents of a product and its quality and keeping quality history. The second is composed by data comprehensible by the final consumers. The OMM relays this data to the QVM, which will compare all commonly indexed data between files and recalculate all quality and keeping quality evaluations in order to detect the existence of non-compliant or fraudulent activity. According to the result of that evaluation, the products marked for validation may or may not be accepted by the QVM. If any is valid, the QVM will relay information to the UCGM, which will uniquely identify valid products and the IFC will be relayed to the RIFC. IDC and IFC will be relayed back to OMM in order to inform the respective company of the ability to trade the validated products. The OMM will then pass the IDC file to the buyer in the second segment. Figure 2 shows the aforementioned flow of information. To avoid a cluttered diagram and increase readability, all flows never cross layer boundaries but follow the same color code presented in Figure 3.2.

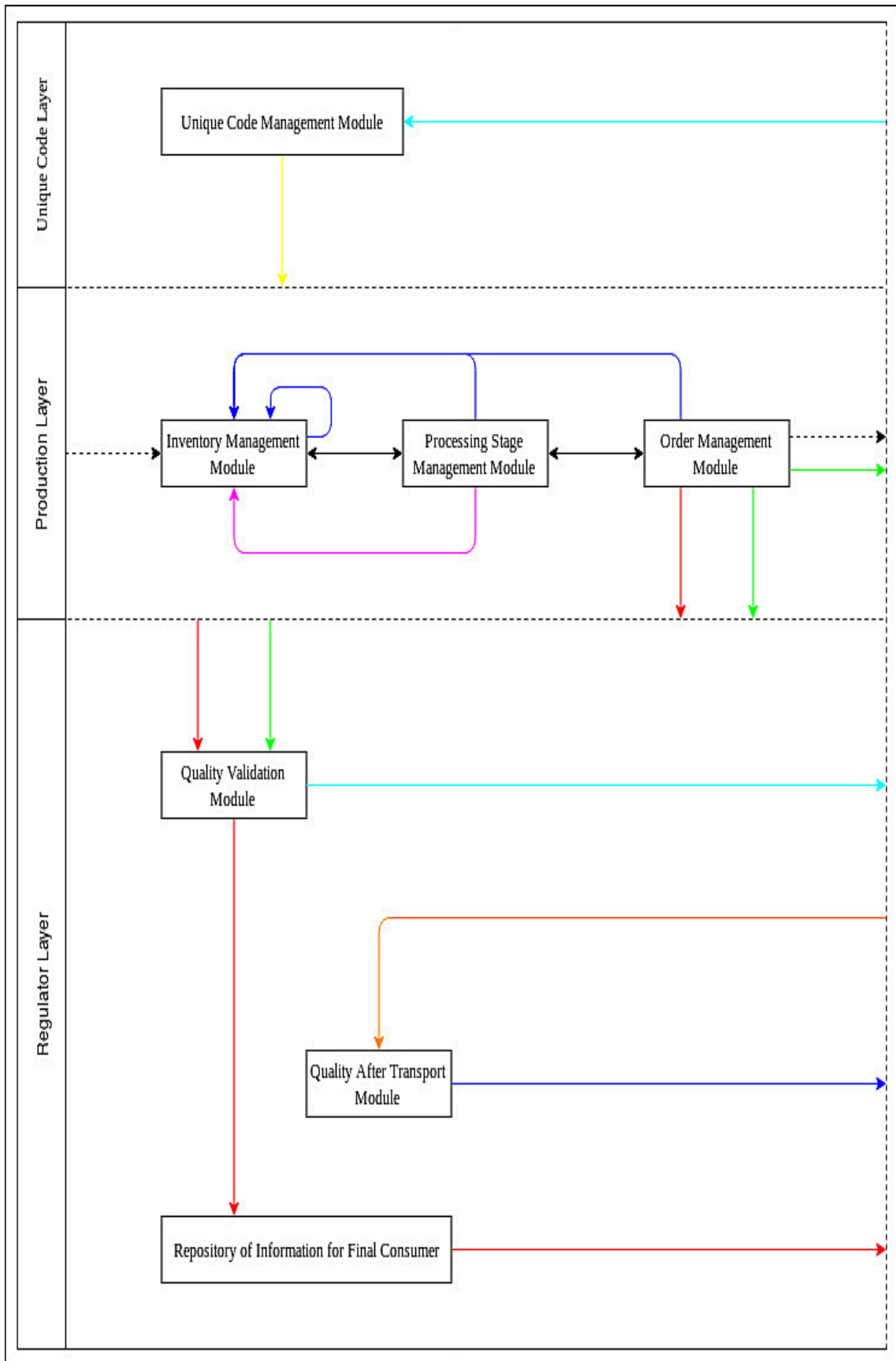


Figure 3.2: Flowchart of the first segment.

Safety and quality based traceability system for food supply chains

3.3.2 Second segment

The only difference of this segment from the former is the use of the QATM in order to compare if the quality and keeping quality of a delivered product corresponds to what was promised by the seller. Data provided from the QATM becomes an input for the IMM. Similarly, to Figure 3.2, Figure 3.3 shows the flow of information applied to the second segment.

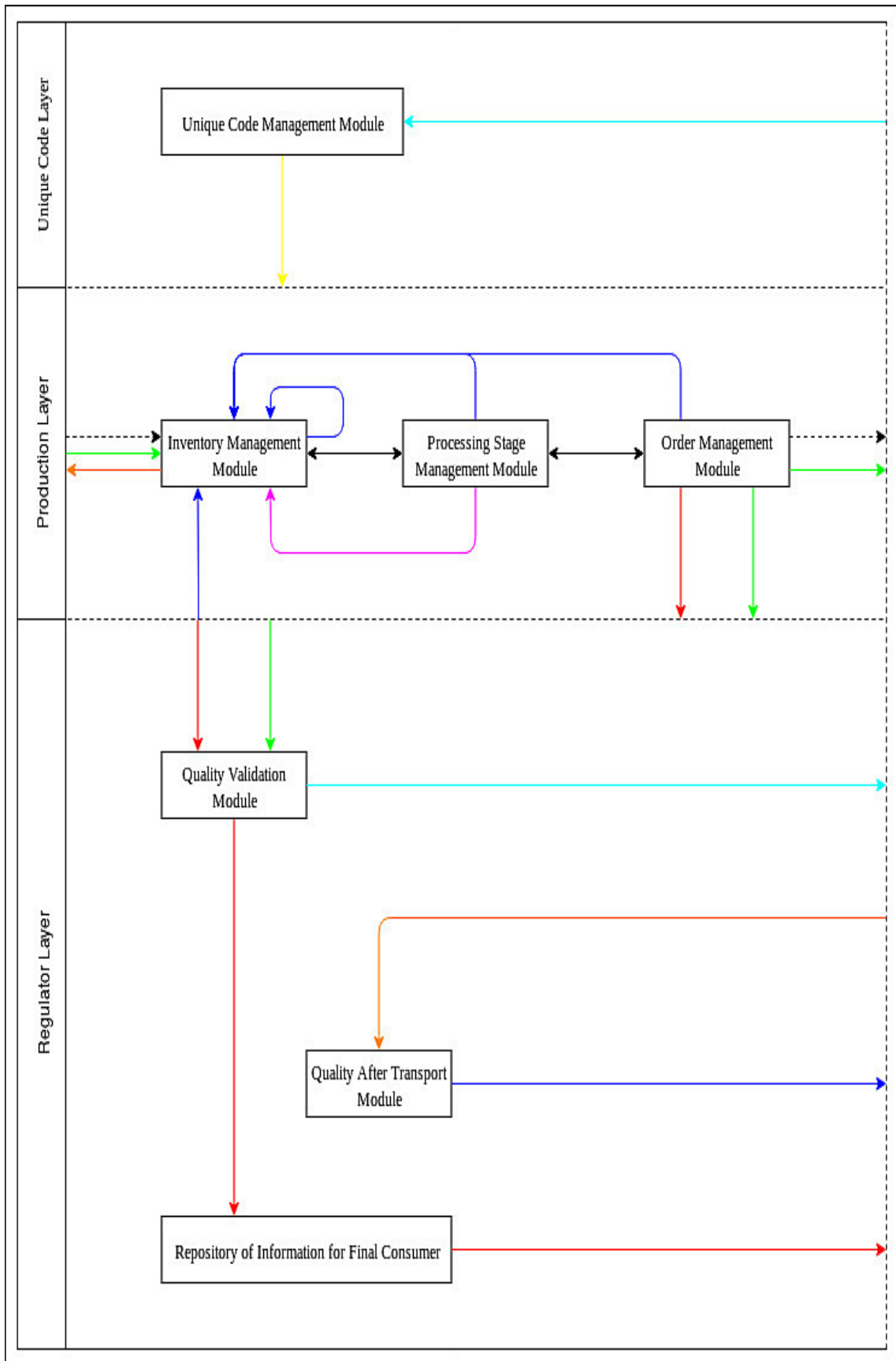


Figure 3.3: Flowchart of the second segment.

3.3.3 Third segment

As in this segment the final consumer is also the direct consumer, it is not considered reasonable to generate both IDC and IFC. Thus, only IFC is generated, but this causes an issue, the absence of commonly indexed data for external evaluation. To solve this problem, an Auxiliary Comparator (AC) is generated just to inform the QVM, which data to evaluate and validate. Figure 3.4 illustrates the information flow of the third segment.

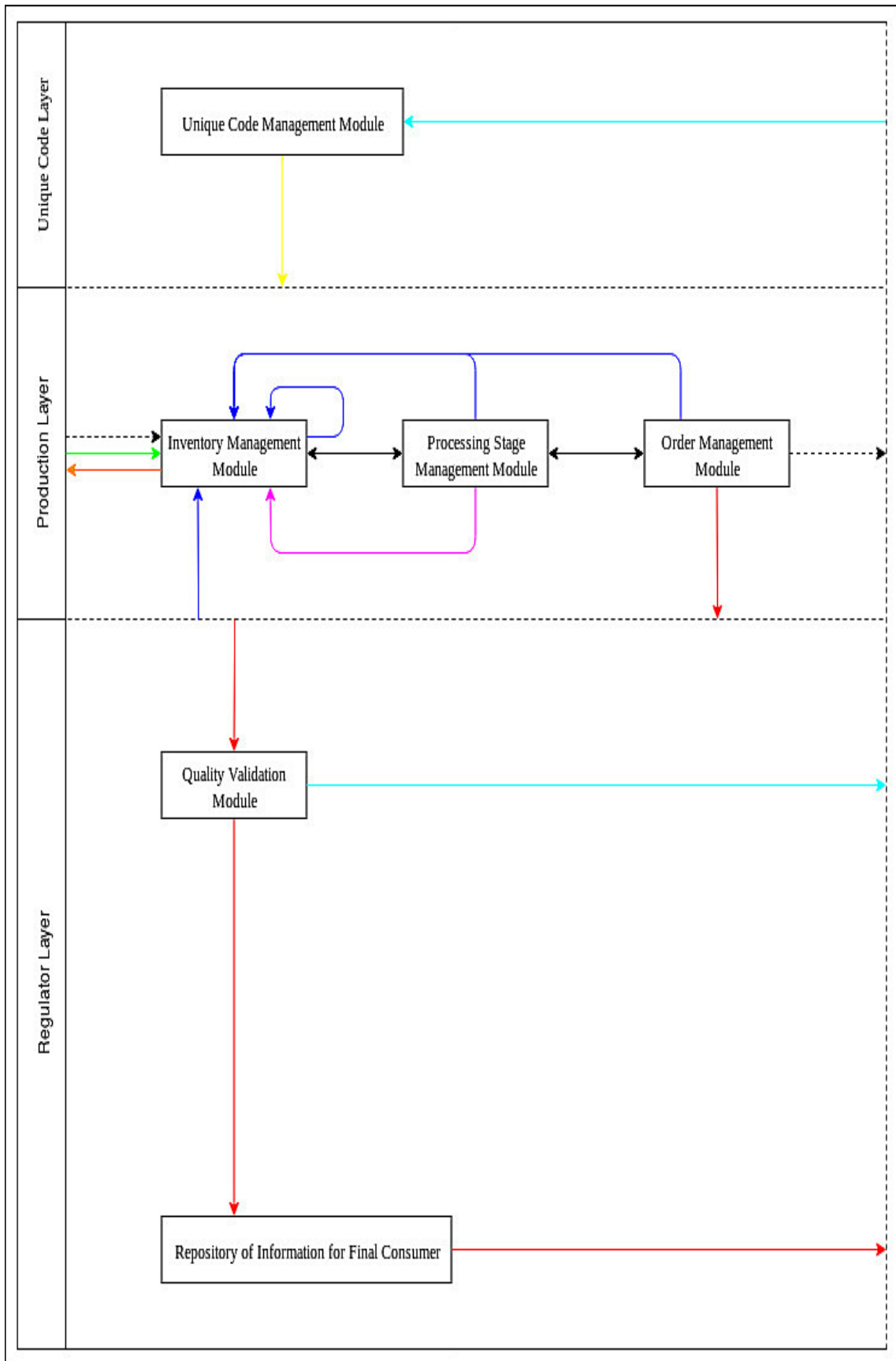


Figure 3.4: Flowchart of the third segment.

3.3.4 Fourth segment

This segment is composed solely by the Final Consumer (FC). The FC will make use of the RFIC to access the cumulative history of a process and will “hop” from IFC to IFC using the external identifiers provided by the UCGM.

3.4 Archetypes for shared files

Although communication between stakeholders can be made very efficient by having a common communication model, it can be made even more effective by sharing standardly structured files. Examples for the structure of the files are given in this section.

3.4.1 Information for final consumers

This file is the less strict in terms of structure and can be used as a marketing tool since the quality and history of the product is verified externally. Marked in bold text in Table 3.1 are the mandatory fields. However, it is highly recommended to include a description of all stages of production and their effect on quality.

To facilitate the comprehension of the table, three materials will be turned into products in a one to one relationship. Raw material X will become A, then D and finally G; similarly, Y and Z will become B and C, then E and F and finally H and I, respectively. Also, only two production stages are demonstrated. If more stages exist, one would need to add more columns to the table with the same structure as presented in the Table 3.1.

The fields of the file are:

- Component ID - this mandatory field indicates the entry ID of a component. This field is mandatory because with this ID a consumer can look for the history of that component on the Information for Final Consumers file of the company who sold it.
- Company ID - this field simply indicates the company who accepted the commodity, and for that reason is mandatory.
- Start Date - indicates when the goods entered the inventory of the above company. This provides the consumer with a better perspective of the age of the perishable, hence its mandatory nature.
- Start Quality - indicates the quality of the goods when they entered inventory. It is also an instrumental value to illustrate the quality of raw materials and so it is mandatory.
- Description - simply serves to describe the commodities.
- Inventory QMP - presents the value of the quality measurement parameter in the inventory when the goods enter it. For simplicity, only one quality measurement parameter was used to illustrate this field. If more parameters were used, then more columns had to be added to the table and each correctly identified.
- Inventory Algorithm - presents the identifier of the algorithm used to assess quality.
- Inventory Exit Quality - presents the goods quality when they leave production to enter processing.

Safety and quality based traceability system for food supply chains

- Production Stage 1 Components - identifies the goods that enter the first processing stage.
- Production Stage 1 Description - describes what happens in the first processing stage.
- Stage 1 Date - indicates when goods enter the first processing stage.
- Stage 1 QMP - indicates the value of the quality measurement parameter relative to this first stage. Likewise Inventory QMP, more columns must be added and identified if more quality measurement parameters are used.
- Stage 1 Algorithm - identifies the algorithm used to assess quality during this stage.
- Stage 1 Exit Quality - presents the quality of a perishable when it leaves the first stage.
- Stage 1 Exit ID - presents the identifier attributed to the perishable when it leaves the first stage and enters inventory as an intermediary product.
- Production Stage N Components - identifies the goods that enter the last stage of processing.
- Production Stage N Description - describes what happens in the last stage of production.
- Stage N Date - indicates when the goods enter the last processing stage.
- Stage N QMP - indicates the value of the quality measurement parameter relative to this last stage. Likewise Inventory QMP, more columns must to be added and identified if more quality measurement parameters are used.
- Stage N Algorithm - identifies the algorithm used to assess quality during this stage.
- Stage N Exit Quality - presents the quality of a perishable when it leaves the last stage.
- Stage N Exit ID - presents the identifier attributed to the perishable when it leaves the last stage and enters inventory as a final product.
- End Date - indicates when the product leaves the company.
- End QMP - indicates the value of the quality measurement parameter when the product leaves the company.
- End Algorithm - identifies the algorithm being used for the final product in inventory at the moment the commodity exits the company.
- End Quality - indicates the quality of the product when it leaves the company. This is another instrumental parameter for an end user to assess and then decide what to buy and so, it is mandatory.
- Exit Internal ID - identifies the commodity internally at the moment of departure from inventory.
- Exit External ID - identifies the commodity as it leaves inventory. This value is crucial for the end user as it is the code that the user will use to search for the history of the product. Thus, it is a mandatory field.

Safety and quality based traceability system for food supply chains

Table 3.1: Information for Final Consumers example.

Component ID	Company ID	Start Date	Start Quality	Description
X ID	Company ID	dd/mm/yy h:m	X hours	Description of X
Y ID	Company ID	dd/mm/yy h:m	Y hours	Description of Y
Z ID	Company ID	dd/mm/yy h:m	Z hours	Description of Z
Inventory QMP	Inventory Algorithm	Inventory Exit Quality	Production Stage 1 Components	Production Stage 1 Description
X QMP value	X Algorithm ID	X hours	X ID	In this stage X was done
Y QMP value	Y Algorithm ID	Y hours	Y ID	In this stage Y was done
Z QMP value	Z Algorithm ID	Z hours	Z ID	In this stage Z was done
Production Stage 1 Date	Production Stage 1 QMP	Production Stage 1 Algorithm	Production Stage 1 Exit Quality	Production Stage 1 Exit ID
dd/mm/yy h:m	X QMP value	X Algorithm ID	A hours	A ID
dd/mm/yy h:m	Y QMP value	Y Algorithm ID	B hours	B ID
dd/mm/yy h:m	Z QMP value	Z Algorithm ID	C hours	C ID
Production Stage N Components	Production Stage N Description	Production Stage N Date	Production Stage N QMP	Production Stage N Algorithm
A ID	In this stage A was done	dd/mm/yy h:m	A QMP value	A Algorithm ID
B ID	In this stage B was done	dd/mm/yy h:m	B QMP value	B Algorithm ID
C ID	In this stage C was done	dd/mm/yy h:m	C QMP value	C Algorithm ID
Production Stage N Exit Quality	Production Stage N Exit ID	End Date	End QMP	End Algorithm
D hours	D ID	dd/mm/yy h:m	D QMP value	D Algorithm ID
E hours	E ID	dd/mm/yy h:m	E QMP value	E Algorithm ID
F hours	F ID	dd/mm/yy h:m	F QMP value	F Algorithm ID
End Quality	End Internal ID	End External ID		
D hours	D ID	G ID		
E hours	E ID	H ID		
F hours	F ID	I ID		

3.4.2 Information for direct consumers

The structure of this file is more rigid. It must include the processing stages and more detailed, less end consumer friendly information about the processes that any given good was subjected to.

The structure of this file is identical to the Information for Final Consumers, albeit with all fields being mandatory. Some of its fields can be used as a marketing tool, but in a different, more appropriate manner since its target is a company and not an end consumer.

Such is not done in the Information for Final Consumers file because if its fields are filled inadequately, it becomes overbearing to the end user.

3.4.3 Quality validation and unique code

This file is the same as IFC and IDC but with the products marked for validation given a unique code if the external quality evaluation returns valid.

3.4.4 Quality after transport and quality verification

The Quality After Transport (QAT) file is to be started at a destination and to be finished at the Quality After Transport module in the regulator layer. The objective of this communication is to externally and scientifically assess the quality of goods that arrived at a company. This allows the company an unbiased evaluation and verification if the quality of goods matches the one that was supposed to be delivered.

The fields to fill in this file are, also shown in Table 3.2:

- Seller Company ID - identifies who sold the perishables.
- Buying Company ID - identifies who bought the perishables.
- Product ID - identifies the products to be evaluated.
- Expected Quality - presents the quality that goods were supposed to have at arrival.
- Algorithm Used - identifies the algorithm used to assess quality.
- Evaluated QMP at Arrival - presents the value of a quality measurement parameter when the product was tested at arrival. As seen before, if more parameters are used, more columns must be used and correctly identified.
- Calculated Quality - this column is filled by the quality after transport module and contains the level of quality as evaluated using the previous field.
- Delta - indicates the difference between the expected quality and the real quality. This column is also filled by the quality after transport module.

Table 3.2: Quality After Transport example.

Selling company ID	Buying company ID	Product ID
Seller ID	Buyer ID	X ID
Seller ID	Buyer ID	Y ID
Seller ID	Buyer ID	Z ID
Expected quality	Algorithm used	Evaluated QMP at arrival
X hours	Algorithm ID	X QMP value
Y hours	Algorithm ID	Y QMP value
Z hours	Algorithm ID	Z QMP value
Calculated quality	Delta	
QX	QX-X	
QY	QY-Y	
QZ	QZ-Z	

3.5 Choice of language and development environment

In order to choose the programming language to develop all modules presented in the traceability model, some requirements must be met. The programming language must be simple to learn

Safety and quality based traceability system for food supply chains

and to teach, extremely versatile as it will be used for a multitude of different tasks and not be inherently computationally intensive. It is considered that the best language that fulfills those criteria is Python [95]. There is a wide variety of free material that can be effectively used to learn the programming language and its syntax is quite simple and easy to begin using. It is widely used for very diverse functions including the web apps and servers, graphical interfaces, data analysis and, perhaps more distinctively, to process the data that led to the image of the black hole obtained by an Event Horizon Telescope team in April 2019 [96].

Although Python 2 is still widely used, support will end in 2020, and Python 3 was used in order to future-proof of concept this project.

The only other required software is an Integrated Development Environment (IDE). In this case there is no selection process as Python is an interpreted programming language and so does not need a compiler. This means that any text editor can be used for development and testing can be done in a terminal. Microsoft Visual Studio Code [97] was used to develop this prototype. It operates as a text editor with a direct connection to a terminal, making development less time consuming.

As an operating system (OS) Ubuntu 18.04 LTS [98] was used. This choice not to use Windows was simply made due to the rock solid stability of this OS which is quite useful for testing purposes and to deploy servers. Although this change is not in any way, shape or form necessary it is recommended.

Concerning the hardware used, an Asus N76VZ-T1171H, with a CPU Intel Core i7 3630QM with the Intel Turbo purposely disabled, thus lowering maximum clock frequency quite drastically. RAM is composed by two sticks of 4GB and 1600MHz each, on a dual channel configuration.

Although resource consumption was not measured, it was always considered by verifying the usage of those two components. It never exceeded or was close to exceeding hardware capabilities at any given time.

As demonstrated, even by restricting the environment quite heavily, the system managed to operate correctly, thus testifying that traceability systems have to be neither expensive nor difficult to operate.

3.6 Operation of the constituents

The following sections illustrate the functioning of all modules. This exposition refers to the templates created. In order to better fit the needs of a user, the presented tools will have to be adapted to a concrete situation. Snippets for the code that led to the tools developed will be presented and detailed in the Annex section of this dissertation. However, a brief summary of all files that compose any given module will be given. To avoid repetition, only the modules that compose the first segment will be presented as they are very similar to the modules in the second and third segment. Still all differences will be remarked when pertinent. To increase the readability of this section, all code snippets that led to the tools presented were moved to the Annex section of this dissertation and will be given the same numeration as the section in which they are presented.

3.6.1 Production layer

As seen before, the production layer is divided into segments in order to better illustrate the information flow. This layer is the most affected by the nature of the templates as there will

have to be, in the very least, as many PSMM's as stages in the HACCP diagram.

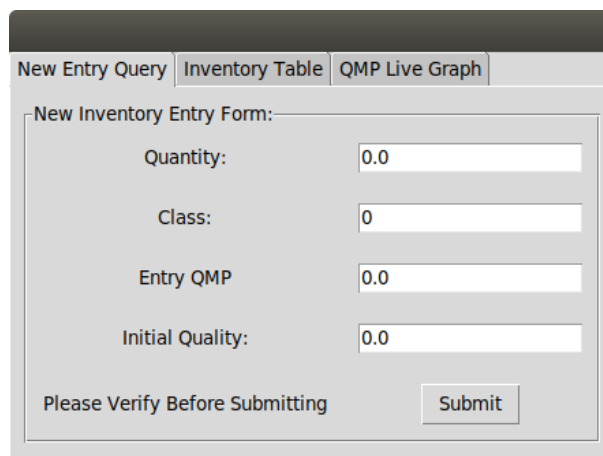
All other templates do not require extensive replication and adaptation to operate properly.

In order to avoid unnecessary repetition only the code for the first segment will be discussed as it is very similar to the other segments.

3.6.1.1 Inventory management module

The template for the inventory management module is constituted by seven files. `MainApp.py` aggregates all other files to create a graphical user interface (GUI) that is intended to be very simple to use and alter. The remaining six files were created in pairs composed by the layout of the elements and their functions.

This GUI is composed by a window with three tabs. The first tab concerns new entries in inventory and the record of initial conditions. This tab makes use of the `Tab1Layout.py` file for the position of all elements in this tab and `Tab1Functions.py` for all functions related to the first tab. Figure 3.5 illustrates the first tab of the IMM's GUI when waiting for an entry to be submitted. This submission tab is very simple in its constitution, requiring only quantity, class, the value of the quality measurement parameter and initial quality as inputs for new entries in inventory.



The screenshot shows a GUI window with three tabs: 'New Entry Query', 'Inventory Table', and 'QMP Live Graph'. The 'New Entry Query' tab is active. Below the tabs is a form titled 'New Inventory Entry Form:'. The form contains four input fields, each with a label and a value: 'Quantity: 0.0', 'Class: 0', 'Entry QMP: 0.0', and 'Initial Quality: 0.0'. At the bottom left of the form is the text 'Please Verify Before Submitting', and at the bottom right is a 'Submit' button.

Figure 3.5: Inventory management module new entry tab.

Submitting a new entry will cause a window to pop-up. This window contains data from the submission form showing all the gathered or calculated data, such as the identifier given to the new entry, necessary parameters for the determination of the keeping quality and its value. An example of this window is given in Figure 3.6.

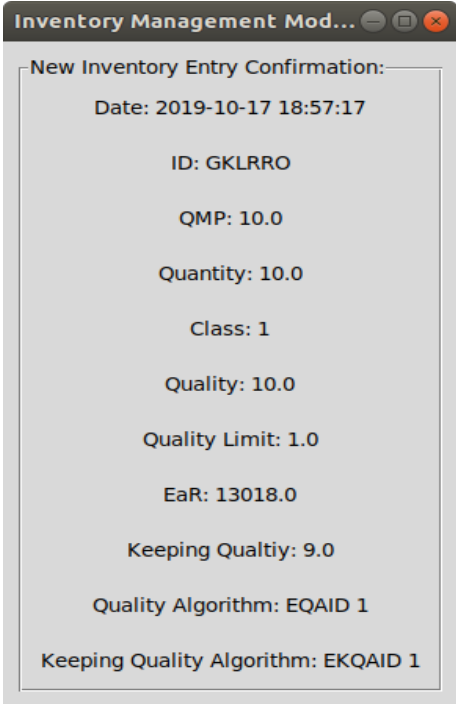


Figure 3.6: Inventory management module entry confirmation.

The second tab in the GUI contains a table of all items in inventory. To further reduce resource consumption, the tab only shows a button at first. The tab can be seen in Figure 3.7.

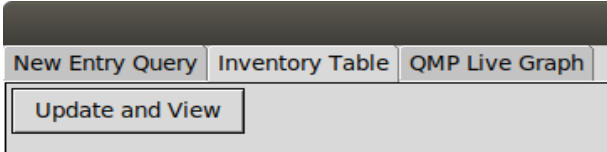


Figure 3.7: Inventory management module inventory tab.

Pressing the button will display the table containing all items and the updated quality and keeping quality values as Figure 3.8 and 3.9 show.

Safety and quality based traceability system for food supply chains

Entry Date	Company ID	Entry ID	Entry OMP	Entry Quantity	Class	Initial Quality	Quality Limit	EaR	Initial Keeping Quality	Initial Quality Algorithm	Initial Keeping Quality Algorithm	Last Update
2019-08-26 21:59:32	E1	M66G2Y	10.0	500.0	1	10000.0	1.0	13018.0	9999.0	EQAID 1	EKQAID 1	2019-10-17
2019-08-26 21:59:37	E1	YFGTN1	10.0	1000.0	1	10000.0	1.0	13018.0	9999.0	EQAID 1	EKQAID 1	2019-10-17
2019-08-26 21:59:39	E1	CPNFZB	10.0	1000.0	1	10000.0	1.0	13018.0	9999.0	EQAID 1	EKQAID 1	2019-10-17
2019-08-26 21:59:41	E1	M3W368	10.0	1000.0	1	10000.0	1.0	13018.0	9999.0	EQAID 1	EKQAID 1	2019-10-17
2019-08-26 21:59:43	E1	EXIDWZ	10.0	1000.0	1	10000.0	1.0	13018.0	9999.0	EQAID 1	EKQAID 1	2019-10-17
2019-08-26 21:59:44	E1	MU32BG	10.0	1000.0	1	10000.0	1.0	13018.0	9999.0	EQAID 1	EKQAID 1	2019-10-17
2019-08-26 21:59:46	E1	70280D	10.0	1000.0	1	10000.0	1.0	13018.0	9999.0	EQAID 1	EKQAID 1	2019-10-17
2019-08-26 21:59:47	E1	TSYACO	10.0	1000.0	1	10000.0	1.0	13018.0	9999.0	EQAID 1	EKQAID 1	2019-10-17
2019-08-26 21:59:49	E1	3WNTM9	10.0	1000.0	1	10000.0	1.0	13018.0	9999.0	EQAID 1	EKQAID 1	2019-10-17
2019-08-26 21:59:51	E1	U28YDF	10.0	1000.0	1	10000.0	1.0	13018.0	9999.0	EQAID 1	EKQAID 1	2019-10-17
2019-08-26 22:00:07	E1	ONKEOV	20.0	1550.0	2	20000.0	1.0	12520.0	4425.339855003893	EQAID 2	EKQAID 2	2019-10-17
2019-08-26 22:00:14	E1	YHAU3Z	20.0	1950.0	2	20000.0	1.0	12520.0	4425.339855003893	EQAID 2	EKQAID 2	2019-10-17
2019-08-26 22:00:15	E1	SRNRJV	20.0	2000.0	2	20000.0	1.0	12520.0	4425.339855003893	EQAID 2	EKQAID 2	2019-10-17
2019-08-26 22:00:17	E1	KYRWHF	20.0	2000.0	2	20000.0	1.0	12520.0	4425.339855003893	EQAID 2	EKQAID 2	2019-10-17
2019-08-26 22:00:18	E1	TTXC8S	20.0	2000.0	2	20000.0	1.0	12520.0	4425.339855003893	EQAID 2	EKQAID 2	2019-10-17
2019-08-26 22:00:20	E1	26ZWMS	20.0	2000.0	2	20000.0	1.0	12520.0	4425.339855003893	EQAID 2	EKQAID 2	2019-10-17
2019-08-26 22:00:21	E1	9NSLX2	20.0	2000.0	2	20000.0	1.0	12520.0	4425.339855003893	EQAID 2	EKQAID 2	2019-10-17
2019-08-26 22:00:23	E1	DR00LJ	20.0	2000.0	2	20000.0	1.0	12520.0	4425.339855003893	EQAID 2	EKQAID 2	2019-10-17
2019-08-26 22:00:24	E1	SOZKJA	20.0	2000.0	2	20000.0	1.0	12520.0	4425.339855003893	EQAID 2	EKQAID 2	2019-10-17

Figure 3.8: Inventory management module inventory table.

Initial Keeping Quality Algorithm	Last Update	Last Update OMP	Last Update Quality	Last Update Keeping Quality	Last Update Quality Algorithm
EQAID 1	2019-10-17 18:57:36	-4.83	9901.918404804543	126919.15119439852	EQAID 1
EQAID 1	2019-10-17 18:57:36	-4.83	9901.918514224593	126919.15119439852	EQAID 1
EQAID 1	2019-10-17 18:57:36	-4.83	9901.918557992614	126919.15119439852	EQAID 1
EQAID 1	2019-10-17 18:57:36	-4.83	9901.918601760635	126919.15119439852	EQAID 1
EQAID 1	2019-10-17 18:57:36	-4.83	9901.918645528654	126919.15119439852	EQAID 1
EQAID 1	2019-10-17 18:57:36	-4.83	9901.918667412665	126919.15119439852	EQAID 1
EQAID 1	2019-10-17 18:57:36	-4.83	9901.918711180686	126919.15119439852	EQAID 1
EQAID 1	2019-10-17 18:57:36	-4.83	9901.918733064696	126919.15119439852	EQAID 1
EQAID 1	2019-10-17 18:57:36	-4.83	9901.918776832716	126919.15119439852	EQAID 1
EQAID 1	2019-10-17 18:57:36	-4.83	9901.918820600737	126919.15119439852	EQAID 1
EQAID 2	2019-10-17 18:57:36	-4.83	19891.906166873097	230336.13882324917	EQAID 2
EQAID 2	2019-10-17 18:57:36	-4.83	19891.90633570003	230336.13882324917	EQAID 2
EQAID 2	2019-10-17 18:57:36	-4.83	19891.90639818167	230336.13882324917	EQAID 2
EQAID 2	2019-10-17 18:57:36	-4.83	19891.906408054434	230336.13882324917	EQAID 2
EQAID 2	2019-10-17 18:57:36	-4.83	19891.906432172567	230336.13882324917	EQAID 2
EQAID 2	2019-10-17 18:57:36	-4.83	19891.906480408834	230336.13882324917	EQAID 2
EQAID 2	2019-10-17 18:57:36	-4.83	19891.906504526967	230336.13882324917	EQAID 2
EQAID 2	2019-10-17 18:57:36	-4.83	19891.906552763234	230336.13882324917	EQAID 2
EQAID 2	2019-10-17 18:57:36	-4.83	19891.906576881367	230336.13882324917	EQAID 2

Figure 3.9: Inventory management module inventory table (cont).

The third tab contains an animated plot of the inventory QMP. For illustration purposes temperature variation is plotted as is one of the most quality relevant parameter in agri-food supply chains. As the update interval can be freely changed, this tab operates as soon as the GUI is started. This means that the respective resource consumption can be easily controlled. As Figure 3.10 shows, only the five more recent values are shown to avoid a cluttered plot. Below the plot there is a toolbar that can increase plot readability if necessary.

Safety and quality based traceability system for food supply chains

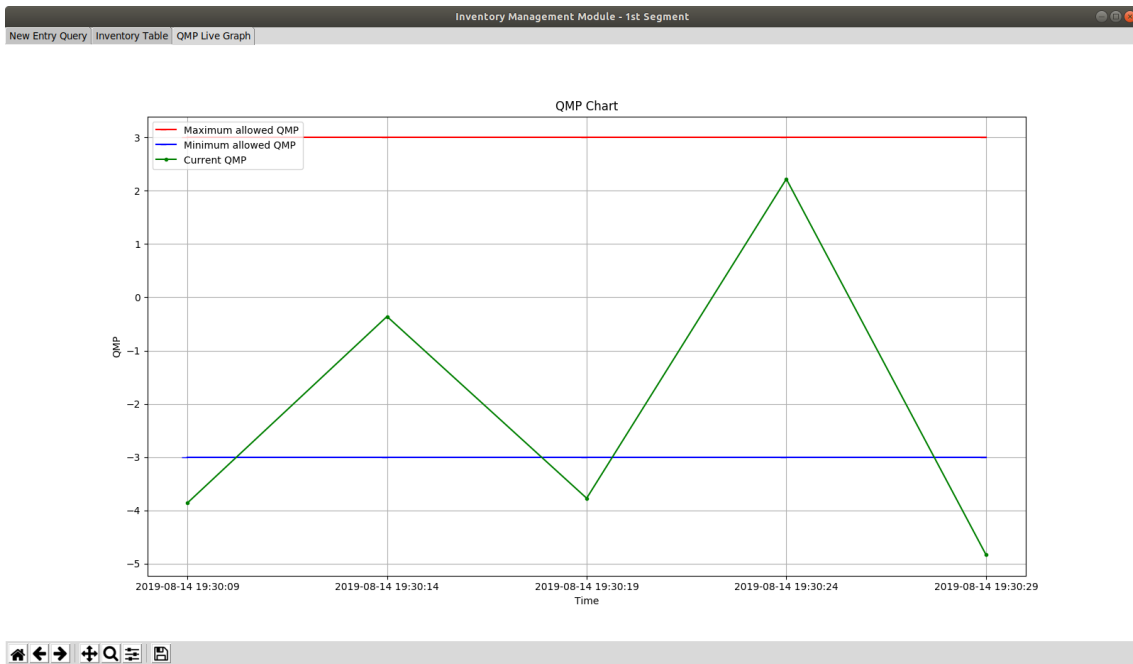


Figure 3.10: Inventory management module QMP plot

There is only one difference from this to the second and third segments' IMM. There is one additional field necessary for entry submission. That field is the external identifier given previously given to the product. This is necessary in order to not lose product traceability. Having this additional field means that the exits of a company can be easily connected to the entries of another.

3.6.1.2 Processing stage management module

This module is composed by seven files, each with the same name as the ones in the previous module. As such, this module is very similar to the previous in the sense that the first tab queries the user for information relative to an operation stage. The second tab keeps record of all operations and the third shows a live plot of quality measurement parameters. In order to fully link product information throughout entire processes it is necessary for a PSMM to exist per HACCP flowchart stage or equivalent health and safety method. Thus, a cumulative product history can be kept. To create a new operation, it is simply necessary to press the "New Operation" button in the top left corner of the first tab (shown in Figure 3.11).

Section	Field	Value
Raw Material Input	Raw Material ID	
	Raw Material Quantity	0.0
Intermediate Product Input	Intermediate Product ID	
	Intermediate Product Quantity	0.0
Exit Output	Exit Quantity	0.0
	Exit Quality	0.0
	Exit Class	0

Figure 3.11: Processing stage management module entry tab.

Safety and quality based traceability system for food supply chains

After submitting the parameters, a simple confirmation window pops-up. This serves as a mere confirmation that a new operation was created. This window is shown in Figure 3.12.

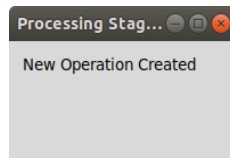
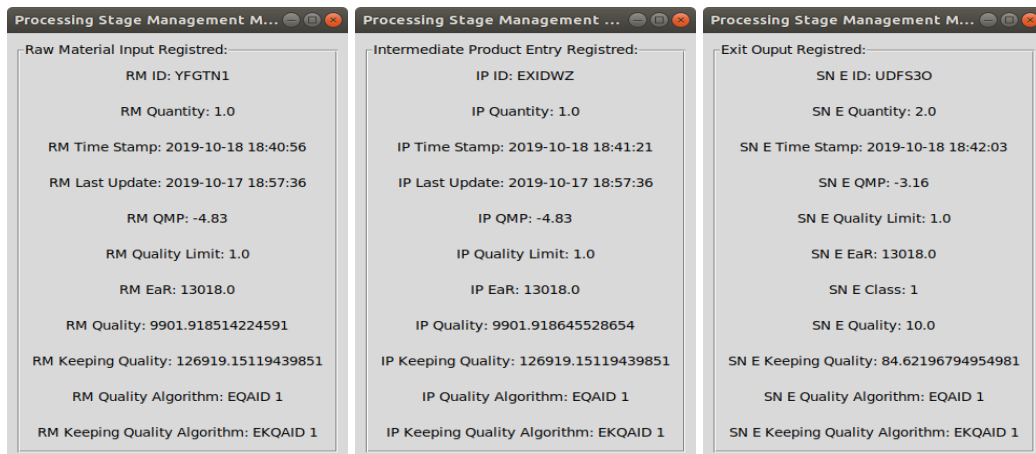


Figure 3.12: Processing stage management module new operation confirmation.

After an operation is created, it will remain on standby until a new operation is created and will associate all entries and exits of that stage to that operation. As information is submitted, top level windows will appear and will contain all information recorded upon submission. The windows can be seen in Figure 3.13a, Figure 3.13b and Figure 3.13c for raw materials, intermediate products and exits respectively.



(a) Raw material entry confirmation.

(b) Intermediate product entry confirmation.

(c) Exit output confirmation.

Figure 3.13: Processing stage management module information input confirmation windows.

Similar to IMM's second tab, to access operational history a button must be pressed. Figure 3.14 shows the second tab when the button has not yet been pressed and Figures 3.15 through 3.17 show the tab after the button was pressed.

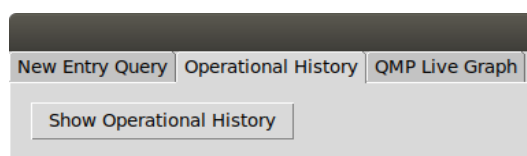


Figure 3.14: Processing stage management module operational history.

Safety and quality based traceability system for food supply chains

Processing Stage Management Module - 1st Segment

New Entry Query | Operational History | QMP Live Graph

Show Operational History

RM ID	RM Quantity	RM Time Stamp	RM Last Update	RM QMP	RM Quality Limit	RM EaR	RM Quality	RM Keeping Quality	RM Quality Algorithm	RM Keeping Quality Algo
M66G2Y	50.0	2019-08-28 03:49:48	2019-08-26 22:00:58	-4.83	1.0	13018.0	9999.998117975125	126919.15119439851	EQAID 1	EKQAID 1
---	---	---	---	---	---	---	---	---	---	---
M66G2Y	50.0	2019-08-28 03:51:18	2019-08-26 22:00:58	-4.83	1.0	13018.0	9999.998117975125	126919.15119439851	EQAID 1	EKQAID 1
---	---	---	---	---	---	---	---	---	---	---
M66G2Y	50.0	2019-08-28 03:51:18	2019-08-26 22:00:58	-4.83	1.0	13018.0	9999.998117975125	126919.15119439851	EQAID 1	EKQAID 1
---	---	---	---	---	---	---	---	---	---	---
RM ID <th>RM Quantity</th> <th>RM Time Stamp</th> <th>RM Last Update</th> <th>RM QMP</th> <th>RM Quality Limit</th> <th>RM EaR</th> <th>RM Quality</th> <th>RM Keeping Quality</th> <th>RM Quality Algorithm</th> <th>RM Keeping Quality Algo</th>	RM Quantity	RM Time Stamp	RM Last Update	RM QMP	RM Quality Limit	RM EaR	RM Quality	RM Keeping Quality	RM Quality Algorithm	RM Keeping Quality Algo
M66G2Y	50.0	2019-08-28 03:49:48	2019-08-26 22:00:58	-4.83	1.0	13018.0	9999.998117975125	126919.15119439851	EQAID 1	EKQAID 1
---	---	---	---	---	---	---	---	---	---	---
M66G2Y	50.0	2019-08-28 03:51:18	2019-08-26 22:00:58	-4.83	1.0	13018.0	9999.998117975125	126919.15119439851	EQAID 1	EKQAID 1
---	---	---	---	---	---	---	---	---	---	---
RM ID <th>RM Quantity</th> <th>RM Time Stamp</th> <th>RM Last Update</th> <th>RM QMP</th> <th>RM Quality Limit</th> <th>RM EaR</th> <th>RM Quality</th> <th>RM Keeping Quality</th> <th>RM Quality Algorithm</th> <th>RM Keeping Quality Algo</th>	RM Quantity	RM Time Stamp	RM Last Update	RM QMP	RM Quality Limit	RM EaR	RM Quality	RM Keeping Quality	RM Quality Algorithm	RM Keeping Quality Algo
YFGTN1	1.0	2019-10-17 20:00:32	2019-10-17 18:57:36	-4.83	1.0	13018.0	9901.918514224591	126919.15119439851	EQAID 1	EKQAID 1
---	---	---	---	---	---	---	---	---	---	---
RM ID <th>RM Quantity</th> <th>RM Time Stamp</th> <th>RM Last Update</th> <th>RM QMP</th> <th>RM Quality Limit</th> <th>RM EaR</th> <th>RM Quality</th> <th>RM Keeping Quality</th> <th>RM Quality Algorithm</th> <th>RM Keeping Quality Algo</th>	RM Quantity	RM Time Stamp	RM Last Update	RM QMP	RM Quality Limit	RM EaR	RM Quality	RM Keeping Quality	RM Quality Algorithm	RM Keeping Quality Algo

Figure 3.15: Processing stage management module operational history(cont).

Processing Stage Management Module - 1st Segment

New Entry Query | Operational History | QMP Live Graph

RM Keeping Quality Algorithm	IP ID	IP Quantity	IP Time Stamp	IP Last Update	IP QMP	IP Quality Limit	IP EaR	IP Quality	IP Keeping Quality	IP Quality Algorithm	IP Keeping Quality
EKQAID 1	---	---	---	---	---	---	---	---	---	---	---
---	ONKE0V	50.0	2019-08-28 03:49:52	2019-08-26 22:00:58	-4.83	1.0	12520.0	19999.998769975184	230336.1388232493	EQAID 2	EKQAID 2
---	---	---	---	---	---	---	---	---	---	---	---
EKQAID 1	---	---	---	---	---	---	---	---	---	---	---
---	ONKE0V	50.0	2019-08-28 03:51:48	2019-08-26 22:00:58	-4.83	1.0	12520.0	19999.998769975184	230336.1388232493	EQAID 2	EKQAID 2
---	---	---	---	---	---	---	---	---	---	---	---
RM Keeping Quality Algorithm	IP ID	IP Quantity	IP Time Stamp	IP Last Update	IP QMP	IP Quality Limit	IP EaR	IP Quality	IP Keeping Quality	IP Quality Algorithm	IP Keeping Quality
EKQAID 1	---	---	---	---	---	---	---	---	---	---	---
---	ONKE0V	50.0	2019-08-28 03:49:52	2019-08-26 22:00:58	-4.83	1.0	12520.0	19999.998769975184	230336.1388232493	EQAID 2	EKQAID 2
---	---	---	---	---	---	---	---	---	---	---	---
EKQAID 1	---	---	---	---	---	---	---	---	---	---	---
---	ONKE0V	50.0	2019-08-28 03:51:48	2019-08-26 22:00:58	-4.83	1.0	12520.0	19999.998769975184	230336.1388232493	EQAID 2	EKQAID 2
---	---	---	---	---	---	---	---	---	---	---	---
RM Keeping Quality Algorithm	IP ID	IP Quantity	IP Time Stamp	IP Last Update	IP QMP	IP Quality Limit	IP EaR	IP Quality	IP Keeping Quality	IP Quality Algorithm	IP Keeping Quality
EKQAID 1	---	---	---	---	---	---	---	---	---	---	---
---	EKIDWZ	1.0	2019-10-17 20:01:03	2019-10-17 18:57:36	-4.83	1.0	13018.0	9901.918645528654	126919.15119439851	EQAID 1	EKQAID 1
---	---	---	---	---	---	---	---	---	---	---	---
RM Keeping Quality Algorithm	IP ID	IP Quantity	IP Time Stamp	IP Last Update	IP QMP	IP Quality Limit	IP EaR	IP Quality	IP Keeping Quality	IP Quality Algorithm	IP Keeping Quality

Figure 3.16: Processing stage management module operational history(cont).

Safety and quality based traceability system for food supply chains

IP Keeping Quality Algorithm	SN E ID	SN E Quantity	SN E Time Stamp	SN E QMP	SN E Quality Limit	SN E EaR	SN E Class	SN E Quality	SN E Keeping Quality	SN E Quality Algorithm	SN E Keeping Quality
---	---	---	---	---	---	---	---	---	---	---	---
EKQAID 2	---	---	---	---	---	---	---	---	---	---	---
---	WPE403	100.0	2019-08-28 03:49:56	-3.16	1.0	17147.0	4	40000.0	765558.7440655511	EQAID 4	EKQAID 4
---	---	---	---	---	---	---	---	---	---	---	---
EKQAID 2	---	---	---	---	---	---	---	---	---	---	---
---	B4TL7H	100.0	2019-08-28 03:51:52	-3.16	1.0	17147.0	4	40000.0	765558.7440655511	EQAID 4	EKQAID 4
---	---	---	---	---	---	---	---	---	---	---	---
IP Keeping Quality Algorithm	SN E ID	SN E Quantity	SN E Time Stamp	SN E QMP	SN E Quality Limit	SN E EaR	SN E Class	SN E Quality	SN E Keeping Quality	SN E Quality Algorithm	SN E Keeping Quality
---	---	---	---	---	---	---	---	---	---	---	---
EKQAID 2	---	---	---	---	---	---	---	---	---	---	---
---	FPE403	100.0	2019-08-28 03:49:56	-3.16	1.0	17147.0	4	40000.0	765558.7440655511	EQAID 4	EKQAID 4
---	---	---	---	---	---	---	---	---	---	---	---
EKQAID 2	---	---	---	---	---	---	---	---	---	---	---
---	F4TL7H	100.0	2019-08-28 03:51:52	-3.16	1.0	17147.0	4	40000.0	765558.7440655511	EQAID 4	EKQAID 4
---	---	---	---	---	---	---	---	---	---	---	---
IP Keeping Quality Algorithm	SN E ID	SN E Quantity	SN E Time Stamp	SN E QMP	SN E Quality Limit	SN E EaR	SN E Class	SN E Quality	SN E Keeping Quality	SN E Quality Algorithm	SN E Keeping Quality
---	---	---	---	---	---	---	---	---	---	---	---
EKQAID 1	---	---	---	---	---	---	---	---	---	---	---
---	DBAB32	2.0	2019-10-17 20:01:29	-3.16	1.0	12520.0	2	10.0	77.66977906131221	EQAID 2	EKQAID 2
---	---	---	---	---	---	---	---	---	---	---	---
IP Keeping Quality Algorithm	SN E ID	SN E Quantity	SN E Time Stamp	SN E QMP	SN E Quality Limit	SN E EaR	SN E Class	SN E Quality	SN E Keeping Quality	SN E Quality Algorithm	SN E Keeping Quality

Figure 3.17: Processing stage management module operational history(cont).

As mentioned, the third tab is a live plot of parameters that influence quality in a specific stage, shown in Figure 3.18.

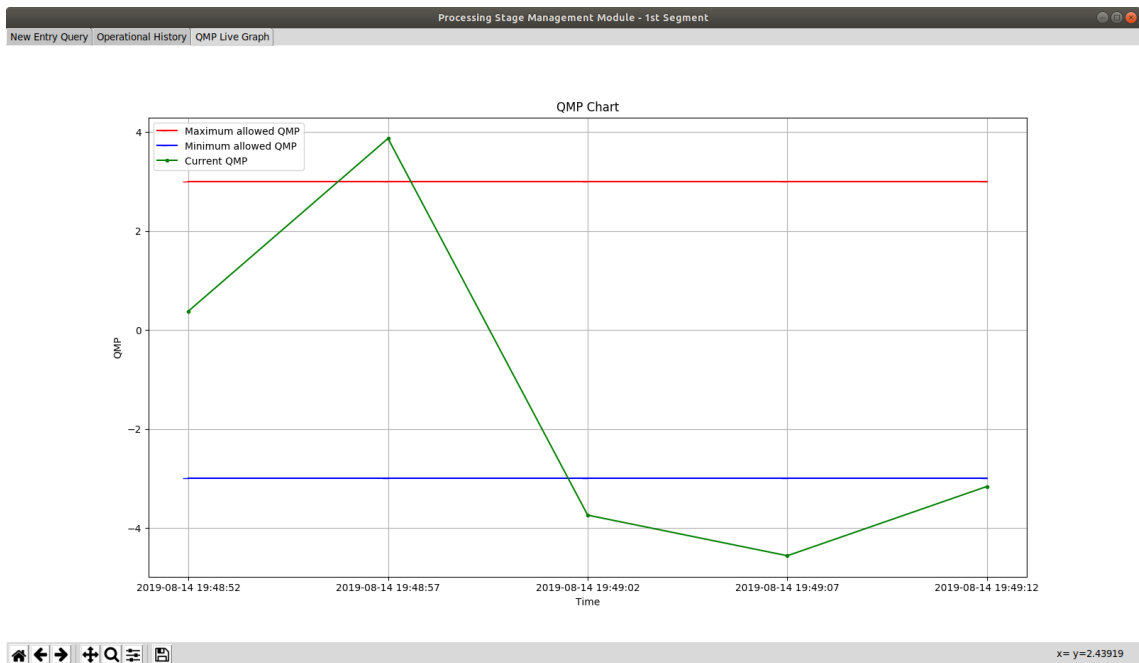


Figure 3.18: Processing stage management module QMP plot.

3.6.1.3 Order management module

This module consists on a web application made using the Flask framework. The rationale behind this choice is extremely simple. Making a web application guarantees access anytime, anywhere and with practically any modern device. Adding to that, Flask is very simple to learn and use, making possible for any entity to easily and cheaply deploy a fully featured application.

Safety and quality based traceability system for food supply chains

As this is a prototype, this module contains only all the functionalities deemed essential for application in an enterprise following the traceability model proposed. This module is divided in two, users and non-users. This division increases greatly the organization of this module, which is, by far, the most complex of this layer. Non-users are all persons who do not have login credentials, i.e., individuals or entities not belonging to any given company. All non-users are restricted to simple functions in the application. They have access to a homepage and a search page. The homepage consists in a simple presentation of a company via price table. Due to the possibility of scaling prices with quality, the price table has three columns. A class identifier, a description of item class and price per quality interval. This is, however, a mere example of how to present a company in a simplistic manner and can be easily altered to better suit each corporation’s necessities. Figure 3.19 shows the homepage.

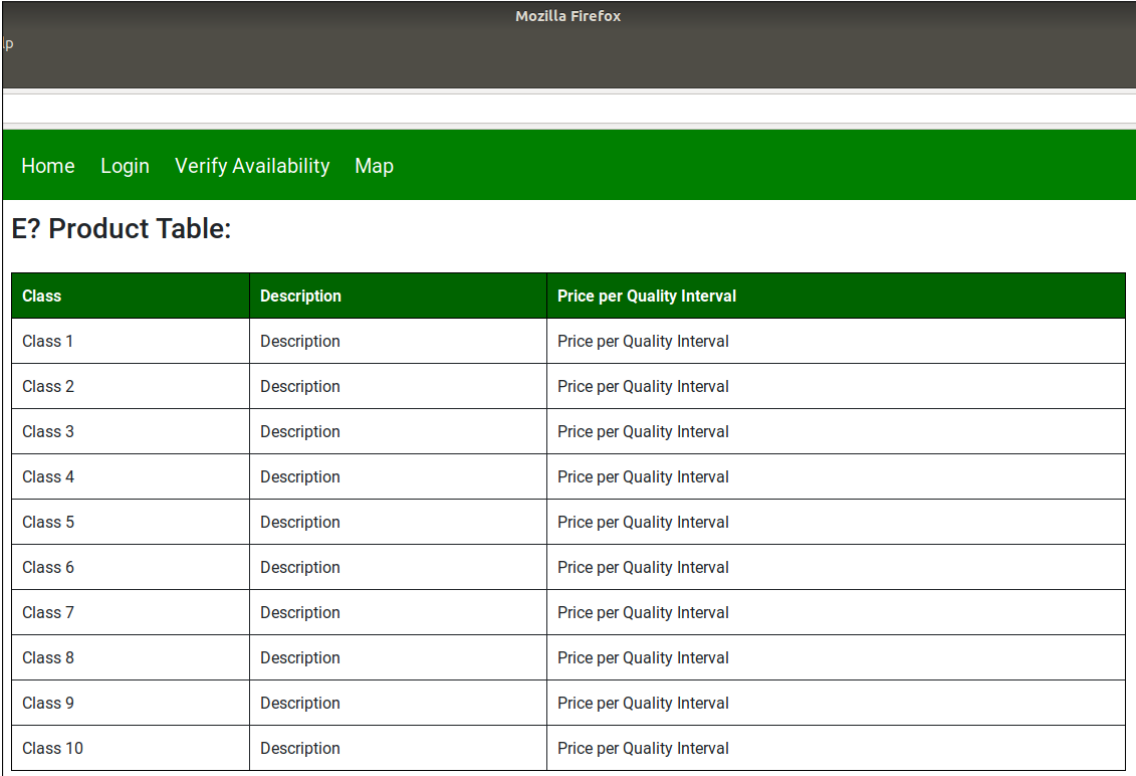


Figure 3.19: Order management module homepage.

The only other function available for non-users is a search function to search for products currently in inventory. This search has two required parameters, the class identifier and the time interval until delivery. Figure 3.20 shows the form and Figure 3.21 shows the result of the form.

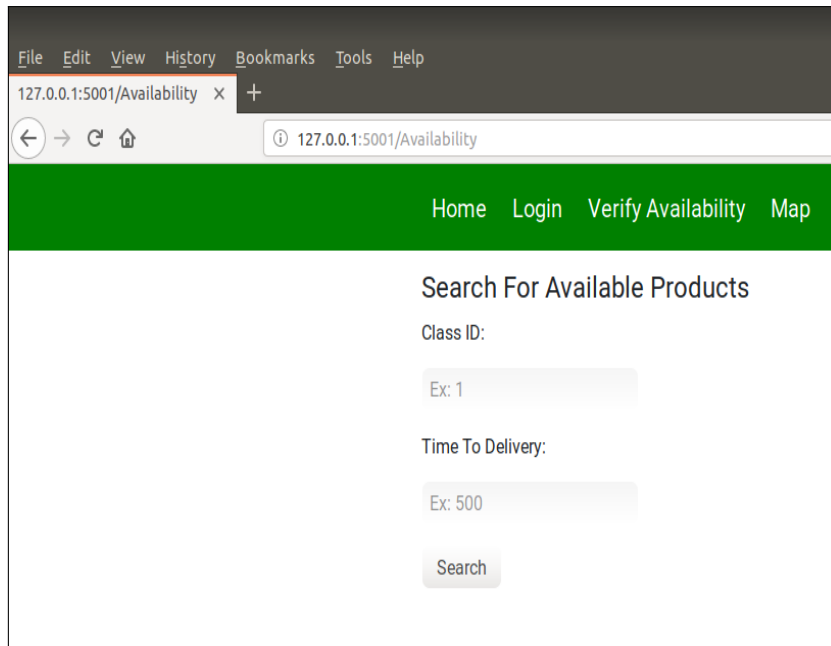


Figure 3.20: Order management module availability form.

availabilityResult

Home Login Verify Availability Map

Search Results:

Entry ID	Entry Quantity	Class	Quality Limit	EaR	Last Update Quality	Last Update Keeping Quality
M66G2Y	500.0	1	1.0	13018.0	9901.918405	126919.151194
YFGTN1	998.0	1	1.0	13018.0	9901.918514	126919.151194
CPNFZB	1000.0	1	1.0	13018.0	9901.918558	126919.151194
M3W368	1000.0	1	1.0	13018.0	9901.918602	126919.151194
EXIDWZ	998.0	1	1.0	13018.0	9901.918646	126919.151194
MU32BG	1000.0	1	1.0	13018.0	9901.918667	126919.151194
7O28OD	1000.0	1	1.0	13018.0	9901.918711	126919.151194
TSYAC0	1000.0	1	1.0	13018.0	9901.918733	126919.151194
3WNTM9	1000.0	1	1.0	13018.0	9901.918777	126919.151194
U28YDF	1000.0	1	1.0	13018.0	9901.918821	126919.151194

Figure 3.21: Order management module availability result.

For users, that is, persons or entities related to a company and with login privileges, more functions are available. These functions are locked behind a login screen as Figure 3.22 illustrates.

Safety and quality based traceability system for food supply chains

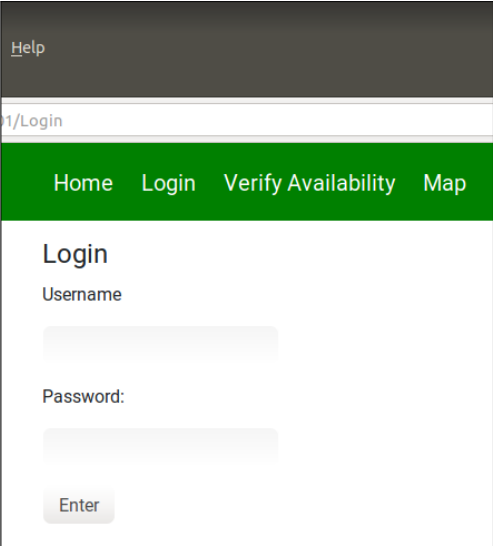


Figure 3.22: Order management module login page.

By providing valid credentials a user is taken to a personal page. This page serves only as a place in which user available functions are aggregated via hyperlinks in a sidebar as Figure 3.23 shows.

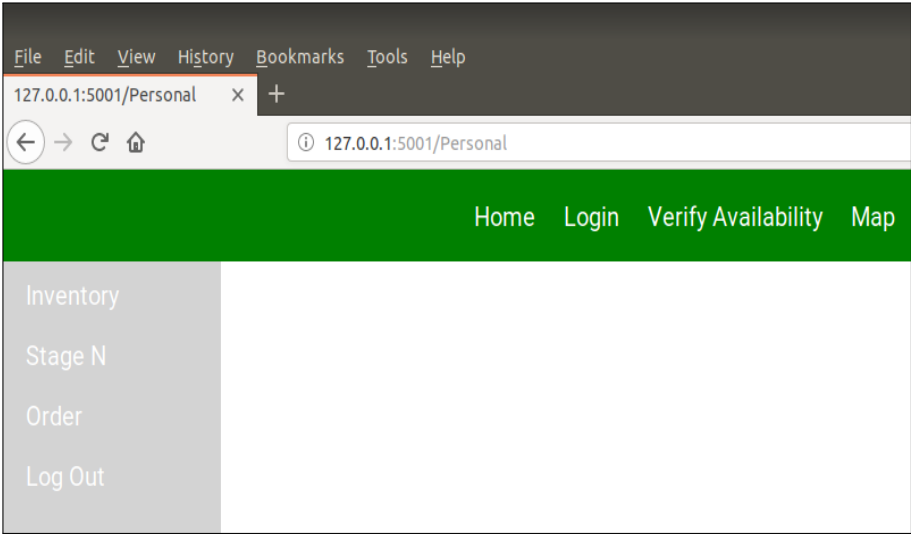


Figure 3.23: Order management module personal page.

The inventory page shows the inventory like in the IMM. The Stage N page shows the operational history as the PSMM and, just as that module, needs to be replicated per production stage described in an HACCP flow chart or equivalent health and safety mechanism. Figure 3.24 shows the inventory page and Figure 3.25 shows the Stage N page.

Mozilla Firefox

File Edit View History Bookmarks Tools Help

127.0.0.1:5001/Inventory

127.0.0.1:5001/Inventory

Home Login Verify Availability Map

Entry Date	Company ID	Entry ID	Entry QMP	Entry Quantity	Class	Initial Quality	Quality Limit	EaR	Initial Keeping Quality	Initial Quality Algorithm	Initial Keeping Quality Algorithm	Last Update	Last Update QMP	Last Update Quality	Last Update Keeping Quality	Last Update Quality Algorithm
2019-08-26 21:59:32	E1	M66G2Y	10.00	500.0	1	10000.0	1.0	13018.0	9999.000000	EQAID 1	EKQAID 1	2019-10-17 18:57:36	-4.83	9901.918405	1.269192e+05	EQAID
2019-08-26 21:59:37	E1	YFGTN1	10.00	998.0	1	10000.0	1.0	13018.0	9999.000000	EQAID 1	EKQAID 1	2019-10-17 18:57:36	-4.83	9901.918514	1.269192e+05	EQAID
2019-08-26 21:59:39	E1	CPNFZB	10.00	1000.0	1	10000.0	1.0	13018.0	9999.000000	EQAID 1	EKQAID 1	2019-10-17 18:57:36	-4.83	9901.918558	1.269192e+05	EQAID
2019-08-26 21:59:41	E1	M3W368	10.00	1000.0	1	10000.0	1.0	13018.0	9999.000000	EQAID 1	EKQAID 1	2019-10-17 18:57:36	-4.83	9901.918602	1.269192e+05	EQAID
2019-08-26 21:59:43	E1	EXIDWZ	10.00	998.0	1	10000.0	1.0	13018.0	9999.000000	EQAID 1	EKQAID 1	2019-10-17 18:57:36	-4.83	9901.918646	1.269192e+05	EQAID
2019-08-26 21:59:44	E1	MU32BG	10.00	1000.0	1	10000.0	1.0	13018.0	9999.000000	EQAID 1	EKQAID 1	2019-10-17 18:57:36	-4.83	9901.918667	1.269192e+05	EQAID
2019-08-26 21:59:46	E1	7O28OD	10.00	1000.0	1	10000.0	1.0	13018.0	9999.000000	EQAID 1	EKQAID 1	2019-10-17 18:57:36	-4.83	9901.918711	1.269192e+05	EQAID
2019-08-26 21:59:47	E1	TSYAC0	10.00	1000.0	1	10000.0	1.0	13018.0	9999.000000	EQAID 1	EKQAID 1	2019-10-17 18:57:36	-4.83	9901.918733	1.269192e+05	EQAID
2019-08-26 21:59:49	E1	3WNTM9	10.00	1000.0	1	10000.0	1.0	13018.0	9999.000000	EQAID 1	EKQAID 1	2019-10-17 18:57:36	-4.83	9901.918777	1.269192e+05	EQAID
2019-08-26 21:59:51	E1	U28YDF	10.00	1000.0	1	10000.0	1.0	13018.0	9999.000000	EQAID 1	EKQAID 1	2019-10-17 18:57:36	-4.83	9901.918821	1.269192e+05	EQAID

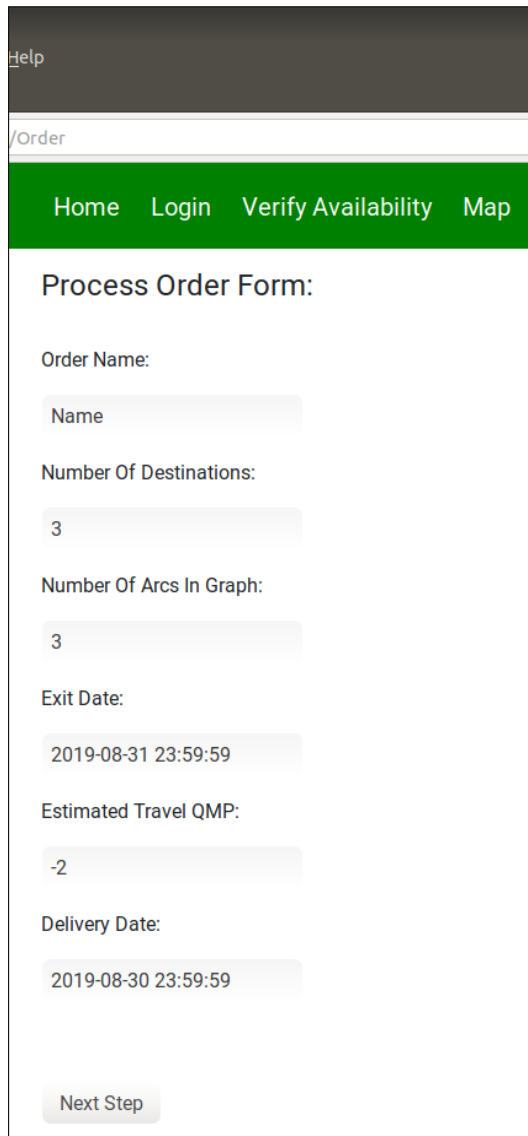
Figure 3.24: Order management module inventory page.

RM ID	RM Quantity	RM Time Stamp	RM Last Update	RM QMP	RM Quality Limit	RM EaR	RM Quality	RM Keeping Quality	RM Quality Algorithm	RM Keeping Quality Algorithm	IP ID	IP Quantity	IP Time Stamp	IP Last Update	IP QMP
M66G2Y	50.0	2019-08-28 03:49:48	2019-08-26 22:00:58	-4.83	1.0	13018.0	9999.998117975125	126919.15119439851	EQAID 1	EKQAID 1	--	--	--	--	--
--	--	--	--	--	--	--	--	--	--	--	ONKE0V	50.0	2019-08-28 03:49:52	2019-08-26 22:00:58	-4.83
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
M66G2Y	50.0	2019-08-28 03:51:18	2019-08-26 22:00:58	-4.83	1.0	13018.0	9999.998117975125	126919.15119439851	EQAID 1	EKQAID 1	--	--	--	--	--
--	--	--	--	--	--	--	--	--	--	--	ONKE0V	50.0	2019-08-28 03:51:48	2019-08-26 22:00:58	-4.83
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
RM ID	RM Quantity	RM Time Stamp	RM Last Update	RM QMP	RM Quality Limit	RM EaR	RM Quality	RM Keeping Quality	RM Quality Algorithm	RM Keeping Quality Algorithm	IP ID	IP Quantity	IP Time Stamp	IP Last Update	IP QMP
M66G2Y	50.0	2019-08-28 03:49:48	2019-08-26 22:00:58	-4.83	1.0	13018.0	9999.998117975125	126919.15119439851	EQAID 1	EKQAID 1	--	--	--	--	--
--	--	--	--	--	--	--	--	--	--	--	ONKE0V	50.0	2019-08-28 03:49:52	2019-08-26 22:00:58	-4.83
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Figure 3.25: Order management module stage N page.

Safety and quality based traceability system for food supply chains

The order page allows for a user to process an order when requested. A form must be filled. This form has three steps, the first two querying the user about the order and the third with a final recap of all information given and a final submission button. Figure 3.26 shows the first step of the form.



The screenshot shows a web application interface for processing an order. At the top, there is a dark grey header with a 'Help' link. Below the header is a white navigation bar with a green background containing the links 'Home', 'Login', 'Verify Availability', and 'Map'. The main content area is titled 'Process Order Form:' and contains several input fields with pre-filled values:

- Order Name: Name
- Number Of Destinations: 3
- Number Of Arcs In Graph: 3
- Exit Date: 2019-08-31 23:59:59
- Estimated Travel QMP: -2
- Delivery Date: 2019-08-30 23:59:59

At the bottom of the form is a 'Next Step' button.

Figure 3.26: Order management module first step of the order form.

The second step contains information relative to the first step plus some more necessary input fields as Figure 3.27 shows.

Process Order Form:

Previous Data:

Order Name: Name

Number Of Destinations: 3

Number Of Arcs In Graph : 3

Exit Date: 2019-08-31 23:59:59

Travel QMP: -2

Delivery Date: 2019-08-30 23:59:59

Summary Table:

Home	1, 2, 3	1, 2, 3
Home	1, 2, 3	1, 2, 3
Home	1, 2, 3	1, 2, 3

Construct Graph:

Home	Destination 1	1
Destination 1	Destination 2	1
Destination 2	Destination 3	1

Make Order

Figure 3.27: Order management module second step of the order form.

The third step is a mere input recap and a final submission button. Figure 3.28 illustrates this step. All created files will be detailed further down this section.

127.0.0.1:5001/OrderResult x + Mozilla Firefox

127.0.0.1:5001/OrderResult

Home Login Verify Availability Map

Order OrderName Documents:

Submit

Order Name	Number Of Destinations	Number Of Arcs In Graph	Exit Date	Travel QMP	Delivery Date
OrderName	3	3	2019/12/30 23:59:59	-2	2019/12/31 23:59:59

Unnamed: 0	Unnamed: 1	Destination 1	Destination 2	Destination 3
Goods To Deliver	ID1 ID2 ID3	ID1 ID2 ID3	ID1 ID2 ID3	NaN
Quantity To Deliver	Q1 Q2 Q3	Q1 Q2 Q3	Q1 Q2 Q3	NaN

ID	Exit Date Quality	Exit Date Keeping Quality	Delivery Date Quality	Delivery Date Keeping Quality
ID1	EDQ1	EDKQ1	DDQ1	DDKQ1
ID2	EDQ2	EDKQ2	DDQ2	DDKQ2
ID3	EDQ3	EDKQ3	DDQ3	DDKQ3

Origins	Destinations	Relative Cost
Destination 2	Destination 3'	'1 1 1

Origin	Destination	Relative Cost
0	1	1
1	2	1
2	3	1

Figure 3.28: Order management module third step of the order form.

Safety and quality based traceability system for food supply chains

Pressing the final submission button, a number of files necessary for an order is created and showed to the user via tables. Some of the files correspond to user input and though they may seem unnecessary due to every input being shown in the application. Its intent is to create an history of all orders. All the other files contain processed information that is necessary for the completion of an order.

The Summary file shows all perishables to deliver and to which destination. Table 3.3 presents the structure of this file.

Table 3.3: Summary file structure.

	Destination 1	Destination 2	Destination 3
Goods to deliver	ID1 ID2 ID3	ID1 ID2 ID3	ID1 ID2 ID3
Quantity to deliver	Q1 Q2 Q3	Q1 Q2 Q3	Q1 Q2 Q3

The Conditions file shows the conditions expected to be subjected to an order. These include, number of destinations, number of possible routes, exit and delivery dates and the relevant conditions inside the transport vehicle. Temperature is considered the most relevant QMP for this study. Table 3.4 shows this file.

Table 3.4: Conditions file structure.

Order name	Number of destinations	Number of arcs in graph
OrderName	3	3
Exit date	Travel QMP	Delivery date
2019/12/30 23:59:59	-2	2019/12/31 23:59:59

The Graph file shows the origins, destinations and relative cost between origin destination pairs. This file is necessary for the application of Prim's algorithm. This algorithm determines the best path for distribution. The structure is shown in Table 3.5.

Table 3.5: Graph file structure.

Origins	Destinations	Relative cost
Home Destination 1 Destination 2	Destination 1 Destination 2 Destination 3	1 1 1

The IFC and IDC files are also created. Due to their structure already discussed, it is considered that there is no necessity to further expose them.

The quality file contains the expected quality and keeping quality of products at delivery using the predicted exit and delivery dates plus the expected travel conditions. Table 3.6 illustrates the structure of this file.

Table 3.6: Quality file structure.

ID	Exit date quality	Exit date keeping quality	Delivery date quality	Delivery date keeping quality
ID1	EDQ1	EDKQ1	DDQ1	DDKQ1
ID2	EDQ2	EDKQ2	DDQ2	DDKQ2
ID3	EDQ3	EDKQ3	DDQ3	DDKQ3

The Route file contains the result of the application of the Prim's algorithm to the given graph and, in this case, returns the cheapest path through all destinations. Table 3.7 shows the structure of this file.

All the files are contained within a folder with the same name as the order.

Table 3.7: Route file structure.

Origin	Destination	Relative cost
0	1	1
1	2	1
2	3	1

3.6.2 Regulator layer

3.6.2.1 Non-user available functions

For non-users, meaning the general public there is only one function available for them. That function corresponds to the RIFC, in which any individual can view the cumulative product history. To do so, the individual has to choose "Search" on the regulator home page as Figure 3.29 illustrates.

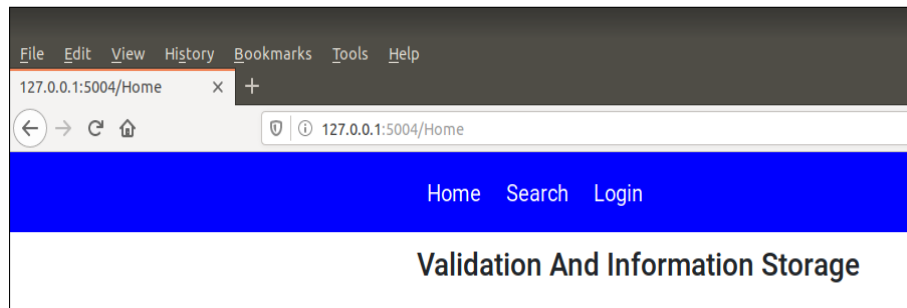


Figure 3.29: Regulator home page.

Then the individual has to input the desired ID in the search box as Figure 3.30 shows.

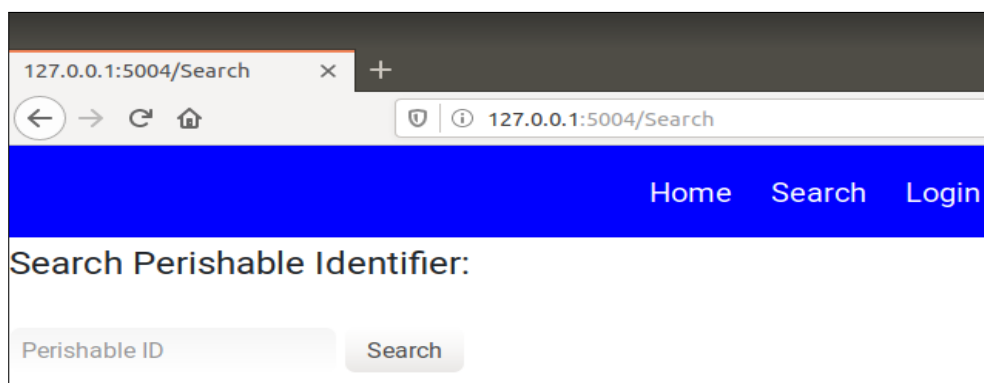


Figure 3.30: Regulator search page.

Finally the results are presented as Figures 3.31 through 3.33 show.

Safety and quality based traceability system for food supply chains

Search Results:

Component ID	Company ID	Start Date	Start Quality	Description
X ID	Company ID	dd/mm/yy h:m	X hours	Description of X
Y ID	Company ID	dd/mm/yy h:m	Y hours	Description of Y
Z ID	Company ID	dd/mm/yy h:m	Z hours	Description of Z
Inventory QMP	Inventory Algorithm	Inventory Exit Quality	Production Stage 1 Components	Production Stage 1 Description
X QMP value	X Algorithm ID	X hours	X ID	In this stage X was done
Y QMP value	Y Algorithm ID	Y hours	Y ID	In this stage Y was done
Z QMP value	Z Algorithm ID	Z hours	Z ID	In this stage Z was done
Production Stage 1 Date	Production Stage 1 QMP	Production Stage 1 Algorithm	Production Stage 1 Exit Quality	Production Stage 1 Exit ID
dd/mm/yy h:m	X QMP value	X Algorithm ID	A hours	A ID
dd/mm/yy h:m	Y QMP value	Y Algorithm ID	B hours	B ID
dd/mm/yy h:m	Z QMP value	Z Algorithm ID	C hours	C ID
Production Stage N Components	Production Stage N Description	Production Stage N Date	Production Stage N QMP	Production Stage N Algorithm
A ID	In this stage A was done	dd/mm/yy h:m	A QMP value	A Algorithm ID
B ID	In this stage B was done	dd/mm/yy h:m	B QMP value	B Algorithm ID
C ID	In this stage C was done	dd/mm/yy h:m	C QMP value	C Algorithm ID
Production Stage N Exit Quality	Production Stage N Exit ID	End Date	End QMP	End Algorithm
D hours	D ID	dd/mm/yy h:m	D QMP value	D Algorithm ID
E hours	E ID	dd/mm/yy h:m	E QMP value	E Algorithm ID
F hours	F ID	dd/mm/yy h:m	F QMP value	F Algorithm ID
End Quality	End Internal ID	End External ID	NaN	NaN
D hours	D ID	G ID	NaN	NaN
E hours	E ID	H ID	NaN	NaN
F hours	F ID	I ID	NaN	NaN

Figure 3.31: Regulator search result page.

Safety and quality based traceability system for food supply chains

Search Results:				
Component ID	Company ID	Start Date	Start Quality	Description
G ID	Company ID	dd/mm/yy h:m	G hours	Description of G
H ID	Company ID	dd/mm/yy h:m	H hours	Description of H
I ID	Company ID	dd/mm/yy h:m	I hours	Description of I
Inventory QMP	Inventory Algorithm	Inventory Exit Quality	Production Stage 1 Components	Production Stage 1 Description
G QMP value	G Algorithm ID	G hours	G ID	In this stage G was done
H QMP value	H Algorithm ID	H hours	H ID	In this stage H was done
I QMP value	I Algorithm ID	I hours	I ID	In this stage I was done
Production Stage 1 Date	Production Stage 1 QMP	Production Stage 1 Algorithm	Production Stage 1 Exit Quality	Production Stage 1 Exit ID
dd/mm/yy h:m	G QMP value	G Algorithm ID	J hours	J ID
dd/mm/yy h:m	H QMP value	H Algorithm ID	K hours	K ID
dd/mm/yy h:m	I QMP value	I Algorithm ID	L hours	L ID
Production Stage N Components	Production Stage N Description	Production Stage N Date	Production Stage N QMP	Production Stage N Algorithm
J ID	In this stage J was done	dd/mm/yy h:m	J QMP value	J Algorithm ID
K ID	In this stage K was done	dd/mm/yy h:m	K QMP value	K Algorithm ID
L ID	In this stage L was done	dd/mm/yy h:m	L QMP value	L Algorithm ID
Production Stage N Exit Quality	Production Stage N Exit ID	End Date	End QMP	End Algorithm
M hours	M ID	dd/mm/yy h:m	M QMP value	M Algorithm ID
N hours	N ID	dd/mm/yy h:m	N QMP value	N Algorithm ID
O hours	O ID	dd/mm/yy h:m	O QMP value	O Algorithm ID
End Quality	End Internal ID	End External ID	NaN	NaN
M hours	M ID	P ID	NaN	NaN
N hours	N ID	Q ID	NaN	NaN
O hours	O ID	R ID	NaN	NaN

Figure 3.32: Regulator search result page.

Safety and quality based traceability system for food supply chains

Search Results:				
Component ID	Company ID	Start Date	Start Quality	Description
P ID	Company ID	dd/mm/yy h:m	P hours	Description of P
Q ID	Company ID	dd/mm/yy h:m	Q hours	Description of Q
R ID	Company ID	dd/mm/yy h:m	R hours	Description of R
Inventory QMP	Inventory Algorithm	Inventory Exit Quality	Production Stage 1 Components	Production Stage 1 Description
P QMP value	P Algorithm ID	P hours	P ID	In this stage P was done
Q QMP value	Q Algorithm ID	Q hours	Q ID	In this stage Q was done
R QMP value	R Algorithm ID	R hours	R ID	In this stage R was done
Production Stage 1 Date	Production Stage 1 QMP	Production Stage 1 Algorithm	Production Stage 1 Exit Quality	Production Stage 1 Exit ID
dd/mm/yy h:m	P QMP value	P Algorithm ID	S hours	S ID
dd/mm/yy h:m	Q QMP value	Q Algorithm ID	T hours	T ID
dd/mm/yy h:m	R QMP value	R Algorithm ID	U hours	U ID
Production Stage N Components	Production Stage N Description	Production Stage N Date	Production Stage N QMP	Production Stage N Algorithm
S ID	In this stage S was done	dd/mm/yy h:m	S QMP value	S Algorithm ID
T ID	In this stage T was done	dd/mm/yy h:m	T QMP value	T Algorithm ID
U ID	In this stage U was done	dd/mm/yy h:m	U QMP value	U Algorithm ID
Production Stage N Exit Quality	Production Stage N Exit ID	End Date	End QMP	End Algorithm
X1 hours	X1 ID	dd/mm/yy h:m	X1 QMP value	X1 Algorithm ID
Y1 hours	Y1 ID	dd/mm/yy h:m	Y1 QMP value	Y1 Algorithm ID
Z1 hours	Z1 ID	dd/mm/yy h:m	Z1 QMP value	Z1 Algorithm ID
End Quality	End Internal ID	End External ID	NaN	NaN
X1 hours	X1 ID	X2 ID	NaN	NaN
Y1 hours	Y1 ID	Y2 ID	NaN	NaN

Figure 3.33: Regulator search result page.

3.6.2.2 User available functions

Just as in the Order Management Module, users will have to login to access certain functions as Figure 3.34

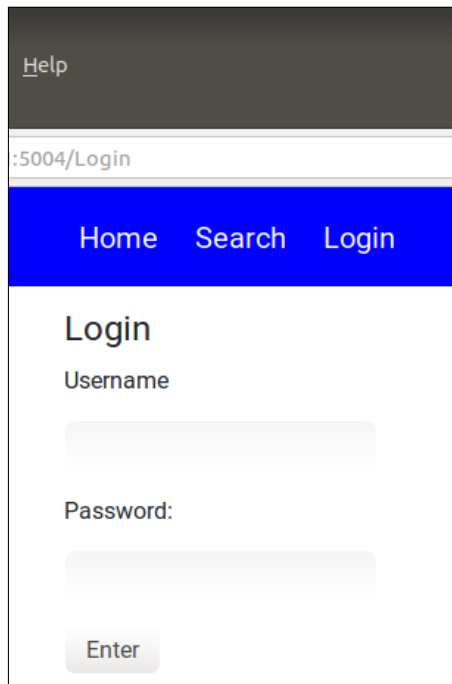


Figure 3.34: Regulator login page.

After logging in, the users will be presented with personal page just as the Order Management Module as show in Figure 3.35

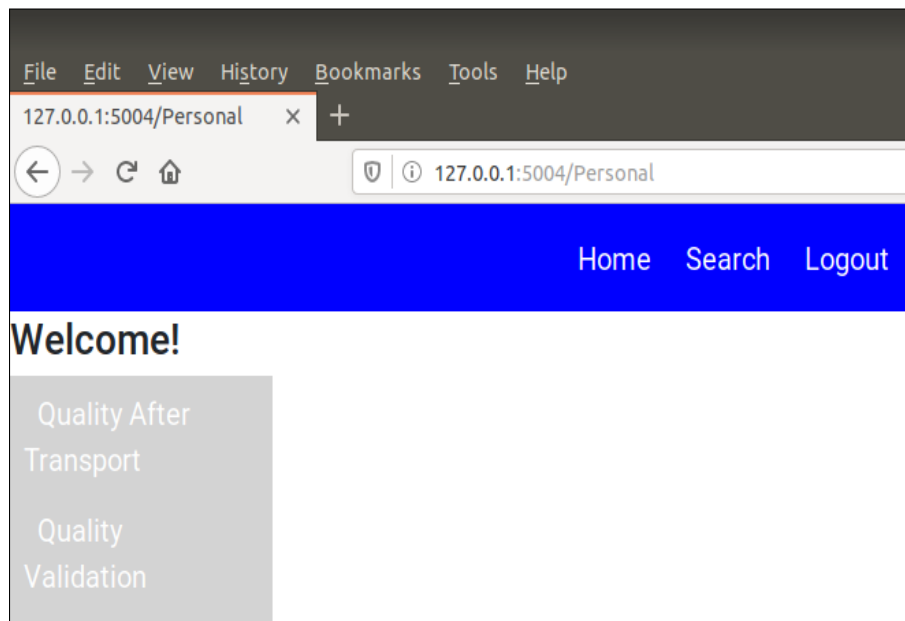


Figure 3.35: Regulator layer personal page.

From the sidebar there are two new functions available. The first corresponds to the Quality After Transport module and the second to the Quality Validation module.

Safety and quality based traceability system for food supply chains

The Quality After Transport module consists in a three step form as Figures 3.36 through 3.38 illustrate.

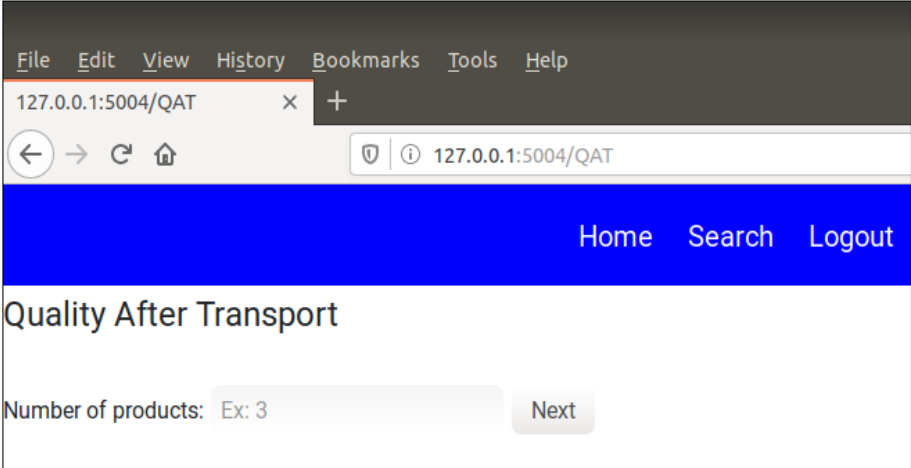


Figure 3.36: Quality after transport step 1.

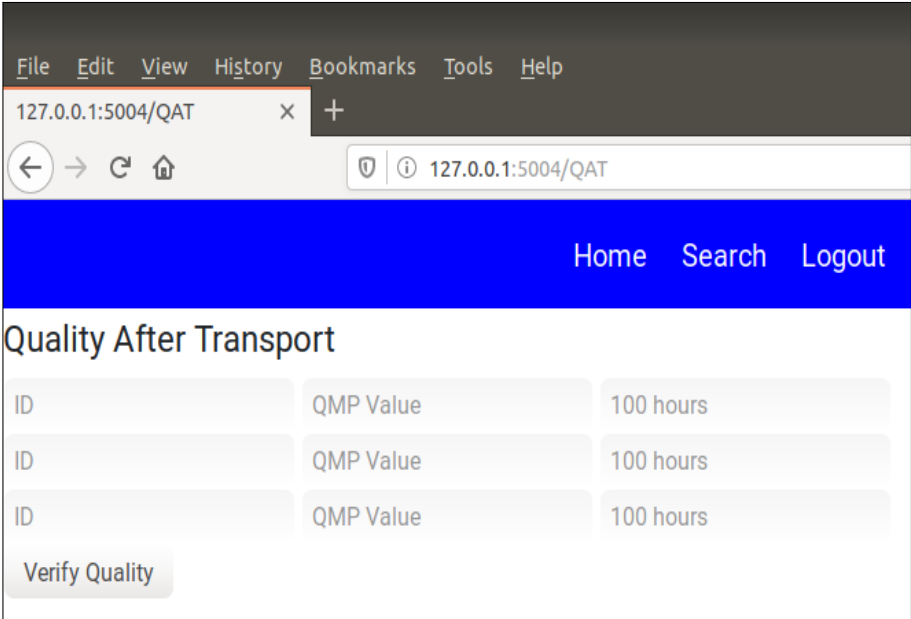


Figure 3.37: Quality after transport step 2.

Safety and quality based traceability system for food supply chains

ID	Initial Quality	EaR	Quality Limit	Last Update	Last Update Quality	Last Update Quality Algorithm	Last Update Keeping Quality	QMP	Calculated Quality	Quality Delta	Expected Keeping Quality	Calculated Keeping Quality	Keeping Quality Delta
ID1	IQ1	EaR1	QL1	LU1	LUQ1	LUQA1	LUKQ1	QMP1	CQ1	QD1	EKQ1	CKQ1	KQD1
ID2	IQ2	EaR2	QL2	LU2	LUQ2	LUQA2	LUKQ2	QMP2	CQ2	QD2	EKQ2	CKQ2	KQD2
ID3	IQ3	EaR3	QL3	LU3	LUQ3	LUQA3	LUKQ3	QMP3	CQ3	QD3	EKQ3	CKQ3	KQD3

Figure 3.38: Quality after transport step 3.

The Quality Validation consists on a form that asks the user for the IFC and IDC files as shown in Figure 3.39 and then warns the user that the validated files are ready, as illustrated by Figure 3.40. There should be a prompt for download of a .zip file that contains both files.

Figure 3.39: Quality validation form.

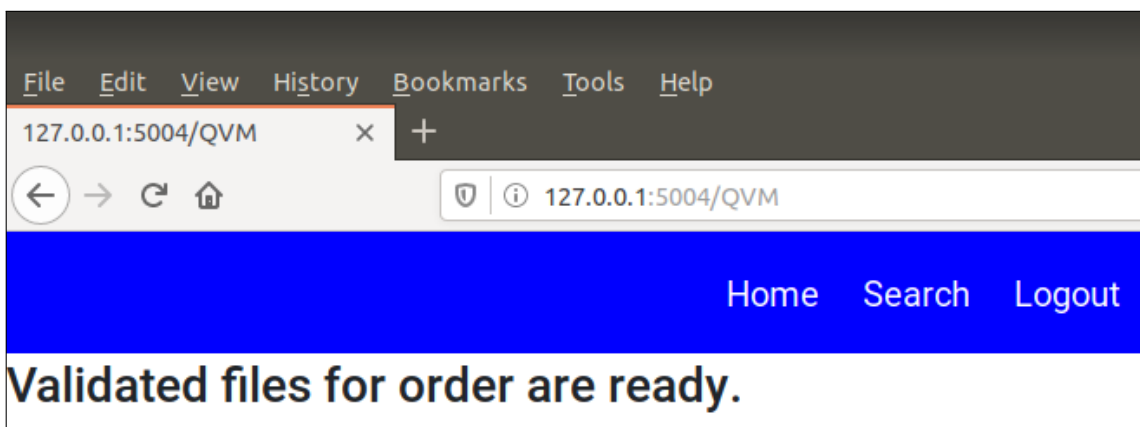


Figure 3.40: Quality validation result.

3.6.3 Unique code layer

This layer does not possess any kind of graphical interface as its only objective is to emulate any kind of entity that is able to uniquely identify a product within the application range of the presented framework.

Chapter 4

Simulation

In order to test the created tools three materials will be followed throughout a supply chain comprehending three different companies, one per segment. As this simulation aims to illustrate quality and keeping quality variations over time and temperature, the materials were subjected to different entry parameters in order to better observe their impact in quality and keeping quality.

First the algorithms used in the created modules are described. However, due to the simplicity of the supply chain simulated, the routing algorithm is irrelevant for simulation and will not be exposed further after this explanation.

Following that presentation, the simulation itself is shown. First segment by segment and concluding with data from the entire chain.

4.1 Algorithms and formulas

This section aims to detail essential methods for the determination of the two most relevant objectives of the created modules, the decay in quality and keeping quality and the determination of transport or distributions routes and their influence in quality and keeping quality.

4.1.1 Quality and keeping quality decay formulas

The linear reaction equations were used from Tijskens and Polderdijk[99]. Although these equation were determined taking into account constant temperature, which is not realistic in the context of food supply chains, they can still be used to evaluate quality and keeping quality since the temperature intervals are small. Even though this assumption induces some error in the evaluation of those parameters. Although, it is considered acceptable for prototyping purposes.

Equation 4.1 is used for the decay of quality:

$$Q = Q_0 - kt \quad (4.1)$$

Where Q symbolizes the value of quality, Q_0 symbolizes the initial quality, k symbolizes the decay rate and t the time elapsed.

Equation 4.2 is used for the evaluation of keeping quality:

$$KQ = \frac{Q_0 - Q_l}{k} \quad (4.2)$$

Where Q_l symbolizes the quality limit.

The value of k varies with temperature and is calculated using Equation 4.3:

$$k = k_{ref} e^{\frac{E_a}{R} \left(\frac{1}{T_{ref}} - \frac{1}{T_{abs}} \right)} \quad (4.3)$$

Safety and quality based traceability system for food supply chains

Where k_{ref} is the reference decay rate (has the value of one), Ea is the energy of activation, R is the gas constant, T_{ref} is the reference temperature (has the value of 10°Celsius) and T_{abs} is the measured absolute temperature.

4.1.2 Routing algorithm

The Prim's algorithm is used to determine transport and distribution routes. To use this algorithm a graph must be made. The graph contains nodes, which are locations, and arches, representing all possible relationships between the nodes. This algorithm determines a route that passes through all nodes in the graph with the lowest relationship between them. Usually the arches represent the travel cost between nodes and so the algorithm will determine the cheapest route that passes through all nodes. Although the arches can symbolize whatever a company values most, here the travel cost was the relevant parameter.

4.2 Entry data

The first material has an initial quality value of 100 and will be subjected to temperatures ranging from 0 to 5°C. The second material has an initial quality of 70 and will be subjected to the same temperatures. The final material has an initial quality of 100 but is subjected to temperatures ranging from 2.5 to 7.5°C.

For the time elapsed between companies, the example from [99] was followed. This example can be seen in Table 4.1.

Table 4.1: Tijskens and Polderdijk [99] Table 4.

Action	Temp(°C)	Time (hr)
From auction to export firm		
Transport	8	1
Packaging	16	1
Storage	8	4
From export firm to distribution centre		
Transport	8	4
Storage	8	2
From distribution centre to retail		
Transport	19	2
Sales	19	48

As the course of action illustrated on the table is not adequate to the developed system, only the time elapsed will be used. For the temperature, initial quality and quality limit the Table 4.2 summarizes the remaining parameters used for simulation

Table 4.2: Entry data for simulation.

Material	Initial quality	Quality limit	Temperature range
1	100	60	0°-5°C
2	70	60	0°-5°C
3	100	60	2.5°-7.5°C

4.3 First segment results

Following the time intervals shown in Table 4.1 but adapting each interval to the modules presented in this dissertation it can be defined:

- 1 hour in inventory with a randomized temperature value within the above limits
- 1 hour in processing with a randomized temperature value within the above limits,
- 4 hours in transport to the second segment with a randomized temperature value within the above limits.

To increase readability, tables will be shown and only contain the parameters necessary for the observation of quality decay. It should also be noted that it is considered that there no waste during processing and transport, thus making quantity irrelevant for simulation.

4.3.1 Inventory management module results

Upon using the values shown in Table 4.1 and Table 4.2, the inventory data concerning initial data is shown in Table 4.3.

Table 4.3: First segment inventory entry.

Material	Entry ID	Entry QMP	Initial quality	Quality limit	Ea/R	Initial keeping quality
1	X8636N	8	100	60	13018	55.48
2	WSZOCQ	8	70	60	13018	13.87
3	MDLE5Z	8	100	60	13018	55.48

Table 4.4 shows the variation in quality and keeping quality after 1 hour in inventory.

Table 4.4: First segment inventory updated values.

Material	Entry ID	Last update QMP	Last update quality	Last update keeping quality
1	X8636N	4.46	99.60	100.25
2	WSZOCQ	4.46	69.60	25.06
3	MDLE5Z	6.47	99.44	71.42

4.3.2 Processing stage management module results

Here it is considered that all items in inventory are input as raw materials for processing for 1 hour. All information for processing input is described previously in Table 4.4. However, a small change was made. As the created modules are general templates that must be adapted to each specific circumstance, it was considered that processing stage exit quality is possible to be determined. In this simulation it is considered that processing does not increase or decrease quality and keeping quality. All changes observed in those parameters are due to temperature and time just as the case of materials resting in inventory. Operational history concerning the process output is described in Table 4.5.

Table 4.5: First segment processing stage management module output.

Material	Exit ID	Exit QMP	Exit quality	Exit keeping quality
1	JAHQDZ	1.24	99.14	173.91
2	FKA5OY	1.24	69.14	43.48
3	31MBRO	7.04	98.77	65.04

4.3.3 Quality validation module results

In this simulation it is assumed that no illegality or lack of care for product quality was made. Thus, this module would merely present the same results as Table 4.5 if the products were immediately validated for sale after processing only adding one unique code for each material. The codes are JDCVJM93JENF, UJT4A91RHVD4 and RRFPDZQAGN4K for Material 1, Material 2 and Material 3 respectively.

4.3.4 Order management module results

Here it is considered that transport takes 4 hours between the first segment company and the second segment company. The predicted values of quality and keeping quality at arrival are shown in Table 4.6.

Table 4.6: First segment order management module prediction.

Material	External Product ID	Predicted travel QMP	Quality prediction	Keeping quality prediction
1	JDCVJM93JENF	4.41	97.62	101.01
2	UJT4A91RHVD4	4.41	67.62	25.25
3	RRFPDZQAGN4K	5.61	97.09	82.47

4.3.5 First segment quality and keeping quality variation

In order to better illustrate the variation of quality and keeping this section contains graphs of their that show more clearly all variations that products suffer in this first segment. Figure 4.1 shows quality decay and Figure 4.2 shows the variation in keeping quality.

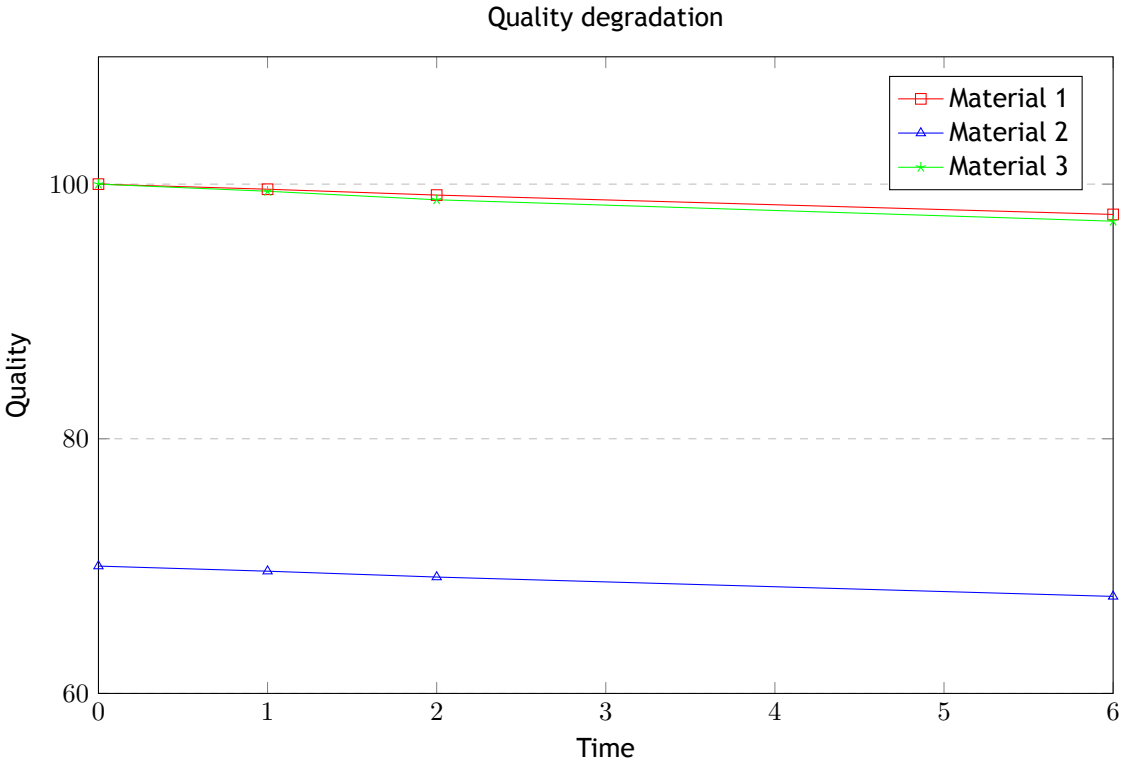


Figure 4.1: Quality degradation throughout the first segment.

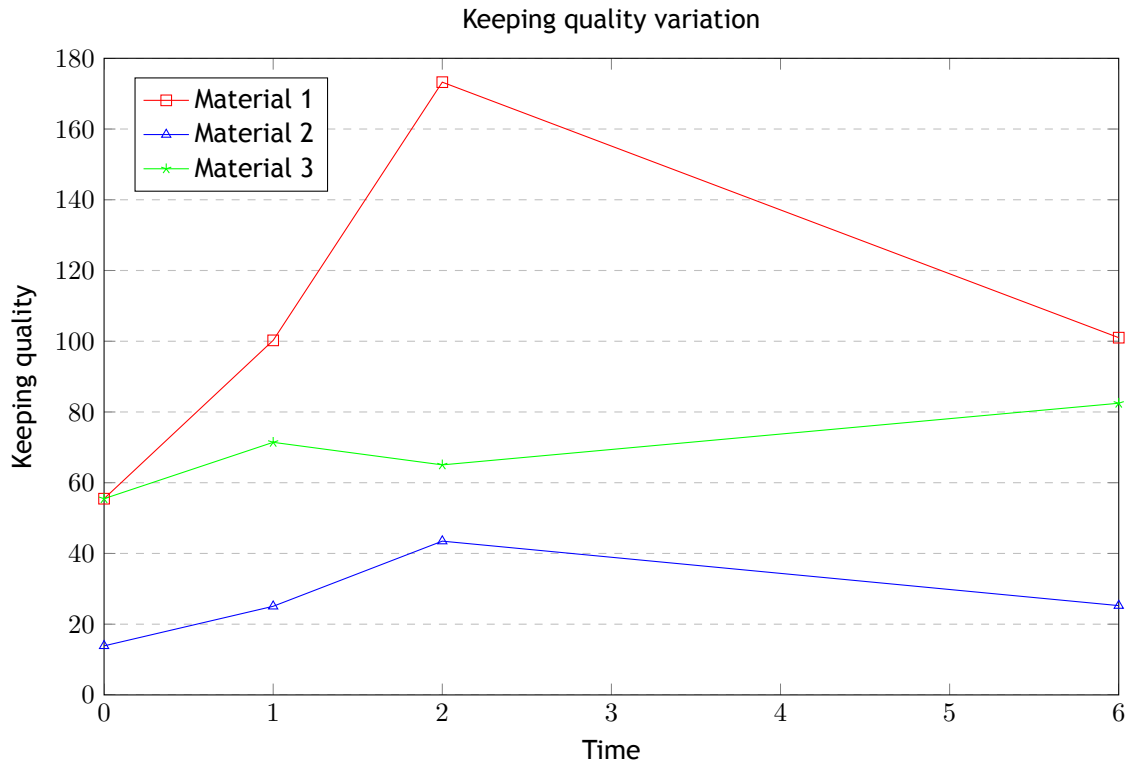


Figure 4.2: Keeping quality variation throughout the first segment.

4.4 Second segment results

In order to better demonstrate the operation of this module, products were received with different quality than expected but still accepted by the company in this segment. This difference in quality comes from worse conditions in transport than expected. For illustrative purposes is the temperature during transport was 1°C higher than predicted.

In this segment the following sequence of events is simulated:

- materials did not rest in inventory, instead they were immediately allocated for processing without considering any time interval between reception, quality verification and movement from inventory to production,
- materials spent 4 hours in processing at a randomized temperature value within the described interval,
- materials spent 2 hours in transport subjected to a randomized temperature value within the limits described.

4.4.1 Quality after transport module results

The Quality After Transport module detected this change and presents it to the second segment company. Tables 4.7 and 4.8 show all relevant information taken from the referred module.

Safety and quality based traceability system for food supply chains

Table 4.7: Quality After Transport module entry data.

Material	External Product ID	Last update quality	Last update keeping quality	Travel QMP
1	JDCVJM93JENF	97.62	101.01	5.41
2	UJT4A91RHVD4	67.62	25.25	5.41
3	RRFPDZQAGN4K	97.09	82.47	6.61

Table 4.8: Quality After Transport module results.

Material	Quality	Quality delta	Keeping quality	Keeping quality delta
1	97.19	0.43	85.29	15.72
2	67.19	0.43	21.32	3.93
3	96.56	0.53	69.81	12.66

4.4.2 Inventory management module results

The results show in Table 4.8 would be the entries to the Inventory Management Module. However, in order to follow the time interval shown in Table 4.1 it is considered that the products were immediately allocated to production and did not rest in inventory.

4.4.3 Processing stage management module results

Table 4.9 shows all relevant information taken from this module after 4 hours in processing.

Table 4.9: Second segment processing stage management module output.

Material	Exit ID	Exit QMP	Exit quality	Exit keeping quality
1	VVHZ33	2.03	96.13	140.87
2	PTCA6W	2.03	66.13	27.23
3	N7MSZG	5.75	94.58	73.71

4.4.4 Quality validation module results

Just as the previous segment, here is considered that the products were immediately validated without any issue as well.

4.4.5 Order management module results

Table 4.10 shows the predictions made in by this module for a transport that lasts 2 hours.

Table 4.10: Second segment order management module prediction.

Material	External Product ID	Predicted travel QMP	Quality prediction	Keeping quality prediction
1	80S5LF80S5LF	0.88	95.89	171.38
2	FHVVMNOLY66G	0.88	65.89	33.13
3	OX45NO7WF0D8	3.60	94.49	105.97

4.4.6 Second segment quality and keeping quality variation

Similarly to the previous section, Figures 4.3 and 4.4 show all variations in quality and keeping quality respectively.

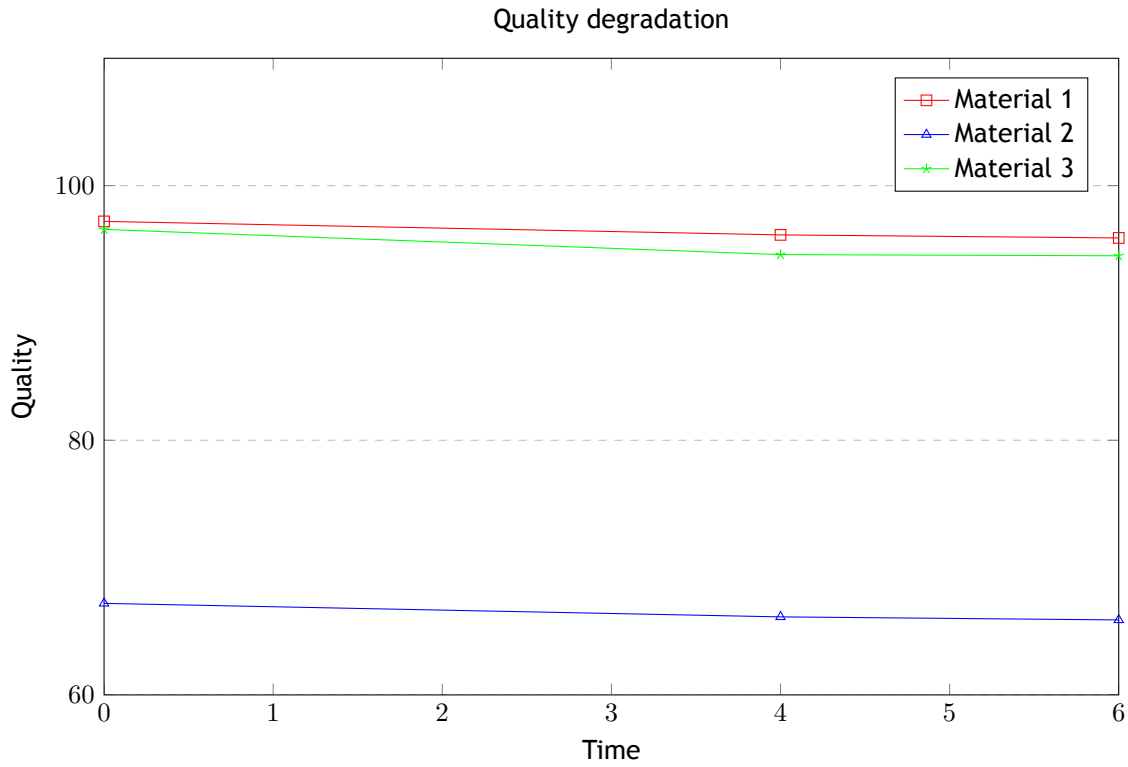


Figure 4.3: Quality degradation throughout the second segment.

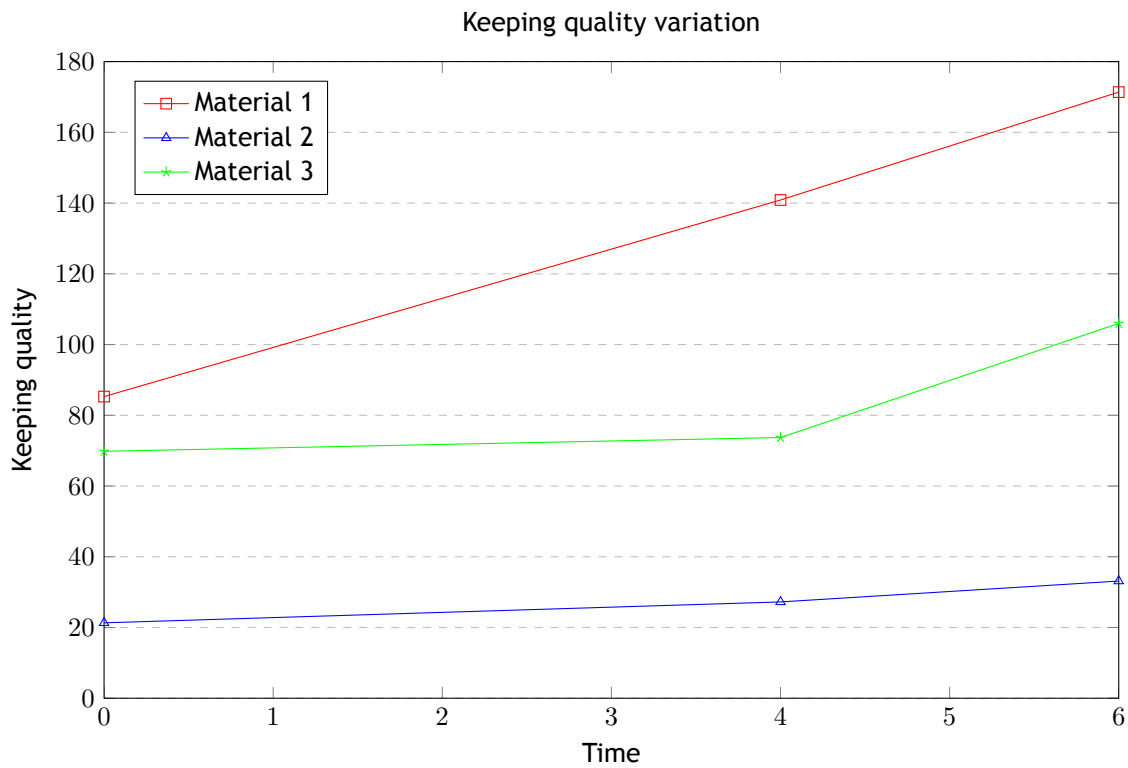


Figure 4.4: Keeping quality variation throughout the second segment.

4.5 Third segment results

Again, it was considered a slight difference between the expected transport temperature and the real transport temperature. For illustrative purposes this difference is 1°C below expected. In this segment the following sequence of events was simulated:

- just as the previous segment, materials were immediately allocated for processing,
- materials spent 2 hours in processing,
- materials spent 48 hours in transit.

4.5.1 Quality after transport module results

Due to the aforementioned temperature variation there were changes in quality and keeping quality. As such the Quality After Transport module presents the changes as shown in Tables 4.11 and 4.12.

Table 4.11: Quality After Transport module entry data.

Material	External Product ID	Last update quality	Last update keeping quality	Travel QMP
1	80S5LF80S5LF	95.89	171.38	-0.88
2	FHVVMNOLY66G	65.89	33.13	-0.88
3	0X45NO7WF0D8	94.49	105.97	2.60

Table 4.12: Quality After Transport module results.

Material	Quality	Quality delta	Keeping quality	Keeping quality delta
1	96.24	-0.35	227.93	-56.55
2	66.24	-0.35	39.25	-6.12
3	94.81	-0.23	119.62	-45.91

4.5.2 Inventory management module results

Again, to match the time intervals shown in Table 4.1, it is considered that the materials do not rest in inventory and move to processing immediately after verification.

4.5.3 Processing stage management module results

Table 4.13 shows all relevant data from this module after materials were processed for 2 hours.

Table 4.13: Third segment processing stage management module output.

Material	Exit ID	Exit QMP	Exit quality	Exit keeping quality
1	Q3YGYGYP	2.54	95.66	125.83
2	6OKVOX	2.54	65.66	21.67
3	EOBBCV	4.69	93.98	83.88

4.5.4 Quality validation module results

Again, it is assumed that there was no lack of care or illegalities committed while handling the materials. As such this module would only present the same results as show in Table 4.13. Here too, unique codes were attributed to the materials, CVY3PL38M03I for Material 1, 5F0IOSW6TGVC for Material 2 and 0ILHRYAS620D for Material 3.

4.5.5 Order management module results

Table 4.14: Third segment order management module prediction.

Material	External Product ID	Predicted travel QMP	Quality prediction	Keeping quality prediction
1	CVY3PL38M03I	1.68	83.79	145.54
2	5F0IOSW6TGVC	1.68	0	0
3	0ILHRYAS620D	6.83	65.11	58.60

Here it can be seen that Material 2 has a quality and keeping quality of zero. This happened because the value of quality was bellow the value of the quality limit. The value of the quality limit is 60 and the predicted quality value was 53.79, while the predicted keeping quality was 25.06 hours. Throughout this simulation the value of the quality limit was not lowered throughout the supply chain. In this scenario such is unreasonable because it was considered that processing did not increase the quality value of the materials used.

4.5.6 Third segment quality and keeping quality variation

Figure 4.5 shows the quality decay that occurred in the third segment and Figure 4.6 shows the variations in keeping quality due to quality decay and temperature.

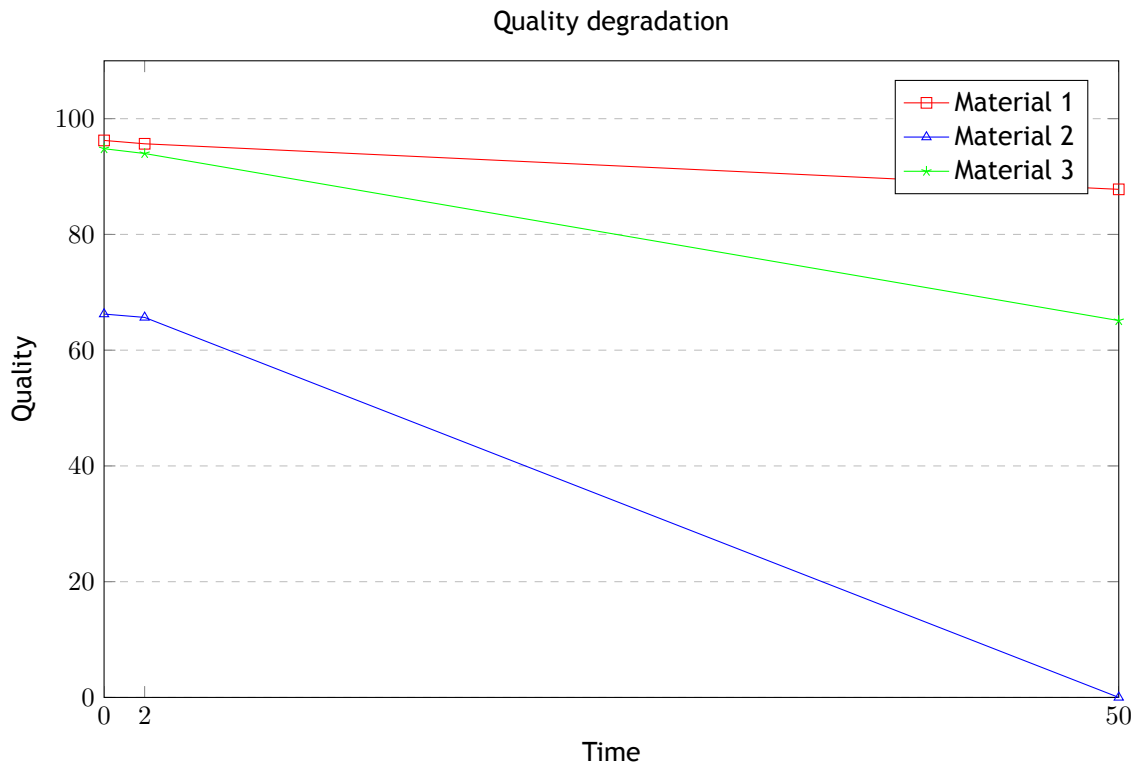


Figure 4.5: Quality degradation throughout the third segment.

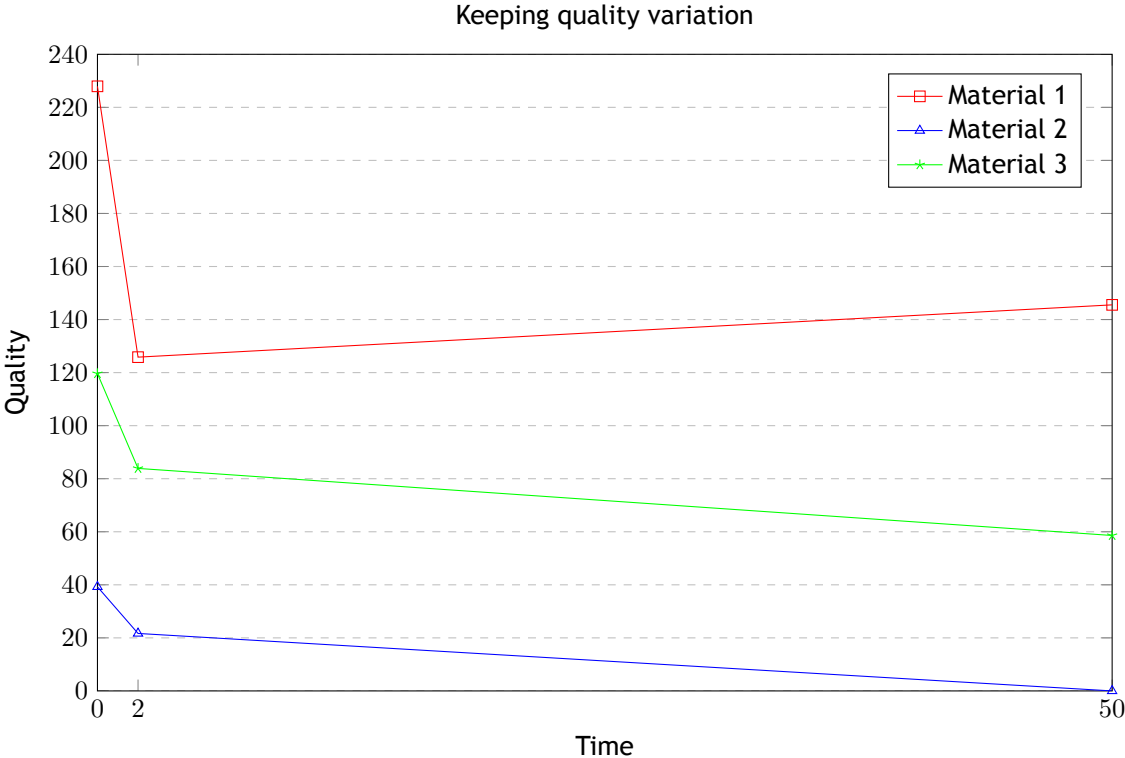


Figure 4.6: Keeping quality variation throughout the third segment.

4.6 Supply chain quality degradation

This section concludes this chapter by presenting cumulative data from the previous sections in order to better demonstrate the results obtained.

4.6.1 Quality

Quality decayed as expected, linearly throughout all measurements with its slope varying according to the temperature that the materials were subjected to. Due to the manner the quality and keeping quality algorithm was applied, the slope of the quality curve only changes when the materials change segment due to the different values of initial quality and temperatures. Figure 4.7 shows the decay of quality throughout the entire simulated food supply chain.

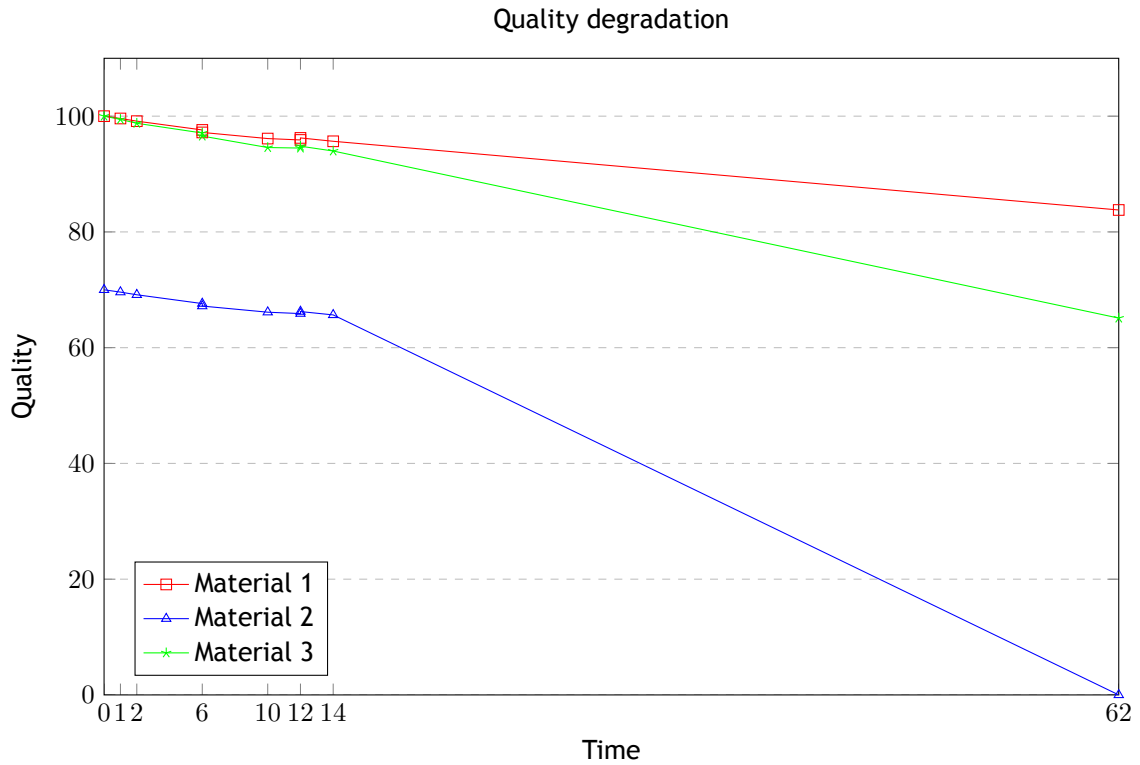


Figure 4.7: Quality degradation throughout the supply chain.

4.6.2 Keeping quality

Keeping quality too, varied linearly throughout all measurements. However, but still as expected, it is extremely sensitive to small variations in temperature. This sensitiveness is easily shown by the keeping quality values obtained throughout this simulation. A good example of this sensitiveness is in the transition between segments, in particular in between the second and third segment for Material 1. Here a 1°C below expectation meant a keeping quality delta of nearly 60 hours above expected. Although this changes are more noticeable in Material 1 due to its starting quality value and lower temperatures, all other materials are too affected by the conditions imposed for the transitions between segments.

Still, throughout this simulation, the only objective was to observe the evaluation of quality and keeping quality and the efficiency of the communication model. This means that certain circumstances were not taken into account, such as the possibility of frost damage to Materials 1 and 2 during the transport from the second segment company to the third segment enterprise when the temperature value was below freezing.

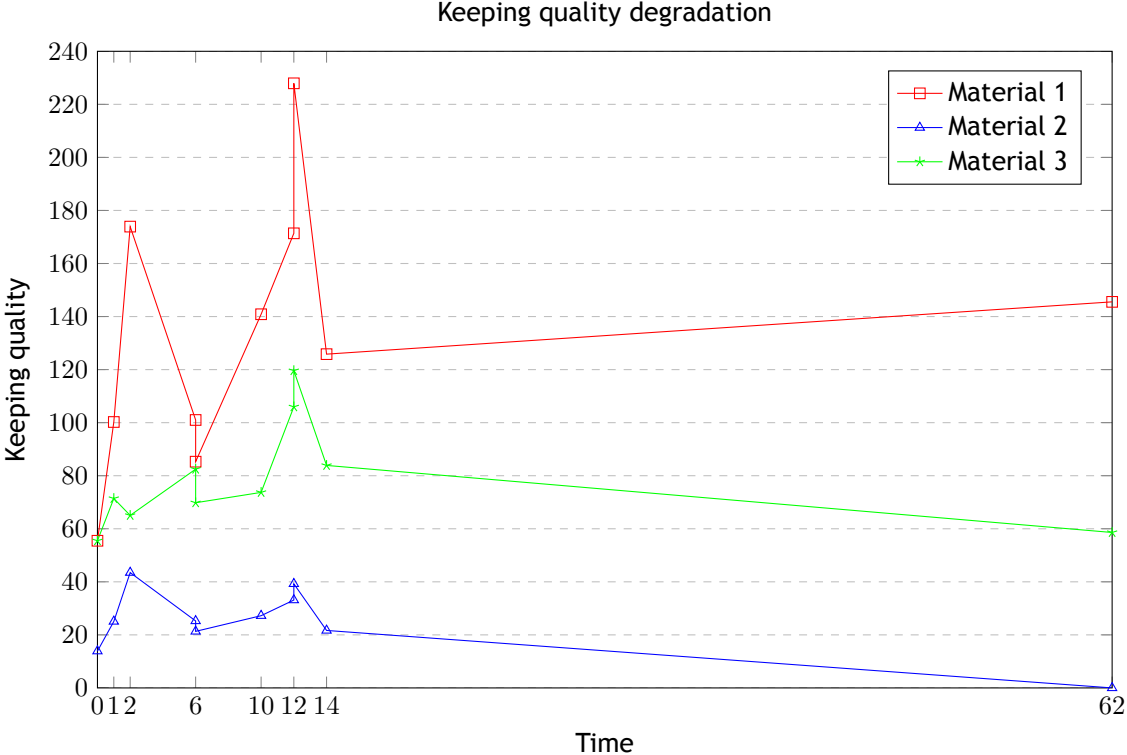


Figure 4.8: Keeping quality variation throughout the supply chain.

Chapter 5

Limitations and performance analysis

Although this template traceability system was developed with comprehensiveness, restrictiveness and low cost in mind as mean to emulate conditions subjected to MSE's and SME's, it is still a prototype and could use some further development in order to become a completely finished product more valuable to MSE's or SME's than the prototype presented.

With the plethora of information available about traceability systems, some pertains to the evaluation of their performance. Although some adaptations were made in order to fit the evaluation methods presented in scientific literature to this prototype in particular, it is still possible to make a general assessment of the capabilities of this prototype.

It is possible to include other elements in this traceability system due to its modular structure. However, it is considered that not all elements should be included in a template like this. That does not mean that those elements are of secondary importance to a MSE or SME, it simply means that their inclusion limits the application range of this prototype by reducing its abstraction as some elements could be useful to an enterprise but not to another. As to demonstrate the potential additions to this traceability system, some potentially useful methods will be described.

5.1 Limitations

This prototype has three main flaws. First and foremost is security. As it is out of scope of this study, the security of information collection, storage and transmission was not considered. If applied in a real scenario, a fair number of precautions would have to be made in order to ensure the safety of privileged data. The second greatest flaw is the manual input of information which greatly limits the input of material flows. The third major flaw is the manner the quality and keeping quality was applied. As temperature varies continuously, assuming a constant temperature through a long period of time induces an error in quality determination. To correctly apply the algorithm, Equation 5.1 should be used instead:

$$\frac{dQ}{dt} = -k \quad (5.1)$$

Where dQ/dt is the variation of quality over time and k the decay rate. This, however, implies constant monitoring which was deemed unfeasible to MSE's and SME's due to the added investment in technology capable to handle continuous monitoring and large data volumes.

One minor flaw is the connection of the regulator layer with the unique code layer as if the first fails, there is no access to the second and products will not be able to receive unique codes. However, such can easily be countered by the existence of several regulator servers.

5.2 Performance analysis

From Mgonja et al. [100] several criteria can be used to assess the performance of the prototype presented in this dissertation.

Table 1 refers to contextual factors. As the prototype is a general template the criteria in this table is not applicable to the system presented in this dissertation.

Table 2 from the aforementioned study contains indicators that allow to assess the design of the system. However not all of them are applicable to this system.

The first indicator is types of TRU identifiers, mode of data registration and location of data storage. Although all data is to be managed electronically, companies are always given a choice on how and how much information to manage as long as it complies with all legal requirements. This means that all answers presented by the authors are possible including the lowest ranking, paper-based systems if the amount and quality of information is low enough to make a fair equivalent. This question is more appropriate to a specific application of a traceability system. The second indicator is appropriateness of the location of information collection point. As the prototype implies a segmentation of the process based on the HACCP system, the most appropriate classification is the highest, T&T information is collected at all appropriate CIP and it is based on HACCP system.

The third indicator is determination of the TRU. All classifications are applicable to fish products only, making this indicator inadequate to assess the design of the prototype.

The fourth indicator is mode of information communication. Due to the necessity external validation and identification for a product to be sold, the highest-ranking classification is the most adequate, System design allows communication via printed material and via electronically e.g., EDI.

The fifth indicator is degree of data standardization. Again, due to external identification, the best possible result, use of international standards such as EAN.UCC standards are the most adequate as is it even possible to use it on an internal level.

Table 3 refers to the operation of the traceability system by humans. This implies an internal evaluation which cannot be applied in this dissertation.

Table 4 from the aforementioned study contains performance indicators relative to performance and food safety.

The final indicator from Table 2 is level of using HACCP system during T&T system design. As the entire system was developed around the usage of HACCP to easily and correctly segment any given process, the highest ranking indicator is the one that suits best, HACCP system is entirely used in all stages of T&T system design and during execution.

The first indicator is how long does it take to trace product information within the company? Although all information relative to each module is kept within the module, the Order Management Module aggregates all information in one place making possible to verify product information with ease. This implies the best possible answer to this question, within four hours.

The second indicator is what is the level of reliability of procedures, tools and information used in the company? Since the presented system is a prototype template for a traceability system meant to be derived by each company to better suit individual needs but needs to follow national and international directives, the most adequate performance metric is the intermediate, use of both local and international approved tools, procedures and information.

The third indicator is what is the degree of accuracy/precision of product batches? All data from operations over any given batches is always recorded by the Processing Stage Management module. As such, the most adequate indicator is the highest ranking one, the actual batch size

is known and is constant at all the times.

Shankar et al.[101] modeled Critical Success Factors (CSFs) for Food Logistics Systems (FLS) by questioning persons capable of evaluating and classifying the CSFs. From the initial sixteen proposed CSFs, twelve were the most relevant and the remainder was excluded. Although the relationships between CSFs is also studied it has no use for the prototype presented in this dissertation as it aimed to template traceability systems. Instead, for evaluation, each of the twelve most relevant CSFs will be individually discussed.

The first CSF is effective transportation management. The Order Management Module is responsible for this task. As is, it only finds the route with the lowest cost and predicts quality and keeping quality at arrival. Route prediction is the most trivial of the described functions but the prediction of quality and keeping quality is not. By being capable of doing it automatically, this module can increase the effectiveness of transportation management by taking in heavy and morose workloads from company employees.

The second CSF is manufacturer branding. Properly using both the IDC and IFC files can help brands distinguish themselves from one another and more easily captivate their target audience.

The third CSF is safe and quality food. As this entire dissertation is built around the HACCP system and regulation enforcement, this CSF is an inherent characteristic of the system.

The fourth CSF is sustainable agricultural practices. As the developed framework and tools are not specific to agri-food products, this CSF cannot be used to evaluate the success of the prototype.

The fifth CSF is government regulations. Again, as the third CSF, this is an inherent characteristic of this system.

The sixth CSF is increased marketing and trading. Increased marketing happens due to the existence of the IDC and ICF files. This, however, does not guarantee increased trading as is dependent on the efficiency of the marketing made in those two files which is impossible to evaluate outside a specific application.

The seventh CSF is proper coordination and transparency. By enforcing external verification and validation using scientific methods, transparency becomes an inherent characteristic. Coordination is dependent on the specific relationships between stakeholders and cannot be evaluated in the context of this dissertation.

The eighth CSF is control of collusive behavior in food logistics. External verification and validation once again takes the role of this CSF. Verification and validation before transaction can heavily hinder this type of behavior that could promote the dissemination of improper products throughout a food chain.

The ninth CSF is logistics competitiveness. This again implies a more specific application of a traceability system as is heavily dependent on particular use. Still, automatic quality and keeping quality assessment can be extremely useful to accelerate logistic processes within a company.

The tenth CSF is risk management strategies in food logistics. This CSF has implications on both internal and external levels. Internally the use of the HACCP to segment a process and monitor each stage has a big influence in reducing risk. Also, internally, the identification of all materials and operations can help the detection of abnormal circumstances or correctly identify products that have to be recalled. Externally, again due to mandatory verification and validation, risk is reduced as improper products are not able to be commercialized between companies.

The eleventh CSF is use of transportation technology. This is dependent on a specific application and cannot be used to evaluate the prototype described.

Safety and quality based traceability system for food supply chains

The final CSF is consumer satisfaction. As consumers value information, providing a tool that helps them access externally and scientifically validated information has the potential of increasing their satisfaction.

Bendaoud et al. [102] lists several functions that a traceability system has to be able to perform in Table 1 of the study.

Table 5.1 lists all functions required in the first column and if and how the prototype achieves them in the second. There can be seen that all functions that do not require internal evaluation are performed by the prototype presented. The performance analysis made in [102] was executed on a poultry processing company making it unusable in the context of this dissertation.

Table 5.1: Bendaoud et al. [102] Table 1 functions and achievements.

Function	Achievement
To create product lots	Both IMM and PSMM can create product lots
To create lot identifiers	IMM and PSMM always identify all operations over lots
To mark the identifier on the product	This implies the evaluation of a specific application
To use identification carriers	This implies the evaluation of a specific application
To collect product traceability data	OMM, QATM and RIFC and all capable of such
To generate product traceability data	IMM, PSMM and OMM are capable of such
To record traceability data in an external support	The mandatory communication between producers and regulators performs this function
To restore product traceability data	According to the communication module, information cannot be lost
To communicate product traceability data	Mandatory according to the communication model

Chapter 6

Conclusion

6.1 General conclusions

Starting the concluding remarks with the literature analysis, it is possible to verify that there are numerous advantages of the application of traceability systems. Some of those potential advantages are: increase in food safety, economic gain, minimization of distribution costs, optimization of production, increase in environmental sustainability, increase in consumer trust, increase in consumer satisfaction, improvement in management in food crisis scenarios, improvement in the contact between stakeholders leading to the creation of more adequate products, emergence of competitive advantages, obtaining more adequate products through quality and price scaling, increase in product value through information access, reduction of operational costs, customization to satisfy each company's requirements, turn quality tangible through quantification and verification, increase in the ease of implementation of other tools that help companies, increased ease in fulfilling regulations and certifications, and inclusion and integrated analysis of the five driving forces.

It is also possible to verify the existence of fundamental issues that make the implementation of comprehensive traceability systems very difficult.

Although some authors offer an economic analysis relative to information value when a traceability system is implemented, this analysis is very complex and applicable only to the traceability system in question or to a very similar one. This analysis is made after implementing a traceability system. It is considered that it would be more useful if that analysis was made in real-time as goods are being processed. This provides a more direct control over production and immediacy in cost and profit analysis.

Determining optimal granularity is a complex process, however it will be left in charge of the companies as it is easier for them to determine the granularity level according to their capabilities. This allows greater flexibility for the companies. Such would not happen if too many rigid rules were imposed from the beginning.

One of the most prevalent issues is the difficulty in dealing with mixing and contamination. This is particularly prevalent when there is not an obvious separation between raw material batches as is the case of the production of liquid food and grain. In liquids is possible to use turbulent diffusion or k-nearest neighbors algorithms. To solid foods there is the possibility to include markers during the process and also k-nearest neighbors algorithms.

There is also some difficulty in accompanying transformations and movements along a supply chain without significantly restructuring the whole process. To alleviate this condition there is the possibility to apply this kind of systems using gateways. This technical solution allows the addition of monitoring technologies without or with minimal interference in existing processes. It becomes possible to apply traceability systems in the short term without compromising the scale factor in the medium to long terms, as well as enabling the transition to a system that allows continuous monitoring. By using this method, a virtual segmentation of the process is achieved and the data can be monitored at each checkpoint. It also allows the introduction of more functionalities beyond simple product history recording.

Safety and quality based traceability system for food supply chains

It is hard to connect the exit of a product from a company to the entry of the same product in another company. This issue is reduced by implementing a unified communication model, as there is the possibility of having a product whose exit code is the same as the entry code in another company.

Information loss is still inevitable but can be greatly reduced by transferring that workload to information technologies that operate autonomously or semi-autonomously. Although, this information loss can lead to security flaws and diminished sustainability, but even so these are reduced due to a better information retention.

The lack of standard practices is one of the larger restrictions to the broad application of traceability systems. This issue can only be solved by the implementation of a standard communication model that dictates how information sharing should be made. It must be highlighted that must be an external entity that imposes and regulates that model and non-compliance sanctions.

By applying broad traceability systems, large data volumes arise. To minimize this issue, the communication model must be able to minimize the information that is mandatory to collect and report without losing functionality. Thus, there is the necessity to maximize the utility of transmitted information. It should also be possible to divide the quantity of information between stakeholders.

Traceability is often seen as an obligation and as a bureaucracy that involves significant investment and diminished returns, and not always tangible, gains. An open-source system that includes inventory, distribution and communication management by default can reduce this drawback. Using open-source system leads to a drastic reduction in investment and allows customization and self-development. However, it must be pointed that this kind of systems may suffer of unreliable and weak data safety and security systems that may jeopardy the data confidentiality.

By providing more information about their products, companies inadvertently open the door to copies, falsification and counterfeiting. To prevent these unwanted situations, it is necessary that the companies themselves are attentive to the information they share, that all identification codes are unique and that there is an external entity to enforce compliance and punish non-compliance.

One significant difficulty is chain wide implementation. A free system and adequate communication model can help minimizing this problem.

Many authors state that there is an existing structure that can accommodate traceability systems in IoT perspective. This structure does not exist and has yet to be built. Its application in the short term requires a very efficient communication model that does not compromise the long-term development.

Often, traceability systems are observed and approached from the consumer's point of view. A large amount of information must be transmitted easily and the best quality possible must be obtained. It is considered that such is not necessary as different consumers have different purchasing power and necessities, as well as not all information obtained is pertinent outside the company. This condition allows companies to better regulate the information they share and diversify product lineups, that benefits both companies and consumers.

Implementing isolated traceability systems can cause others to value their products without investing in traceability systems. This inconvenience happens because a company that implemented a traceability system reveals value increasing information and posteriorly sells the goods to a company that has not implemented a traceability system. Due to the value increasing information disclosed by the first company, the second can use that same information and value

their own products. A free and comprehensive traceability system applicable to all entities in a supply chain can identify from who is the increased value, thus giving merit and recognition to them, but cannot fully avert others from increasing the value of their own products as all that is necessary is declaring the use of materials from a more reputable source.

Loss of control of processing by the companies will occur if the communication model or system contains many excessively rigid rules. A customizable system and an effective communication model can reduce the interference in company authority leaving them subjected to the self-regulatory nature of their markets.

Following the principles here described in the review, an open-source traceability framework focused on safety and quality was developed. This framework uses HACCP flowcharts to define gateways for quality evaluation and encompasses external verification and product history maintenance. This strategy can reduce the initial cost, limiting it to the acquisition of technology. MSEs and SMEs can benefit from such a system by the tighter quality control involved and possible future integration of any tools supposed valuable. The system was developed mostly with MSE's and SME's in mind and thus it will not require mature IT capabilities.

There are several advantages to this framework. As information monitoring is required, implementation costs will be offset by the validation of the quality and by the consequent ability to better compose a product line according to corporate capabilities and objectives. Information monitoring also translates into process optimization as the parameters that affect any given stage are monitored and so flaws and defects can be effectively counteracted due to the disclosure of their causes. Being able to transmit externally validated information also allows to better price products according to target audience which will translate to more consumer satisfaction and trust as well as to reduce losses from waste. Traceability frameworks must be able to increase profit. If such does not happen, there is not enough incentive to adopt one and the corporations will combat the implementation of a framework as in those circumstances, it will only make operations more cumbersome. Therefore, cooperation between all stakeholders is mandatory and regulators must become an active agent. This condition concerns security and crisis management as well, and help regulatory compliant corporations to profit and punish non compliance.

Although this framework is capable to solve several issues associated with traceability, it is not, however, completely free of flaws, the most obvious being the interchangeable nature of companies. This means that it is difficult to assign them to a specific segment as there may be the need to purchase products from another company. Because of this difficulty, segmentation is made by what constitutes the most significant mean of acquisition of raw material. Another major issue is the amount of power possessed by companies in the first segment as the initial quality and keeping quality of a raw material as those values are not those dictated by the QATM. Such is an open door to fraudulent activity as all that is necessary in the input of false data. The solution of this issue depends on the parameters used by the QATM to evaluate quality and these in turn depend on the parameters used by the second segment enterprise.

Simulating the tools created proved that they can evaluate quality and keeping quality and effectively transmit all necessary information.

According to the performance analysis, the system ranked highly in the proposed objectives. Concluding these final remarks with a few notes on the software created, it can be inferred that opting for an open-source semi-abstract software is a good choice for traceability systems. It dramatically reduces implementation costs while still being able to respect and comply with an external communication model without overly interfering with internal and external development while keeping each of those naturally cohesive and easy to derive from a template

according to whichever necessity and capability that may arise.

6.2 Suggestions for future work

As mentioned before, there are multiple additions that can be made to this template in order to make it better suited the needs of a company. Below are some examples useful additions that were not included in order to not restrict application range but could be useful to adapt to a more specific set of circumstances.

Alumur & Kara [103] surveyed and classified network hub location models. This study can serve as a compendium of methods for planning hub locations. This planning can be accompanied by the implementation of a compatible and perhaps more cohesive traceability system to the network as a whole.

Zhou et al. [104] developed a model that varies price based on consumption. It is a very interesting perspective and maybe applicable in the future, however, it is a self-learning system whose necessities are considered to great in computational resources to comply with the objective of this study. So, it will not be applied as it is, consider that it would make a system based on this framework too costly to implement in the short term.

Blackburn & Scudder ([105] developed a method to help fresh produce supply chain decisions based on the loss of product value over time. This method may seem directly beneficial to the prototype system presented. It is more applicable to a more sizeable company than the typical MSE or SME because the application of this model is easier when the same company deals with production and distribution.

Ahumada & Villalobos [106] reviewed models applicable to agri-food supply chains planning. As this is an aggregation of models, can be used as a compendium for applicable models if necessary when making changes or creating an enterprise.

Akkerman et al. [107] developed a method that aims to help determine what materials to store or to process into intermediate products and then postpone in a flour mill in order to minimize costs while taking into account quality attributes. When quality is already being monitored by a traceability system, this method or similar can help to further reduce costs and optimize productions.

Rong et al. [108] developed a mixed-integer linear programming model for production and distribution planning. This model incorporates quality degradation in the same manner as Tijssens & Polderdijk (1996) and adds the other mentioned factors.

Guericke et al. [109] developed a model for postponement strategies in distribution networks. By having product flows well determined, it becomes easier to implement a traceability system adequate to the situation, thus limiting costs with robustness that will not be necessary for the system to properly function.

Lieckens & Vandaele [110] developed a model for logistic networks planning considering decay and uncertainty. This kind of model can prove to be quite useful when designing a network from the ground up or when expanding a company.

Baghalian et al. [111] developed a method to design supply chain networks that aims to prevent against disruption while dealing with uncertainty from stakeholders, whether suppliers or buyers. For reasons already described, this type of models can be useful as they to help planning traceability systems, although indirectly.

Lin et al. [112] developed a model for the transport of perishable commodities. Contains a

decision-making system that is fed by real-time data and transport decisions are can be changed in order to reduce waste.

Tsang et al. [113] developed an IoT based method to ensure product quality in temperature sensitive distribution. Models like this can replace the Prim's algorithm used for vehicle routing but are more demanding to implement.

Although very useful, none of these additions are directly necessary to correctly implement a traceability system. Still, there are circumstances in which they may prove very useful to a company and help delineate a better suited traceability system by defining more specific priorities and necessities that would be quite useful if accommodated by the accompanying system. All planning models help in the definition of priorities and necessities and that in turn helps defining the traceability system that virtualizes the entire company. This planning models can be useful when creating, expanding or redefining an enterprise. More specific functions of a traceability system can be replaced by more robust models as it becomes practical for a company to implement.

Thus, there is ample space to grow and specialize the prototype presented as necessity and availability dictate. This section is a mere illustration of the enormous amount of material readily available to be adapted to a traceability system.

Bibliography

- [1] K. M. Karlsen, B. Dreyer, P. Olsen, and E. O. Elvevoll, "Literature review: Does a common theoretical framework to implement food traceability exist?" *Food Control*, vol. 32, no. 2, pp. 409-417, 2013. 1
- [2] T. Moe, "Perspectives on traceability in food manufacture," *Trends in Food Science & Technology*, vol. 9, no. 5, pp. 211-214, 1998. 1
- [3] I. O. for Standardization, "Quality management and quality assurance – Vocabulary," International Organization for Standardization, Standard, 1994. 1
- [4] —, "Quality management systems – Fundamentals and vocabulary," International Organization for Standardization, Standard, 2015. 1
- [5] —, "Traceability in the feed and food chain – General principles and basic requirements for system design and implementation," International Organization for Standardization, Standard, 2007. 1
- [6] J. F. C. A. Commission., *Codex alimentarius / Joint FAO/WHO Food Standards Programme, Codex Alimentarius Commission*, 21st ed. Food and Agriculture Organization of the United Nations: World Health Organization, 2013. 1
- [7] P. Olsen and M. Borit, "How to define traceability," *Trends in Food Science & Technology*, vol. 29, no. 2, pp. 142-150, 2013. 1
- [8] S. Charlebois, B. Sterling, S. Haratifar, and S. K. Naing, "Comparison of Global Food Traceability Regulations and Requirements," *Comprehensive Reviews in Food Science and Food Safety*, vol. 13, no. 5, pp. 1104-1123, 2014. 1
- [9] J. R. Lupien, "Food quality and safety: Traceability and labeling," *Critical Reviews in Food Science and Nutrition*, vol. 45, no. 2, pp. 119-123, 2005. 1
- [10] K. Souali, O. Rahmaoui, and M. Ouzzif, "An Overview of Traceability: Towards a general multi-domain model," *Advances in Science, Technology and Engineering Systems Journal*, vol. 2, no. 3, pp. 356-361, 2017. 1
- [11] G. S. Walker, "Food authentication and traceability: An Asian and Australian perspective," *Food Control*, vol. 72, no. Part B, pp. 168-172, 2017. 1
- [12] L. U. Opara, "Traceability in agriculture and food supply chain: A review of basic concepts, technological implications, and future prospects," *Food, Agriculture & Environment*, vol. 1, no. 1, pp. 101-106, 2003. 2
- [13] J. Curto and P. Gaspar, "Traceability in food supply chains: Review and sme focused analysis - part 1," submitted. 5
- [14] A. J. Beulens, D.-F. Broens, P. Folstar, and G. J. Hofstede, "Food safety and transparency in food chains and networks. Relationships and challenges," *Food Control*, vol. 16, no. 6, pp. 481-486, 2005. 5, 9

Safety and quality based traceability system for food supply chains

- [15] A. Bollen, C. Riden, and N. Cox, "Agricultural supply system traceability, Part I: Role of packing procedures and effects of fruit mixing," *Biosystems Engineering*, vol. 98, no. 4, pp. 391-400, 2007. 5, 9
- [16] T. Skoglund and P. Dejmek, "Fuzzy traceability: A process simulation derived extension of the traceability concept in continuous food processing," *Food and Bioproducts Processing*, vol. 85, no. 4, pp. 354-359, 2007. 6, 9
- [17] S. Frosch, M. Randrup, and M. Thorup Frederiksen, "Opportunities for the herring industry to optimize operations through information recording, effective traceability systems, and use of advanced data analysis," *Journal of Aquatic Food Product Technology*, vol. 17, no. 4, pp. 387-403, 2008. 6, 9
- [18] L. Wang, S. K. Kwok, and W. H. Ip, "A radio frequency identification and sensor-based system for the transportation of food," *Journal of Food Engineering*, vol. 101, no. 1, pp. 120-129, 2010. 6, 9
- [19] M. Thakur, B. J. Martens, and C. R. Hurburgh, "Data modeling to facilitate internal traceability at a grain elevator," *Computers and Electronics in Agriculture*, vol. 75, no. 2, pp. 327-336, 2011. 7, 9
- [20] K. M. Karlsen, K. A. Donnelly, and P. Olsen, "Granularity and its importance for traceability in a farmed salmon supply chain," *Journal of Food Engineering*, vol. 102, no. 1, pp. 1-8, 2011. 7, 9
- [21] M. Borit and P. Olsen, "Evaluation framework for regulatory requirements related to data recording and traceability designed to prevent illegal, unreported and unregulated fishing," *Marine Policy*, vol. 36, no. 1, pp. 96-102, 2012. 7, 9
- [22] F. Huang, S. Zhang, and H. Zhao, "Design and application of quality traceability system based on RFID technology for red jujubes," *IFIP Advances in Information and Communication Technology*, vol. AICT-368, no. Part I, pp. 371-380, 2012. 7, 9
- [23] T. Pizzuti, G. Mirabelli, F. Gómez-González, and M. A. Sanz-Bobi, "Modeling of an Agro-Food Traceability System : The Case of the Frozen Vegetables," in *Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management*, 2012, pp. 1065-1074. 7, 10
- [24] V. Lavelli, "High-warranty traceability system in the poultry meat supply chain: A medium-sized enterprise case study," *Food Control*, vol. 33, no. 1, pp. 148-156, 2013. 7, 10
- [25] J. Hu, X. Zhang, L. M. Moga, and M. Neculita, "Modeling and implementation of the vegetable supply chain traceability system," *Food Control*, vol. 30, no. 1, pp. 341-353, 2013. 8, 10
- [26] M. Trebar, M. Lotrič, I. Fonda, A. Pleteršek, and K. Kovačič, "RFID Data Loggers in Fish Supply Chain Traceability," *International Journal of Antennas and Propagation*, vol. 2013, no. 3-4, pp. 1-9, 2013. 8, 10
- [27] A. Parreño-Marchante, A. Alvarez-Melcon, M. Trebar, and P. Filippin, "Advanced traceability system in aquaculture supply chain," *Journal of Food Engineering*, vol. 122, pp. 99-109, 2014. 8, 10

- [28] F. Liu, Y. Wang, Y. Jia, S. Hu, L. Tu, and C. Tang, "The egg traceability system based on the video capture and wireless networking technology," *International Journal of Sensor Networks*, vol. 17, no. 4, pp. 211-216, 2015. 8, 10
- [29] G. Alfian, J. Rhee, H. Ahn, J. Lee, U. Farooq, M. F. Ijaz, and M. A. Syaekhoni, "Integration of RFID, wireless sensor networks, and data mining in an e-pedigree food traceability system," *Journal of Food Engineering*, vol. 212, pp. 65-75, 2017. 8, 10
- [30] X. Wang, D. Fu, G. Fruk, E. Chen, and X. Zhang, "Improving quality control and transparency in honey peach export chain by a multi-sensors-managed traceability system," *Food Control*, vol. 88, pp. 169-180, 2018. 8, 10
- [31] M. Li, J.-P. Qian, X.-T. Yang, C.-H. Sun, and Z.-T. Ji, "A PDA-based record-keeping and decision-support system for traceability in cucumber production," *Computers and Electronics in Agriculture*, vol. 70, no. 1, pp. 69-77, 2010. 9
- [32] M. H. Jansen-Vullers, C. A. Van Dorp, and A. J. M. Beulens, "Managing traceability information in manufacture," *International Journal of Information Management*, vol. 23, no. 5, pp. 395-413, 2003. 10, 16
- [33] M. Bogataj, L. Bogataj, and R. Vodopivec, "Stability of perishable goods in cold logistic chains," *International Journal of Production Economics*, vol. 93-94, pp. 345-356, 2005. 11, 16
- [34] A. Regattieri, M. Gamberi, and R. Manzini, "Traceability of food products: General framework and experimental evidence," *Journal of Food Engineering*, vol. 81, no. 2, pp. 347-356, 2007. 11, 16
- [35] G. H. Gessner, L. Volonino, and L. A. Fish, "One-up, one-back ERM in the food supply chain," *Information Systems Management*, vol. 24, no. 3, pp. 213-222, 2007. 11, 16
- [36] R. Jedermann and W. Lang, "Semi-passive RFID and beyond: steps towards automated quality tracing in the food chain," *International Journal of Radio Frequency Identification Technology and Applications*, vol. 1, no. 3, pp. 247-259, 2007. 11, 16
- [37] Y. C. Hsu, A. P. Chen, and C. H. Wang, "A RFID-enabled traceability system for the supply chain of live fish," in *2008 IEEE International Conference on Automation and Logistics*, 2008, pp. 81-86. 11, 16
- [38] K. A. M. Donnelly, K. M. Karlsen, and P. Olsen, "The importance of transformations for traceability - A case study of lamb and lamb products," *Meat Science*, vol. 83, no. 1, pp. 68-73, 2009. 11, 16
- [39] P. Chrysochou, G. Chrysochoidis, and O. Kehagia, "Traceability information carriers. The technology backgrounds and consumers' perceptions of the technological solutions," *Appetite*, vol. 53, no. 3, pp. 322-331, 2009. 11, 16
- [40] K. M. Karlsen, C. F. Sørensen, F. Forås, and P. Olsen, "Critical criteria when implementing electronic chain traceability in a fish supply chain," *Food Control*, vol. 22, no. 8, pp. 1339-1347, 2011. 11, 16
- [41] J. V. der Vorst, O. Van Kooten, and P. Luning, "Towards a diagnostic instrument to identify improvement opportunities for quality controlled logistics in agrifood supply chain

Safety and quality based traceability system for food supply chains

- networks,” *International Journal on Food System Dynamics*, vol. 2, no. 1, pp. 94 - 105, 2011. 11, 16
- [42] T. Bosona and G. Gebresenbet, “Food traceability as an integral part of logistics management in food and agricultural supply chain,” *Food Control*, vol. 33, no. 1, pp. 32-48, 2013. 12, 16
- [43] J. Storoy, M. Thakur, and P. Olsen, “The TraceFood Framework - Principles and guidelines for implementing traceability in food value chains,” *Journal of Food Engineering*, vol. 115, no. 1, pp. 41-48, 2013. 12, 16
- [44] M. M. Aung and Y. S. Chang, “Traceability in a food supply chain: Safety and quality perspectives,” *Food Control*, vol. 39, pp. 172-184, 2014. 12, 16
- [45] D. Asioli, A. Boecker, and M. Canavari, “On the linkages between traceability levels and expected and actual traceability costs and benefits in the Italian fishery supply chain,” *Food Control*, vol. 46, pp. 10-17, 2014. 12, 16
- [46] F. Dabbene, P. Gay, and C. Tortia, “Traceability issues in food supply chain management: A review,” *Biosystems Engineering*, vol. 120, pp. 65-80, 2014. 13, 16
- [47] G. Aiello, M. Enea, and C. Muriana, “The expected value of the traceability information,” *European Journal of Operational Research*, vol. 244, no. 1, pp. 176-186, 2015. 13, 16
- [48] M. Germani, M. Mandolini, M. Marconi, E. Marilungo, and A. Papetti, “A system to increase the sustainability and traceability of supply chains,” *Procedia CIRP*, vol. 29, pp. 227-232, 2015. 13, 16
- [49] M. Thakur and E. Forås, “EPCIS based online temperature monitoring and traceability in a cold meat chain,” *Computers and Electronics in Agriculture*, vol. 117, pp. 22-30, 2015. 13, 16
- [50] E. Forås, M. Thakur, K. Solem, and R. Svarva, “State of traceability in the Norwegian food sectors,” *Food Control*, vol. 57, pp. 65-69, 2015. 13, 17
- [51] H.-I. Hsiao and K.-L. Huang, “Time-temperature transparency in the cold chain,” *Food Control*, vol. 64, pp. 181-188, 2016. 13, 17
- [52] V. M. Alonso-Rorís, L. Álvarez-Sabucedo, J. M. Santos-Gago, and M. Ramos-Merino, “Towards a cost-effective and reusable traceability system. A semantic approach,” *Computers in Industry*, vol. 83, pp. 1-11, 2016. 14, 17
- [53] K. Dandage, R. Badia-Melis, and L. Ruiz-García, “Indian perspective in food traceability: A review,” *Food Control*, vol. 71, pp. 217-227, 2017. 14, 17
- [54] N. Raak, C. Symmank, S. Zahn, J. Aschemann-Witzel, and H. Rohm, “Processing- and product-related causes for food waste and implications for the food supply chain,” *Waste Management*, vol. 61, pp. 461-472, 2017. 14, 17
- [55] P. Olsen and M. Borit, “The components of a food traceability system,” *Trends in Food Science & Technology*, vol. 77, pp. 143-149, 2018. 14, 17
- [56] D. E. Matzembacher, I. do Carmo Stangherlin, L. A. Slongo, and R. Cataldi, “An integration of traceability elements and their impact in consumer’s trust,” *Food Control*, vol. 92, pp. 420-429, 2018. 14, 17

- [57] N. Ndraha, H.-I. Hsiao, J. Vljajic, M.-F. Yang, and H.-T. V. Lin, "Time-temperature abuse in the food cold chain: Review of issues, challenges, and recommendations," *Food Control*, vol. 89, pp. 12-21, 2018. 14, 17
- [58] S. Stranieri, A. Cavaliere, and A. Banterle, "The determinants of voluntary traceability standards. The case of the wine sector," *Wine Economics and Policy*, vol. 7, no. 1, pp. 45-53, 2018. 14, 17
- [59] K. Óskarsdóttir and G. V. Oddsson, "Towards a decision support framework for technologies used in cold supply chain traceability," *Journal of Food Engineering*, vol. 240, pp. 153-159, 2019. 15, 17
- [60] M. Sloof, P. Tijskens, and E. C. Wilkinson, "Concepts for modelling the quality of perishable products," *Trends in Food Science and Technology*, vol. 7, no. 5, pp. 165-171, 1996. 17, 22
- [61] A. Bechini, M. Cimino, B. Lazzerini, F. Marcelloni, and A. Tomasi, "A General Framework for Food Traceability," in *2005 Symposium on Applications and the Internet Workshops (SAINT 2005 Workshops)*, 2006, pp. 366-369. 17, 22
- [62] R. Jedermann, C. Behrens, D. Westphal, and W. Lang, "Applying autonomous sensor systems in logistics-Combining sensor networks, RFIDs and software agents," *Sensors and Actuators, A: Physical*, vol. 132, no. 1, pp. 370-375, 2006. 18, 22
- [63] C.-I. Hsu, S.-F. Hung, and H.-C. Li, "Vehicle routing problem with time-windows for perishable food delivery," *Journal of Food Engineering*, vol. 80, no. 2, pp. 465-475, 2007. 18, 22
- [64] T. Kelepouris, K. Pramataris, and G. Doukidis, "RFID-enabled traceability in the food supply chain," *Industrial Management and Data Systems*, vol. 107, no. 2, pp. 183-200, 2007. 18, 22
- [65] H. S. Heese, "Inventory Record Inaccuracy, Double Marginalization, and RFID Adoption," *Production and Operations Management*, vol. 16, no. 5, pp. 542-553, 2007. 18, 22
- [66] L. Xiaofeng, O. Tang, and P. Huang, "Dynamic pricing and ordering decision for the perishable food of the supermarket using RFID technology," *Asia Pacific Journal of Marketing and Logistics*, vol. 20, no. 1, pp. 7-22, 2008. 18, 22
- [67] S. K. Kwok, A. H. Tsang, J. S. Ting, W. B. Lee, and B. C. Cheung, *An intelligent RFID-based electronic anti-counterfeit system (InRECS) for the manufacturing industry*. IFAC, 2008, vol. 41, no. 2. 18, 22
- [68] A. Bechini, M. G. C. A. Cimino, F. Marcelloni, and A. Tomasi, "Patterns and technologies for enabling supply chain traceability through collaborative e-business," *Information and Software Technology*, vol. 50, no. 4, pp. 342-359, 2008. 18, 22
- [69] S. H. Woo, J. Y. Choi, C. Kwak, and C. O. Kim, "An active product state tracking architecture in logistics sensor networks," *Computers in Industry*, vol. 60, no. 3, pp. 149-160, 2009. 18, 22
- [70] J. G. van der Vorst, S.-O. Tromp, and D.-J. van der Zee, "Simulation modelling for food supply chain redesign; Integrated decision making on product quality, sustainability and

Safety and quality based traceability system for food supply chains

- logistics,” *International Journal of Production Research*, vol. 47, no. 23, pp. 6611-6631, 2009. 18, 22
- [71] W. Zhou, G. Kapoor, and S. Piramuthu, “RFID-enabled item-level product information revelation,” *European Journal of Information Systems*, vol. 18, pp. 570-577, 2009. 19, 22
- [72] Z. Hu, Z. Jian, P. Shen, Z. Xiaoshuan, and M. Weisong, “Modeling method of traceability system based on information flow in meat food supply chain,” *WSEAS Transactions on Information Science and Applications*, vol. 6, pp. 1094-1103, 2009. 19, 22
- [73] M. Thakur and C. R. Hurburgh, “Framework for implementing traceability system in the bulk grain supply chain,” *Journal of Food Engineering*, vol. 95, no. 4, pp. 617-626, 2009. 19, 22
- [74] P. Olsen and M. Aschan, “Reference method for analyzing material flow, information flow and information loss in food supply chains,” *Trends in Food Science and Technology*, vol. 21, no. 6, pp. 313-320, 2010. 19, 22
- [75] M. Thakur, C.-F. Sørensen, F. O. Bjørnson, E. Forås, and C. R. Hurburgh, “Managing food traceability information using EPCIS framework,” *Journal of Food Engineering*, vol. 103, no. 4, pp. 417-433, 2011. 19
- [76] K. M. Karlsen, B. Dreyer, P. Olsen, and E. O. Elvevoll, “Granularity and its role in implementation of seafood traceability,” *Journal of Food Engineering*, vol. 112, no. 1-2, pp. 78-85, 2012. 19, 22
- [77] M. Bakker, J. Riezebos, and R. H. Teunter, “Review of inventory systems with deterioration since 2001,” *European Journal of Operational Research*, vol. 221, no. 2, pp. 275-284, 2012. 20, 22
- [78] X. Wang and D. Li, “A dynamic product quality evaluation based pricing model for perishable food supply chains,” *Omega*, vol. 40, no. 6, pp. 906-917, 2012. 20, 22
- [79] S. O. Tromp, H. Rijgersberg, F. Pereira Da Silva, and P. Bartels, “Retail benefits of dynamic expiry dates - Simulating opportunity losses due to product loss, discount policy and out of stock,” *International Journal of Production Economics*, vol. 139, no. 1, pp. 14-21, 2012. 20, 22
- [80] M. Grunow and S. Piramuthu, “RFID in highly perishable food supply chains - Remaining shelf life to supplant expiry date?” *International Journal of Production Economics*, vol. 146, no. 2, pp. 717-727, 2013. 20, 23
- [81] C. N. Verdouw, A. J. Beulens, and J. G. van der Vorst, “Virtualisation of floricultural supply chains: A review from an internet of things perspective,” *Computers and Electronics in Agriculture*, vol. 99, pp. 160-175, 2013. 20, 23
- [82] S. Piramuthu, P. Farahani, and M. Grunow, “RFID-generated traceability for contaminated product recall in perishable food supply networks,” *European Journal of Operational Research*, vol. 225, no. 2, pp. 253-262, 2013. 20, 23
- [83] T. Pizzuti, G. Mirabelli, M. A. Sanz-Bobi, and F. Gómez-González, “Food Track & Trace ontology for helping the food traceability control,” *Journal of Food Engineering*, vol. 120, pp. 17-30, 2014. 20, 23

- [84] J. Pahl and S. Voß, "Integrating deterioration and lifetime constraints in production and supply chain planning: A survey," *European Journal of Operational Research*, vol. 238, no. 3, pp. 654-674, 2014. 20, 23
- [85] M. L. A. T. M. Hertog, I. Uysal, B. M. Verlinden, U. McCarthy, and B. M. Nicolai, "Shelf life modelling for first-expired-first-out warehouse management," *Philosophical transactions of THE ROYAL SOCIETY*, vol. Series A, 2014. 20, 23
- [86] R. Jedermann, M. Nicometo, I. Uysal, and W. Lang, "Reducing food losses by intelligent food logistics Reducing food losses by intelligent food logistics," *Philosophical transactions of THE ROYAL SOCIETY*, vol. Series A,, no. 372(2017), 2014. 20, 23
- [87] R. Badia-Melis, P. Mishra, and L. Ruiz-García, "Food traceability: New trends and recent advances. A review," *Food Control*, vol. 57, pp. 393-401, 2015. 21, 23
- [88] A. E. Saak, "Traceability and reputation in supply chains," *International Journal of Production Economics*, vol. 177, pp. 149-162, 2016. 21, 23
- [89] J. Qian, B. Fan, X. Wu, S. Han, S. Liu, and X. Yang, "Comprehensive and quantifiable granularity: A novel model to measure agro-food traceability," *Food Control*, vol. 74, pp. 98-106, 2017. 21, 23
- [90] G. Gaukler, M. Ketzenberg, and V. Salin, "Establishing dynamic expiration dates for perishables: An application of RFID and sensor technology," *International Journal of Production Economics*, vol. 193, pp. 617-632, 2017. 21, 23
- [91] R. Accorsi, M. Bortolini, G. Baruffaldi, F. Pilati, and E. Ferrari, "Internet-of-things Paradigm in Food Supply Chains Control and Management," *Procedia Manufacturing*, vol. 11, pp. 889-895, 2017. 21, 23
- [92] K. Óskarsdóttir and G. V. Oddsson, "Towards a decision support framework for technologies used in cold supply chain traceability," *Journal of Food Engineering*, vol. 240, pp. 153-159, 2019. 21, 23
- [93] J. Curto and P. Gaspar, "Sme and quality focused traceability architecture to increase sustainability and profit," submitted. 25
- [94] —, "Traceability in food supply chains: Sme focused traceability framework for chain wide quality and safety - part 2," submitted. 25
- [95] G. van Rossum. Python. [Online]. Available: <https://www.python.org/> 38
- [96] D. Castelvecchi, "Black hole pictured for first time — in spectacular detail," *Nature*, vol. 568, 04 2019. 38
- [97] Microsoft. Visual studio code. [Online]. Available: <https://code.visualstudio.com/> 38
- [98] Canonical. Ubuntu. [Online]. Available: <https://ubuntu.com/> 38
- [99] L. Tijssens and J. Polderdijk, "A generic model for keeping quality of vegetable produce during storage and distribution," *Agricultural Systems*, vol. 51, no. 4, pp. 431 - 452, 1996. 63, 64

Safety and quality based traceability system for food supply chains

- [100] J. T. Mgonja, P. Luning, and J. G. Van Der Vorst, "Diagnostic model for assessing traceability system performance in fish processing plants," *Journal of Food Engineering*, vol. 118, no. 2, pp. 188-197, 2013. 76
- [101] R. Shankar, R. Gupta, and D. K. Pathak, "Modeling critical success factors of traceability for food logistics system," *Transportation Research Part E: Logistics and Transportation Review*, vol. 119, pp. 205-222, 2018. 77
- [102] M. Bendaoud, C. Lecomte, and B. Yannou, "A methodological framework to design and assess food traceability systems," *International Food and Agribusiness Management Review*, vol. 15, no. 1, pp. 103-126, 2012. 78
- [103] S. Alumur and B. Y. Kara, "Network hub location problems: The state of the art," *European Journal of Operational Research*, vol. 190, no. 1, pp. 1-21, 2008. 82
- [104] W. Zhou, Y. J. Tu, and S. Piramuthu, "RFID-enabled item-level retail pricing," *Decision Support Systems*, vol. 48, no. 1, pp. 169-179, 2009. 82
- [105] J. Blackburn and G. Scudder, "Supply chain strategies for perishable products: The case of fresh produce," *Production and Operations Management*, vol. 18, no. 2, pp. 129-137, 2009. 82
- [106] O. Ahumada and J. R. Villalobos, "Application of planning models in the agri-food supply chain: A review," *European Journal of Operational Research*, vol. 196, no. 1, pp. 1-20, 2009. 82
- [107] R. Akkerman, D. Van Der Meer, and D. Pieter Van Donk, "Make to stock and mix to order: Choosing intermediate products in the food-processing industry," *International Journal of Production Research*, vol. 48, no. 12, pp. 3475-3492, 2010. 82
- [108] A. Rong, R. Akkerman, and M. Grunow, "An optimization approach for managing fresh food quality throughout the supply chain," *International Journal of Production Economics*, vol. 131, no. 1, pp. 421-429, 2011. 82
- [109] S. Guericke, A. Koberstein, F. Schwartz, and S. Voß, "A stochastic model for the implementation of postponement strategies in global distribution networks," *Decision Support Systems*, vol. 53, no. 2, pp. 294-305, 2012. 82
- [110] K. Lieckens and N. Vandaele, "Multi-level reverse logistics network design under uncertainty," *International Journal of Production Research*, vol. 50, no. 1, pp. 23-40, 2012. 82
- [111] A. Baghalian, S. Rezapour, and R. Z. Farahani, "Robust supply chain network design with service level against disruptions and demand uncertainties: A real-life case," *European Journal of Operational Research*, vol. 227, no. 1, pp. 199-215, 2013. 82
- [112] X. Lin, R. R. Negenborn, and G. Lodewijks, "Towards Quality-aware Control of Perishable Goods in Synchronodal Transport Networks," *IFAC-PapersOnLine*, vol. 49, no. 16, pp. 132-137, 2016. 82
- [113] Y. P. Tsang, K. L. Choy, C. H. Wu, G. T. S. Ho, H. Y. Lam, and V. Tang, "An intelligent model for assuring food quality in managing a multi-temperature food distribution centre," *Food Control*, vol. 90, pp. 81-97, 2018. 83

Appendix A

First segment modules code snippets

A.1 Inventory Management Module

A.1.1 MainApp.py

The first two lines of the `MainApp.py` file import all necessary modules. `Frame` is where all elements will be placed, `Tk` is necessary to run the interface and `Notebook` aggregates all tabs.

```
from tkinter import Frame, Tk
from tkinter.ttk import Notebook
```

After those imports, all created tab layouts are imported.

```
from Tab1Layout import Tab1
from Tab2Layout import Tab2
from Tab3Layout import Tab3
```

With all imports made, the main window is defined. The main window consists on a `Frame` containing a `Notebook`. To create the main window, the constructor method is used. After, the `Frame` is created using the constructor method again. Then, the `Notebook` is created inside the `Frame`. The “pages” of the `Notebook` are the tabs created and imported previously and for that reason were called “Page#”. Those “pages” were then added to the `Notebook` in the order they should appear in the interface.

```
class MainApp(Frame):

    def __init__(self, parent):

        Frame.__init__(self, parent)

        self.Notebook = Notebook(self)
        self.Notebook.pack(fill='both', expand=True)

        Page1 = Tab1(self.Notebook)
        Page2 = Tab2(self.Notebook)
        Page3 = Tab3(self.Notebook)

        self.Notebook.add(Page1, text='New Entry Query')
        self.Notebook.add(Page2, text='Inventory Table')
        self.Notebook.add(Page3, text='QMP Live Graph')
```

The following code block is necessary to start the interface. It consists on creating and executing a root window where the `MainApp.py` class will be placed with all its content.

```
if __name__ == '__main__':
```

```
Root = Tk()
Root.geometry('1600x900')
Root.title('Inventory Management Module - 1st Segment')
MainApp(Root).pack(fill='both', expand=True)
Root.mainloop()
```

A.1.2 Tab1Layout.py

As the name indicates, the `Tab1Layout.py` file is responsible for the layout of this “page” in the Notebook. As per usual, the first lines correspond to the import of all necessary elements. Imported here is `Button`, to create buttons, `DoubleVar`, to create float like objects, `Entry`, to create entry boxes, `Frame`, to place elements, `IntVar`, to create int like objects, `Label`, to create labels, and finally `LabelFrame`, here used to better highlight inputs. `CheckInfo` is a function imported from this tab’s functions and will be explored later.

```
from tkinter import (Button, DoubleVar, Entry, Frame, IntVar, Label,
                    LabelFrame)
```

```
from Tab1Functions import CheckInfo
```

A class was created to create a tab. This allows for its import by the `MainApp`. Similarly, to `MainApp` the constructor method is used twice with the same purpose. After, all variables used to input information are initialized. A `LabelFrame` is then created to better group and highlight the input and submission area of the tab. Then, labels and entry boxes are created and disposed in a grid where in front of the label there will be the matching entry box with the correct variable associated. Finally, a button is created and is responsible for the submission of information using the `CheckInfo` function.

```
class Tab1(Frame):

    def __init__(self, parent):
        Frame.__init__(self, parent)

        Quantity = DoubleVar()
        Class = IntVar()
        EntryQmp = DoubleVar()
        InitialQuality = DoubleVar()

        self.Group = LabelFrame(self, text='New Inventory Entry Form:')
        self.Group.grid(row=0, column=0, padx=10, pady=10)

        self.Label1 = Label(self.Group, text='Quantity:')
        self.Label1.grid(row=1, column=0, padx=10, pady=10)
        self.Label2 = Label(self.Group, text='Class:')
        self.Label2.grid(row=2, column=0, padx=10, pady=10)
        self.Label3 = Label(self.Group, text='Entry QMP')
        self.Label3.grid(row=3, column=0)
        self.Label4 = Label(self.Group, text='Initial Quality:')
        self.Label4.grid(row=4, column=0, padx=10, pady=10)
```

```

self.Label5 = Label(self.Group, text='Please Verify Before
                    Submitting')
self.Label5.grid(row=5, column=0, padx=10, pady=10)

self.Entry1 = Entry(self.Group, textvariable=Quantity)
self.Entry1.grid(row=1, column=1, padx=10, pady=10)
self.Entry2 = Entry(self.Group, textvariable=Class)
self.Entry2.grid(row=2, column=1, padx=10, pady=10)
self.Entry3 = Entry(self.Group, textvariable=EntryQmp)
self.Entry3.grid(row=3, column=1, padx=10, pady=10)
self.Entry4 = Entry(self.Group, textvariable=InitialQuality)
self.Entry4.grid(row=4, column=1, padx=10, pady=10)

self.Button1 = Button(self.Group,
                      text='Submit',
                      command=lambda: CheckInfo(Quantity.get(),
                                                Class.get(),
                                                EntryQmp.get(),
                                                InitialQuality.
                                                get()))

self.Button1.grid(row=5, column=1, padx=10, pady=10)

```

A.1.3 Tab1Functions.py

This file contains all the functions necessary for the first tab to operate properly. Starting with the imports, `csv` creates and edits csv files, `datetime` creates datetime objects necessary for inventory input, `os` will allow to verify if the inventory file is empty, `random` and `string` will be used for the creation of a random code, `exp` will be used to exponentiate some values in a formula, `Toplevel` will be used to create a top level windows for the interface and `showerror` to show an error window in the interface.

```

import csv
import datetime
import os
import random
import string
from math import exp
from tkinter import Label, LabelFrame, Toplevel
from tkinter.messagebox import showerror

```

`CheckInfo`, mentioned previously, verifies the validity of the input. If invalid, a warning window is shown and the input is not recorded. Otherwise the `NewEntry` function is called.

```

def CheckInfo(Quantity,
              Class,
              EntryQmp,

              if Quantity <= 0:
                  showerror('Error', 'Quantity Equal Or Bellow 0!')

```

```
elif Class not in [1, 2, 3, 4, 5]:
    showerror('Error', 'Invalid Class')

elif InitialQuality <= 0:
    showerror('Error', 'Quality Equal Or Bellow 0')

else:
    NewEntry(Quantity,
             Class,
             EntryQmp,
             InitialQuality)
```

The `NewEntry` function writes the input to a csv file. Firstly, a location for the file is given and data is collected. Another function is called for data collection, the `RandomCode` function. With that data, the `KeepingQualityFormula` function is called. After its execution, the file is opened and if empty, a header will be created using the `fieldnames` variable. Data is then written into the file according to the input, the result of `KeepingQualityFormula` and `RandomCode`. Data is addressed by matching each string in the `fieldnames` variable to the data variables. A window is show with all information written to the file after input.

```
def NewEntry(Quantity, Class, EntryQmp, InitialQuality):

    InventoryFile = '/home/joao/Desktop/Python/Production Layer/
1st Segment
/4 - Docs/Inventory/Inventory.csv'

    EntryDate = datetime.datetime.now().replace(microsecond=0)
    CompanyID = 'E1'
    EntryID = RandomCode()
    EaR, Qlim, QA, KQA, KQ = KeepingQualityFormula(Class,
                                                    InitialQuality,
                                                    EntryQmp)

    with open(InventoryFile, mode='a') as Inventory:
        fieldnames = ['Entry Date',
                     'Company ID',
                     'Entry ID',
                     'Entry QMP',
                     'Entry Quantity',
                     'Class',
                     'Initial Quality',
                     'Quality Limit',
                     'EaR',
                     'Inital Keeping Quality',
                     'Initial Quality Algorithm',
                     'Initial Keeping Quality Algorithm',
                     'Last Update',
```



```

        'Last Update QMP',
        'Last Update Quality ',
        'Last Update Keeping Quality ',
        'Last Update Quality Algorithm ',
        'Last Update Keeping Quality Algorithm ']

writer = csv.DictWriter(Inventory, fieldnames=fieldnames)

if os.stat(InventoryFile).st_size == 0:
    writer.writeheader()

writer.writerow({'Entry Date': EntryDate,
                'Company ID': CompanyID,
                'Entry ID': EntryID,
                'Entry QMP': EntryQmp,
                'Entry Quantity': Quantity,
                'Class': Class,
                'Initial Quality': InitialQuality,
                'Quality Limit': Qlim,
                'EaR': EaR,
                'Inital Keeping Quality': KQ,
                'Initial Quality Algorithm': QA,
                'Initial Keeping Quality Algorithm': KQA,
                'Last Update': EntryDate,
                'Last Update QMP': EntryQmp,
                'Last Update Quality': InitialQuality,
                'Last Update Keeping Quality': KQ,
                'Last Update Quality Algorithm': QA,
                'Last Update Keeping Quality Algorithm': KQA})

TopInv = Toplevel()

GroupTop = LabelFrame(TopInv, text='New Inventory Entry
                        Confirmation:')
GroupTop.grid(row=0, column=0, padx=10, pady=10)

Info1 = 'Date: {}'.format(EntryDate)
InfoInv1 = Label(GroupTop, text=Info1)
InfoInv1.grid(row=1, column=0, padx=10, pady=10)

Info2 = 'ID: {}'.format(EntryID)
InfoInv2 = Label(GroupTop, text=Info2)
InfoInv2.grid(row=2, column=0, padx=10, pady=10)

Info3 = 'QMP: {}'.format(EntryQmp)
InfoInv3 = Label(GroupTop, text=Info3)
InfoInv3.grid(row=3, column=0, padx=10, pady=10)

```

```
Info4 = 'Quantity: {}'.format(Quantity)
InfoInv4 = Label(GroupTop, text=Info4)
InfoInv4.grid(row=4, column=0, padx=10, pady=10)

Info5 = 'Class: {}'.format(Class)
InfoInv5 = Label(GroupTop, text=Info5)
InfoInv5.grid(row=5, column=0, padx=10, pady=10)

Info6 = 'Quality: {}'.format(InitialQuality)
InfoInv6 = Label(GroupTop, text=Info6)
InfoInv6.grid(row=6, column=0, padx=10, pady=10)

Info7 = 'Quality Limit: {}'.format(Qlim)
InfoInv7 = Label(GroupTop, text=Info7)
InfoInv7.grid(row=7, column=0, padx=10, pady=10)

Info8 = 'EaR: {}'.format(EaR)
InfoInv8 = Label(GroupTop, text=Info8)
InfoInv8.grid(row=8, column=0, padx=10, pady=10)

Info9 = 'Keeping Quality: {}'.format(KQ)
InfoInv9 = Label(GroupTop, text=Info9)
InfoInv9.grid(row=9, column=0, padx=10, pady=10)

Info10 = 'Quality Algorithm: {}'.format(QA)
InfoInv10 = Label(GroupTop, text=Info10)
InfoInv10.grid(row=10, column=0, padx=10, pady=10)

Info11 = 'Keeping Quality Algorithm: {}'.format(KQA)
InfoInv11 = Label(GroupTop, text=Info11)
InfoInv11.grid(row=11, column=0, padx=10, pady=10)

TopInv.mainloop()
```

The KeepingQuality function was developed as described in Tijssens & Polderdijk (1996). The objective is to evaluate quality and keeping quality variation according to temperature. The QMP is used for simulation in this study.

```
def KeepingQualityFormula(Class, Q0, QMP):

    if Class == 1:
        EaR = 13018.0
        QA = 'EQAID 1'
        KQA = 'EKQAID 1'
    elif Class == 2:
        EaR = 12520.0
        QA = 'EQAID 2'
```

```

        KQA = 'EKQAID 2'
    elif Class == 3:
        EaR = 6669.0
        QA = 'EQAID 3'
        KQA = 'EKQAID 3'
    elif Class == 4:
        EaR = 17147.0
        QA = 'EQAID 4'
        KQA = 'EKQAID 4'
    elif Class == 5:
        EaR = 9817.0
        QA = 'EQAID 5'
        KQA = 'EKQAID 5'

    Qlim = 1.0
    k = 1.0 * exp(EaR * ((1 / 283.15) - (1 / (QMP + 273.15))))
    KQ = (Q0 - Qlim) / k

    return EaR, Qlim, QA, KQA, KQ

```

The RandomCode functions returns a randomly generated code used for identifying inputs.

```

def RandomCode():

    Size = 6
    Characters = string.ascii_uppercase + string.digits
    Code = ''.join(random.choice(Characters) for _ in range(Size))

    return Code

```

A.1.4 Tab2Layout.py

This file operates similarly to Tab1Layout.py. Other than the imports already detailed some others are imported. Canvas will be used to create a place for variable size content and Scrollbar will be used to scroll through that content. ResetScroll and ShowInventory will be detailed later.

```

from tkinter import Button, Canvas, Frame, Scrollbar

from Tab2Functions import ResetScroll, ShowInventory

```

Again, a class is created to allow for import. A parent Frame is created with a child Canvas inside. Inside the Canvas a child Frame is created. Vertical and horizontal scrollbars are created in the Canvas. Then a window in the child Frame is created. Aside from a Button, all content will be displayed inside this window, which will be variable in size. The button is used to call the ShowInventory function.

```

class Tab2(Frame):

    def __init__(self, parent):
        Frame.__init__(self, parent)

```

```
self.Canvas = Canvas(self)

self.Frame = Frame(self.Canvas)

self.VSB = Scrollbar(self,
                     orient='vertical',
                     command=self.Canvas.yview)
self.Canvas.configure(yscrollcommand=self.VSB.set)
self.VSB.pack(side='right', fill='y')

self.HSB = Scrollbar(self,
                     orient='horizontal',
                     command=self.Canvas.xview)
self.Canvas.configure(xscrollcommand=self.HSB.set)
self.HSB.pack(side='bottom', fill='x')

self.Canvas.pack(side='left', fill='both', expand=True)

self.Canvas.create_window((4, 4),
                          window=self.Frame,
                          anchor='nw')

self.Frame.bind('<Configure>', lambda event,
               canvas=self.Canvas: ResetScroll)

self.Button1 = Button(self.Frame,
                      text='Update and View',
                      command=lambda: ShowInventory(self.Frame)
                      )
self.Button1.grid(row=0, column=0)
```

A.1.5 Tab2Functions.py

The only unknown import in this file is pandas. It will open, read and update the inventory csv file.

```
import csv
import datetime
from math import exp
from tkinter import Label

import pandas as pd
```

The show inventory function will display all information in inventory after its update. The file is opened by the `csv` module and looped through. Every word in that file will become a label displayed in the interface.

```
def ShowInventory(Frame):
```

```

UpdateQuality ()

InventoryFile = '/home/joao/Desktop/Python/Production Layer/
1st Segment/4 - Docs/Inventory/Inventory.csv '

with open(InventoryFile , mode='r') as Inventory:
    Reader = csv.reader(Inventory)

    r = 1
    for col in Reader:
        c = 0
        for row in col:
            # Show items in inventory as labels
            LabelInv = Label(Frame, text=row)
            LabelInv.grid(row=r , column=c , padx=10, pady=10)
            c += 1
        r += 1

```

The UpdateQuality function opens the inventory file and the QMP record file as DataFrames. Here, pandas DataFrames are used as they are less cumbersome than csv to read and update. The inventory is looped through and quality parameters updated using a similar function to KeepingQualityFormula seen previously. Items in inventory with zero quantity and quality are removed from inventory. Quality is set to zero by this function if its updated value falls below the defined quality limit.

```

def UpdateQuality ():

    InventoryFile = '/home/joao/Desktop/Python/Production Layer/
1st Segment/4 - Docs/Inventory/Inventory.csv '
    InventoryQmp = '/home/joao/Desktop/Python/Production Layer/
1st Segment/4 - Docs/Inventory/QMP History/QmpLog.csv '

    df = pd.read_csv(InventoryFile)
    df2 = pd.read_csv(InventoryQmp)

    LastRow = len(df.index)

    for row in range(0, LastRow, 1):
        Now = datetime.datetime.now().replace(microsecond=0)
        Now = Now.strftime('%Y-%m-%d %H:%M:%S')
        df.at[row, 'Last Update'] = Now
        df.to_csv(InventoryFile , index=False)

    for row in range(0, LastRow, 1):
        QMP = df2['Real'].iloc[-1]
        df.at[row, 'Last Update QMP'] = QMP
        Q0 = df.at[row, 'Initial Quality']

```

Safety and quality based traceability system for food supply chains

```
t1 = df.at[row, 'Entry Date']
t1 = datetime.datetime.strptime(t1, '%Y-%m-%d %H:%M:%S')
t2 = df.at[row, 'Last Update']
t2 = datetime.datetime.strptime(t2, '%Y-%m-%d %H:%M:%S')
t = abs(t2-t1).total_seconds()/3600
Class = df.at[row, 'Class']
Qlim = df.at[row, 'Quality Limit']
QA, KQA, Q, KQ = QualityFormula(QMP, Class, Q0, t, Qlim)
if Q <= Qlim:
    df.at[row, 'Last Update Quality'] = 0
    df.at[row, 'Last Update Keeping Quality'] = 0
    df.at[row, 'Last Update Quality Algorithm'] = QA
    df.at[row, 'Last Update Keeping Quality Algorithm'] = KQA
else:
    df.at[row, 'Last Update Quality'] = Q
    df.at[row, 'Last Update Keeping Quality'] = KQ
    df.at[row, 'Last Update Quality Algorithm'] = QA
    df.at[row, 'Last Update Keeping Quality Algorithm'] = KQA

df.drop(df[df['Entry Quantity'] <= 0].index, inplace=True)

df.drop(df[df['Last Update Quality'] <= 0].index, inplace=True)

df.to_csv(InventoryFile, index=False)
```

The `ResetScroll` function allows for the scrollbars in the interface to adapt to the variable size of the inventory.

```
def ResetScroll(Canvas):

    Canvas.configure(scrollregion=Canvas.bbox("all"))
```

A.1.6 Tab3Layout.py

In this file the only unknown imports are `matplotlib.pyplot`, `FuncAnimation`, `FigureCanvasTkAgg` and `NavigationToolbar2Tk`. Their function is to allow for a real time QMP plotting with `matplotlib.pyplot` being responsible for creating a plot, `FuncAnimation` for animated plotting, `FigureCanvasTkAgg` as a place for the plot to reside in the interface and `NavigationToolbar2Tk` for a navigation bar to create a navigation bar for the plot. `Animate` will be discussed later.

```
from tkinter import Frame

import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg,
                                                NavigationToolbar2Tk)

from Tab3Functions import Animate
```

The constructor method is used with the exact same purpose as before. `Location` is a variable to be passed into the `Animate` function. A figure where the plot will be located is created. A Canvas like object is created inside the figure using `FigureCanvasTkAgg`. A toolbar is created and will be located inside the Canvas using `NavigationToolbar2Tk`. A subplot called `Ax` is placed inside the figure. This subplot `Ax` will be updated by the `Animate` function.

```
class Tab3(Frame):

    def __init__(self, parent):
        Frame.__init__(self, parent)

        Location = self

        self.Fig = plt.Figure()
        self.Canvas = FigureCanvasTkAgg(self.Fig, self)
        self.NavBar = NavigationToolbar2Tk(self.Canvas, self)
        self.Canvas.get_tk_widget().pack(side='left',
                                          fill='both',
                                          expand=True)

        self.Ax = self.Fig.add_subplot(1, 1, 1)
        self.Ani = FuncAnimation(self.Fig,
                                 Animate,
                                 interval=1000,
                                 fargs=(Location,))
```

A.1.7 Tab3Functions.py

This file consists only in one function and is only dependent on one import. This import allows to read the QMP records file.

```
from pandas import read_csv
```

The `Animate` function is responsible by animating the plot displayed in the interface. The QMP record file location is given and evaluated in size. Only the five last QMP values will be used in the plot to prevent labels from clipping. `Ax` is cleared of the previous curves and then the maximum, current and minimum values for QMP will be plotted. To increase visual perception a grid, a legend and timestamps are added to the plot along with a title and axis labels. The `Location` variable defined in `Tab3Layout.py` is used here to set the location of the `Ax` variable in which the plot is updated.

```
def Animate(i, Location):

    InventoryQmp = '/home/joao/Desktop/Python/Production Layer/
1st Segment/4 - Docs/Inventory/QMP History/QmpLog.csv'

    Data = read_csv(InventoryQmp)

    LastRow = len(Data.index)
```

```
if LastRow <= 5:
    Max = Data.iloc[:, 0]
    Min = Data.iloc[:, 1]
    Real = Data.iloc[:, 2]
    Time = Data.iloc[:, 3]
else:
    Data = Data.tail(5)
    Max = Data.iloc[:, 0]
    Min = Data.iloc[:, 1]
    Real = Data.iloc[:, 2]
    Time = Data.iloc[:, 3]

Location.Ax.cla()

Location.Ax.plot_date(Time,
                      Max,
                      label='Maximum allowed QMP',
                      color='r',
                      linestyle='-',
                      marker=False)

Location.Ax.plot_date(Time,
                      Min,
                      label='Minimum allowed QMP',
                      color='b',
                      linestyle='-',
                      marker=False)

Location.Ax.plot_date(Time,
                      Real,
                      label='Current QMP',
                      color='g',
                      linestyle='-',
                      marker='.')

Location.Ax.grid()

Location.Ax.legend(loc='upper left')

Location.Ax.set_title('QMP Chart')

Location.Ax.set_xlabel('Time')

Location.Ax.set_ylabel('QMP')
```


A.2 Processing stage management module

A.2.1 MainApp.py

This file is coded in the exact same way as IMM's MainApp.py.

A.2.2 Tab1Layout.py

The layout of this tab is like IMM's first tab. The only difference in imports are the functions imported from this file's pair.

```
from tkinter import (Button, DoubleVar, Entry, Frame, IntVar, Label,
                    LabelFrame, StringVar)

from Tab1Functions import CheckExit, CheckInt, CheckRaw, NewOperation
```

One difference in the layout is the inclusion of three groups for input. One for raw materials, one for intermediate product and one for stage exit. The other difference is the inclusion of a button to create a new operation. Outside those two differences the code is written in a similar manner to IMM's Tab1Layout.py.

```
class Tab1(Frame):

    def __init__(self, parent):
        Frame.__init__(self, parent)

        self.Button1 = Button(self,
                              text='New Operation',
                              command=NewOperation)
        self.Button1.grid(row=0, column=0, padx=10, pady=10)

        RawID = StringVar()
        RawQuantity = DoubleVar()

        self.Group1 = LabelFrame(self, text='Raw Material Input')
        self.Group1.grid(row=1, column=0, padx=10, pady=10)

        self.Label1 = Label(self.Group1, text='Raw Material ID:')
        self.Label1.grid(row=2, column=0, padx=10, pady=10)
        self.Label2 = Label(self.Group1, text='Raw Material Quantity:')
        self.Label2.grid(row=3, column=0, padx=10, pady=10)

        self.Entry1 = Entry(self.Group1, textvariable=RawID)
        self.Entry1.grid(row=2, column=1, padx=10, pady=10)
        self.Entry2 = Entry(self.Group1, textvariable=RawQuantity)
        self.Entry2.grid(row=3, column=1, padx=10, pady=10)

        self.Button2 = Button(self.Group1,
                              text='Submit',
```

```

        command=lambda: CheckRaw(RawID.get(),
                                RawQuantity.get()
                                ()))
self.Button2.grid(row=4, column=1, padx=10, pady=10)

IntID = StringVar()
IntQuantity = DoubleVar()

# Intermediate product label frame
self.Group2 = LabelFrame(self,
                          text='Intermediate Product Input')
self.Group2.grid(row=1, column=2, padx=10, pady=10)

self.Label3 = Label(self.Group2,
                    text='Intermediate Product ID:')
self.Label3.grid(row=2, column=2, padx=10, pady=10)
self.Label4 = Label(self.Group2,
                    text='Intermediate Product Quantity:')
self.Label4.grid(row=3, column=2, padx=10, pady=10)

self.Entry3 = Entry(self.Group2, textvariable=IntID)
self.Entry3.grid(row=2, column=3, padx=10, pady=10)
self.Entry4 = Entry(self.Group2, textvariable=IntQuantity)
self.Entry4.grid(row=3, column=3, padx=10, pady=10)

self.Button3 = Button(self.Group2,
                      text='Submit',
                      command=
                          lambda: CheckInt(IntID.get(),
                                             IntQuantity.get()
                                             ()))
self.Button3.grid(row=4, column=3, padx=10, pady=10)

ExitQuantity = DoubleVar()
ExitQuality = DoubleVar()
ExitClass = IntVar()

self.Group3 = LabelFrame(self, text='Exit Output')
self.Group3.grid(row=1, column=4, padx=10, pady=10)

self.Label5 = Label(self.Group3, text='Exit Quantity:')
self.Label5.grid(row=2, column=4, padx=10, pady=10)
self.Label6 = Label(self.Group3, text='Exit Quality:')
self.Label6.grid(row=3, column=4, padx=10, pady=10)
self.Label7 = Label(self.Group3, text='Exit Class:')
self.Label7.grid(row=4, column=4, padx=10, pady=10)

```

```

self.Entry5 = Entry(self.Group3, textvariable=ExitQuantity)
self.Entry5.grid(row=2, column=5, padx=10, pady=10)
self.Entry6 = Entry(self.Group3, textvariable=ExitQuality)
self.Entry6.grid(row=3, column=5, padx=10, pady=10)
self.Entry7 = Entry(self.Group3, textvariable=ExitClass)
self.Entry7.grid(row=4, column=5, padx=10, pady=10)

self.Button4 = Button(self.Group3,
                      text='Submit',
                      command=
                        lambda: CheckExit(ExitQuantity.get(),
                                           ExitQuality.get(),
                                           ExitClass.get()))
self.Button4.grid(row=5, column=5, padx=10, pady=10)

```

A.2.3 Tab1Functions.py

All the imports used in this file were already explained and will not be shown. The NewOperation function writes a new header in the operational record file and displays a confirmation window.

```

def NewOperation():

    HistoryN = '/home/joao/Desktop/Python/Production Layer/1st Segment/
4 - Docs/Stage N History/StageN.csv'

    with open(HistoryN, mode='a') as StageN:

        Fieldnames = NDictKeys()

        Writer = csv.DictWriter(StageN, fieldnames=Fieldnames)

        Writer.writeheader()

    TopNew = Toplevel()
    TopNew.geometry('200x100')
    Info = Label(TopNew, text='New Operation Created')
    Info.grid(row=0, column=0, padx=10, pady=10)
    TopNew.mainloop()

```

The CheckRaw, CheckInt and CheckExit functions are extremely similar. Thus, only the CheckRaw function will be analyzed. It starts by opening the inventory file and deleting zero quantity and quality rows. This last function is not necessary, it is just a last-resort safety measure used in the case that it did not operate previously. Data input is evaluated and if valid, more information is collected from inventory. The used quantity is removed from inventory and the RawInput function is called. If invalid, a window appears showing the first detected error.

```

def CheckRaw(RawID, RawQuantity):

    InventoryFile = '/home/joao/Desktop/Python/Production Layer/

```

Safety and quality based traceability system for food supply chains

```
1st Segment/4 - Docs/Inventory/Inventory.csv '

df = pd.read_csv(InventoryFile)
df.drop(df[df['Entry Quantity'] <= 0].index, inplace=True)
df.drop(df[df['Last Update Quality'] <= 0].index, inplace=True)
df.to_csv(InventoryFile, index=False)

LastRow = len(df.index)

if RawID in df.values and RawQuantity > 0:

    for row in range(0, LastRow, 1):
        if df.at[row, 'Entry ID'] == RawID:

            if RawQuantity <= df.at[row, 'Entry Quantity']:
                df.at[row, 'Entry Quantity'] -= RawQuantity
                df.to_csv(InventoryFile, index=False)
            else:
                showerror('Error', 'Excessive Quantity')
                break

            LU = 'Last Update'
            LUQMP = 'Last Update QMP'
            LUQ = 'Last Update Quality'
            LUKQ = 'Last Update Keeping Quality'
            LUQA = 'Last Update Quality Algorithm'
            LUKQA = 'Last Update Keeping Quality Algorithm'

            ID = RawID
            Quantity = RawQuantity
            LastUpdate = df.at[row, LU]
            Time = TimeStamp()
            QMP = df.at[row, LUQMP]
            Quality = df.at[row, LUQ]
            KQuality = df.at[row, LUKQ]
            QAlgorithm = df.at[row, LUQA]
            KQAlgorithm = df.at[row, LUKQA]

            RawInput(ID,
                    Quantity,
                    Time,
                    LastUpdate,
                    QMP,
                    Quality,
                    KQuality,
                    QAlgorithm,
                    KQAlgorithm)
```

```

else :
    if RawID not in df.values:
        showerror('Error', 'Item Not In Inventory')
    elif RawQuantity <= 0:
        showerror('Error', 'Quantity Bellow 0')

```

RawInput, IntInput and ExitOutput are very similar and thus only RawInput will be discussed. This function writes all relevant data either from user input or from collection from inventory. This function terminates after calling RawInputInfo.

```

def RawInput(ID,
             Quantity,
             Time,
             LastUpdate,
             QMP,
             Quality,
             KQuality,
             QAlgorithm,
             KQAlgorithm):

    HistoryN = '/home/joao/Desktop/Python/Production Layer/1st Segment/
4 - Docs/Stage N History/StageN.csv'
    InventoryFile = '/home/joao/Desktop/Python/Production Layer/
1st Segment/4 - Docs/Inventory/Inventory.csv'

    Inventory = pd.read_csv(InventoryFile)

    with open(HistoryN, mode='a') as StageN:

        for row in range(0, len(Inventory)):
            if Inventory.at[row, 'Entry ID'] == ID:
                Qlim = Inventory.at[row, 'Quality Limit']
                EaR = Inventory.at[row, 'EaR']

        Fieldnames = NDictKeys()

        Writer = csv.DictWriter(StageN, fieldnames=Fieldnames)

        Writer.writerow({'RM ID': ID,
                        'RM Quantity': Quantity,
                        'RM Time Stamp': Time,
                        'RM Last Update': LastUpdate,
                        'RM QMP': QMP,
                        'RM Quality Limit': Qlim,
                        'RM EaR': EaR,
                        'RM Quality': Quality,
                        'RM Keeping Quality': KQuality,

```

Safety and quality based traceability system for food supply chains

```
        'RM Quality Algorithm ': QAlgorithm ,
        'RM Keeping Quality Algorithm ': KQAlgorithm})

RawInputInfo(ID ,
             Quantity ,
             Time ,
             LastUpdate ,
             QMP,
             Qlim ,
             EaR,
             Quality ,
             KQuality ,
             QAlgorithm ,
             KQAlgorithm)
```

Again, the RawInputInfo, IntInputInfo and ExitOutputInfo were created similarly and for the same purpose. That purpose is to display a window with all corresponding data written to the operational history file. Once more only the raw material variant will be displayed in this section.

```
def RawInputInfo(ID ,
                Quantity ,
                Time ,
                LastUpdate ,
                QMP,
                Qlim ,
                EaR,
                Quality ,
                KQuality ,
                QAlgorithm ,
                KQAlgorithm):

    TopRaw = Toplevel()

    GroupRaw = LabelFrame(TopRaw, text='Raw Material Input Registred:')
    GroupRaw.grid(row=0, column=0, padx=10, pady=10)

    Info1 = 'RM ID: {}'.format(ID)
    InfoRaw1 = Label(GroupRaw, text=Info1)
    InfoRaw1.grid(row=1, column=0, padx=10, pady=10)

    Info2 = 'RM Quantity: {}'.format(Quantity)
    InfoRaw2 = Label(GroupRaw, text=Info2)
    InfoRaw2.grid(row=2, column=0, padx=10, pady=10)

    Info3 = 'RM Time Stamp: {}'.format(Time)
    InfoRaw3 = Label(GroupRaw, text=Info3)
    InfoRaw3.grid(row=3, column=0, padx=10, pady=10)
```

```

Info4 = 'RM Last Update: {}'.format(LastUpdate)
InfoRaw4 = Label(GroupRaw, text=Info4)
InfoRaw4.grid(row=4, column=0, padx=10, pady=10)

Info5 = 'RM QMP: {}'.format(QMP)
InfoRaw5 = Label(GroupRaw, text=Info5)
InfoRaw5.grid(row=5, column=0, padx=10, pady=10)

Info6 = 'RM Quality Limit: {}'.format(Qlim)
InfoRaw6 = Label(GroupRaw, text=Info6)
InfoRaw6.grid(row=6, column=0, padx=10, pady=10)

Info7 = 'RM EaR: {}'.format(EaR)
InfoRaw7 = Label(GroupRaw, text=Info7)
InfoRaw7.grid(row=7, column=0, padx=10, pady=10)

Info8 = 'RM Quality: {}'.format(Quality)
InfoRaw8 = Label(GroupRaw, text=Info8)
InfoRaw8.grid(row=8, column=0, padx=10, pady=10)

Info9 = 'RM Keeping Quality: {}'.format(KQuality)
InfoRaw9 = Label(GroupRaw, text=Info9)
InfoRaw9.grid(row=9, column=0, padx=10, pady=10)

Info10 = 'RM Quality Algorithm: {}'.format(QAlgorithm)
InfoRaw10 = Label(GroupRaw, text=Info10)
InfoRaw10.grid(row=10, column=0, padx=10, pady=10)

Info11 = 'RM Keeping Quality Algorithm: {}'.format(KQAlgorithm)
InfoRaw11 = Label(GroupRaw, text=Info11)
InfoRaw11.grid(row=11, column=0, padx=10, pady=10)

TopRaw.mainloop()

```

Some other functions are in this file. TimeStamp is one. It has the purpose of assigning the current timestamp to variable when called. using a function for a menial task such as this is justified using timestamps by different functions and writing this function avoids repetition.

```

def TimeStamp():

    return datetime.datetime.now().replace(microsecond=0)

```

Another function is NDictKeys, whose purpose is to facilitate addressing to data contained or to be written to the operational history file, since it will have to be opened in different occasions.

```

def NDictKeys():

    Keys = ['RM ID',

```

```
'RM Quantity ',
'RM Time Stamp ',
'RM Last Update ',
'RM QMP',
'RM Quality Limit ',
'RM EaR',
'RM Quality ',
'RM Keeping Quality ',
'RM Quality Algorithm ',
'RM Keeping Quality Algorithm ',
'IP ID ',
'IP Quantity ',
'IP Time Stamp ',
'IP Last Update ',
'IP QMP',
'IP Quality Limit ',
'IP EaR',
'IP Quality ',
'IP Keeping Quality ',
'IP Quality Algorithm ',
'IP Keeping Quality Algorithm ',
'SN E ID ',
'SN E Quantity ',
'SN E Time Stamp ',
'SN E QMP',
'SN E Quality Limit ',
'SN E EaR',
'SN E Class ',
'SN E Quality ',
'SN E Keeping Quality ',
'SN E Quality Algorithm ',
'SN E Keeping Quality Algorithm ']
```

return Keys

Like `NDictKeys`, there is `IDictKeys`. It has the same purpose as `NDictKeys`, but it is applied to the inventory file. Since it is written in the exact same manner, but with different fields, it is unnecessary to describe it. Another function that does not need further exposition is `RandomCode`, used to identify exits from stage to inventory.

A.2.4 Tab2Layout.py

The code for this tab is extremely similar to IMM's `Tab2Layout.py`.

A.2.5 Tab2Functions.py

All the functions used in this file are akin to IMM's `Tab2Functions.py`, thus not needing further exposition.

A.2.6 Tab3Layout.py

The code for this file is identical to IMM's Tab3Layout.py.

A.2.7 Tab3Functions.py

Just as the previous section, the functions in this file do not need further description. The section containing IMM's Tab3Functions.py can be referred for the explanation of the functions.

A.3 Order management module

For simplicity and readability, all relevant characteristics of each folder and file will be presented in the same order as presented in the tree shown in Figure A.1

A.3.1 Run.py

The purpose of this file is to start the web server using all content in the OMMServer folder. As this is only a prototype server, the host's address is 127.0.0.1, also known as localhost. Due to the multitude of servers having to coexist for testing, each was assigned a different port.

```
from OMMServer import CreateApp

App = CreateApp()

if __name__ == '__main__':
    App.run(debug=True, port=5001)
```

A.3.2 OMMServer

This folder contains all contents that will be displayed in the website and how it will be processed.

A.3.2.1 __ini__.py

This file is always necessary when the folder content is or will be imported by another file unless the module is added to PYTHONPATH, which is, summarily, where the modules location is and allows for them to be imported easily. The PYTHONPATH process was not used as the content of OMMServer is to be used only by the web server. Thus, making it unnecessary to turn the modules globally available. Although this file can be left empty in many cases, here it is convenient to write code that will influence all aggregated content. The imports for this file Flask, the used framework, and LoginManager, which is necessary as not all functions should be accessed by others than a company's workers. The Config.py file will be analyzed in the next subsection.

```
from flask import Flask
from flask_login import LoginManager

from OMMServer.Config import Config
```

Safety and quality based traceability system for food supply chains

The `LoginManager` is initialized and is configured for the case someone tries to access a login restricted function of the website to force the redirection to the login page.

```
LoginManager = LoginManager()
LoginManager.login_view = 'Users.Login'
```

The `CreateApp` function aggregates all content. Some imports are made inside the function to avoid errors when launching the server. The `Config` file is used here to pass some settings to the web app.

```
def CreateApp(config_class=Config):
    App = Flask(__name__)
    App.config.from_object(Config)

    LoginManager.init_app(App)

    from OMMServer.Users.Routes import Users
    from OMMServer.NonUsers.Routes import NonUsers
    App.register_blueprint(Users)
    App.register_blueprint(NonUsers)

    return App
```

A.3.2.2 Config.py

As shown before, this file passes settings to the web app. In this case, only a secret key was passed. The key was obtained using the `os` module to create a randomly generated value. The `os` module was used instead of the `random` module as this key is used for security purposes and the `random` module is only pseudo-random.

```
import os

class Config:
    SECRET_KEY = os.urandom(16)
```

A.3.2.3 Models.py

This file contains clients addresses and their respective value, the user loader for the `LoginManager` and the `User` class. This last two are necessary parts for the login ability. To make the user class, `UserMixin` is used as a parent class.

```
from OMMServer import LoginManager
from flask_login import UserMixin

Addresses = {'Home': 0,
            'Destination 1': 1,
            'Destination 2': 2,
            'Destination 3': 3,
            'Destination 4': 4,
```

```

        'Destination 5': 5,
        'Destination 6': 6,
        'Destination 7': 7,
        'Destination 8': 8,
        'Destination 9': 9}

@login_manager.user_loader
def load_user(user_id):
    return User(user_id)

class User(UserMixin):
    def __init__(self, id):
        self.id = id

```

A.3.2.4 Templates

This folder contains all necessary html files. The StageN.html file will have to be replicated in the same manner PSMM has to, to reflect all processing stages.

Availability.html

This file contains the code for the availability page of the website. This page has a small query that non-users can have access in order to view available products. The first line corresponds to the use of the Layout.html that is used by all pages.

```

{% extends 'Layout.html' %}

{% block content %}
<div class='container'>
  <form action='{{ url_for('NonUsers.AvailabilityResult') }}'
  method='POST'>
    <fieldset>
      <legend> Search For Available Products </legend>

      <p> Class ID: </p>
      <p>
        <input type='text' name='ClassID'
        placeholder='Ex: 1'
        required>
      </p>

      <p> Time To Delivery: </p>
      <p>
        <input type='text' name='TimeToDelivery'
        placeholder='Ex: 500' required>
      </p>

```

Safety and quality based traceability system for food supply chains

```
        <p>
            <input type='submit' value='Search'>
        </p>
    </fieldset >
</form>
</div>
{% endblock %}
```

AvailabilityResult.html

This file simply displays a table with the results from the query of the availability page.

```
{% extends 'Layout.html' %}

{% block content %}

    {% for table in tables %}
        <div class = 'container'>
            <h3> Search Results:</h3>
            {{ table|safe }}
        </div>
    {% endfor %}
{% endblock %}
```

Home.html

This page shows the company. As an example, a generic enterprise name and a price table is presented.

```
{% extends 'Layout.html' %}

{% block content %}

    <div class='container'>
        <h3>
            E? Product Table:
        </h3>
    </div>

    {% for table in tables %}
        <div class='container'>
            {{ table|safe }}
        </div>
    {% endfor %}
{% endblock content %}
```

Inventory.html

This page shows nothing else but the inventory table of a company.

```
{% extends 'Layout.html' %}

{% block content %}

    {% for table in tables %}
        <div class='container'>
            {{ table|safe }}
        </div>
    {% endfor %}
{% endblock %}
```

Layout.html

The Layout.html file is the general layout of all web pages. Having this file and extending it to others allows to avoid repetition of the exact same code. The code block section is what will be overridden by other files.

```
<!DOCTYPE html>
<html>
<head>

    <meta charset='utf-8'>
    <meta name='viewport '
        content='width=device - width ,
        initial - scale=1,
        shrink - to - fit=no'>

    <link rel='stylesheet '
        href='https://stackpath.bootstrapcdn.com/bootstrap/
        4.3.1/css/bootstrap.min.css '
        integrity='sha384 - ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY /
        iJTQUOhcWr7x9JvoRxT2MZw1T'
        crossorigin='anonymous'>

    <link rel="stylesheet" type="text/css"
        href="{% url_for('static', filename='Style.css') %}">

</head>

<body>

<div class = 'topnav'>
    <div class = 'container'>
        <a href = '{% url_for('NonUsers.Home') %}'>
            Home
        </a>

        <a href = '{% url_for('Users.Login') %}'>
```

```
        Login
    </a>

    <a href = '{{ url_for('NonUsers.Availability') }}'>
        Verify Availability
    </a>

    <a href = '{{ url_for('NonUsers.Map') }}'>
        Map
    </a>
</div>
</div>

{% block content %}
{% endblock %}

<script src='https://code.jquery.com/jquery-3.3.1.slim.min.js'
    integrity='sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5s
    mXKp4YfRvH+8abtTE1Pi6jizo'
    crossorigin='anonymous'>
</script>
<script src='https://cdnjs.cloudflare.com/ajax/libs/popper.js/
    1.14.7/umd/popper.min.js'
    integrity='sha384-UO2eT0CpHqdSJK6hJty5KVphtPhzWj9WO1clHTM
    Ga3JDZwrnQq4sF86dIHNDz0W1'
    crossorigin='anonymous'>
</script>
<script src='https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/
    bootstrap.min.js'
    integrity='sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/
    nJGzIxFDsf4x0xIM+B07jRM'
    crossorigin='anonymous'>
</script>
</body>
</html>
```

Login.html

This file contains the login form.

```
{% extends 'Layout.html' %}

{% block content %}
<div class='container'>
    <form action='' method='POST'>
        <fieldset>
            <legend> Login </legend>
```

```

        <p> Username </p>
        <p>
            <input type='text' name='Username' required>
        </p>

        <p> Password: </p>
        <p>
            <input type='text' name='Password'>
        </p>

        <p>
            <input type='submit' value='Enter'>
        </p>
    </fieldset>
</form>
</div>
{% endblock content %}

```

Map.html

This file was created for testing purposes. It contains hyperlinks for all pages in the website. This file simply serves to speed up the access to some pages while testing and should not be included in a non-development server as is.

```

{% extends 'Layout.html' %}

{% block content %}
    <div class='sidenav'>
        <a href = '{{ url_for('NonUsers.Home') }}'>
            Home
        </a>
        <a href = '{{ url_for('Users.Order') }}'>
            Order
        </a>
        <a href = '{{ url_for('Users.OrderResult') }}'>
            OrderResult
        </a>
        <a href = '{{ url_for('Users.Personal') }}'>
            Personal
        </a>
        <a href = '{{ url_for('Users.StageN') }}'>
            StageN
        </a>
        <a href = '{{ url_for('NonUsers.Availability') }}'>
            Availability
        </a>
        <a href = '{{ url_for('NonUsers.AvailabilityResult') }}'>

```

Safety and quality based traceability system for food supply chains

```
        AvailabilityResult
    </a>
    <a href = '{{ url_for('Users.Inventory') }}'>
        Inventory
    </a>
    <a href = '{{ url_for('Users.Login') }}'>
        Login
    </a>
    <a href = '{{ url_for('Users.Logout') }}'>
        Logout
    </a>
    <a href = '{{ url_for('NonUsers.Map') }}'>
        Map
    </a>
</div>
{% endblock %}
```

Order.html

This file contains a multi-step form for the creation of an order for expedition.

```
{% extends 'Layout.html' %}

{% block content %}
    <div class='container'>

        {% if Step == 'Step1' %}
            <form action='{{ url_for('Users.Order') }}' method='POST'>
                <fieldset>
                    <legend> Process Order Form:</legend>
                    <br>
                    <p> Order Name: </p>
                    <p>
                        <input type='text'
                            name='OrderName'
                            placeholder='Order Name' required>
                    </p>
                    <p> Number Of Destinations: </p>
                    <p>
                        <input type='text'
                            name='NumberOfDestinations'
                            placeholder='Ex: 3' required>
                    </p>
                    <p> Number Of Arcs In Graph: </p>
                    <p>
                        <input type='text' name='NumberOfArcsInGraph'
                            placeholder='Ex: 4' required>
                    </p>
                </fieldset>
            </form>
        {% else %}
            <div class='text-align: center;'>
                <p> Order Form: </p>
            </div>
        {% endif %}
    </div>
{% endblock %}
```



```

<p> Exit Date: </p>
<p>
  <input type='text '
    name='ExitDate '
    placeholder='YYYY-MM-DD HH:MM:SS' required >
</p>
<p> Estimated Travel QMP: </p>
<p>
  <input type='text '
    name='TravelQMP '
    placeholder='Ex: -2' required >
</p>
<p> Delivery Date: </p>
<p>
  <input type='text '
    name='DeliveryDate '
    placeholder='YYYY-MM-DD HH:MM:SS' required >
</p>
<br>
<br>
<p>
  <input type='hidden ' name='Step ' value='Step2'>
  <input type='submit ' value='Next Step'>
</p>
</fieldset >
</form>

{% elif Step == 'Step2' %}
  <form action='{{ url_for('Users.Order') }}' method='POST'>
    <fieldset >
      <legend> Process Order Form: </legend>
      <br>
      <p> Previous Data: </p>
      <p> Order Name: {{ OrderName }} </p>
      <p> Number Of Destinations :
        {{ NumberOfDestinations }}
      </p>
      <p> Number Of Arcs In Graph :
        {{ GraphLength }}
      </p>
      <p> Exit Date: {{ ExitDate }} </p>
      <p> Travel QMP: {{ TravelQMP }} </p>
      <p> Delivery Date: {{ DeliveryDate }} </p>
      <br>
      <p> Summary Table: </p>
      {% for i in range(0, NumberOfDestinations, 1) %}
        <p>

```

```

        <select name='DestinationAddress'>
            {% for Address in Addresses %}
                <option
                    value="{{ Address }}">{{ Address }}
                </option>
            {% endfor %}
        </select>
        <input type='text '
            name='GoodsToDeliver '
            placeholder='Ex: ID1, ID2, ID3, ID4 '
            required>
        <input type='text '
            name='QuantityToDeliver '
            placeholder='Ex: 100, 200, 300, 400 '
            required>
    </p>
{% endfor %}
<br>
<p> Construct Graph:</p>
{% for i in range(0, GraphLength, 1) %}
    <p>
        <select name='GraphOrigins'>
            {% for Address in Addresses %}
                <option
                    value="{{ Address }}">{{ Address }}
                </option>
            {% endfor %}
        </select>
        <select name='GraphDestinations'>
            {% for Address in Addresses %}
                <option
                    value="{{ Address }}">{{ Address }}
                </option>
            {% endfor %}
        </select>
        <input type='text '
            name='GraphRelativeCost '
            placeholder='Ex: 10'>
    </p>
{% endfor %}
<br>
<br>
<p>
    <input type='hidden ' name='Step ' value='Step3'>
    <input type='submit ' value='Make Order'>
</p>
</fieldset >

```

```

        </form>

        {% elif Step == 'Step3' %}
        <form action='{{ url_for('Users.OrderResult') }}'
        method='POST'>
            <fieldset>
                <legend> Order Info: </legend>
                <br>
                <p> Previous Data: </p>
                <p> Order Name: {{ OrderName }} </p>
                <p> Number Of Destinations:
                    {{ NumberOfDestinations }}
                </p>
                <p> Number Of Arcs In Graph :
                    {{ GraphLength }}
                </p>
                <p> Exit Date: {{ ExitDate }} </p>
                <p> Travel QMP: {{ TravelQMP }} </p>
                <p> Delivery Date: {{ DeliveryDate }} </p>
                <p> Destinations Addresses:
                    {{ DestinationsAddresses }}
                </p>
                <p> Goods To Deliver: {{ GoodsToDeliver }} </p>
                <p> Quantity To Deliver:
                    {{ QuantityToDeliver }}
                </p>
                <p> Graph Origins: {{ GraphOrigins }} </p>
                <p> Graph Destinations:
                    {{ GraphDestinations }}
                </p>
                <p> Graph Relative Cost:
                    {{ GraphRelativeCost }}
                </p>
                <br>
                <br>
                <input type='submit' value='View Documents'>
            </fieldset>
        </form>
        {% endif %}

    </div>
{% endblock %}

```

OrderResult.html

This file contains the code necessary to display all the files created when an order is submitted.

```
{% extends 'Layout.html' %}
```

Safety and quality based traceability system for food supply chains

```
{% block content %}
  <fieldset >
    <legend> Order {{ OrderName }} Documents:
    </legend>

    <form action='Users.OrderResult' method=POST>
      <input type='submit' value='Submit'>
    </form>

    {% for table in tables %}
      <div class='container'>
        {{ table|safe }}
      </div>
    {% endfor %}

  </fieldset >
{% endblock %}
```

Personal.html

This is the personal page displayed when a user is logged in. It contains hyperlinks to access materials deemed to be only accessible by company collaborators.

```
{% extends 'Layout.html' %}

{% block content %}

  <div class = 'sidenav'>

    <p>
      <a href = '{{ url_for('Users.Inventory') }}'>
        Inventory
      </a>
    </p>

    <p>
      <a href = '{{ url_for('Users.StageN') }}'>
        Stage N
      </a>
    </p>

    <p>
      <a href = '{{ url_for('Users.Order') }}'>
        Order
      </a>
    </p>

  </div>

{% endblock %}
```

```

        </p>

        <p>
            <a href = '{{ url_for('Users.Logout') }}'>
                Log Out
            </a>
        </p>
    </div>
{% endblock%}

```

StageN.html

Just as Inventory.html, this file exposes a processing stage operational history. This file will have to be replicated once per stage and some modifications to other components of the website will have to be made in to give access to each stage.

```

{% extends 'Layout.html' %}

{% block content %}

    {% for table in tables %}
        <div class='container'>
            {{ table|safe }}
        </div>
    {% endfor %}
{% endblock %}

```

A.3.2.5 Static

This folder includes all static files to be used. As no images or videos were necessary in the context of this dissertation and thus, only contains one file.

Style.css

The purpose of this file is to style the html tags used in the templates.

```

div {
    padding-top: 10px;
    padding-bottom: 10px;
    padding-right: 10px;
    padding-left: 10px;
}

table {
    border: 1px solid black;
    text-align: left;
    border-collapse: collapse;
    width: 100%;
}

```

```
td {
    text-align: left;
    padding: 10px;
}

th {
    text-align: left;
    background-color: darkgreen;
    border: 1px solid black;
    color: white;
    padding: 10px;
}

tr:hover {
    background-color: lightgray;
}

.topnav {
    background-color: green;
    width: 100%;
    padding-top: 10px;
    padding-bottom: 10px;
    padding-right: 10px;
    padding-left: 10px;
}

.sidenav {
    background-color: lightgray;
    width: 10%;
    height: 100%;
    position: fixed;
}

a {
    color: white;
    padding-top: 10px;
    padding-bottom: 10px;
    padding-right: 10px;
    padding-left: 10px;
    text-align: center;
    font-size: 20px
}

a:visited {
    color: white;
}
```

```
a: hover {
    background-color: limegreen;
    color: white;
}
```

A.3.2.6 Users

This folder contains files respecting to content only users can access.

`__init__.py`

This file is left empty. It is necessary as it allows to import the `Routes.py` file by the `CreateApp` function.

`Routes.py`

This file contains all functionalities available for a company's collaborators. In this file there are some imports that have not yet been discussed. `Blueprint` serves to create divisions inside the web server. In this case the divisions are users and non-users. The `redirect` import is to redirect from one page to the other, `render_template` renders the html files already shown, `request` pulls data from the queries for processing and `url_for` creates the routes that allow navigation from one page to another.

```
import datetime

import pandas as pd
from flask import (Blueprint, redirect, render_template, request,
                  url_for)
from flask_login import (current_user, login_required, login_user,
                        logout_user)
```

The other imports are functions required to the processing of input data. To be able to import these functions as “regular” imports, the `__init__.py` file is required.

```
from OMMServer.Models import Addresses, User
from OMMServer.Users.MasterFunctions.Login import ValidateCredentials
from OMMServer.Users.MasterFunctions.Order import OrderMaster
from OMMServer.Users.MasterFunctions.RetrieveTables
    import RetrieveTables
```

After this, a location for the templates and static files is given and the `Blueprint` is created. This `Blueprint` is one of two shown in the `CreateApp` function.

```
Templates = '/home/joao/Desktop/Python/Production Layer/1st Segment/
3 - OMM/OMMServer/Templates '

Static = '/home/joao/Desktop/Python/Production Layer/1st Segment/
3 - OMM/OMMServer/Static '

Users = Blueprint('Users',
                  __name__,
```

Safety and quality based traceability system for food supply chains

```
template_folder=Templates ,
static_folder=Static)
```

After doing all that, the web pages themselves can be created. Decorators are used to create routes for this specific Blueprint instead of doing it for the whole web app. Using `request`, data is pulled from forms and used to call the functions that process them. Here can be seen the use of global variables. They are used to pass variables between steps in a function without stopping the function itself, which is necessary for the multistep order form.

```
@Users.route('/Login', methods=['GET', 'POST'])
def Login():

    # Get data from
    if request.method == 'POST':
        Username = request.form['Username']
        Password = request.form['Password']

        Credentials = ValidateCredentials(Username, Password)

        if Credentials == True:
            login_user(User(Username))
            return redirect(url_for('Users.Personal'))

    # Renders login page template
    return render_template('Login.html',
                           title='Login')

@Users.route('/Personal')
@login_required
def Personal():
    """Displays user only available hyperlinks.
    """

    return render_template('Personal.html',
                           title='Personal')

@Users.route('/Order', methods=['GET', 'POST'])
@login_required
def Order():
    """Processes a new order
    """

    if 'Step' not in request.form:

        return render_template('Order.html',
                               title='Order',
```



```

        Step='Step1 ')

elif request.form['Step'] == 'Step2':

    global OrderName
    OrderName = request.form['OrderName']

    global TravelQMP
    TravelQMP = request.form['TravelQMP']
    TravelQMP = int(TravelQMP)

    global DeliveryDate
    DeliveryDate = request.form['DeliveryDate']
    DeliveryDate = datetime.datetime.strptime(
        DeliveryDate, '%Y-%m-%d %H:%M:%S')

    global NumberOfDestinations
    NumberOfDestinations = request.form['NumberOfDestinations']
    NumberOfDestinations = int(NumberOfDestinations)

    global GraphLength
    GraphLength = request.form['NumberOfArcsInGraph']
    GraphLength = int(GraphLength)

    global ExitDate
    ExitDate = request.form['ExitDate']
    ExitDate = datetime.datetime.strptime(ExitDate,
        '%Y-%m-%d %H:%M:%S')

    return render_template('Order.html',
        title='Order',
        Step='Step2',
        Addresses=Addresses,
        OrderName=OrderName,
        TravelQMP=TravelQMP,
        DeliveryDate=DeliveryDate,
        NumberOfDestinations
            =NumberOfDestinations,
        GraphLength=GraphLength,
        ExitDate=ExitDate)

elif request.form['Step'] == 'Step3':

    DestinationsAddresses = request.form.getlist(
        'DestinationAddress', type=str)

    GoodsToDeliver = request.form.getlist('GoodsToDeliver',

```

```

                                type=str)

QuantityToDeliver = request.form.getlist('QuantityToDeliver',
                                           type=str)

GraphOrigins = request.form.getlist('GraphOrigins', type=str)

GraphDestinations = request.form.getlist('GraphDestinations',
                                           type=str)

GraphRelativeCost = request.form.getlist('GraphRelativeCost',
                                           type=str)

OrderMaster(OrderName,
            TravelQMP,
            DeliveryDate,
            NumberOfDestinations,
            GraphLength,
            ExitDate,
            DestinationsAddresses,
            GoodsToDeliver,
            QuantityToDeliver,
            GraphOrigins,
            GraphDestinations,
            GraphRelativeCost)

return render_template('Order.html',
                      title='Order',
                      Step='Step3',
                      Addresses=Addresses,
                      OrderName=OrderName,
                      TravelQMP=TravelQMP,
                      DeliveryDate=DeliveryDate,
                      NumberOfDestinations
                        =NumberOfDestinations,
                      GraphLength=GraphLength,
                      ExitDate=ExitDate,
                      DestinationsAddresses
                        =DestinationsAddresses,
                      GoodsToDeliver=GoodsToDeliver,
                      QuantityToDeliver=QuantityToDeliver,
                      GraphOrigins=GraphOrigins,
                      GraphDestinations=GraphDestinations,
                      GraphRelativeCost=GraphRelativeCost)

@Users.route('/OrderResult', methods=['GET', 'POST'])
```

```

@login_required
def OrderResult():

    Conditions, Summary, Quality, Graph, Route, ICD, IFC
        = RetriveTables(OrderName)

    return render_template('OrderResult.html',
                           title='Order Result',
                           OrderName=OrderName,
                           tables=[Conditions.to_html(index=False),
                                   Summary.to_html(index=False),
                                   Quality.to_html(index=False),
                                   Graph.to_html(index=False),
                                   Route.to_html(index=False),
                                   ICD.to_html(index=False),
                                   IFC.to_html(index=False)])

@Users.route('/Inventory')
@login_required
def Inventory():

    Inventory = '/home/joao/Desktop/Python/Production Layer/
1st Segment/4 - Docs/Inventory/Inventory.csv'

    Inventory = pd.read_csv(Inventory)

    return render_template('Inventory.html',
                           title='Inventory',
                           tables=[Inventory.to_html(index=False)])

@Users.route('/StageN')
@login_required
def StageN():

    StageN = '/home/joao/Desktop/Python/Production Layer/1st Segment/
4 - Docs/Stage N History/StageN.csv'

    StageN = pd.read_csv(StageN)

    return render_template('StageN.html',
                           title='Stage N',
                           tables=[StageN.to_html(index=False)])

@Users.route('/Logout')

```

Safety and quality based traceability system for food supply chains

```
@login_required
def Logout():

    logout_user()

    return redirect(url_for('NonUsers.Home'))
```

MasterFunctions

Master functions are functions that simply call other functions, slave functions, to process data for them. In that sense, master functions delegate functions to slave functions. This condition made simple for the purpose of better organization.

__init__.py

This file is left empty. It is necessary for the imports seen in the User section.

Login.py

This file contains a simple script that returns either true or false if the credentials pulled from the login form match the ones in the dictionary. Keeping credentials in a dictionary is not safe and will have to be changed to a safer method for real world scenarios. However, it is a valid approach for testing purposes.

```
def ValidateCredentials(Username, Password):

    UsernamePassword = {' ': ' ',
                        'Admin': 'Admin',
                        'Worker1': 'Worker1'}

    if (Username, Password) in UsernamePassword.items():
        Credentials = True
    else:
        Credentials = False

    return Credentials
```

Order.py

This file contains a function that simply calls many other slave functions. These functions create all necessary documentation for processing an order in a specific path determined by the name given that order and removes the quantity ordered from inventory.

```
from OMMServer.Models import Addresses
from OMMServer.Users.SlaveFunctions.ConditionsCSV
import MakeConditionsCSV
from OMMServer.Users.SlaveFunctions.GraphCSV import MakeGraphCSV
from OMMServer.Users.SlaveFunctions.IDCCSV import MakeIDCCSV
from OMMServer.Users.SlaveFunctions.IFCCSV import MakeIFCCSV
from OMMServer.Users.SlaveFunctions.MakeFolder import OrderFolder
from OMMServer.Users.SlaveFunctions.QualityCSV import MakeQualityCSV
```

```

from OMMServer.Users.SlaveFunctions.RouteCSV import MakeRouteCSV
from OMMServer.Users.SlaveFunctions.SummaryCSV import MakeSummaryCSV
from OMMServer.Users.SlaveFunctions.RemoveQuantity
    import RemoveQuantity

def OrderMaster( OrderName ,
                 TravelQMP ,
                 DeliveryDate ,
                 NumberOfDestinations ,
                 GraphLength ,
                 ExitDate ,
                 DestinationsAddresses ,
                 GoodsToDeliver ,
                 QuantityToDeliver ,
                 GraphOrigins ,
                 GraphDestinations ,
                 GraphRelativeCost ):

    Path = '/home/joao/Desktop/Python/Production Layer/1st Segment/
    4 - Docs/Orders '

    OrderFolder( Path ,
                 OrderName )

    MakeConditionsCSV( Path ,
                      OrderName ,
                      NumberOfDestinations ,
                      GraphLength ,
                      ExitDate ,
                      TravelQMP ,
                      DeliveryDate )

    MakeSummaryCSV( Path ,
                   OrderName ,
                   DestinationsAddresses ,
                   GoodsToDeliver ,
                   QuantityToDeliver )

    MakeGraphCSV( Path ,
                  OrderName ,
                  GraphOrigins ,
                  GraphDestinations ,
                  GraphRelativeCost )

    MakeRouteCSV( Path ,
                  OrderName ,

```

Safety and quality based traceability system for food supply chains

```
        Addresses ,
        NumberOfDestinations ,
        GraphOrigins ,
        GraphDestinations ,
        GraphRelativeCost)

MakeQualityCSV(Path ,
               OrderName ,
               GoodsToDeliver ,
               ExitDate ,
               DeliveryDate ,
               TravelQMP)

MakeIFCCSV(Path ,
            OrderName ,
            GoodsToDeliver)

MakeIDCCSV(Path ,
            OrderName ,
            GoodsToDeliver)

RemoveQuantity(GoodsToDeliver ,
               QuantityToDeliver)
```

RetrieveTables.py

This is a simple script to retrieve and return all files previously created into tables to be shown in the OrderResult page.

```
import pandas as pd

def RetriveTables(OrderName):

    Path = '/home/joao/Desktop/Python/Production Layer/1st Segment/
    4 - Docs/Orders '

    ConditionsPath = Path+'/' +OrderName+'/' +OrderName+'_Conditions.csv '
    SummaryPath = Path+'/' +OrderName+'/' +OrderName+'_Summary.csv '
    QualityPath = Path+'/' +OrderName+'/' +OrderName+'_Quality.csv '
    GraphPath = Path+'/' +OrderName+'/' +OrderName+'_Graph.csv '
    RoutePath = Path+'/' +OrderName+'/' +OrderName+'_Route.csv '
    ICDPath = Path+'/' +OrderName+'/' +OrderName+'_ICD.csv '
    IFCPATH = Path+'/' +OrderName+'/' +OrderName+'_IFC.csv '

    Conditions = pd.DataFrame(ConditionsPath)
    Summary = pd.DataFrame(SummaryPath)
    Quality = pd.DataFrame(QualityPath)
    Graph = pd.DataFrame(GraphPath)
```

```
Route = pd.DataFrame(RoutePath)
ICD = pd.DataFrame(ICDPath)
IFC = pd.DataFrame(IFCPath)

return Conditions, Summary, Quality, Graph, Route, ICD, IFC
```

SlaveFunctions

These functions are called by the master functions and serve to better separate and organize all tasks necessary to the processing of query information.

__init__.py

Another empty file to allow for importing functions.

ConditionsCSV

This file is a simple script to generate a .csv file detailing the expected travel conditions.

```
import csv

def MakeConditionsCSV(Path,
                     OrderName,
                     NumberOfDestinations,
                     GraphLength,
                     ExitDate,
                     TravelQMP,
                     DeliveryDate):

    with open(Path+'/' + OrderName + '/' + OrderName + '_Conditions.csv',
              mode='w') as Conditions:

        Fieldnames = ['Order Name',
                     'Number Of Destinations',
                     'Number Of Arcs In Graph',
                     'Exit Date',
                     'Travel QMP',
                     'Delivery Date']

        Writer = csv.DictWriter(Conditions, fieldnames=Fieldnames)

        Writer.writeheader()

        Writer.writerow({'Order Name': OrderName,
                        'Number Of Destinations':
                            NumberOfDestinations,
                        'Number Of Arcs In Graph': GraphLength,
                        'Exit Date': ExitDate,
                        'Travel QMP': TravelQMP,
```

```
'Delivery Date': DeliveryDate}))
```

GraphCSV.py

This script turns the travel graph from the order query into a table.

```
import pandas as pd

def MakeGraphCSV(Path,
                 OrderName,
                 GraphOrigins,
                 GraphDestinations,
                 GraphRelativeCost):

    GraphDict = {'Origins ': GraphOrigins,
                 'Destinations ': GraphDestinations,
                 'Relative Cost ': GraphRelativeCost}

    Graph = pd.DataFrame.from_dict(GraphDict)

    Graph.to_csv(Path+'/' + OrderName+'/' + OrderName+'_Graph.csv',
                 index=False)

    return Graph
```

IDCCSV.py

This file makes the Information for Direct Consumer file.

```
import pandas as pd

def MakeIDCCSV(Path,
               OrderName,
               GoodsToDeliver):

    InventoryFile = '/home/joao/Desktop/Python/Production Layer/
1st Segment/4 - Docs/Inventory/Inventory.csv'

    StageNFile = '/home/joao/Desktop/Python/Production Layer/
1st Segment/4 - Docs/Stage N History/StageN.csv'

    Inventory = pd.read_csv(InventoryFile)

    GoodsToDeliver = [
        item for items in GoodsToDeliver for item in items.split()]

    IDCI = pd.DataFrame(data=[],
```



```

        columns=['Entry Date',
                'Company ID',
                'Entry ID',
                'Entry QMP',
                'Entry Quantity',
                'Class',
                'Initial Quality',
                'Quality Limit',
                'EaR',
                'Initial Keeping Quality',
                'Initial Quality Algorithm',
                'Initial Keeping Quality
                Algorithm',
                'Last Update',
                'Last Update QMP',
                'Last Update Quality',
                'Last Update Keeping Quality',
                'Last Update Quality Algorithm',
                'Last Update Keeping Quality
                Algorithm'])
for row in range(0, len(Inventory.index)):
    for item in range(0, len(GoodsToDeliver)):
        if Inventory.at[row, 'Entry ID'] == GoodsToDeliver[item]:
            IDCIRow = Inventory.loc[row]
            IDCI = IDCI.append(IDCIRow, ignore_index=True)
Mark = pd.DataFrame(columns=['MARKED FOR VALIDATION'])
for row in range(0, len(IDCI), 1):
    Mark.at[row, 'MARKED FOR VALIDATION'] = 'Marked For Validation'
IDCI = IDCI.join(Mark)

StageN = pd.read_csv(StageNFile)
OperationID = pd.DataFrame(data=[], columns=['OPERATION ID'])
OpID = 0
for row in range(0, len(StageN)):
    OperationID.at[row, 'OPERATION ID'] = OpID
    if StageN.at[row, 'SN E ID'] == 'SN E ID':
        OpID += 1
OperationID = OperationID.join(StageN)

temp = pd.DataFrame(data=[],
                    columns=['OPERATION ID',
                            'RM ID',
                            'RM Quantity',
                            'RM Time Stamp',
                            'RM Last Update',
                            'RM QMP',
                            'RM Quality Limit',

```

```

'RM EaR',
'RM Quality',
'RM Keeping Quality',
'RM Quality Algorithm',
'RM Keeping Quality Algorithm',
'IP ID',
'IP Quantity',
'IP Time Stamp',
'IP Last Update',
'IP QMP',
'IP Quality Limit',
'IP EaR',
'IP Quality',
'IP Keeping Quality',
'IP Quality Algorithm',
'IP Keeping Quality Algorithm',
'SN E ID',
'SN E Quantity',
'SN E Time Stamp',
'SN E QMP',
'SN E Quality Limit',
'SN E EaR',
'SN E Class',
'SN E Quality',
'SN E Keeping Quality',
'SN E Quality Algorithm',
'SN E Keeping Quality Algorithm'])
for row in range(0, len(OperationID.index)):
    for item in range(0, len(GoodsToDeliver)):
        if OperationID.at[row, 'SN E ID'] == GoodsToDeliver[item]:
            tempRow = OperationID.loc[row]
            temp = temp.append(tempRow, ignore_index=True)

IDCSN = pd.DataFrame(data=[],
                    columns=['RM ID',
                            'RM Quantity',
                            'RM Time Stamp',
                            'RM Last Update',
                            'RM QMP',
                            'RM Quality Limit',
                            'RM EaR',
                            'RM Quality',
                            'RM Keeping Quality',
                            'RM Quality Algorithm',
                            'RM Keeping Quality Algorithm',
                            'IP ID',
                            'IP Quantity',

```

```

        'IP Time Stamp',
        'IP Last Update',
        'IP QMP',
        'IP Quality Limit',
        'IP EaR',
        'IP Quality',
        'IP Keeping Quality',
        'IP Quality Algorithm',
        'IP Keeping Quality Algorithm',
        'SN E ID',
        'SN E Quantity',
        'SN E Time Stamp',
        'SN E QMP',
        'SN E Quality Limit',
        'SN E EaR',
        'SN E Class',
        'SN E Quality',
        'SN E Keeping Quality',
        'SN E Quality Algorithm',
        'SN E Keeping Quality Algorithm'])

for row in range(0, len(OperationID)):
    if OperationID.at[row, 'OPERATION ID']
        == temp.at[0, 'OPERATION ID']:
        IDCSNRow = StageN.loc[row]
        IDCSN = IDCSN.append(IDCSNRow, ignore_index=True)
IDCSN.drop(IDCSN.tail(1).index, inplace=True)

IDCC = pd.DataFrame(data=[],
                    columns=['Entry Date',
                            'Company ID',
                            'Entry ID',
                            'Entry QMP',
                            'Entry Quantity',
                            'Class',
                            'Initial Quality',
                            'Quality Limit',
                            'EaR',
                            'Initial Keeping Quality',
                            'Initial Quality Algorithm',
                            'Initial Keeping Quality Algorithm',
                            'Last Update',
                            'Last Update QMP',
                            'Last Update Quality',
                            'Last Update Keeping Quality',
                            'Last Update Quality Algorithm',
                            'Last Update Keeping Quality
                                Algorithm'])

```

Safety and quality based traceability system for food supply chains

```
for row in range(0, len(IDCSN)):
    for entry in range(0, len(Inventory)):
        if IDCSN.at[row, 'RM ID'] == Inventory.at[entry,
                                                    'Entry ID']:
            IDCCRow = Inventory.loc[entry]
            IDCC = IDCC.append(IDCCRow, ignore_index=True)
        elif IDCSN.at[row, 'IP ID'] == Inventory.at[entry,
                                                    'Entry ID']:
            IDCCRow = Inventory.loc[entry]
            IDCC = IDCC.append(IDCCRow, ignore_index=True)

IDC = pd.DataFrame()
IDC = IDC.append(IDCI, ignore_index=True)
IDC = IDC.append(IDCC, sort=False)
IDC = IDC.join(IDCSN)

IDC.to_csv(Path+'/' + OrderName + '/' + OrderName + '_IDC.csv',
           index=False)
```

IFCCSV.py

This file is very similar to the previous. For testing purposes, it was assumed that the Information for Final Consumers and Information for Direct Consumers is the same. In a real use case scenario this file would need to be adapted, however, it is built in a similar manner simply with different information. For this reason, it is not necessary to discuss this file in particular.

MakeFolder.py

This function creates a folder to contain all the documents. The folder created has the same name as the order. To avoid errors, if a directory with the same name already exists it is then deleted and a new one created. The `os` import is used to evaluate if the directory exists and to create a new one and the `shutil` import to delete an existent directory.

```
import os
import shutil

def OrderFolder(Path, OrderName):

    FolderDir = Path + '/' + OrderName

    if os.path.exists(FolderDir):
        shutil.rmtree(FolderDir)

    os.mkdir(FolderDir)
```

QualityCSV.py

This file is very similar to the quality evaluation already detailed in other files joined with the

creation of files that was already discussed. This function determines the expected variation of quality during transport.

RemoveQuantity.py

This function alters the inventory file according to the exit of products due to an order.

```
import pandas as pd

def RemoveQuantity(GoodsToDeliver ,
                  QuantityToDeliver ):

    InventoryFile = '/home/joao/Desktop/Python/Production Layer/
1st Segment/4 - Docs/Inventory/Inventory.csv '
    Inventory = pd.read_csv(InventoryFile)

    GoodsToDeliver = [
        item for items in GoodsToDeliver for item in items.split()]
    QuantityToDeliver = [
        quantity for quantities in QuantityToDeliver
        for quantity in quantities.split()]

    for item in range(0, len(GoodsToDeliver), 1):
        for row in range(0, len(Inventory), 1):
            if Inventory.at[row, 'Entry ID'] == GoodsToDeliver[item]:
                Inventory.at[row,
                    'Entry Quantity'] -=
                    float(QuantityToDeliver[item])

    Inventory.to_csv(InventoryFile , index=False)
```

RouteCSV.py

This file determines the most appropriate route for delivery of orders according to the input graph in the Order form. To determine the route, the Prim's algorithm is used. In summary, this algorithm determines a minimum spanning tree that passes through all nodes. To determine this route, the cost from node to node is taking into consideration. However, it is possible to use other values if information input is correct for those values.

```
import numpy as np
import pandas as pd

def MakeRouteCSV(Path ,
                OrderName ,
                Addresses ,
                NumberOfDestinations ,
                GraphOrigins ,
                GraphDestinations ,
```

```

        GraphRelativeCost):

NumberOfNodes = NumberOfDestinations + 1

Graph = []

Graph.append(GraphOrigins)
Graph.append(GraphDestinations)
Graph.append(GraphRelativeCost)

Graph = np.array(Graph).T.tolist()

for Row in range(0, len(Graph), 1):
    for (key, value) in Addresses.items():

        if Graph[Row][0] == key:
            Graph[Row][0] = value

        if Graph[Row][1] == key:
            Graph[Row][1] = value

for Row in range(0, len(Graph), 1):
    Graph[Row][2] = int(Graph[Row][2])

AdjacencyMatrix = []

for i in range(0, NumberOfNodes, 1):
    AdjacencyMatrix.append([])
    for j in range(0, NumberOfNodes, 1):
        AdjacencyMatrix[i].append(0)

for i in range(0, len(Graph), 1):
    AdjacencyMatrix[Graph[i][0]][Graph[i][1]] = Graph[i][2]
    AdjacencyMatrix[Graph[i][1]][Graph[i][0]] = Graph[i][2]

Node = 0
Route = []
Edges = []
Visited = []
MinimumEdge = [None, None, float('inf')]

while len(Route) != NumberOfNodes - 1: iterations

    Visited.append(Node)

    for Edge in range(0, NumberOfNodes, 1):
        if AdjacencyMatrix[Node][Edge] != 0:

```

```

Edges.append([Node, Edge, AdjacencyMatrix[Node][Edge]])

for Edge in range(0, len(Edges), 1):
    if Edges[Edge][2] < MinimumEdge[2]
    and Edges[Edge][1] not in Visited:
        MinimumEdge = Edges[Edge]

Edges.remove(MinimumEdge)

Route.append(MinimumEdge)

Node = MinimumEdge[1]
MinimumEdge = [None, None, float('inf')]

for Row in range(0, len(Route), 1):
    for (key, value) in Addresses.items():

        if Route[Row][0] == value:
            Route[Row][0] = key

        if Route[Row][1] == value:
            Route[Row][1] = key

Route = pd.DataFrame(columns=['Origin ',
                             'Destination ',
                             'Relative Cost'],
                    data=Route)
Route.to_csv(Path+'/' + OrderName+'/' + OrderName+'_Route.csv',
            index=False)

return Route

```

SummaryCSV.py

This file creates a simple summary of the orders. This summary includes the name of the order, the destinations, the goods and quantity to deliver per destination.

```

import pandas as pd

def MakeSummaryCSV(Path,
                  OrderName,
                  DestinationsAddresses,
                  GoodsToDeliver,
                  QuantityToDeliver):

    Summary = pd.DataFrame(data=[GoodsToDeliver, QuantityToDeliver],
                          columns=[DA for DA

```

Safety and quality based traceability system for food supply chains

```
                in DestinationsAddresses],
                index=['Goods To Deliver ',
                       'Quantity To Deliver '])

Summary.to_csv(Path+'/' + OrderName+'/' + OrderName+'_Summary.csv')
```

A.3.2.7 NonUsers

This folder contains all content available to all non-collaborators. Due to the simplicity of this content there was no necessity to create master and slave functions.

`__init__.py`

Again, this is an empty file to allow for importing.

`AvailabilityFunctions.py`

This function processes the input of the availability form and returns a table with the result.

```
import pandas as pd

def AvailabilitySearch(ClassID, TimeToDelivery):

    Inventory = '/home/joao/Desktop/Python/Production Layer/
    1st Segment/4 - Docs/Inventory/Inventory.csv'

    df = pd.read_csv(Inventory)

    Columns = ['Entry Date',
               'Company ID',
               'Entry QMP',
               'Initial Quality',
               'Initial Keeping Quality',
               'Initial Quality Algorithm',
               'Initial Keeping Quality Algorithm',
               'Last Update',
               'Last Update QMP',
               'Last Update Quality Algorithm',
               'Last Update Keeping Quality Algorithm']

    df.drop(columns=Columns, axis=1, inplace=True)

    df.drop(df[df['Class'] != ClassID].index, inplace=True)

    df.drop(df[df['Last Update Keeping Quality']
              < TimeToDelivery].index, inplace=True)

    return df
```


Routes.py

As the Routes.py file for the users this contains all routes for non-users.

```

from flask import (render_template, Blueprint, request, redirect,
                  url_for)
import pandas as pd

from OMMServer.NonUsers.AvailabilityFunction import AvailabilitySearch

Templates = '/home/joao/Desktop/Python/Production Layer/1st Segment/
3 - OMM/OMMServer/Templates'

Static = '/home/joao/Desktop/Python/Production Layer/1st Segment/
3 - OMM/OMMServer/Static'

NonUsers = Blueprint('NonUsers',
                    __name__,
                    template_folder=Templates,
                    static_folder=Static)

@NonUsers.route('/')
def Home():

    PriceTable = '/home/joao/Desktop/Python/Production Layer/
1st Segment/4 - Docs/PriceTable.csv'

    df = pd.read_csv(PriceTable)

    return render_template('Home.html',
                          title='Home Page',
                          tables=[df.to_html(index=False)])

@NonUsers.route('/Availability', methods=['GET', 'POST'])
def Availability():

    return render_template('Availability.html',
                          title='Availability')

@NonUsers.route('/AvailabilityResult', methods=['GET', 'POST'])
def AvailabilityResult():

    if request.method == 'POST':
        ClassID = request.form['ClassID']
        ClassID = int(ClassID)

```

Safety and quality based traceability system for food supply chains

```
TimeToDelivery = request.form['TimeToDelivery']
TimeToDelivery = float(TimeToDelivery)
Search = AvailabilitySearch(ClassID, TimeToDelivery)

return render_template('AvailabilityResult.html',
                       title='Availability Result',
                       tables=[Search.to_html(index=False)])

@NonUsers.route('/Map')
def Map():

    return render_template('Map.html',
                           title='Map')
```

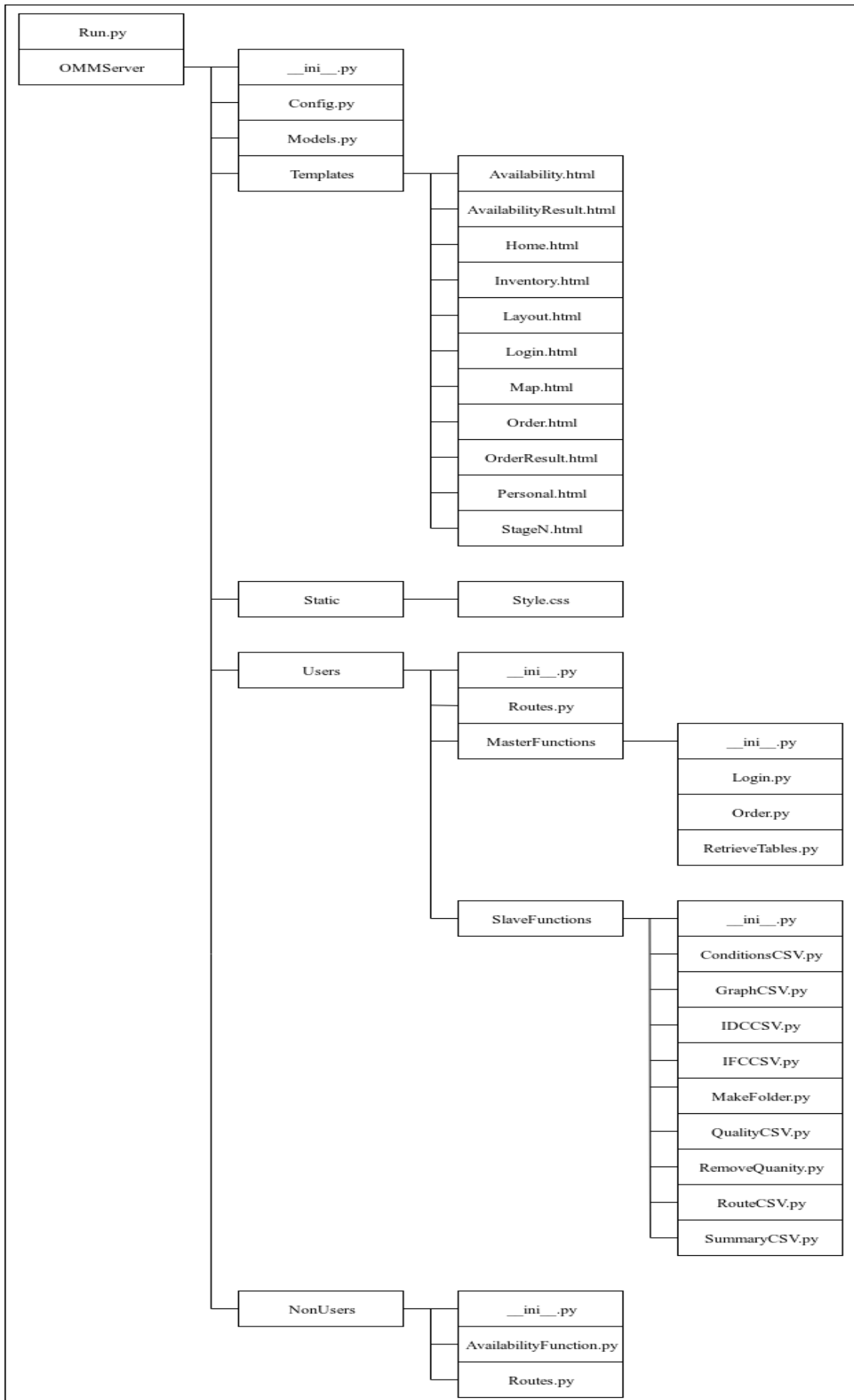


Figure A.1: Order management module file hierarchy.

Appendix B

Regulator layer code snippets

For simplicity and readability, all relevant characteristics of each folder and file will be presented in the same order as presented in the tree shown in Figure B.1

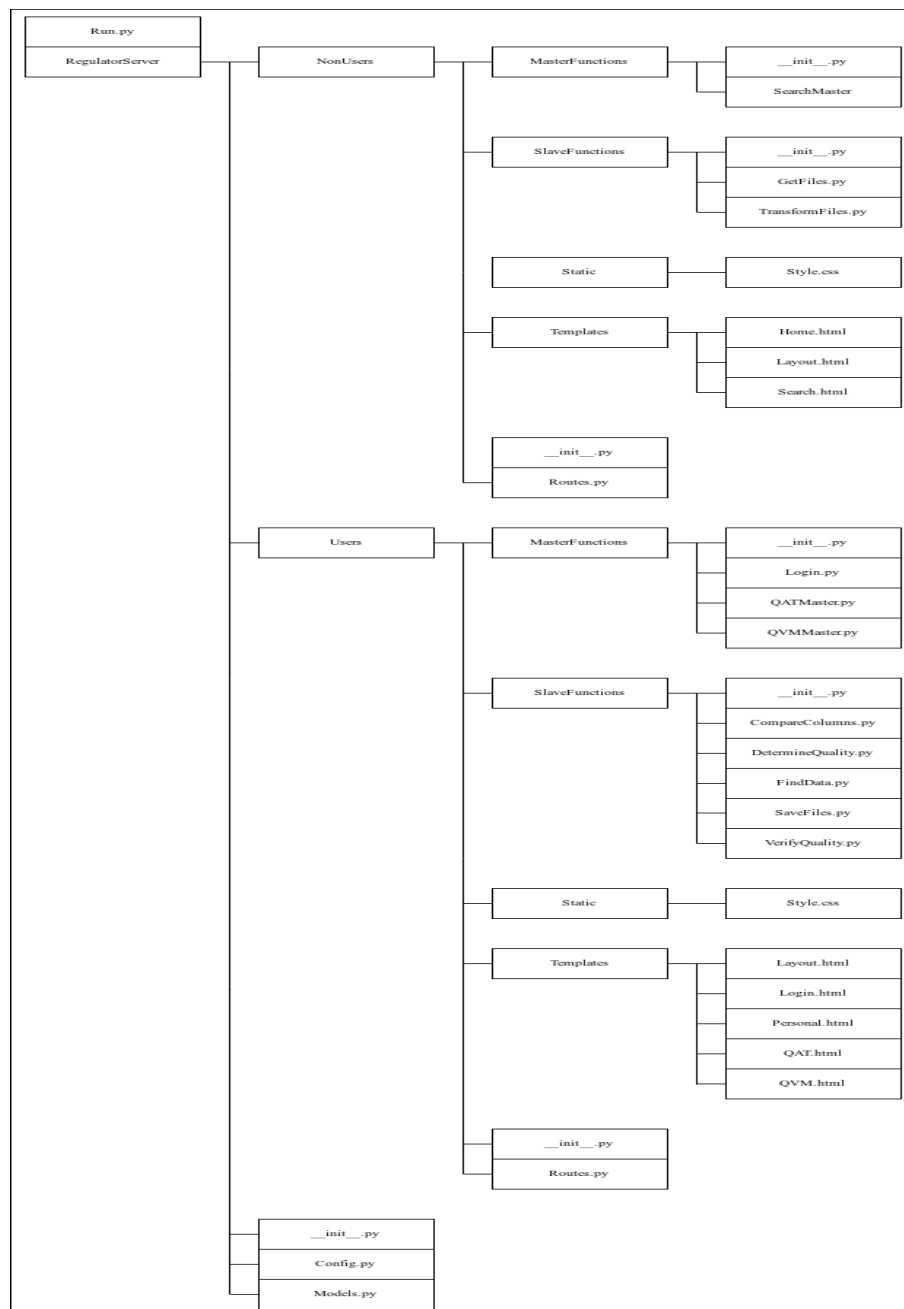


Figure B.1: Regulator layer file hierarchy.

B.1 Run.py

This file is made similar to OMM's Run.py and does not need further exposition.

B.2 Regulator server

B.2.1 __init__.py

This file is similar to OMM's __init__.py and it is unnecessary to expose it further.

B.2.2 Config.py

Just as the previous file this is similar to OMM's Config.py thus not needing more exposition.

B.2.3 Models.py

Similarly to the previous two files, this is almost identical to OMM's Models.py and will to be exposed further.

B.2.4 NonUsers

B.2.4.1 __init__.py

This is an empty file to allow for import.

B.2.4.2 Routes.py

This file contains all paths available to non-users.

Just as the equally named file already presented, all imports are made, a Blueprint is created and the routes created.

```
from flask import (Blueprint, redirect, render_template, request,
                  url_for)
from RegulatorServer.NonUsers.MasterFunctions.SearchMaster import
SearchMaster

Templates = '/home/joao/Desktop/Python/Regulator Layer/RegulatorServer/
NonUsers/Templates'

Static = '/home/joao/Desktop/Python/Regulator Layer/RegulatorServer/
NonUsers/Static'

NonUsers = Blueprint('NonUsers',
                    __name__,
                    template_folder=Templates,
                    static_folder=Static)
```

```
@NonUsers.route('/')
def Home():

    return render_template('Home.html',
title='Home')

@NonUsers.route('/Search')
def Search():

    if 'Step' not in request.form:

        return render_template('Search.html',
title='Search', Step='Step1')

    if request.form['Step'] == 'Step2':

        ID = request.form['ID']

        History = SearchMaster(ID)

    return render_template('Search.html', title='Search',
ID=ID, tables=[History.to_html(index=False)])
```

MasterFunctions

This folder contains all master functions for functionalities available to non-users.

__init__.py

Empty file to allow for import

SeachMaster.py

This file imports all slave functions necessary to construct an historic file for any given product.

```
from RegulatorServer.NonUsers.SlaveFunctions.GetFiles import GetFiles
from RegulatorServer.NonUsers.SlaveFunctions.TransformFiles import
TransformFiles

def SearchMaster(ID):

    Pieces = GetFiles(ID)

    History = TransformFiles(Pieces)

    return History
```

Safety and quality based traceability system for food supply chains

SlaveFunctions

This folder contains all files necessary for the correct functioning of SearchMaster.py

`__init__.py`

Another empty file to allow for import.

`GetFiles.py`

This file search all files and retrieves all relevant pieces of history necessary to construct any given products full history.

```
from os import listdir
import pandas as pd

def GetFiles(ID):

IFCPath = '/home/joao/Desktop/Python/Regulator Layer/Docs/ValidPostUC/
PostValUCIFC'

Pieces = []

for ifc in listdir(IFCPath):

    IFC = pd.read_csv(ifc)

    for row in range(len(IFC.index)):

        if IFC.at[row, 'Entry ID'] == ID or IFC.at[row, 'MARKED FOR
VALIDATION'] == ID:

Pieces.append(IFCPath+'/' + ifc)

return Pieces
```

`TransformFiles.py`

This file compiles all the pieces coming from GetFiles.py into a concise history file.

```
import pandas as pd

def TransformFiles(Pieces):

    History = []

    for ifc in range(len(History)):

        History[ifc] = pd.read_csv(History[ifc])

    History = pd.concat(Pieces)
```



```
return History
```

Static

Folder containing the `Style.css` file used for styling. As this file is very similar to its OMM eponymous and does not need further description.

Templates

This folder contains all template that present non-user available content.

Home.html

File for the regulators homepage.

```
{% extends 'Layout.html' %}

{% block content %}
    <div class='container'>
        <h3>
            Validation And Information Storage
        </h3>
    </div>
{% endblock %}
```

Layout.html

Just as in the Order Management Module, this file contains all common presentations that other templates inherit.

```
<!DOCTYPE html>
<html>
<head>

<meta charset='utf-8'>
<meta name='viewport '
content='width=device - width , initial - scale=1, shrink - to - fit=no'>

<link rel='stylesheet '
href='https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/
bootstrap.min.css '
integrity='sha384 - ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY /
iJTQUOhcWr7x9JvoRxT2MZw1T'
crossorigin='anonymous'>

<link rel="stylesheet" type="text/css" href="/Static/Style.css">

</head>

<body>
```

Safety and quality based traceability system for food supply chains

```
<div class = 'topnav'>
  <div class = 'container'>
    <a href = '{{ url_for('NonUsers.Home') }}'>
      Home
    </a>

    <a href='{{ url_for('NonUsers.Search') }}'>
      Search
    </a>

    <a href='{{ url_for('Users.Login') }}'>
      Login
    </a>
  </div>
</div>

{% block content %}
{% endblock %}

<script src='https://code.jquery.com/jquery-3.3.1.slim.min.js '
integrity='sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH
+8abtTE1Pi6jizo '
crossorigin='anonymous'>
</script>
<script src='https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/
umd/popper.min.js '
integrity='sha384-UO2eT0CpHqdSJK6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrn
Qq4sF86dIHNDz0W1 '
crossorigin='anonymous'>
</script>
<script src='https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/
bootstrap.min.js '
integrity='sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4
x0xIM+B07jRM '
crossorigin='anonymous'>
</script>
</body>
</html>
```

Search.html

This file presents the search form necessary to find product history.

```
{% extends 'Layout.html' %}

{% block content %}
```

```
{% if Step == 'Step1' %}
    <form action='{{ url_for('NonUsers.Search') }}' method='POST'>
        <fieldset>
            <legend> Search Perishable Identifier:</legend>
            <br>
            <p>
                <input type='text' name='ID'
                    placeholder='Perishable ID'
                    required>
                <input type='hidden' name='Step' value='Step2'>
                <input type='submit' value='Search'>
            </p>
        </fieldset>
    </form>

{% elif Step == 'Step2' %}
    <h3> Results for {{ ID }} </h3>
    {% for table in tables %}
        <div class = 'container'>
            <h3> Search Results:</h3>
            {{ table|safe }}
        </div>
    {% endfor %}

{% endblock %}
```

B.2.5 Users

This folder contains all content available to users.

B.2.5.1 __init__.py

Empty file to allow for import.

B.2.5.2 Routes.py

Just as its non-users counterpart, this file contains all functions and paths available to users and is constructed in the exact same manner.

```
from os import listdir
from zipfile import ZipFile

import pandas
from flask import (Blueprint, redirect, render_template, request,
                  send_file, url_for)
from flask_login import (current_user, login_required, login_user,
                          logout_user)
```

Safety and quality based traceability system for food supply chains

```
from RegulatorServer.Models import User
from RegulatorServer.Users.MasterFunctions.Login import
ValidateCredentials
from RegulatorServer.Users.MasterFunctions.QATMaster import QATMaster
from RegulatorServer.Users.MasterFunctions.QVMMaster import QVMMaster

Templates = '/home/joao/Desktop/Python/Regulator Layer/RegulatorServer/
Users/Templates'

Static = '/home/joao/Desktop/Python/Regulator Layer/RegulatorServer/
Users/Static'

Users = Blueprint('Users',
                  __name__,
                  template_folder=Templates,
                  static_folder=Static)

@Users.route('/Login')
def Login():

    if request.method == 'POST':
        Username = request.form['Username']
        Password = request.form['Password']

        Credentials = ValidateCredentials(Username, Password)

        if Credentials == True:
            login_user(User(Username))
            return redirect(url_for('Users.Personal'))

    return render_template('Login.html',
                           title='Login')

@Users.route('/Personal')
@login_required
def Personal():

    return render_template('Personal.html',
                           title='Personal')

@Users.route('/QVM')
@login_required
def QVM():
```

```

if 'Step' not in request.form:

    return render_template('QVM.html',
                           title='Quality Validation',
                           Step='Step1')

elif request.form['Step'] == 'Step2':

    IDC = request.files['IDC']
    IFC = request.files['IFC']

    FileName = QVMMaster(IDC, IFC)

# Dummy function to send files to UCGM
# Dummy function to receive files from UCGM

IDCPath = '/home/joao/Desktop/Python/Regulator Layer/Docs/
ValidPostUC/PostValUCIDC'
IFCPath = '/home/joao/Desktop/Python/Regulator Layer/Docs/
ValidPostUC/PostValUCIFC'

ZipDir = '/home/joao/Desktop/Python/Regulator Layer/Docs/Zip'

with ZipFile(ZipDir + '/' + FileName + '_Docs.zip', mode='w') as
ZipDocs:
    ZipDocs.write(IDCPath + '/' + FileName + '_IDC.csv')
    ZipDocs.write(IFCPath + '/' + FileName + '_IFC.csv')

    send_file(ZipDir + '/' + FileName + '_Docs.zip',
              as_attachment=True)

return render_template('QVM.html',
                       title='Quality Validation',
                       Step='Step2',
                       FileName=FileName)

@Users.route('/QAT')
@login_required
def QAT():

    if 'Step' not in request.form:

        return render_template('QAT.html',
                               title='Quality After Transport',
                               Step='Step1')

```

```
elif request.form['Step'] == 'Step2':

    global NumberOfProducts
    NumberOfProducts = request.form['NumberOfProducts']

    return render_template('QAT.html',
                           title='Quality After Transport',
                           Step='Step2',
                           NumberOfProducts=NumberOfProducts)

elif request.form['Step'] == 'Step3':

    IDs = request.form.getlist(IDs, type=str)
    QMPs = request.form.getlist(QMPs, type=str)
    ExpectedKeepingQuality = request.form.getlist(
        ExpectedKeepingQuality, type=str)

    DeltaFrame = QATMaster(IDs, QMPs, ExpectedKeepingQuality)

    return render_template('QAT.html',
                           title='Quality After Transport',
                           Step='Step3',
                           tables=[DeltaFrame.to_html(index=False)])

@Users.route('/Logout')
@login_required
def Logout():

    logout_user()

    return redirect(url_for('NonUsers.Home'))
```

B.2.5.3 MasterFunctions

Folder containing all functions necessary for back end processing.

__init__.py

Empty file to allow for import.

Login.py

This file is very similar to OMM's Login.py and will not be described.

QATMaster.py

This file calls all slave functions necessary for the correct functioning of the Quality After Transport module. This function works by joining two partial DataFrames.

```

import pandas as pd
from RegulatorServer.Users.SlaveFunctions.FindData import FindData
from RegulatorServer.Users.SlaveFunctions.DetermineQuality import
DetermineQuality

def QATMaster(IDs, QMPs, ExpectedKeepingQuality):

DeltaFrameP1 = FindData(IDs)

DeltaFrame = DetermineQuality(DeltaFrameP1,
                              QMPs,
                              ExpectedKeepingQuality)

return DeltaFrame

```

QVMMaster.py

This function is responsible for validating products for transaction. As there was no files sent between serves, which is acceptable for prototyping. All functions that would perform that task were replaced by comments indicating their position in the code.

```

from pandas import read_csv
from RegulatorServer.Users.SlaveFunctions.CompareColumns import
CompareColumns
from RegulatorServer.Users.SlaveFunctions.VerifyQuality import
VerifyQuality
from RegulatorServer.Users.SlaveFunctions.SaveFiles import SaveFiles

def QVMMaster(IDC, IFC):

    IDC = read_csv(IDC)
    IFC = read_csv(IFC)

    IDC, IFC = CompareColumns(IDC, IFC)

    IDC, IFC = VerifyQuality(IDC, IFC)

    FileName = SaveFiles(IDC, IFC)

    # Send files to the unique code layer
    # Dummy function

    # Receive files from the unique code layer
    # Dummy function

    return FileName

```

Safety and quality based traceability system for food supply chains

B.2.5.4 SlaveFunctions

This folder contains all necessary sub-functions. These functions are called by their masters and perform all partial tasks.

`__init__.py`

Empty file to allow for import.

`CompareColumns.py`

This file compares the information contained in commonly named columns between the IDC and IFC files to ensure that no discrepancies occur between direct and final consumers.

```
import pandas as pd

def CompareColumns(IDC, IFC):

    IFCColumns = list(IFC.columns)

    IDCColumns = list(IDC.columns)

    CommonColumns = list(set(IFCColumns).intersection(IDCColumns))

    for col in CommonColumns:
        for row in len(IFC.index):
            if IFC.at[row, 'MARKED FOR VALIDATION'] ==
                'Marked For Validation':
                if IFC.at[row, col] == IDC.at[row, col]:
                    IFC.at[row, 'MARKED FOR VALIDATION'] = 'VALID'
                    IDC.at[row, 'MARKED FOR VALIDATION'] = 'VALID'
                else:
                    IFC.at[row, 'MARKED FOR VALIDATION'] = 'INVALID'
                    IDC.at[row, 'MARKED FOR VALIDATION'] = 'INVALID'

    return IDC, IFC
```

`FindData.py`

This file creates the first partial DataFrame necessary for the QATMaster. Consists in searching for already existing information about any given product.

```
from os import listdir
import pandas as pd

def FindData(IDs):

    IDs = list(IDs.split())
```



```

IFCDir = '/home/joao/Desktop/Python/Regulator Layer/Docs/
ValidPostUC/PostValUCIFC'

Columns = ['ID',
           'Initial Quality',
           'EaR',
           'Quality Limit',
           'Last Update',
           'Last Update Quality',
           'Last Update Quality Algorithm',
           'Last Update Keeping Quality']

DeltaFrameP1 = pd.DataFrame(data=[], columns=Columns)

ListLU = []
ListLUQ = []
ListLUQA = []
ListLUKQ = []
ListInitialQuality = []
ListEaR = []
ListQualityLimit = []

for ID in IDs:
    for ifc in listdir(IFCDir):
        IFC = pd.read_csv(ifc)
        for row in range(len(IFC.index)):

            if IFC.at[row, 'MARKED FOR VALIDATION'] == ID:
                LU = IFC.at[row, 'Last Update']
                LUQ = IFC.at[row, 'Last Update Quality']
                LUQA = IFC.at[row, 'Last Update Quality Algorithm']
                LUKQ = IFC.at[row, 'Last Update Keeping Quality']
                IQ = IFC.at[row, 'Initial Quality']
                EaR = IFC.at[row, 'EaR']
                Qlim = IFC.at[row, 'Quality Limit']

ListLU.append(LU)
ListLUQ.append(LUQ)
ListLUQA.append(LUQA)
ListLUKQ.append(LUKQ)
ListInitialQuality.append(IQ)
ListEaR.append(EaR)
ListQualityLimit.append(Qlim)

for i in range(len(IDs)):
    ID = IDs[i]
    LU = ListLU[i]

```

Safety and quality based traceability system for food supply chains

```
LUQ = ListLUQ[i]
LUQA = ListLUQA[i]
LUKQ = ListLUKQ[i]
IQ = ListInitialQuality[i]
EaR = ListEaR[i]
Qlim = ListQualityLimit[i]
row = [ID, LU, LUQ, LUQA, LUKQ, IQ, EaR, Qlim]
DeltaFrameP1.append(row)

return DeltaFrameP1
```

DetermineQuality.py

This file makes the second partial DataFrame and joins it with the first part. This second part calculates quality at arrival and presents it along with the quality difference between what was received and what was promised.

```
import pandas as pd
import datetime
from math import exp

def DetermineQuality(DeltaFrameP1, QMPs, ExpectedKeepingQuality):

    QMPs = list(QMPs.split())

    ExpectedKeepingQuality = list(ExpectedKeepingQuality.split())

    Columns = ['QMP',
               'Calculated Quality',
               'Quality Delta',
               'Expected Keeping Quality',
               'Calculated Keeping Quality',
               'Keeping Quality Delta']
    DeltaFrameP2 = pd.DataFrame(data=[], columns=Columns)

    for row in range(len(DeltaFrameP1.index)):
        if DeltaFrameP1.at[row, 'Last Update Quality Algorithm'] ==
            'EQAID 1':
            EaR = DeltaFrameP1.at[row, 'EaR']
            Qlim = DeltaFrameP1.at[row, 'Quality Limit']
            Q0 = DeltaFrameP1.at[row, 'Initial Quality']
            t1 = DeltaFrameP1.at[row, 'Last Update']
            t1 = datetime.datetime.strptime(t1, '%Y-%m-%d %H:%M:%S')
            t2 = datetime.datetime.now().replace(microsecond=0)
            t = abs(t1 - t2).total_seconds()/3600
            QMP = QMPs[row, 'Last Update QMP']
            ExpectedKeepingQuality = ExpectedKeepingQuality[row,
```

```

'Last Update Keeping Quality']

k = 1.0 * exp(EaR * ((1 / 283.15) - (1 / (QMP + 273.15))))
Q = Q0 - k * t
DeltaQ = Q - DeltaFrameP1.at[row, 'Last Update Quality']
KQ = (Q0 - Qlim) / k
DeltaKQ = KQ - ExpectedKeepingQuality

row = [QMP, Q, DeltaQ, ExpectedKeepingQuality, KQ, DeltaKQ]

DeltaFrameP2.append(row)

elif DeltaFrameP1.at[row, 'Last Update Quality Algorithm'] ==
'EQAID 2':
    EaR = DeltaFrameP1.at[row, 'EaR']
    Qlim = DeltaFrameP1.at[row, 'Quality Limit']
    Q0 = DeltaFrameP1.at[row, 'Initial Quality']
    t1 = DeltaFrameP1.at[row, 'Last Update']
    t1 = datetime.datetime.strptime(t1, '%Y-%m-%d %H:%M:%S')
    t2 = datetime.datetime.now().replace(microsecond=0)
    t = abs(t1 - t2).total_seconds()/3600
    QMP = QMPs[row, 'Last Update QMP']
    ExpectedKeepingQuality = ExpectedKeepingQuality[row,
'Last Update
Keeping Quality']

    k = 1.0 * exp(EaR * ((1 / 283.15) - (1 / (QMP + 273.15))))
    Q = Q0 - k * t
    DeltaQ = Q - DeltaFrameP1.at[row, 'Last Update Quality']
    KQ = (Q0 - Qlim) / k
    DeltaKQ = KQ - ExpectedKeepingQuality

    row = [QMP, Q, DeltaQ, ExpectedKeepingQuality, KQ, DeltaKQ]

    DeltaFrameP2.append(row)

elif DeltaFrameP1.at[row, 'Last Update Quality Algorithm'] ==
'EQAID 3':
    EaR = DeltaFrameP1.at[row, 'EaR']
    Qlim = DeltaFrameP1.at[row, 'Quality Limit']
    Q0 = DeltaFrameP1.at[row, 'Initial Quality']
    t1 = DeltaFrameP1.at[row, 'Last Update']
    t1 = datetime.datetime.strptime(t1, '%Y-%m-%d %H:%M:%S')
    t2 = datetime.datetime.now().replace(microsecond=0)
    t = abs(t1 - t2).total_seconds()/3600
    QMP = QMPs[row, 'Last Update QMP']
    ExpectedKeepingQuality = ExpectedKeepingQuality[row,

```

```

    'Last Update Keeping Quality']

    k = 1.0 * exp(EaR * ((1 / 283.15) - (1 / (QMP + 273.15))))
    Q = Q0 - k * t
    DeltaQ = Q - DeltaFrameP1.at[row, 'Last Update Quality']
    KQ = (Q0 - Qlim) / k
    DeltaKQ = KQ - ExpectedKeepingQuality

    row = [QMP, Q, DeltaQ, ExpectedKeepingQuality, KQ, DeltaKQ]

    DeltaFrameP2.append(row)

elif DeltaFrameP1.at[row, 'Last Update Quality Algorithm'] ==
'EQAID 4':
    EaR = DeltaFrameP1.at[row, 'EaR']
    Qlim = DeltaFrameP1.at[row, 'Quality Limit']
    Q0 = DeltaFrameP1.at[row, 'Initial Quality']
    t1 = DeltaFrameP1.at[row, 'Last Update']
    t1 = datetime.datetime.strptime(t1, '%Y-%m-%d %H:%M:%S')
    t2 = datetime.datetime.now().replace(microsecond=0)
    t = abs(t1 - t2).total_seconds()/3600
    QMP = QMPs[row]
    ExpectedKeepingQuality = ExpectedKeepingQuality[row,
    'Last Update Keeping Quality']

    k = 1.0 * exp(EaR * ((1 / 283.15) - (1 / (QMP + 273.15))))
    Q = Q0 - k * t
    DeltaQ = Q - DeltaFrameP1.at[row, 'Last Update Quality']
    KQ = (Q0 - Qlim) / k
    DeltaKQ = KQ - ExpectedKeepingQuality

    row = [QMP, Q, DeltaQ, ExpectedKeepingQuality, KQ, DeltaKQ]

    DeltaFrameP2.append(row)

elif DeltaFrameP1.at[row, 'Last Update Quality Algorithm'] ==
'EQAID 5':
    EaR = DeltaFrameP1.at[row, 'EaR']
    Qlim = DeltaFrameP1.at[row, 'Quality Limit']
    Q0 = DeltaFrameP1.at[row, 'Initial Quality']
    t1 = DeltaFrameP1.at[row, 'Last Update']
    t1 = datetime.datetime.strptime(t1, '%Y-%m-%d %H:%M:%S')
    t2 = datetime.datetime.now().replace(microsecond=0)
    t = abs(t1 - t2).total_seconds()/3600
    QMP = QMPs[row]
    ExpectedKeepingQuality = ExpectedKeepingQuality[row,
    'Last Update Keeping Quality']

```

```

k = 1.0 * exp(EaR * ((1 / 283.15) - (1 / (QMP + 273.15))))
Q = Q0 - k * t
DeltaQ = Q - DeltaFrameP1.at[row, 'Last Update Quality']
KQ = (Q0 - Qlim) / k
DeltaKQ = KQ - ExpectedKeepingQuality

row = [QMP, Q, DeltaQ, ExpectedKeepingQuality, KQ, DeltaKQ]

DeltaFrameP2.append(row)

DeltaFrame = DeltaFrameP1.join(DeltaFrameP2)

return DeltaFrame

```

SaveFiles.py

This file is responsible for saving the IFC and IDC files into the regulator layer.

```

import pandas as pd
from string import ascii_uppercase, digits
import random

def RandomCodeGenerator():
    Size = 8
    Characters = ascii_uppercase + digits
    Code = ''.join(random.choice(Characters) for char in range(Size))

    return Code

def SaveFiles(IDC, IFC):

    FileName = RandomCodeGenerator()

    IDCSaveDir = '/home/joao/Desktop/Python/Regulator Layer/Docs/
ValidPreUC/PreValUCIDC'
    IFCSaveDir = '/home/joao/Desktop/Python/Regulator Layer/Docs/
ValidPreUC/PreValUCIFC'

    IDC.to_csv(IDCSaveDir + '/' + FileName, index=False)
    IFC.to_csv(IFCSaveDir + '/' + FileName, index=False)

    return FileName

```

VerifyQuality.py

This file is responsible for recalculating quality and returning any marked product as valid or invalid if the result matches the information given.

Safety and quality based traceability system for food supply chains

```
import pandas as pd
import datetime
from math import exp

def VerifyQuality(IDC, IFC):

    for row in range(len(IFC.index)):

        if IFC.at[row, 'MARKED FOR VALIDATION'] == 'Marked For
Validation ':

            if IFC.at[row, 'Initial Quality Algorithm'] == 'EQAID 1':
                EaR = IFC.at[row, 'EaR']
                Qlim = IFC.at[row, 'Quality Limit']
                QMP = IFC.at[row, 'Last Update QMP']
                Q0 = IFC.at[row, 'Initial Quality']
                t1 = IFC.at[row, 'Entry Date']
                t1 = datetime.datetime.strptime(t1,
                                                '%Y-%m-%d %H:%M:%S')
                t2 = IFC.at[row, 'Last Update']
                t2 = datetime.datetime.strptime(t2,
                                                '%Y-%m-%d %H:%M:%S')
                t = abs(t2-t1).total_seconds()/3600
                k = 1.0 * exp(EaR * ((1 / 283.15)
- (1 / (QMP + 273.15))))

                Q = Q0 - k * t
                KQ = (Q0 - Qlim) / k
                if Q != IFC.at[row, 'Last Update Quality'] or
                KQ != IFC.at[row, 'Last Update Keeping Quaity']:

                    IFC.at[row, 'MARKED FOR VALIDATION'] == 'INVALID'

            elif IFC.at[row, 'Initial Quality Algorithm'] == 'EQAID 2':
                EaR = IFC.at[row, 'EaR']
                Qlim = IFC.at[row, 'Quality Limit']
                QMP = IFC.at[row, 'Last Update QMP']
                Q0 = IFC.at[row, 'Initial Quality']
                t1 = IFC.at[row, 'Entry Date']
                t1 = datetime.datetime.strptime(t1,
                                                '%Y-%m-%d %H:%M:%S')
                t2 = IFC.at[row, 'Last Update']
                t2 = datetime.datetime.strptime(t2,
                                                '%Y-%m-%d %H:%M:%S')
                t = abs(t2-t1).total_seconds()/3600
```

```

k = 1.0 * exp(EaR * ((1 / 283.15)
- (1 / (QMP + 273.15))))

KQ = (Q0 - Qlim) / k
Q = Q0 - k * t
KQ = (Q0 - Qlim) / k
if Q != IFC.at[row, 'Last Update Quality']
or KQ != IFC.at[row, 'Last Update Keeping Quaity']:

    IFC.at[row, 'MARKED FOR VALIDATION'] == 'INVALID'

elif IFC.at[row, 'Initial Quality Algorithm'] == 'EQAID 3':
    EaR = IFC.at[row, 'EaR']
    Qlim = IFC.at[row, 'Quality Limit']
    QMP = IFC.at[row, 'Last Update QMP']
    Q0 = IFC.at[row, 'Initial Quality']
    t1 = IFC.at[row, 'Entry Date']
    t1 = datetime.datetime.strptime(t1,
                                     '%Y-%m-%d %H:%M:%S')
    t2 = IFC.at[row, 'Last Update']
    t2 = datetime.datetime.strptime(t2,
                                     '%Y-%m-%d %H:%M:%S')
    t = abs(t2-t1).total_seconds()/3600
    k = 1.0 * exp(EaR * ((1 / 283.15)
- (1 / (QMP + 273.15))))

    KQ = (Q0 - Qlim) / k
    Q = Q0 - k * t
    KQ = (Q0 - Qlim) / k
    if Q != IFC.at[row, 'Last Update Quality']
    or KQ != IFC.at[row, 'Last Update Keeping Quaity']:

        IFC.at[row, 'MARKED FOR VALIDATION'] == 'INVALID'

elif IFC.at[row, 'Initial Quality Algorithm'] == 'EQAID 4':
    EaR = IFC.at[row, 'EaR']
    Qlim = IFC.at[row, 'Quality Limit']
    QMP = IFC.at[row, 'Last Update QMP']
    Q0 = IFC.at[row, 'Initial Quality']
    t1 = IFC.at[row, 'Entry Date']
    t1 = datetime.datetime.strptime(t1,
                                     '%Y-%m-%d %H:%M:%S')
    t2 = IFC.at[row, 'Last Update']
    t2 = datetime.datetime.strptime(t2,
                                     '%Y-%m-%d %H:%M:%S')
    t = abs(t2-t1).total_seconds()/3600
    k = 1.0 * exp(EaR * ((1 / 283.15)

```

```
- (1 / (QMP + 273.15))))

KQ = (Q0 - Qlim) / k
Q = Q0 - k * t
KQ = (Q0 - Qlim) / k
if Q != IFC.at[row, 'Last Update Quality']
or KQ != IFC.at[row, 'Last Update Keeping Quaity']:

    IFC.at[row, 'MARKED FOR VALIDATION'] == 'INVALID'

if IFC.at[row, 'Initial Quality Algorithm'] == 'EQAID 5':
    EaR = IFC.at[row, 'EaR']
    Qlim = IFC.at[row, 'Quality Limit']
    QMP = IFC.at[row, 'Last Update QMP']
    Q0 = IFC.at[row, 'Initial Quality']
    t1 = IFC.at[row, 'Entry Date']
    t1 = datetime.datetime.strptime(t1,
                                    '%Y-%m-%d %H:%M:%S')
    t2 = IFC.at[row, 'Last Update']
    t2 = datetime.datetime.strptime(t2,
                                    '%Y-%m-%d %H:%M:%S')

    t = abs(t2 - t1).total_seconds()/3600
    k = 1.0 * exp(EaR * ((1 / 283.15)
                        - (1 / (QMP + 273.15))))

    KQ = (Q0 - Qlim) / k
    Q = Q0 - k * t
    KQ = (Q0 - Qlim) / k
    if Q != IFC.at[row, 'Last Update Quality']
    or KQ != IFC.at[row, 'Last Update Keeping Quaity']:

        IFC.at[row, 'MARKED FOR VALIDATION'] == 'INVALID'

return IDC, IFC
```

B.2.5.5 Static

Again, this folder contains the `Style.css` file responsible for the aesthetic of the content available to users. As it is very similar to its non-users' counterpart it does not need exposition.

B.2.5.6 Templates

Just as the non-users' counterpart, it contains the files that present the content to the users.

Layout.html

This file has the same purpose of its non-user counterpart and is made similar, thus not needing further exposition.

Login.html

Similar to Login.py files already shown. Further exposition is unnecessary.

Personal.html

This file represents a personal page in the regulator layer. In this case the "person" is an entire company. As this file is similar to OMM's Personal.html it does not need further exposition.

QAT.html

This file presents the Quality After Transport module and consists in a multi-step form that asks users for QMP inputs and returns a result table.

```
{% extends 'Layout.html' %}

{% block content %}

    {% if Step == 'Step1' %}
    <form action='{ { url_for('User.QAT') } }' method='POST'>
        <fieldset>
            <legend> Quality After Transport </legend>
            <br>
            Number of products:
            <input type='text' name='NumberOfProducts'
                placeholder='Ex: 3' required>
            <input type='hidden' name='Step' value='Step2'>
            <input type='submit' value='Next'>
        </fieldset>
    </form>

    {% elif Step == 'Step2' %}
    <form action='{ { url_for('User.QAT') } }' method='POST'>
        <fieldset>
            <legend> Quality After Transport </legend>
            {% for i in range(0, NumberOfProducts, 1) %}
                <input type='text' name='IDs' placeholder='ID'
                    required>
                <input type='text' name='QMPs' placeholder='QMP'
                    required>
                <input type='text' name='ExpectedKeepingQuality'
                    placeholder='100 hours' required>
            {% endfor %}
            <input type='hidden' name='Step' value='Step3'>
            <input type='submit' value='Verify Quality'>
        </fieldset>
    </form>

    {% elif Step == 'Step3' %}
        {% for table in tables %}
```

Safety and quality based traceability system for food supply chains

```
        {{ table|safe }}
    {% endfor %}

{% endif%}

{% endblock %}
```

B.3 Docs

This folder represents the RFIC and contains all relevant files plus an extra folder to allow for files to be downloaded together in a .zip file.

Appendix C

Unique code layer code snippets

This module is composed by one file that modifies the IDC and IFC files as necessary. Once again, the first lines correspond to the necessary imports.

```
import csv
import random
from os import listdir, path
from string import ascii_uppercase, digits

import pandas as pd
```

A random code generator is created in the same manner as the one already presented. As such, does not need further exposition.

Then the files are opened in pairs and modified accordingly using the code bellow.

```
for csvfile in listdir(PreValDirIFC):
    IDC = pd.read_csv(PreValDirIDC + '/' + csvfile)
    IFC = pd.read_csv(PreValDirIFC + '/' + csvfile)

for row in range(len(IFC.index)):

    ValidationCode = RandomCodeGenerator()

    if IFC.at[row, 'MARKED FOR VALIDATION'] == 'VALID':
        IDC.at[row, 'MARKED FOR VALIDATION'] = ValidationCode
        IFC.at[row, 'MARKED FOR VALIDATION'] = ValidationCode
```

