DEPARTAMENTO DE ENGENHARIA
INFORMÁTICA

# Gorgeous Food

ARQSOFT

2019 / 2020

**Grupo 204**

**Filipe Ferreira | 1160826**

**Pedro Ferreira | 1140953**

**Pedro Santos | 1120653**

isep Instituto Superior de Engenharia do Porto

# Índice

# 1 Introduction

This paper describes the proposed decomposition of a system called Gorgeous Food into microservices. This chapter presents the contextualization of the analyzes performed as well as the information produced during the decomposition process.

# 2 Decomposition

For this project were used tree types of strategies for decomposition into microservices, Business Capability Decomposition, Decomposition with Service Cutter Tool and Business Subdomain Decomposition. The results of this analysis are explicit in the following subtopics.

## 2.1 Business Capability Decomposition

The business capability pattern focuses on the identification of what the business does. This decomposition helps to identify features/processes that may be converted to microservices.

The business capability is a description of what a business does, independently of how or why.

After the analyses of the company objectives and structure, five business capabilities were identified, **Inventory Management**, **Meal Management**, **Descriptor Creation**, **Report Generation** and **POS Management**. They are explicit in the following tables.

| | |
|---|---|
| ID | C1 |
| Type | Inventory Management |
| Definition | The company manages the inventory of their meals. It can add and update the meals in the inventory. Also includes the POS (point of sale) id. |
| Inputs | 1. A meal is produced, and a member of the staff adds that meal to the inventory. 2. A sale occurs and the specific item is updated. 3. The item reaches its expiration date, so it is updated/removed. |
| Outputs | |
| Artefacts | UC1-4, P2D4 |

| | |
|---|---|
| ID | C2 |
| Type | Meal Management |
| Definition | The company provides information about the ingredients, allergens and nutrition data for the available meals. |
| Inputs | 1. A user requests information about a meal. |
| Outputs | 1. A list of information about the meal is provided, including the ingredients, allergens and nutrition. |
| Artefacts | P2D2, P2D5 |

| | |
|---|---|
| ID | C3 |
| Type | Descriptor Creation |
| Definition | A meal descriptor formula is used to generate the meal descriptor. The company has the capability to change/update that formula |
| Inputs | 1. A admin/staff updates the formula for the meal descriptor. |
| Outputs | 1. The new formula is adopted in the creation of meals. |
| Artefacts | UC6-7, UC9 |

| | |
|---|---|
| ID | C4 |
| Type | Report Generation |
| Definition | The company creates reports about the several business activities. |
| Inputs | 1. A report is requested with specific parameters (date, items, meals, sales, etc) |
| Outputs | 1. A report is generated and made available. |
| Artefacts | UC8, UC12 |

| | |
|---|---|
| ID | C5 |
| Type | POS Management |
| Definition | The company as several POS that have different locations and have different stats. |
| Inputs | 1. A admin/staff updates the POS location. 2. A admin/staff adds a new POS. 3. A ServiceUser consults the near POS. |
| Outputs | 1. A POS location is made available. |
| Artefacts | P2D4 |

## 2.2 Decomposition with Service Cutter Tool

This tool is a prototype that offers a 16 coupling criterion and uses these rules to propose a decomposition of a system into microservices. The tool uses well-established artefacts of the system, like domain models and use cases[1].

After the installation of the tool two files are needed to express the current system.

A *json* model that has the system domain, data, operations, and artefacts and a user representation file that expresses the use case and relations of the system entities and responsible roles.

The tool as two algorithms for decomposition. The Leung and Girvan-Newman algorithms, and the results of the application of these are in the Figure 1 and Figure 2.

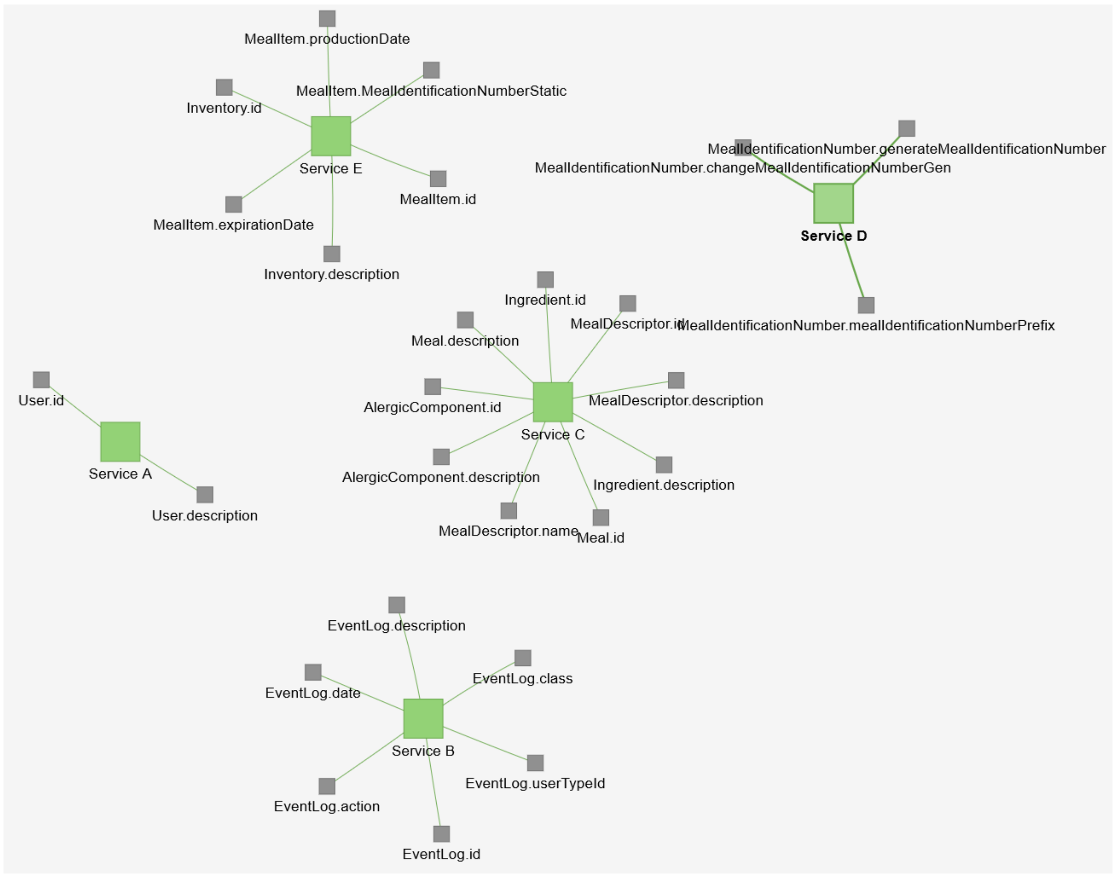

*Figure 1 – Girvan-Newman algorithm*

The Girvan-Newman algorithm suggests five microservices for the system.

**Service A->** for the user management;

**Service B->** for the log management;

**Service C ->** for the meal management;

**Service D ->** for the meal descriptor management;

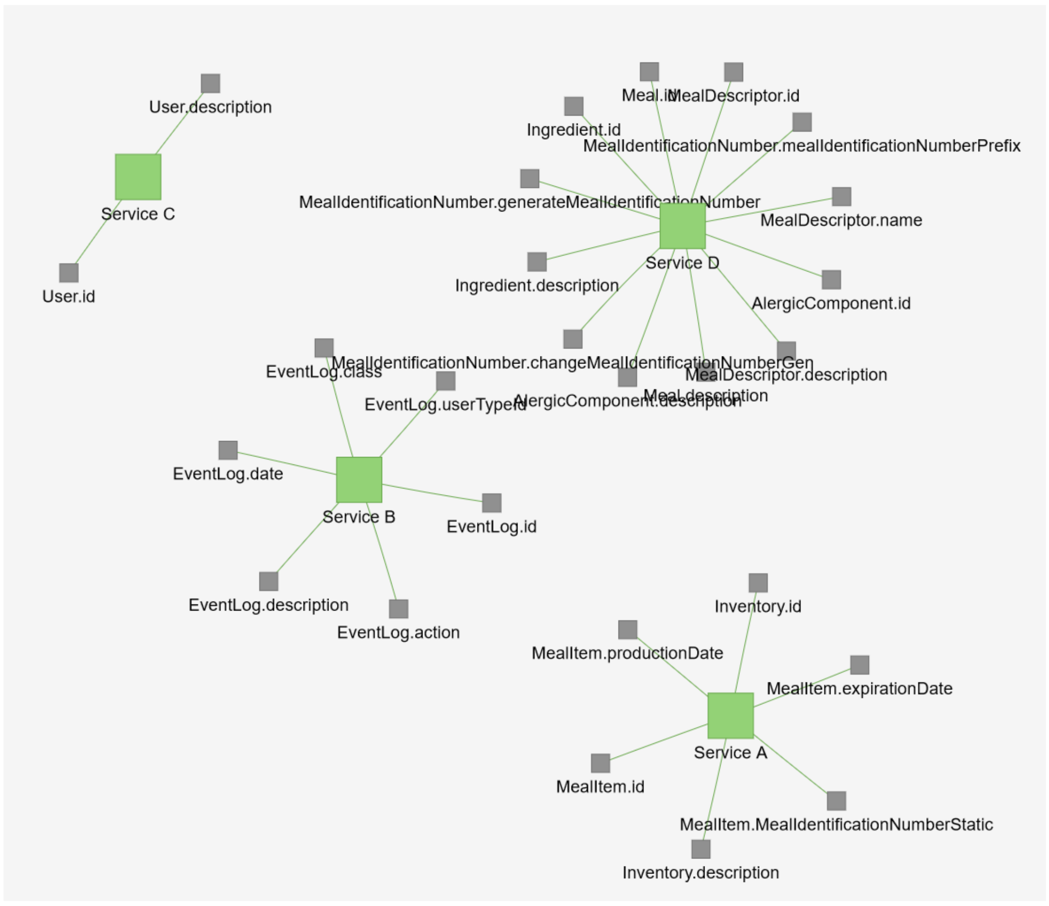

*Figure 2 - Leung algorithm.*

The Leung algorithm suggests four microservices for the system.

**Service A ->** for inventory management.

**Service B ->** for the log management.

**Service C ->** for the user management.

**Service D ->** for the meal and descriptor management.

## 2.3 Business Subdomain Decomposition

The subdomain decomposition pattern is useful to compare their results to the previous decompositions, so a better general decomposition of the system can be achieved. This decomposition consists in the identification of the bounded contexts of the system and judge the connection between them. Then the identified bounded contexts can be considered microservices. For the system it were identified six bounded contexts, that are explicit in the next.

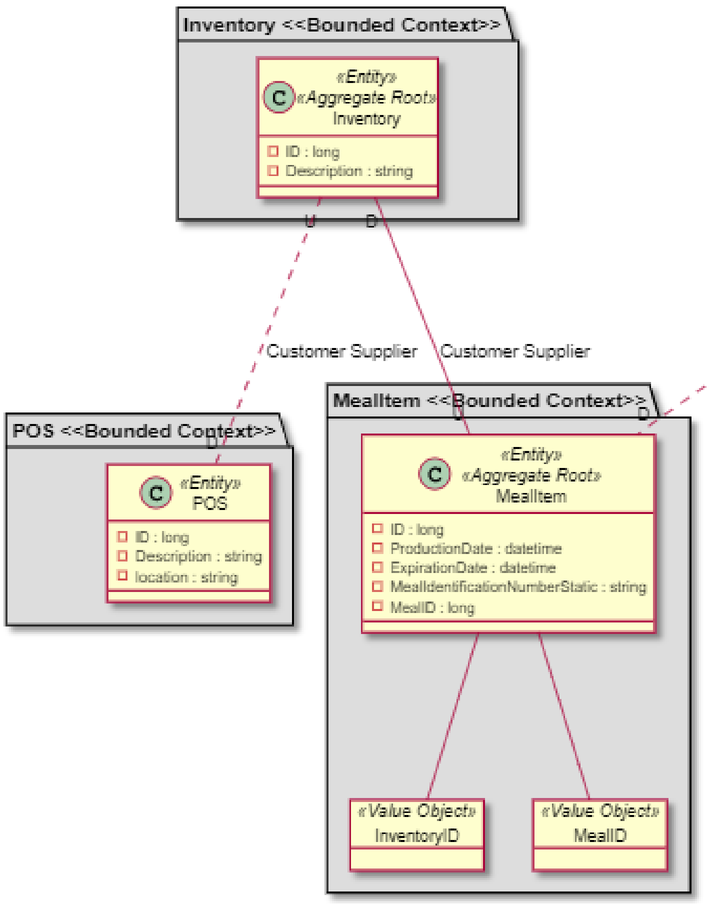

*Figure 3 - Inventory, POS and MealItem bounded contexts*

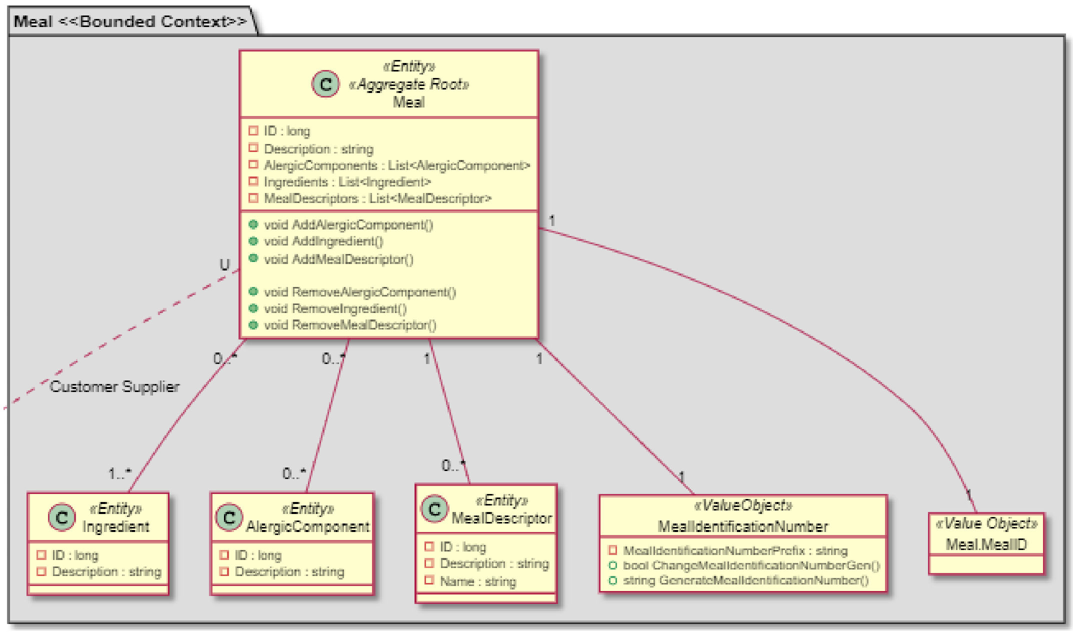

*Figure 4 - Meal Bounded Context*

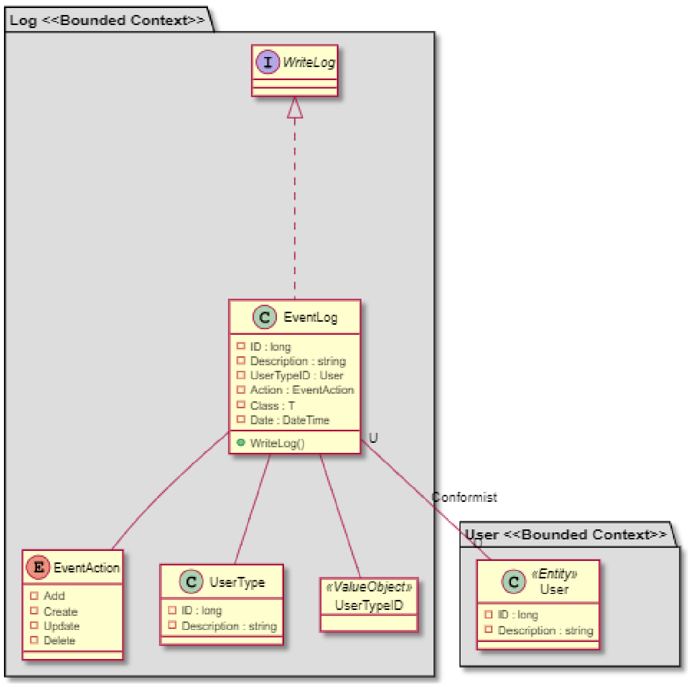

*Figure 5 - Log and User Bounded Context*

# 3 Used Technologies

This chapter describes the technologies applied in this project as well as technologies considered interesting and strong possibilities.

All technologies used in the project development are ‘open-source’. This allow it’s use without implying any licence payment.

| Component | Technology |
|---|---|
| **Gorgeous Food User Interface** | React |
| **Gorgeous Food Application Programming Interface** | .NET Core 3.1 + Entity Framework Core |
| **Meal Database** | Sql Server 2019 |
| **IDE** | Visual Studio Code |
| **Other** | Windows terminal + Microsoft Powershell |
| **Decomposition Tool** | Service Cutter |
| **Container Platform** | Docker |

*Table 1 - Project used technologies*

# 4 *Attribute-Driven Design* (ADD) P2

## 4.1 Interaction 1

This iteration’s objective was recreating the previously established architecture. This time since we are operating in an already develop project, it is considered ‘Brownfield’.

This time, the main objective was, besides adding some new functionalities, adopting a new software architecture called ‘Microservice Architecture’.

Microservice architecture implies deep planning and restructure of the whole system since we need to migrate a Monolithic system to a complete distributed system, where each piece has it’s own concerns and responsibilities as well as lack of dependencies to other services.

Since the team’s knowledge on that area was a bit limited, we based of on the example provided by Microsoft called ‘eShopOnContainers’ (Figure 3).

This example contains innumerous ways and best practices when applying and developing software with this architecture.

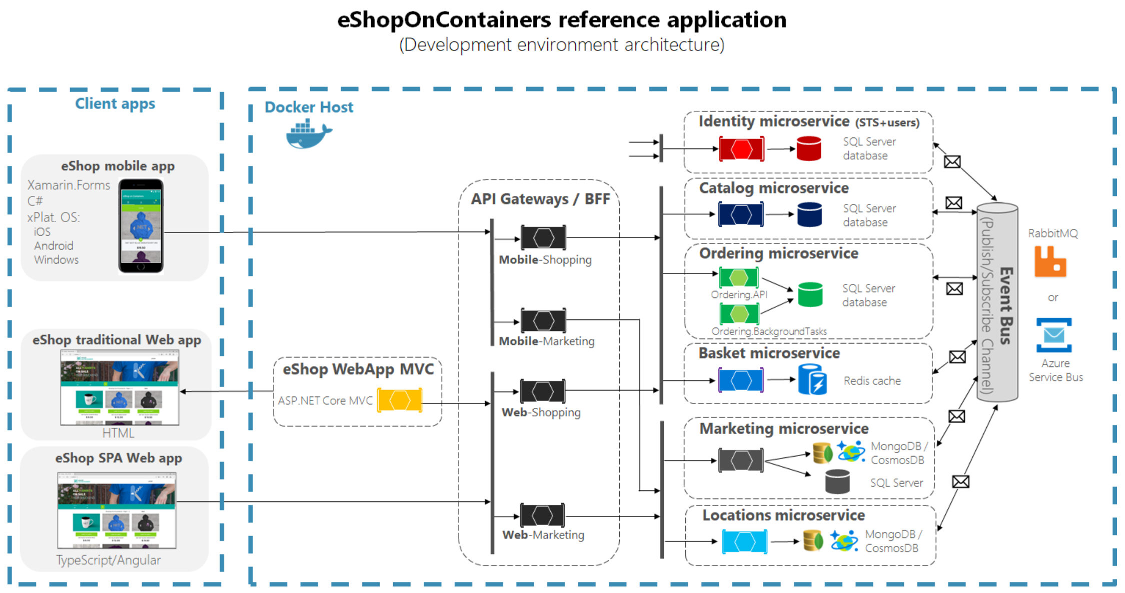

Figure 6 – Diagram representing the Microservice architecture present in Microsoft’s example project [2].

The team started by dividing and reevaluating the entities present in the before monolithic API called ‘GorgeousFood.API’. This resulted in two smaller contexts called ‘Meal Bounded Context’ and ‘MealItem Bounded Context’.

Afterwards we saw the dependencies between those two bounded contexts. Since the 'MealItem B.C.' has a deep link with the Meal entity, we decided to apply the Gateway Pattern. Gateway pattern involves a gateway that operates between the UI and the numerous microservices that are present in the system. This pattern also allow us to accomplish the preestablished rules present in the microservices architecture, namely the segregation principle that says that microservices cannot have direct dependency between each other.

In the end the team came up with the following architecture:

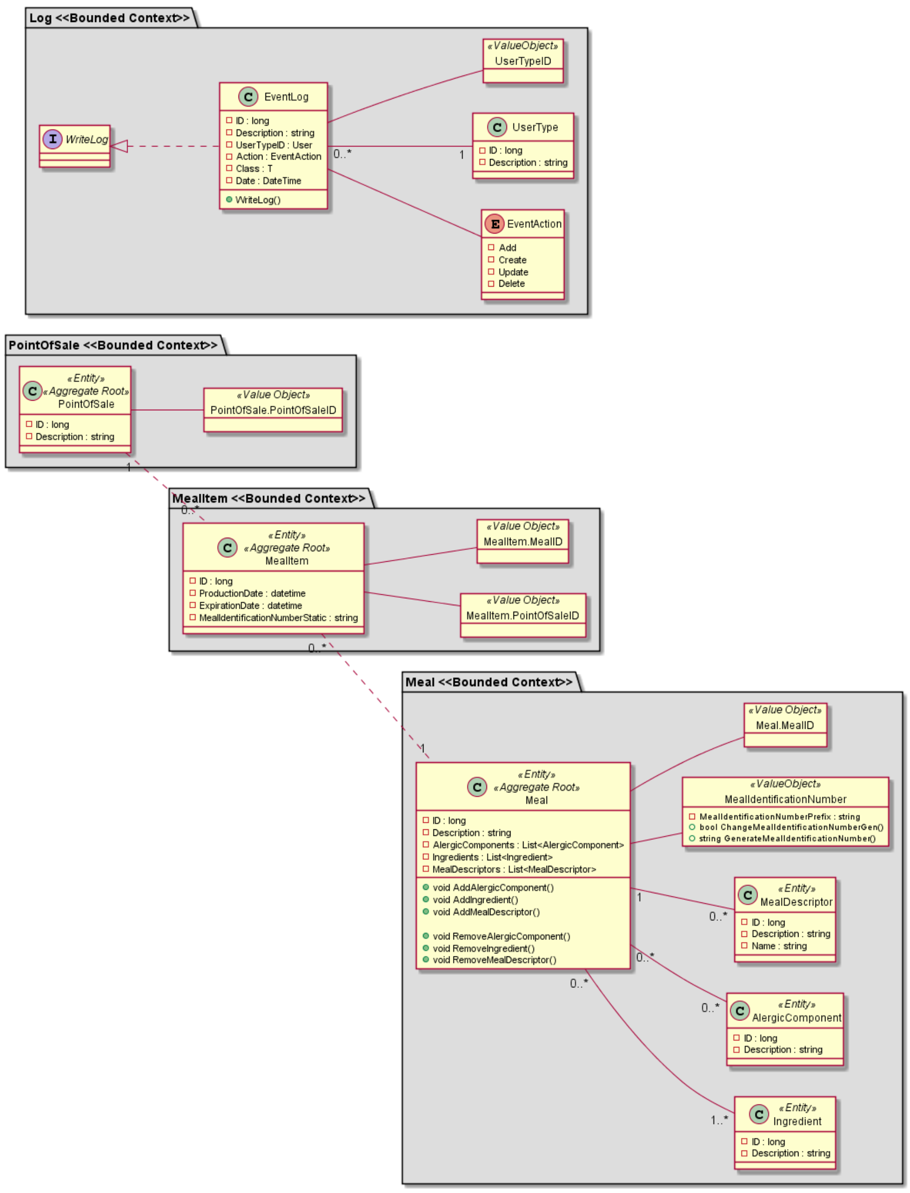
Log <<Bounded Context>>
WriteLog
EventLog
ID : long
Description : string
UserTypeID : User
Action : EventAction
Class : T
Date : DateTime
WriteLog()
«ValueObject» UserTypeID
UserType
ID : long
Description : string
0..*
1
EventAction
Add
Create
Update
Delete
PointOfSale <<Bounded Context>>
«Entity» «Aggregate Root» PointOfSale
ID : long
Description : string
«Value Object» PointOfSale.PointOfSaleID
1
MealItem <<Bounded Context>>
0..*
«Entity» «Aggregate Root» MealItem
ID : long
ProductionDate : datetime
ExpirationDate : datetime
MealIdentificationNumberStatic : string
«Value Object» MealItem.MealID
«Value Object» MealItem.PointOfSaleID
0..*
Meal <<Bounded Context>>
1
«Value Object» Meal.MealID
«Entity» «Aggregate Root» Meal
ID : long
Description : string
AlergicComponents : List<AlergicComponent>
Ingredients : List<Ingredient>
MealDescriptors : List<MealDescriptor>
void AddAlergicComponent()
void AddIngredient()
void AddMealDescriptor()
void RemoveAlergicComponent()
void RemoveIngredient()
void RemoveMealDescriptor()
«ValueObject» MealIdentificationNumber
MealIdentificationNumberPrefix : string
bool ChangeMealIdentificationNumberGen()
string GenerateMealIdentificationNumber()
1
0..*
«Entity» MealDescriptor
ID : long
Description : string
Name : string
0..*
0..*
«Entity» AlergicComponent
ID : long
Description : string
0..*
1..*
«Entity» Ingredient
ID : long
Description : string

Figure 7 – Diagram representing the software's Bounded Contexts

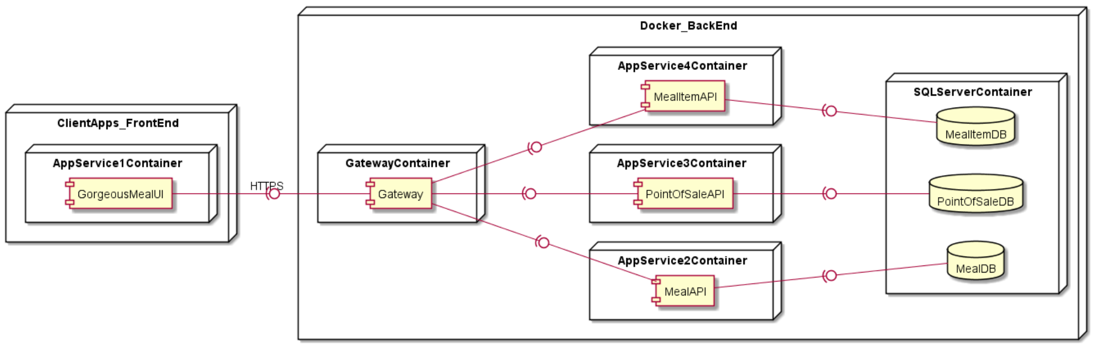

Figure 8 – Diagram representing the software's various components distributed.

# 5 References

[1] “Service Cutter.” [Online]. Available: https://servicecutter.github.io/. [Accessed: 01-Dec-2019].

[2] M. Pope, *i DOWNLOAD available at : https://aka.ms/microservicesebook Co-Authors : Editors : Participants and reviewers :* 2018.