Mariana Martins Corte Real Gonçalves

WORKFORCE MANAGEMENT

SIMULATION OF WORKFORCE REQUIREMENTS IN A CONTACT CENTRE

UNIVERSIDADE Đ COIMBRA

1 2 9 0

1 2 9 0

UNIVERSIDADE Đ
COIMBRA

Mariana Martins Corte Real Gonçalves

WORKFORCE MANAGEMENT

SIMULATION OF WORKFORCE REQUIREMENTS IN A CONTACT CENTRE ENVIRONMENT

Final Report in the context of the Master's degree in Informatics Engineering, Specialisation in Software Engineer advised by Nuno Antunes (University of Coimbra) and Bruno Batista(Talkdesk,Lda) and presented to the Faculty of Sciences and Technology / Department of Informatics Engineering.

July 2020

Faculty of Sciences and Technology

Department of Informatics Engineering

Workforce Management

Simulation of workforce requirements in a Contact Centre environment

Mariana Martins Corte Real Gonçalves

Final Report in the context of the Master's degree in Informatics Engineering, Specialisation in Software Engineer advised by Nuno Antunes (University of Coimbra) and Bruno Batista (Talkdesk,Lda) and presented to the Faculty of Sciences and Technology / Department of Informatics Engineering.

July 2020

UNIVERSIDADE Ð COIMBRA

This page is intentionally left blank.

Acknowledgements

Agradeço o apoio aos familiares que estiveram presentes no meu percurso académico, cujo capítulo encerro, no final deste mestrado a que me propûs.

Uma palavra especial de reconhecimento ao meu orientador Bruno Batista, por toda a ajuda e orientação dada.

Ao meu amigo, João Campos, por toda a orientação e ajuda neste projecto. Desde o primeiro semestre ao final do segundo, esteve sempre presente nos momentos diíceis.

Aos meus avós, Madalena e Zé, por toda a compreensão e apoio com que me acompanharam e sem os quais seria dificil concluir este trabalho.

Agradeço, em especial ao Nuno Santos, pela disponibilidade e ajuda que desmonstrou nas minhas dificuldades, pela paciência e carinho demonstrada durante todo o meu percurso.

This page is intentionally left blank.

Abstract

This document is the result of all the work developed during an internship that took place in the academic year 2019/2020, proposed by Talkdesk, in partnership with the University of Coimbra. Talkdesk is a company that develops solutions for clients in areas as diverse as commerce, financial systems or health systems, using artificial intelligence and simultaneously provides software for cloud contact centre services.
The development of the software aims to help optimising resources, and it is employing agents. The software seeks to increase the company's productivity by validating the created solution provided by the company. The work proposed and developed during the internship period resulted in the creation of two components:

- A segment that in the face of four variables (agent, schedule, shift, queue), inserted by a user in the system, allows the user to subsequently change it (creation, reading, update, deletion).
- A second segment, receiver of the four variables (agent, schedule, shift, queue) sent by the first segment, representing the characteristics of a given environment imitating a contact centre. It interacts, through simulation, with a specific volume of calls generated within the system.
  The result, the behaviour, will be measured and compared with the expected, observing and evaluating the differences, either by defect or by excess, given the expected parameters.

  The contact centre variables interact, by simulation, with a specific volume of calls, generated within the system, this behaviour is evaluated in its capacity.

The intern develops all the steps required to conclude this project with success.

This document aims to serve the reader as a source of guidance. In this possible guide, he/she can have access to information related to the management of contact centres, reference to the technologies used and methodologies associated with the scope of the proposed project.

The reader will be confronted with a description of the project's functionality, as well as with the details of the current implementation. The validation details guarantee that the final product is in line with the specificity of the product.

Keywords

Contact center management. Microservice. Forecast. Scheduling.

This page is intentionally left blank.

Resumo

O documento apresentado é o resultado de todo um trabalho desenvolvido, durante o estágio que decorreu no ano lectivo 2019/ 2020, proposto pela Talkdesk, em parceria com a Universidade de Coimbra.

Talkdesk é uma empresa internacional que desenvolve soluções para clientes de áreas tão distintas como comércio, sistemas financeiros e/ou sistemas de saúde, usando inteligência artificial e fornece, simultâneamente, software orientado a serviços de centros de contacto*cloud*. O desenvolvimento dos software tem como objectivo a otimização de recursos e os seus agentes empregadores actuam com a finalidade de aumentar a produtividade da empresa.

O trabalho que foi proposto, durante o período de estágio, traduziu-se na criação de dois segmentos de desenvolvimento e respectiva análise:

- Um segmento que perante quatro variáveis (agente, horário, turno, fila de espera), inseridas por um utilizador no sistema, permite ao utilizador, posteriormente, a sua alteração (criação, leitura, actualização, eliminação).
- Um segundo segmento, receptor, que a partir das quatro variáveis referidas (agente, horário, turno, fila de espera) enviadas pelo primeiro segmento, correspondem a características de um determinado ambiente representando um centro de contacto. Interage, por simulação, a um volume específico de chamadas, gerado dentro do sistema.

  Este comportamento, após análise, é avaliado na sua capacidade. O resultado desse comportamento é mensurado, comparado com o expectável, observando e avaliando as diferenças, quer por defeito, quer por excesso, em face aos parâmetros esperados.

O interno desenvolveu de raiz todos os algoritmos necessário para a conclusão com sucesso do projecto.

Assim, ao ser apresentado o documento elaborado, este terá como objectivo servir ao leitor de fonte de orientação, um possível guia onde poderá ter acesso à informação relacionada com a gestão de centros de atendimento, referência às tecnologias utilizadas e metodologias de denvolvimento, no âmbito do projecto proposto.

O leitor confrontar-se-á, ainda, através da leitura deste trabalho com a descrição da funcionalidade do projecto, com a actual implementação que é utilizada e a sua validação como garantia que o produto final está em consonância com a especificidade do produto.

Palavras-Chave

Gestão de centro de atendimento. Microserviço. Previsão. Escalonamento.

This page is intentionally left blank.

Contents

This page is intentionally left blank.

Acronyms

AHT	Average handle Time. 15
API	Application programming interface. 17
IVR	Interactive Voice Response. 12
REST	Representational state transfer. 37
RT	Response Time. 15
SDLC	Software development life cycle. 43
SOA	Service-Oriented Architecture. 17
UI	User interface. 41
WIP	Work In Progress. 25

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

The document is a demanded artefact by the course "Dissertation/Internship in Software Engineering” mandatory in the Master’s Degree in Software Engineering at the University of Coimbra. In the next chapters, it is possible to read all the information and research made to best support the development of the project proposed by the host company Talkdesk.

1.1 Hosting Company

Talkdesk is an artificial intelligence and cloud-based software provider. This company was founded after the creators of the company entered a contest. This was when the first version of Talkdesk (TD) software was created and showcased. The company was founded in 2011 in Portugal, after Tiago Paiva and Cristina Fonseca won the contest and thus joined the 500 Startup third batch. Workforce management is a project in progress that focuses on providing more accurate tools and forecasting, that allows contact centres to prevent excessive expenditure through a range of issues, ranging from over staffing problems or even with understaffed problems. The project gives an opportunity to help the contact centre improving its revenue and customer satisfaction over time. Workforce management not only focuses on the client’s point of view(improving client satisfaction), but also improves the agent’s point of view, by creating schedules according to agents choice and the company’s needs.

1.2 Problem Statement

Nowadays, the software currently being used in most contact centres in the global market is incomplete and not user friendly. To have a functional system, the different companies need to license various types of software, and none of them improves or is even focused on forecasting and profit/customer satisfaction improvement.

Since the forecasting is not precise, there is a negative impact that comes from a wrong calculation. Talkdesk’s focus is on forecast accuracy of future afluence of calls that helps in the future management of a Contact Centre, however there was a lack of services available to validate the current approach. The main goal of this Internship is to provide a system that can help in the validation of the generated forecast.

The Internship's primary focus is the creation of a service that simulates a contact centre management environment and return a comparison with the expected results with the outcome from the simulation. With this service, Talkdesk can have functionality that comments a given generated/received environment variables. Some forecast solution created by the company can now be validated with the intended project of this Internship.

1.2.1 Solution

The project will simulate with different variables (agents, shifts, schedule and queues) and uses distinct resources with unique specifications.

The application will, with the help of an algorithm developed by the intern, perform a simulation with the previously defined variables (working staff, shifts, schedules and queues) and a specific/required affluence of calls.

The final result is to create a simulation that returns a comparison of the simulation data results when the expected outcomes.

The project is essential to test out different ramifications possible (different ways that an environment can behave when submitted to different affluence of calls) in the development and simulation of contact centre management.

It helps Talkdesk by having a service that can validate any generated data. The service can be added to the other projects that the company has developed.

The company has new functionality. Talkdesk can use this project to validate forecasts and evaluate it in possibles behaviours.

1.3 Structure Of The Document

This document is divided into six chapters. These best describe the different steps necessary for the development of this project.

1. **Introduction**: In this chapter is identified the existing **problem** and the **solution** created that best resolve the issue.
2. **Background**: The second chapter has all the contextualization related to the technologies used during the internship. Besides technologies, this chapter also references essential terms and ideas to know when working with a project related to contact centre management.
3. **Requirement Analysis**: In this chapter, it is possible to see identified the necessary functionalities to complete the project with success. It is also possible to have a better understanding of how the project will be developed.
4. **Software Architecture**: In this third chapter has all the decision reached when structuring the project and an explanation of the interaction between each component.
5. **Project Planning**: In this fourth is possible to find the planification of the different tasks.

6. **Risk Assessment**: In this fifth chapter, all the identified risks to the project are described. Is possible to find the initial risk and the risk identified after the development, each risk is analysed and a mitigation plan is set.

7. **Implementation specification**: In this seventh chapter is described all the details needed to be defined for effective implementation.

8. **Validation**: In this seventh chapter is described the validation made to the project to ensure that the project works as expected.

9. **Conclusion**: In the last chapter shows the reflection on the development of this project and future development/application with the project.

This page is intentionally left blank.

Chapter 2

Background

In this chapter, we will be looking at all the relevant research data/work necessary and gathered the information needed before starting the development of this project. This chapter also describes the methodologies used in the organisation of the software development of the project.

2.1 Contact Centre Management

The session is mainly divided in three subsections:

- **Data**
- **Forecast**
- **Scheduling**

These are the three main steps for a good Contact Centre management. Each part is described in detail below and explained its importance. First of all, to understand this topic there are some concepts that you are required to know. The term Contact Centre can be described as setting the right number of properly skilled people and supporting resources in place at the perfect time to handle an accurately forecasted workload, at an efficient service level and with quality.[3]

The main goal of today's Contact Centre is to be able to forecast workload, having the right people and resources at an appropriate time. Reaching accurate budgets, avoiding overspending and being able to meet customer's demands and needs. A company with this goal in mind has as target to maintain a good brand reputation and for that it is required to value its customers. Three core values are very important for the company:

- **Efficiency**: In this level the main goal is to be able to deliver a good service to the customer;
- **Customer Satisfaction And Loyalty**: Costumer satisfaction is created with each interaction with the company;
- **Strategic Value**: With every interaction, obtain something from the client experience. It is possible to know from the client point of view what is needed to improve with the centre's products or services.

To efficiently manage a Contact Centre, certain principles are important, that not only affect the organisation but also the people that work there and their customers. It is mandatory to have in mind these **seventeen principle points**[3]:

1. Leadership And Management
2. Communication - Writing, Speaking And Interpersonal
3. Project Management
4. Performance Assessments
5. Quantitative Analysis
6. Customer Behaviour
7. Cultural Aptitude
8. Random Workload Arrival
9. Workload Forecasting
10. Queuing Theory And Statistical Competency
11. Staffing And Scheduling
12. Communication Norms
13. Technology Basis
14. Organisational Basics
15. Organisational Behaviour
16. Ergonomic Work Environment
17. Industry Vocabulary

Each point will have an important role in the final result of the impact not only for the customer, but to the brand reputation and consequently to the customer. The organisation requires a coordinated system of people, technologies and processes that provides synchronised information, resources through the channels of communication by supplying interactions that create value for the organisation and the customer. For every interaction, it is required to create a strategy of how the customer and organisation interacts, it needs a set of **standards and guidelines**. This strategy is called **Customer access strategy**[3]. The customer, in every point of contact, has expectations of the service and because of that it is needed to revisit this strategy often enough to keep it current or it will fall out of alignment with what customers need/expect or how they choose to interact.

1. **Customers Segments**: How customers and their prospects are segmented and how they will be reached. Usually customer segmentation is defined by the marketing strategy of the organisation.
2. **Type Of Interaction**: Each different type of interaction, needs to be analysed for opportunities to build value and to see what needs to be improved for better customers satisfaction and continued loyalty.

3. **Access Channels And Communities**: Nowadays there are many possible channels of communication. It is required to see all the channels that can be used by the company.

4. **Hours Of Operation**: In every channel of communication possible, different hours can be applied to be available to the customer. Communication though self-service channels can be available twenty-four hours every day. This does not involve any working agent.

5. **Service Level And Response Time Objectives**: Different channels of communication are often appropriated for different approaches of interaction and customer segments.

6. **Routing Methodology**: After an interaction arrives in the system, the customer will be treated depending on the original reason for interaction. Because of different interactions, different required action is needed to define a routing system and how these different types of interactions can be distributed.

7. **People/Technology Resources Required**: With every interaction, different actions are required and for that different people may be required to finalise the interaction. For that to happen we need to have the right resources and people available at the right times.

8. **Information Required**: It is important to define what type of information or services will be available to the agent and the same thing goes to the customer. It is fundamental to define what type of information from the services/product available in the organisation is available to the agent and the customer.

9. **Analysis/Improvement**: In every interaction there is information that can be extracted. It is important to know what type of information should be captured during a contact with a customer. This information will be used to improve customers interactions and/or product or services.

10. **Guidelines For Deploying New Services**: This last component is essential because with time; different products and services will appear or even the existing ones will evolve, different skills from the agents may also be required or budget updates so it is important to keep updating the customer access strategy and it is important to define how/when the strategy will change.

Customers' demands are constantly evolving due to constant improvements in services and products. Customers after a while, get used to the product and they expect changes or improvements to meet their ever evolving demands. Below are the **ten different key customers expectations**[3] that the organisation needs to have in mind:

1. Be Accessible

2. Treat Me Courteously

3. Be Responsive To What I Need And Want

4. Do What I Ask Promptly

5. Provide Well-Trained And Informed Employees

6. Tell Me What To Expect

7. Meet Your Commitments And Keep Your Promises
8. Do It Right The First Time
9. Follow Up
10. Be Socially Responsible And Ethical

The best way to control customer expectations is to create a good relationship between the client and the brand of the organisation. It is necessary to always meet the commitments that were promised to the client, always let the customer know what to expect. The first release of the product/service should meet the minimum requirements requested by the customer and check if everything is alright with the product/service given to the customer. To ensure that the customer's expectation is in the right direction regarding the plans of the company, and needs to manage the team thoroughly and make sure that they understand the 10 basic customer expectations. Always listen through social channels, during interactions with agents, through surveys to know how the agents are doing regarding the 10 basic customer expectations. Build cross-functional teams to ensure a common focus on serving customers. Other things that are really important when talking about ensuring that customer expectations are met according to the plans of the organisation, is to always involve customer expectations in the context in which it develops its customer access strategy. There are three important driving forces that can affect the environment of the Contact Centre management organisation:

- **Workload**: The quantity of the incoming calls flow are never constant. There are three types of workload possible: smooth, random or peak. A smooth flow is a constant amount of received calls with not many spikes in the amount of the calls received by the Contact Centre. Random workload is characterised by unexpected spikes of calls. Lastly, peak workload is similar to smooth workload but at the same time an unforeseen amount of calls occur.

- **Queue**: There are two options to how the queue will operate. It is possible to have a visible queue, meaning that the customer in queue will know in which position he/she is on. Invisible queue is the option where the customer does not know if he/she is in the last or first position.

- **Customer Tolerance**: The customer is not always willing to wait, or there is always competition available and more friendly to the user. There are seven factors of customer tolerance that affects the customer experience with the organisation system:

  1. Degree Of Motivation
  2. Availability Of Substitutes
  3. Competitions Service Level
  4. Time Available
  5. Who Is Paying For The Call
  6. Human Behaviour

  All of these six factors can lead to abandonment; but the organisation depending on how accessible it is, can change the percentage of abandonment.

2.1.1 Choose goals and collect information

Choose Service Level And Response Time Objectives

Another important core value of an effective Contact Centre is accessibility. The Contact Centre's accessibility can be measured with two terms: **Service level** and **Response time**.

- **Service level** is a term used to measure the performance of the system, all the goals of a Contact Centre are defined with this term. Inbound calls and walk-in service are counted as part of service level, it is all the calls or request to the contact/support centre that requires immediate action. This term reports how many calls were answered in a specific moment.
  - To determine server levels objectives, it is necessary to gather feedback from the customer (survey, focus group, comments), to relate with the competition or similar organisations and the usage of a combined approach from where the organisation is located, what is the agents behaviour and how are they reacting to the received request.
  - It is also important that supervisors, managers and those with supporting roles know what the service level objective is.
- **Response time** means the same as service level but it refers to those types of contacts that can be handled at a later time, that do not have to be answered the moment they arrive.
  - Choosing response time objectives is necessary to establishing an appropriate service level having always in mind the seven customer tolerance factors.
  - Response time objectives are divided in the automated reply from the system that generates the response, the response that the customer receives and finally when the problem is resolved.
  - There are two types of response, scheduled or rolling. The first type of answer is geared around blocks of time. The second is in a specific time each message arrives.
  - Response time objective also requires internal communication, it is very important to have levels of priority and appropriate responses for internal email/contact. Choosing service level and response time objectives is not an exact science and may need adjustments according to the defined targets because with time it will determine the required resources and calculate cost.

    Choosing service level and response time objectives is not an exact science and may need adjust according with the defined targets because with time you will determine the required resources and calculate cost.

Data

For a Contact Centre to be effective, data must be collected. Without good information the organisation has inaccurate forecasts, variable quality, interaction and unpredictable cost. The best way to identify any gaps in getting and using data in resource planning is to create a flow chart of the nine-step planning processes. In this chart, it is necessary to identify the information needed for each step, the form that the information is received, where the information is coming from, who and what produces it, when the company needs it and how and when it fits into the planning process.

The **nine-step planning processes**[3] are divided in:

1. Choose Service Level And Response Time Objectives
2. Collect Data
3. Forecast Workload
4. Calculate Base Staff
5. Calculate System Resources
6. Calculate Shrinkage
7. Organise Schedules
8. Calculate Cost
9. Repeat For Higher And Lower Service Levels

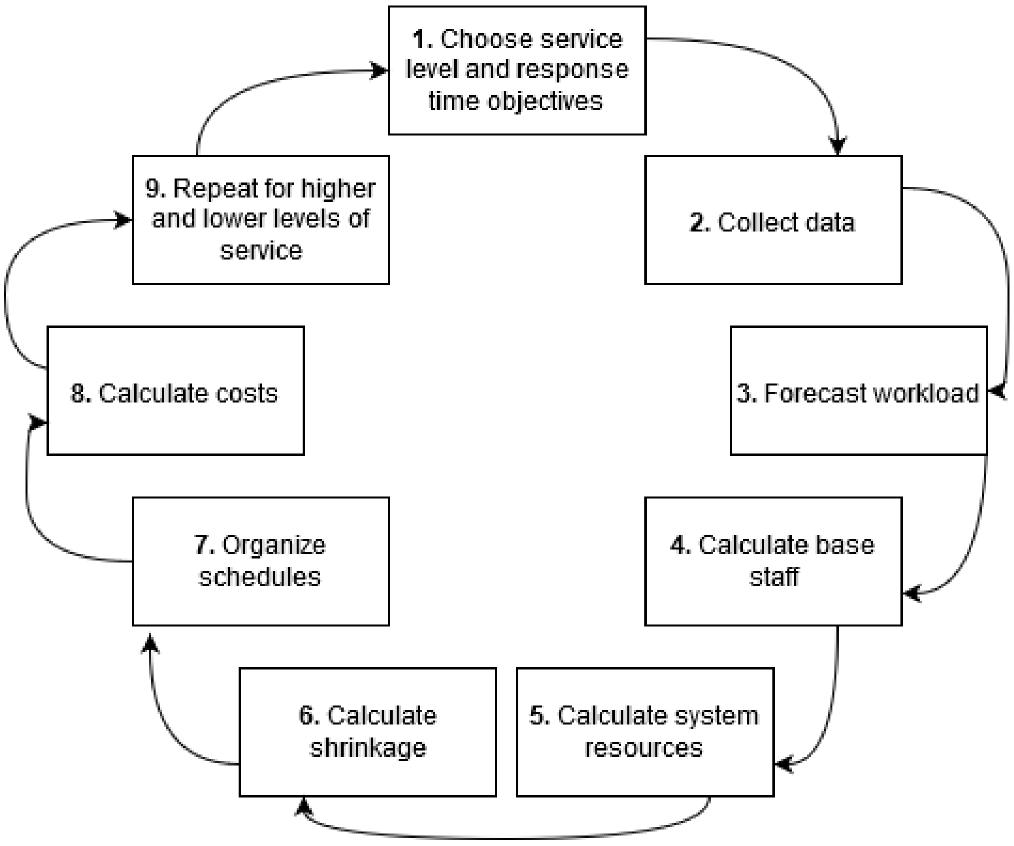

1. Choose service level and response time objectives
2. Collect data
3. Forecast workload
4. Calculate base staff
5. Calculate system resources
6. Calculate shrinkage
7. Organize schedules
8. Calculate costs
9. Repeat for higher and lower levels of service

Figure 2.1: Flow chart of the nine-step planning processes

It will identify missing links and lead to ideas for developing a more integrated and collaborative approach. Needs regular updates. As referred before, building a cross-functional team/processes is a good approach to helping set up an effective Contact Centre management. This approach helps define and make a clear customer access strategy. Helps define the customer segments, agent group structures, service level objectives and all the information needed to handle contact/interactions. Also it is important to implement regular meetings with representatives from all the different departments throughout the organisation who review customer input and implications in future/current projects. Having dedicated planning manager positions with primary function enhance cross-functional communication within the organisation. Applying all of these considerations helps to better forecast/plan. With the use of reporting tools it is possible to have quality monitoring, text, speech analytic, performance management tools. Anyone who is part of the organisation information system can view, extract, print and store real time and historical information. This is helpful in raising the Contact Centre's profile internally and to colleagues across the organisation. From the review of all the information obtained, it is possible to ensure that those working in the centre are getting and judiciously using the right information at the right times for the right purposes. This is a key leadership responsibility. Data collection is a planning step that needs to be continually reassessed and improved. All the information collected helps support key activities and helps build the principles and implement the mission of the organisation. This information can be collected from internal or external sources throughout the cross-functional effort.

2.1.2 Forecast

Forecast Workload

Having the right notion of the resources needed to a specific amount of workload is an essential step for an effective management of a Contact Centre. The ideal environment is having the right number of skilled people and supporting resources in place, at the right times to handle an accurately forecasted workload at level service and with quality. For that environment to become a reality, it is important to determine the amount of staff needed and the requirement of other resources. The art of forecasting how many contacts the organisation is going to get in a future period is dependent on three different components: determination of patterns, consideration of possible trends and lastly the handling times of interactions.

There are two types of forecasts:

- **Longer-Term Forecasts**: this type of forecasts helps to estimate future annual budgets, also helps to establish long-term hiring plans and grants the possibility of definition of future system needs.
- **Shorter-term Forecasts**: this is another type of forecasts that supports the adjustment of schedule requirements and the anticipation of seasonal staffing needs. Helps predict holidays and the determination of imminent hiring necessities.

To be able to successfully forecast there is essential data needed. It is important to know how many contacts the organisation received in the past, when they arrived and how long they took to be handled.

In this environment of forecasting there are **four essentials terms**[3] that the company is required to understand:

- **Talk Time**: talk time is considered the time that the agent is connected with the agent.
- **After-Call Work**: this term refers to the time that agents spend completing transactions after disconnecting from the caller.

- **Average Handling Time**: is the mix of the average time spent in talk time and the average of after call work time.
- **Call Load**: in a given period of time, is the mix of the average talk time and the average after call work.

Besides the fact that it is needed to have in mind these four terms it is mandatory to clean the data. It is required to do appropriate adjustments that reflect what is likely to happen, this way we ensure that the forecasts are built on a solid foundation. When the calls reach the system there are four actions that they encounter: the call can get the busy signal, the call can be answered by the system, the call can be abandoned before reaching an agent or the call can reach an agent. The calls that are abandoned are important to have in mind the seven factors of customer tolerance that influence how long callers will tolerate in queue. This will lead to forecasts with overstate demand. So to ensure that this does not happen it is important to include most of the 70% or more of calls abandoned in that data. If the workload is underestimated this will lead to insufficient staffing and more abandoned calls. There is one other tool/system that can be used to provide customers with a self-service option avoiding abandoned calls and help routing calls. This system is call *Interactive Voice Response (IVR)*. With this system a call starts out in one place and ends up in another place depending on real-time circumstances/choices. Even with the use of this tool, it is important to predict the workload that will be handled by the network and systems so that the organisation can engineer them correctly.

There are three dominant patterns in forecasting:

- **Month/Year/Seasonality**: if the organisation is going through dramatic changes, the organisation will usually detect seasonality in its arrival patterns (three years of information will provide good reading).
- **Day of the Week**: when pent-up demand happens (move of calls bigger than normal amount), the pattern is highly predictable from one week to the next (four to five weeks will reveal the pattern).
- **Half an Hour of the Day**: sometimes exceptions happen and they need to be identified and removed from the data (one or two weeks worth of data are enough to reveal the pattern).

Forecasts can be influenced by different aspects. Future sales, marketing campaigns at a certain time. All of these aspects if predicted can help forecast the affluence of calls. Sales forecast can improve the call load and the need to know the average sales value in the Contact Centre. This forecast can provide a good sanity check to ensure that the contact use response rates to forecast call load, the volume is high in the initial days and tapers down overtime. Besides the use of call, there are other channels used in Contact Centres to help the customer. Even in these channels, it is necessary to anticipate the number of customer sessions/conversations. It is also necessary to know the time required to handle them and the number of customers an agent can simultaneously handle. These channels also provide monitoring tools that allow to have a regular check up on what is being said about the company helping identify the influences, prominent opinions and what is having a big impact on the perceptions of its brand. When talking of forecasting the average handling time, there is a lot of problems and decisions that are required to have in mind. First of all, when calculating the average handling time the volume of calls is meaningless, what really matters is the call load.

$$(ATT + AACW) * VP$$

ATT: Average talk time
AACW: Average after-call work
VP: Volume in a given period of time.

For a good calculation/result on the average handling time there are some prerequisites to have in mind:

- **Look For Patterns**: it is necessary to identify how average talk time and average after-call work vary; can exist a different pattern for a day of the week, season of the year, billing cycles, marketing campaigns.
- **Train Your Agents To Use Work Mode Consistently**: each agent has an impact on the components of handling time (data used in forecasting), is also necessary to define what type of work should follow calls and what type of work can wait.
- **Identify The Average Handling Time For Different Call Types**: define and categorise calls by type, this way it is possible to accurately track calls based on categorising.
- **Assess the Impact of New Agents, Languages Handled and Process Changes**: it is also required to have in mind that less experienced agents require more time to handle a call/contact/process/system, and is important to have in mind the impact of these variables on AHT (average handling time).

Forecast is hardly followed, instead it is partially guessed from the hard data. Forecast is the mix of **quantitative** and **judgemental** forecasting. Quantitative forecast is the usage of hard data in the process. Judgmental forecast belongs to the realm of intuition. The trick for a good forecasting is to combine both approaches. This collaborative approach is most effective when team members who are accountable for staffing take an active role in forecasting. There is another question that needs to be asked *"How accurate should the forecast be?"*, since the forecast impacts the staffing calculation, schedules, budgets and the service that is delivered. It is important to have a table that summarises the percentage of intervals that fall within various ranges of accuracy. This way it is possible to see the behaviour in a week/month and adjust what is necessary.

To finalise this chapter, one last thing that is important to have in mind are these **ten most common mistakes connected to forecasting**[3]:

1. No Systematic Process In Place
2. An Assumption That *"the forecast software know best"*
3. Not Forecasting At The Agent Group Level
4. The Forecasting Is Taken Lightly
5. Events That Should Be Exceptions Become A Part Of The Forecast
6. Ongoing Communication With Other Departments

7. Planning Is Done Around Goals, Not Reality
8. No One Is Accountable
9. Agents Are Mixing Flexible Activities Into Work Modes
10. Not Making The Connection With Staffing

2.1.3 Scheduling

Calculate Base Staff And Calculate System Resources

To have an efficient Contact Centre, it is important to **achieve the service level and response time set**, these goals need to be supported by **sufficient system resources** and have a **reasonably accurate forecast** and base staff calculations.

To understand this concept there are **three new terms**:

- **Delay**: the trunk is sized to the point the caller is connected to an agent.
- **Agent Load**: is the mix of the talk time with the time of after call work.
- **Trunk Load**: trunk load or caller's load are all the aspects of the interaction except the time after call work.

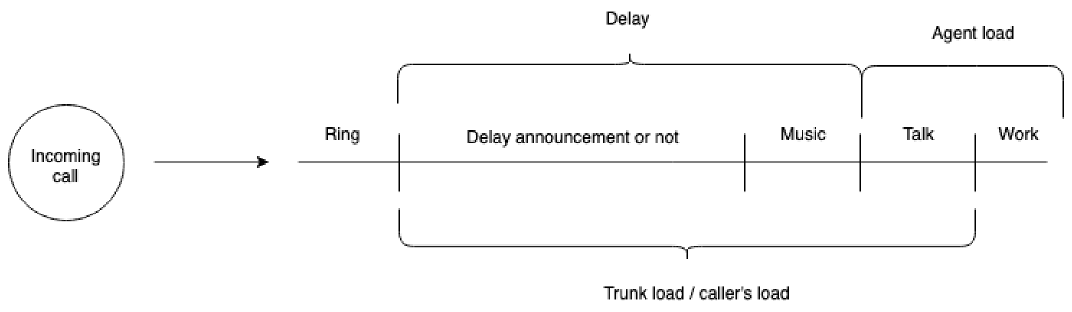


Figure 2.2: Definition of Delay/Agent load/Trunk load

As we can see from what is drawn in the image above, the trunk load is higher depending on the delay surfer by the caller. From that it is possible to see the relation between the staff and trunk load. With more staffing there will be less delay.

With the **formula Erlang C**is possible to determine resources in just about any situation where people might be waiting in queue for service. This formula is built into virtually all of the commercially available Work Force Management (WFM). This formula calculates predicted waiting times based on the number of agents, the number of people waiting and the average amount of time that it takes to serve. This formula also assumes that callers have infinite patience and that there are infinite trunking/system capabilities, overestimating the staff is of utmost importance. Another alternative is the use of computer simulation to help the reduction of mistakes in live environment. This option is meant for modelling/design/verification. It is not for forecasting or scheduling. Besides the fact that it takes more time or expertise to set up then Erlang, it is possible to run over and over to identify potential results.

The *Erlang C* formula illustrates queue dynamics and resources trade off well so it can be used as a great educational tool. This formula also outlines basic staffing requirements and trade offs in a Contact Centre setting. Erlang C requires the following four variables:

- Average talk time, in seconds.
- Average after-call work, in seconds.
- Number of calls.
- Service level objective, in seconds.

When the type of contact arrives and it is needed to be handled later in time, different calculation for staffing is required. When calculating using the formula Erlang C uses volume (quantity of interactions it must handle), Average Hour Time (AHT) (average amount of time it takes agents to handle) and Response Time (RT) (time you have to respond to customers after receiving their messages). **By applying the formula the result will be:**

$$\frac{Volume}{\frac{ResponseTime(RT)}{AveragehandleTime(AHT)}}$$

After applying the formula it is possible to **get the number of agents advised** by the formula.

There are many ways to achieve our objective and many ways to mix the agents and their schedules. This basic *Erlang C* formula assumes a "static" amount of work to be completed, it is defined as an amount of work that has already arrived and is waiting to be processed. It is also important to consider breaks, absenteeism and other activities that keep agents from the work and for that they need to be added to base staff calculations.

When calculating base staff is fundamental to acknowledge the **efficiency factor**. Agents cannot handle one transaction after another with no breathing time in between and because of that is required to think in every element that will influence the productivity of the agent.

To meet the objective set for the response time is necessary to forecast these types of interactions, within time-frames specific enough to calculate base staff needed. There are many ways to achieve our objective and many ways to mix the agents and their schedules. This basic Erlang C formula assumes a "static" amount of work to be completed, it is defined as an amount of work that has already arrived and is waiting to be processed. It is also important to consider breaks, absenteeism and other activities that keep agents from the work and for that they need to be added to base staff calculations. When calculating base staff, it is fundamental to acknowledge the efficiency factor. Agents cannot handle one transaction after another with no breathing time in between and because of that it is required to think of every element that will influence the productivity of the agent. To meet the objective set for the response time it is necessary to forecast these types of interactions, within time-frames specific enough to calculate base staff needed. Another option when talking about staffing is staffing by skills. This allows the organisation skills-based routing, the capability designed to match each caller with the agent who has the skill set best suited to handle the call on real-time basis. To define a skill-based routing it is required to identify and define the skills required for each type of call and individual agent skills. Accordingly, with the existing skills, it is possible to prioritise specific agents based on individual skills and competency levels. After defining all of this aspect it is only necessary to devise and program an appropriate routing plan into the ACD (automatic call distributor). This type of architecture is best suited to small groups where multiple skills are required, and also helps quickly integrate new agents by sending only certain calls to them. It has all the downsides, it is required to always have the right people at the right time and it is also important to develop a contingency plan in case the agent is unavailable or the call load of a specific call type is greater than expected. It is necessary to have enough trunk to carry the delay that callers experience and the conversation time.

The general method for calculating trunk requires four steps:

1. It is necessary to forecast the call load (work load) to be handled for the busiest half hour in the foreseeable future.
2. Compute the number of agents required to handle the forecasted call load at the company service level objective.
3. Determine the trunk load according to the call load it will be handling and the service level the organisation can realistically achieve. The trunk load represents how much time, in hours, callers are in queue or connected to agents over an hour.
4. Determine the number of trunks required to handle the calculated trunk load, using an appropriate formula.

$$P = \frac{\frac{A^N!}{2}}{\sum_{n=0}^{\infty} \frac{A^x}{x!}}$$

- P = grade of service.
- A = total traffic in *Erlang*.
- N = number of trunks.

Formula	**Assumptions**
Erlang B	Assumes that if callers get busy signals, they go away forever, never retry. Since some callers retry, Erlang B can underestimate trunks required.
Poisson	Assumes that if callers get busy signals, they keep trying until they successfully get through. Since some callers won't keep retrying, Poisson can overestimate trunks required.
Retrial tables	Used less frequently by traffic engineers, but correctly assumes that some callers retry and others go away.
Bandwidth Calculations	More a genre of methods than a formula. Bandwidth calculators consider variables such as voice compression and coding algorithms to estimate requirements.

Table 2.1: Different types of formula that focus in specific details to improve the efficiency of the final results in a Contact Centre management.[3]

Organise Schedules And Calculate Costs

The main goal of scheduling is to get the right amount of people and supporting resources in the right place at the right time. The problem with overstaffed centres, unnecessarily high staffing costs, underutilised agents, boredom of agents, loss of credibility in budgeting. Understaffed is not a very good option, it creates unhappy customers, abandoned calls, longer calls, additional or simultaneous contacts, more errors and rework, higher telephone network usage and cost, staff stress and burnout and negative impact on brand reputation.

The principles of scheduling[3]:

- Correctly Identifying And Categorising The Different Types Of Activities.
- Making Accurate Base Staff Calculations.
- Anticipating Shrinkage Requirements.
- Using Scheduling Alternatives That Provide Necessary Flexibility.
- Ensuring That Things Go As Planned.

Following all these principles, in a high level forecasting it is possible to predict who needs to be, where and what is needs to be done and when it is needed.

2.1.4 Repeat For Higher And Lower Levels Of Service

This cycle is repeated to try to improve the best possible way for each stage.

2.2 Microservice

Microservice is a structured software system, that is well known for delivery of large and complex applications. This type of system is a service-oriented architecture(SOA) that is developed to work has a single function. Different microservices communicate thought web Application programming interface (API) or messaging queues.

This type of software development has advantages and disadvantages. Fortunately the advantage overlaps the disadvantage.

2.2.1 How Can A Microservice System Help A Project?

There are three main proprieties that can describe a microservice:

- **Flexibility**: Working as part of a team, this architecture grants a fast development. Since the architecture system accepts the division, as a whole or part of a project, dividing the project in pieces without affecting other systems. Besides the fact that it is possible to divide one big project into little pieces, each piece can be developed in different programming languages or even using different databases.
- **Reliability**: If one function/microservice breaks or stops functioning for any reason, the project will continue to be functional. The system is also easily recoverable. When the error is fixed, only the failed service needs to be deployed again.

- **Scalability**: This type of architecture system, not only gives this application ease of use but also allows end users the ability to scale projects, if it is needed to scale the project. It is only necessary to add a new microservice with the necessary function to the project without compromising the performance of the system.
- **Maintenance And Upgrade**: Since a microservice allows the division of the project in parts, in case of a necessary change or even an improvement, it is easy to focus on a specific part without having to go throughout the all project. Not only is it possible to have specific people working on the project without knowledge of the rest of the project, it is also important to recognise that these proprieties also facilitate the debug and testing of the application/specific microservice.

Microservice has a lot of advantage, but also has some disadvantages

As a downside a microservice architecture is complex compared to a monolithic application. This type of architecture adds up more complexity and deals with different kinds of programming languages and the load balance between multiple platform services.

- **Performance Reduce And Complexity Rise**: Increasing the number of microservices available increases effective effort made to integrate and manage these different microservices.
- **Problems With Maintenance**: As the number of microservices increases, it is harder to ensure that when changing one, it does not affect the rest since these communications are interconnected with other services. When trying to change one of them there is a big chance that a simple fix changes how the output of one is received in the service. When changing or updating one microservice it is necessary to have in mind the integration/communication that all the microservices have with one another.
- **Communication Across Microservice** and **Integration With Others Microservice**: This point is very important, having a stable architecture and standards of communication defines how the contact is made. Not only should this interface be shared between teams but it also helps prevent future issues. By sharing it between teams it helps it have a consistent modulation of how each services/microservices can/will interact.
- **Debugging and monitoring**: When many Microservices (MS) are active, each microservice produces a logger file or each own report. When trying to debug it is necessary to pass all the logger files to be handled to understand the error. Debugging/Monitoring is not always doable because of the complexity involved in this process.

2.2.2 How Can A Microservice Help This Project?

In the first chapter it is possible to read the main goal of the development of this project. The use of a microservice will facilitate the scalability of the microservice developed in case of arrival of large affluence. This type of structured system will grant more asymmetric scalability. Having multiple MS allows to scale instances differently. Also, we can switch particular microservice implementations because of clearly defined interfaces. In case of failure of one of the components of the system, since the system is not all directly connected this ensures the system recovers quickly.

2.3 Scrum

Get your team to work faster and smarter. Rank priorities so your new startup adhere to lean methods.[6] Scrum is an agile approach to work and conducts a project. When developing an agile software with scrum, scrum is often noticed as a methodology.

Scrum is a framework that mainly works over these twelve principles in the Agile manifesto:

- The customer is the highest priority. The customer regularly receives all the valuable software made so far.
- This approach allows and encourages the change of requirements even in a stage of development.
- This third principle says that in a short timescale software should be delivered. It could be weeks or months but it is necessary to deliver working software.
- The project should always be worked on a daily basis. Business people or developers must work together daily throughout the project.
- Having a motivated environment, support and trust is fundamental to helping individuals work effectively on their projects.
- Face-to-face conversation are considered the most efficient and effective method for sending important information across the team and within the development team.
- All the working software is the primary measure of progress.
- All the agile steps promote sustainable development. The users, developers and sponsors need to maintain a constant pace for an indefinite time.
- This framework works with the idea of having a continuous attention to technical excellence and good design enhances agility.
- Simplification is essential, it maximises the amount of work not done.
- With regular breaks, the team reflects on how they can be more effective. Adjust their behaviour accordingly.

Scrum is a team approach that focuses on continuous improvement, team input, to deliver quality products. This framework relies on self-organisation and cross-functional team. The team is self-organising in a way that it is not necessary to have a team leader to make decisions or to solve problems. All of these issues are decided by the whole team. Which element of a team needs to take a feature from idea to implementation. With agile development, scrum teams are supported by **three** important concepts.

- **Scrum Master**, is a person who coaches the team, helps the team members to not only use the scrum process but also to have the best performance. This role does not provide day-to-day directions of assignment of individual tasks.
- **Product Owner**, this person represents the customers/users and the business. He guides the team to produce the right product/goal. This role creates a compelling vision of the product, then allows the team to make that vision in the product backlog. The product owner is the one that prioritises the backlog during scrum development.

- **Scrum Team**, titles in a team are insignificant, each person makes their own contribution to complete the work in every sprint.

2.3.1 What Is Scrum Development?

The Scrum model recommends that the project divide the progress in sprints. These sprints are built having in mind the agile methodology, sprints are timed to no more than a month, the longest use is two weeks.
Scrum advises the planning of a meeting at the start of each sprint. In this meeting it will be decided/defined how many items it is possible to commit to and is also created a sprint backlog (list of tasks to perform during the sprint). Throughout the meeting, the team sets some features from the original idea to be coded and to test its own functionality. When these features are done, (tested and coded) they integrate the evolving product/system.
Every day of the sprint, the team has a daily scrum meeting including the Scrum Master and the product owner. This meeting should not take more than fifteen minutes. In this meeting all team members should share what they did the day before and what they will do on that day. When a sprint finally ends, the team reviews what was done and presents it to the product owner or/and any other stakeholder. This way it is possible to get feedback on everything that was made. This feedback will/could influence the next sprint.
This loop of feedback resulting from the scrum software development can result in changes in functionality in the final vision of the product, by adding or removing and even modifying certain elements already implemented.
Finally, after each sprint is made an investigation of how things went. The team is present and participates in the meeting. This meeting is an opportunity to identify the same needed changes/improvements.
Scrum framework focuses mainly on prioritising requirements and daily showing the process. Scrums encourages requirement changes with regular analysis of the impact/progress made by the team with the product owner and/or stakeholders.

2.3.2 Which Are The Main Artefacts Of The Scrum Process?

There are three main artefacts in the scrum process.

- **Product**: In scrum development the principal artefact is the product itself. As expected, from the scrum model, the team to build the product/system to a potent state to release at the end of each sprint.
- **Product Backlog**: This backlog is the full complete list of the functionality that remains to be developed in the product. Each functionality is prioritised so that the team always works on the most valuable features first. This prioritisation is made by the product owner. The most effective way of creating the product backlog is by using scrum methodology and creating user stories. User stories are short descriptions of each feature described by the point of view by the customer/user. When replicating the scrum project management, in the beginning of each sprint and during the planning meeting, the team members create the sprint backlog. A sprint backlog is the list of tasks that are needed to perform in order to deliver the functionality that was decided to be delivered during the sprint.
- **Sprint Burn-Down Chart And Release Burn-down chart**: Burn-down charts are an effective tool in scrum software development to see the remaining amount of

work necessary in each sprint/release. It is helpful to plan the remaining work or to check if the project/tasks go in schedule.

2.3.3 User Stories

The backlog is the heart of scrum. A good product starts with a customer need and a vision for how to solve it.[6]

The backlog is the result of the improvement of the vision into specific features.

Throughout the entire process the creation of the backlog can be a complex experience. The backlog is the prioritisation of the list of requirements, or stories.

A story is a composition of six fields:

- **ID**: represents the identification of each story. Each story has a unique number.
- **Name**: brief description of the story. Needs to be clear enough that developers understand what the story is about and clear enough to be different from others.
- **Importance**: Value set by the product owner that defines the importance of this story.
- **Initial Estimate**: Estimation of how much work the story requires to be implemented.
- **Notes**: Information or clarifications that references other sources.

2.3.4 Sprint Planning

Before the sprint planning meeting,

- It is very important that the backlog is shipshape.
- The product owner must understand each story in the backlog and the reason why the story was created. The main reason for that is that the product owner is the only one who will rate the importance level per story.

Sprint planning is a fundamental meeting and considered the most important event in scrum. One bad organised sprint can mess up the whole sprint. The main goal of this meeting is to give the team enough information for them to work. Before every meeting it is important to establish a preliminary schedule to reduce the risk of exceeding the initial time set for the meeting. Sometimes when this schedule is not set the meeting can be dragged on and nothing is accomplished. After each sprint specific outcomes are defined[6]:

- The Sprint Goal.
- The List Of All Team Members That Are Committed.
- Date Of The Sprint Demo.
- Daily Date Of Each Daily Scrum.

During all sprints the product owner is required to be there, the scope and the importance of each item in the backlog is set by him, only the estimate is set by the team. All of these variables, scope, estimate and importance, are defined through face-to-face dialogue between the team and the product owner.

2.3.5 Spring Length

One important aspect is the length of each sprint. A long sprint allows the team to recover from the adversity found as well as allows the team to meet the sprint goal. The length of a sprint is a compromise between the product owner and the developers. The sprint needs to be short enough to give business agility and long enough for the team to accomplish the flow set and in some cases recover from the problems that are found during the sprint. It is adviced to initially experiment with different sprint lengths. After the length that best fits the team and the product owner has been decided, it is best to keep/stick to it for an extended period of time.

2.3.6 Spring Goal

There is a fundamental question for this session *"What is the point of doing this sprint?"*. The main sprint goal is to allow the product owner to answer that question. It is important to have a wiki page or similar where the main goal and the list of the sprint goals are listed. This way anyone in the company can access it, even the team if needed.

2.3.7 Sprint Stories

The first important detail to notice is that it is the team and not the product owner that decides how many stories will be included in the sprint backlog. The team will commit to that list of stories during that sprint. Now how will the team choose which stories to include and how can the product owner affect their decision? There are three options on how product owners affect the choice of stories. In case the product is not satisfied with an inclusion of a specific story, it is possible to re-prioritise a specific story or change the scope of that story. The last solution is always possible to split a story to be able to include another story and still not change the estimation made/agreed to that sprint. The team uses two different techniques to choose the stories to include.

- Gut Feel
- Velocity Calculations

Gut feel works well in small teams and short sprints. To use the velocity of calculations it is necessary to calculate how many stories are being added. Velocity is an analysis of the amount of work necessary. Each item has a specific load of estimate work. A story needs to have an estimation of work, prioritisation, and clarification of what is context. Index cards are used to discuss priorities and details of each story. People stand up and walk around and everyone feels more personally involved, multiple stories are edited at the same time. This interface is much superior compared to a computer or projector. After each sprint meeting, the scrum master is responsible for making the changes and updates necessary decided in the meeting. Task breakdown is the division of a story into different tasks that are required to be done to finish/complete the story. This step allows us to see what it takes to complete each story, this makes the time estimates easier because it is visually easy to see what each story needs. In this step the product owner is not required to be present.

2.3.8 Estimation

Some stories involve different people with different abilities to get involved. It is not possible to predict who is going to implement every story and which parts. To be able to provide an estimation, it is necessary that every member interpret the story in the same way. For that it is necessary to personally ask everybody to estimate each story, this facilitates the reciprocal help between team members and increases the possibility that important questions occur early. To facilitate the classification of the estimation of each story, a technique called Planning poker is used. Each element gets a set of thirteen cards, these cards will have a value that when chosen represent the time estimate. At the end of the round, when every element has chosen a card, everyone will show it at the same time. This obliges every team member to think for themselves and nothing is influenced. In case the values are very different, the team discusses the differences and tries to create a common picture of how much work is involved.

2.3.9 Sprint

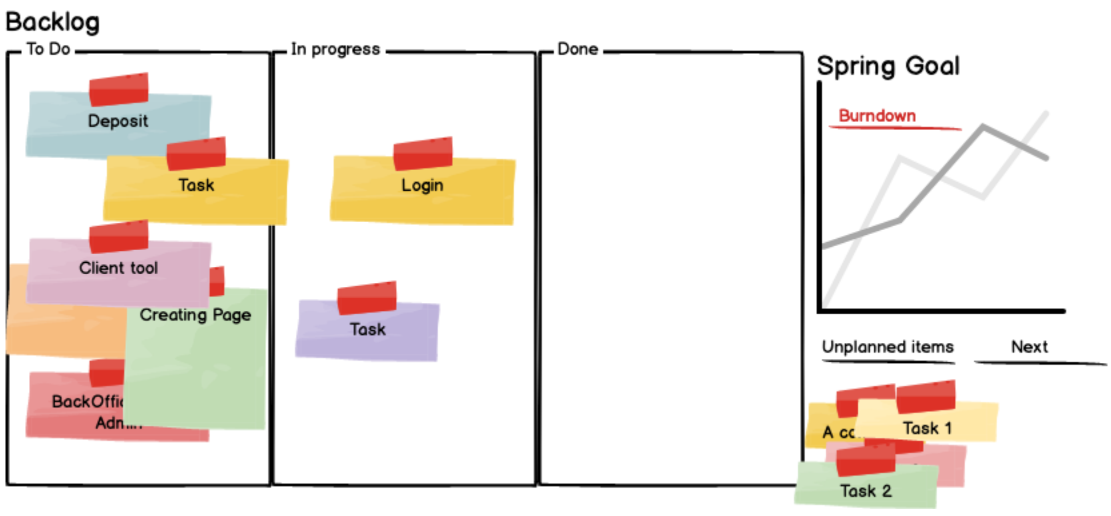


Figure 2.3: Example of a Backlog composition

There are different software that helps with the organisation of sprints backlogs like per example *Jira*, excel or even a physical task board.

There are different softwares that help with the organisation of sprints backlogs like for example Jira, excel or even a physical task board. The most effective way to organise the sprint backlog is with the division of four columns. The first one has all the tasks required for this sprint, the second has all the active tasks (tasks that are in process) and the third one is all the tasks that are considered completed. The last column represents the graphic of the story points required by the number of days remaining and all the unplanned tasks that are being found and all the tasks that were moved to the next sprint. The board needs to give to anyone who sees it an indication of the process of the sprint. The scrum master is responsible for indicating any warning sign if necessary.

2.3.10 Daily Scrums

Normally this type of meeting is a stand up meeting because it helps control the time of the meeting, and should not take more than 15 min. The meeting is important because it helps with the synchronisation within the team of any problems or active tasks. There are three important questions that should be answered in every meeting:

1. *What did I do yesterday that helped our team meet the sprint goal?*
2. *What will I do today to help our team meet the sprint goal?*
3. *Do I see any impediments that prevent me or our team from meeting the sprint goal?*

The task board is updated during this meeting. In case of an unplanned item, this item will be added in the correct session. The advantage of each person updating each respective task is that the Scrum master does not spend time with administrating stuff and the whole team is aware of the status of the sprint.

2.3.11 Sprint Review

Every sprint ends with a demo. A demo allows the team and others to check the accomplishments made by the team, and attracts essential feedback from stakeholders. With a sprint review, other teams are assisting and is an event that allows interaction with each other and a chance to discuss their work. Doing these reviews forces the team to try to finish all the work previously defined. The review is important to present the sprint goal and to present the main goal of the product to people that do not have a clue about it. This review also needs to be fast and if the external people of the team want to try it out this should also be possible.

2.3.12 Sprint Retrospectives

After every sprint it is important to have sprint retrospectives. Retrospective meetings allows the team to discover or even to recognise the mistakes that were made, and prevent them from happening in the next sprint. In this meeting, the product owner and the whole team are necessary. In this meeting the scrum master summarises the sprint and shows all components of that backlog sprint. Each person takes turns to speak and in case of a big difference between the estimated effort and the actual velocity an analyse is made to try to find out the problem. The meeting ends with the scrum master summarising all the suggestions made that can be applied in the next sprint.

2.3.13 Release Planning

When trying to define the release it is necessary to plan the sprints ahead of time, this way avoids the risk of signing something that would not be possible to deliver on time. For an effective organisation it is important to define the limits of what should be included, what is required and what might not be needed. It is fundamental when planning the release that the product owner estimates at least all the stories included in the contract, this is as the sprint planning, this estimation is a cooperative work between the product owner and the team. After every sprint, if the velocity of the task is very different from the estimation it is required to revise the estimated velocity in future sprints and if necessary update the release plan. There are a lot of possibilities that can apply to help fulfilling the contract and at the same time increase velocity. Removing any issue identified during a sprint is always a possibility.

2.3.14 Testing

Testing a project can be very different between different organisations. It depends on how many testers there are and how much automation can be applied. It is fundamental that the testers are not a part of the team. The testers will access the system in exactly the same way as the users would. The test team will catch bugs and the scrum team will create the bug-fix release version. The stage of releasing, catching bugs and re-release is known as the Acceptance-test phase. It is important to minimise this phase. Minimise the amount of time for this stage. The tester has a very important rule. Nothing will be considered done/finished until they agree with that.

2.4 Kanban

Kandan is a methodology that helps organise upcoming work. This process helps visualise the workflow of a specific project. The main goal of this methodology is to recognise the inconsistencies in the process and to find a way to overcome them.

Kanban follow **four essentials principles**[2]:

- **Start with what you are doing now**: This principle expressly says that it is important for the team/organisation to maintain the current methodology of work. In case of a necessary change, this change should only be applied to the active work flow.
- **Agree to pursue incremental, evolutionary change**: This principle strengthens the concept of small progression instead of extreme changes that can start problems between the team or organisations.
- **Initially respect current roles, responsibilities and job-titles**: The concept of no changes unless is necessary still holds in this principle. In case of vital changes, all the team should identify them. This way, any resistance or any fear of change can be overcome.
- **Encourage acts of leadership at all levels**: This last principle says that anyone within the organisation, no matter the role, can help improve the product/service of the organisation.

With this **six practise**[2] are also fundamental in the kandan method:

- **Visualise the flow of work**: Is important to have an idea of what must be done and what is in the process of being completed. The best way to have a notion of the state of the work is by having a graphic board, this allows the team to always see the progress.
- **Limit Work In Progress (WIP)**: This practice creates a limit of active tasks within the team. This forces the team to finish the active work before taking any

more new work. This step also helps give an idea of the capacity of work that the teams can handle in relation to the customer/stakeholders.

- **Manage flow**: In every stage of the work you can always keep track of what is being done and what still needs to be done. A workflow is set when WIP limits are set. It is possible to see what is the behaviour of the system with the established workflow. In case any adjustments are needed to reduce the time to complete the work and to improve the workflow, kanban allows that visualisation and easy identification of the issue/possible upgrade.

- **Make process policies explicit**: The creating of guidelines of visualisation, helps the team know the rules of how the work proceeds. When defined how the organisation of the work flow will happen, it is easy for everything to follow the same line of thought. By dividing the board in two/three columns where one of the column is the work to be done, the second is the work in progress and the third is all the work completed, a kanban board organisation is a good example.

- **Implement feedback loops**: Every time there is a feedback there is a change to improve. Having feedback loops allows a persistent system for improvement, fixing any issues that were identified in the middle. There is always a change to things not being clear enough for everyone, this creates an opportunity to resolve or explain any doubt created.

- **Improve collaboratively, Evolve Experimentally(using the scientific method)**: Kanban method is considered an improvement procedure. The main goal is to keep checking the progress of the workflow and always look for any needed improvement.

The Kandan graphic board allows full transparency of the work load distribution and current bottlenecks. With these principles and best practices, Kandan methodology encourages progressive improvements in the workflow of the organisation. This approach will not only improve the work flow but also create a reduction in the period of development (since the customer defines what needs to be done to deliver the final project) by adding value for the costumer.

Chapter 3

Requirement Analysis

In this chapter, the reader will find all the information of this system regarding all the specific requirements of the system. The chapter is essential to highlight the work required to be done in order to reach all the objectives proposed. The chapter is also divided in three main different chapters:

- **Functional Requirements**: This first section has all the functional requirements for this internship project to be successful.
- **Quality Attributes**: This second section will describe the architectural specifications of the system and will help evaluate it.
- **Product Constraints**: This last section will describe how the system must be developed.

3.1 Functional Requirements

Following the scrum methodology, the functional requirements will also include a note as stories.
As reference in the chapter 2, a requirement is divided with:

- **ID**: represents the identification of each story. Each story has a unique number.
- **Name**: brief description of the story. Needs to be clear enough so that developers comprehend what the story is about and distinct enough to be different from others.
- **Importance**: value set by the product owner that defines the importance of this story.
  - **High**: The requirement is necessary to have a working system. Without this requirement, the system can not be completed.
  - **Medium**: A requirement identified with medium priority, the system will improve significantly with the completion of this requirement. The system can still be completed without this requirement. This requirement must only be completed after all high requirements are completed.
  - **Low**: Low importance requirement has a small impact on the overall project. This requirement must only be considered after all medium requirements are completed.

- **Initial Estimate**: estimation of how much work the story will require to be implemented (this measure uses days as a metric).
- **Description**: in this field it is possible to find a small description of each requirement, and is possible to see who will have access to this requirement, what is the main goal and that is a benefit that can be taken from it and any other relevant information.
- **Notes**: in this session it is possible to find a brief description of the requirement in the following form:
  - **AS a** ‹user_that_most_benefit_from _this›.
  - **I want** ‹be_able_to_do_something›.
  - **So that** ‹can_take_this_benefit_from _it›.

3.1.1 Contextualisation

The project for this internship was to develop a system that can **evaluate a representation of a Contact Centre environment.** The hosting company required the following variables to be taken into consideration:

- **Schedule**: represents a set of shifts and agents available during a specific period.
- **Agent**: represents a working agent.
- **Shift**: represents a particular period of work that an agent has.
- **Queue**: represents a resource available in the Contact Centre that has the objective to process income calls.

3.1.2 Requirements identification

ID	1
Name	**Management of simulation input data that is responsible for representing a Contact Centre environment**
Importance	High
Initial estimate	3 days
Description	The system has to have a component responsible for the creation of relevant data(queue, shift, agent, schedule). This interaction to the system must be possible thought an defined endpoint.
Notes	**As the** user **I (user) want to** be able to insert and manage the information within the system. **So that** more workload is generated.

Table 3.1: Requirement 1: Creation of information

ID	1.1
Name	**Insertion of simulation input data that represents a Contact Centre from input**
Importance	High
Initial estimate	3 days
Description	The component responsible for the management of data(queue, shift, agent, schedule) has to be able to receive input by the user. This interaction to the system must be possible thought an defined endpoint.
Notes	**As the** user **I (user) want to** insert new information into the system. **So that** more workload is generated.

Table 3.2: Requirement 1.1: Insertion of information from input

ID	1.1.1
Name	**Modification of the simulation input data that represents a Contact Centre environment**
Importance	High
Initial estimate	3 days
Description	The component has to allow the user the destruction, insertion, access and modification of the different structure of data(queue, shift, agent, schedule). This interaction to the system must be possible thought an defined endpoint.
Notes	**As the** User **I (User) want to** modify any important structure of data in the system. **So that** the information is used with the updated changes.

Table 3.3: Requirement 1.1.1: Modification of the information

Queue

ID	1.1.1.1
Name	**Creation of a Queue**
Importance	High
Initial estimate	3 days
Description	The system must be able to create a new queue upon receiving a request on a defined endpoint.
Notes	**As the** user **I (user) want to** create a new queue **So that** more workload is generated.

Table 3.4: Requirement 1.1.1.1: Creation of a Queue

ID	1.1.1.2
Name	**Modification of a Queue**
Importance	High
Initial estimate	1 days
Description	The system should be able to modified a queue upon receiving a request on a defined endpoint.
Notes	**As the** User **I (User) want to** modify a queue **So that** the queue will satisfy all the requirements

Table 3.5: Requirement 1.1.1.2: Modification of a Queue

ID	1.1.1.3
Name	**Delete of a Queue**
Importance	High
Initial estimate	1 days
Description	The system must allow the deletion of a queue when receiving a request on a defined endpoint. The system must promptly validate the request The system must allow the deletion of a queue when receiving a request on a defined endpoint.
Notes	**As** the User. **I(User) want to** delete a queue. **So that** workload does not have unnecessary data

Table 3.6: Requirement 1.1.1.3: Delete of a Queue

ID	1.1.1.4
Name	**Read of a Queue**
Importance	High
Initial estimate	1 days
Description	The system must be able to access a queue details and it components. The system must allow this action after receiving a request on a defined endpoint.
Notes	**As the** User. **I(User)** want to access a queue and all her information. **So that** workload can be also checked and validated.

Table 3.7: Requirement 1.1.1.4: Read of the Queue

Agent

ID	1.1.1.5
Name	**Creation of a new Agent**
Importance	High
Initial estimate	3 days
Description	The system must be able to create an agent upon receiving a request on a defined endpoint.
Notes	**As the** User. **I(User) want** to create a new an agent. **So that** workload is generated and different types of agents can be used within the simulation.

Table 3.8: Requirement 1.1.1.5: Creation of a new Agent

ID	1.1.1.6
Name	**Modification of an Agent**
Importance	High
Initial estimate	3 days
Description	The system should allow the modification of an agent details/components. This task should be performed after receiving a request on a defined endpoint and validating the request.
Notes	**As the** User. **I(User) want to** modify an agent. **So that** workload data is correctly checked and modified.

Table 3.9: Requirement 1.1.1.6: Modification of an Agent

ID	1.1.1.7
Name	**Delete of an Agent**
Importance	High
Initial estimate	1 days
Description	The system must be able to delete an agent from the data. This interaction to the system must be possible thought an defined endpoint.
Notes	**As the** User. **I(User) want to** delete an agent. **So that** unnecessary workload is excluded from the system.

Table 3.10: Requirement 1.1.1.7: Delete of an Agent

ID	1.1.1.8
Name	**Read from an Agent**
Importance	High
Initial estimate	1 days
Description	The system must be able to access the detail of an agent after receiving a request on a defined endpoint.
Notes	**As the** User. **I(User) want to** access an agent. **So that** the data generated can be validated.

Table 3.11: Requirement 1.1.1.8: Read from an Agent

Shift

ID	1.1.1.9
Name	**Creation of a Shift**
Importance	High
Initial estimate	3 days
Description	The system must be able to create a shift. This task is important to be performed after receiving a request on a defined endpoint.
Notes	**As the** User. **I(User) want to** create a new shift. **So that** workload is generated.

Table 3.12: Requirement 1.1.1.9: Creation of a Shift

ID	1.1.1.10
Name	**Modification of a Shift**
Importance	High
Initial estimate	1 days
Description	The system should grant the possibility to modified an shift upon receiving a request on a defined endpoint.
Notes	**As the** User. **I(User) want to** modify a shift. **So that** workload data is correct.

Table 3.13: Requirement 1.1.1.10: Modification of a Shift

ID	1.1.1.11
Name	**Delete of a Shift**
Importance	High
Initial estimate	1 days
Description	The system must be able to delete a shift after receiving a request on a defined endpoint.
Notes	**As the** User. **I(User) want to** delete a shift from the system. **So that** unnecessary workload can be deleted from the system.

Table 3.14: Requirement 1.1.1.11: Delete of a Shift

ID	1.1.1.12
Name	**Read of a Shift**
Importance	High
Initial estimate	1 days
Description	The system must be able to access a shift details and components after receiving a request on a predetermined endpoint.
Notes	**As the** User. **I(User) want to** access a shift. **So that** I can check and valid all created data.

Table 3.15: Requirement 1.1.1.12: Read of a Shift

Schedule

ID	1.1.1.13
Name	**Creation of a Schedule**.
Importance	High.
Initial estimate	3 days
Description	The system must be able to create a schedule and generate it necessary components upon receiving a request on a defined endpoint.
Notes	**As the** User. **I(User) want to** create a new schedule. **So that** workload is generated.

Table 3.16: Requirement 1.1.1.13: Creation of a Schedule

ID	1.1.1.14
Name	**Modification of a Schedule**
Importance	High
Initial estimate	1 days
Description	The system should be able to handle request that required a schedule modification. This task should be performed after the system receives an endpoint request.
Notes	**As the** User. **I(User want to** modify the details from a schedule. **So that** workload matches the required data.

Table 3.17: Requirement 1.1.1.14: Modification of a Schedule

ID	1.1.1.15
Name	**Delete a Schedule**
Importance	High
Initial estimate	1 days
Description	The system must be able to allow the deletion of a schedule, this task is demanded by a request on a defined endpoint.
Notes	**As the** microservice management. **I want** to delete a schedule. **So that** unnecessary workload can be deleted.

Table 3.18: Requirement 1.1.1.15: Delete a Schedule

ID	1.1.1.16
Name	**Read from a Schedule**
Importance	High
Initial estimate	1 days
Description	The system must allow to external communication to access a schedule details when received a request throughout a defined endpoint.
Notes	**As the** User. **I(User) want to** read from a schedule. **So that** a can access all the data generated in that schedule.

Table 3.19: Requirement 1.1.1.16: Read from a Schedule

ID	1.2
Name	**Transmission of simulation input data**
Importance	High
Initial estimate	3 days
Description	The component responsible for the management of data(queue, shift, agent, schedule) has to be able to send those data for the component responsible for the simulation. This interaction to the system must be possible thought an defined endpoint.
Notes	**As the** system. **I(system)** want to sent the necessary information to the component simulation. **So that** the workload is validate thought a simulation.

Table 3.20: Requirement 1.2: Transmission of information

ID	1.3
Name	**Receive simulation input data from the first component(component responsible for the management of data)**
Importance	High
Initial estimate	3 days
Description	The component responsible for the management of data(queue, shift, agent, schedule) has to be able to receive the results from the component responsible for the simulation. This interaction to the system must be possible thought an defined endpoint.
Notes	**As the** component. **I(User) want to** be able to receive the results from a previous simulation request. **So that** workload is generated.

Table 3.21: Requirement 1.3: Receive information from another component

ID	1.4
Name	**Storage of simulation input data**
Importance	High
Initial estimate	3 days
Description	The component responsible for the management of data(queue, shift, agent, schedule) has to be able to persist all data received by the second component.
Notes	**As** the system. **I(system) want to** storage all the information that i receive. **So that** the workload is always store for future comparation.

Table 3.22: Requirement 1.4: Storage of information

ID	2
Name	**Simulation of the input data that represents a Contact Centre environment**
Importance	High
Initial estimate	3 days
Description	The system has to have a component responsible for the simulation of data(queue, shift, agent, schedule).
Notes	**As the** User. **I(system) want** validate the information receive by simulate the data has a real situation. **So that** workload is generated and different types of agents can be used within the simulation.

Table 3.23: Requirement 2: Simulation of the information

ID	2.1
Name	**Receiving inter-service communication**
Importance	High
Initial estimate	3 days
Description	The component responsible for the simulation has to be able to receive data(queue, shift, agent, schedule) from the component of the creation of data. This interaction to the system must be possible thought an defined endpoint.
Notes	**As the** User. **I(system) want to** be able to receive information from a different component. **So that** workload is generated.

Table 3.24: Requirement 2.1: Receiving inter-service communication

ID	2.2
Name	**Simulation input data processing**
Importance	High
Initial estimate	10 days
Description	The component responsible for the simulation has to be able to simulate by processing the data(queue, shift, agent, schedule) from the information received. This interaction to the system must be possible thought an defined endpoint.
Notes	**As the** system. **I(system) want to** process the information receive to validate the information after using it within a simulation. **So that** the system creates the final result of the processed data.

Table 3.25: Requirement 2.2: Data processing

ID	2.3
Name	**Inter-service communication reply**
Importance	High
Initial estimate	3 days
Description	The component for the simulation has to be able to send the results of the previous simulation to The component responsible for the management of data. This interaction to the system must be possible thought an defined endpoint.
Notes	**As the** system. **I(system) want to** send back the result of my simulation. **So that** the user can access this result from the component of the creation of information.

Table 3.26: Requirement 2.3: Inter-service communication reply

3.2 Quality Attributes

Quality attributes enables the evaluation of how well defined and developed the system is. These attributes can evaluate the system behaviour in any given category, it is possible to describe what the system will do and this enables us to see the system conduct through a specific action. We can see the different impacts that wish category may have in part of the system as a whole. There is one quality attributes that is important in the context of this work:

1. **Reliability**: this quality attribute represents the availability and fault tolerance given by certain components of the system.
2. **Availability**: this attribute represents the system accessibility when acessed externaly by a endpoint.

ID	1
Description	The system can not be blocked by a faulty/failed request.
Quality attribute	**Reliability**
Validation	The use of an messaging-broker will ensure this property.

Table 3.27: The system can not be blocked by a faulty/failed request.

ID	2
Description	The system has to be active for twenty-four hours/seven days a week.
Quality attribute	**Availability**
Validation	The system has to be able to receive upcoming at any time requests. The system has to be available 99.999% of time.

Table 3.28: The system has to be active for twenty-four hours/seven days a week.

3.3 Product Constraints

There were same product constraints related with the product to be developed:

- **Technology**:
  - The system must be developed in Java 11[5], the product needs to be developed using the Quarkus[8] framework and the usage of Postgres[7] database.
  - All components need to be communicate by Web service REST.
  - The company in question requires the use of GitHub as a way to share code and to facilitate the organisation of tasks within a team.
  - The implementation needs to follow reference implementation given by the company. All code needs to be loaded to given/created repository in GitHub[4] shared with the advisor from the company.
  - The two component need to be developed using microservice architecture.
- **Time**:
  - The deliver of this document is defined at 29/06/2020. The product must be complete until then.

This page is intentionally left blank.

Chapter 4

Software Architecture

This chapter will describe all the decisions that were made to define the architecture of the system. The chapter is also divided into three parts:

- **Decision**: This section will identify all the technologies used in the solution.
- **General View**: Explain how each system module works and how each module interacts.
- **Components And Connectors View**: This section describes the different interactions between components and layers.

4.1 Decisions

This chapter explains what decisions **were made** and which technologies **were used**. The design of the architecture takes into consideration the technologies used by the hosting company. Three technologies are chosen for the three main topics:

- Communication
- Information storage
- The architecture of software development

Communication

RabbitMQ is an message queue broker. This queue allows communication between connected applications with the use of a channel. It is possible to have more than one channel in each application. In each channel, there is a sender and a receiver. If there are two applications expecting a message with the same channel, this protocol automatically balances the load between these two applications. In case of scaling the project, RabbitMQ allows the automatic balance of workload between existing microservices.

Storage information

For storage purposes, a **Postgres** database was chosen. Postgres is a relational database system. Related databases can manage concurrency and work with different types of workloads. It allows multiple users and interaction occurring at the same time.

The architecture of software development

Each component of this project is designed as a **microservice**. Microservices have a lot of advantages and disadvantages as already discussed in the second chapter of this document. The use of this structure was not only a requirement by the hosting company, but also has a tremendous beneficial effect on the processing of the information and its validation.

API framework

Quarkus has a lot of benefits that also help this project. Quarkus allows a fast start of an application, helps by optimising container-native applications, improves java applications performance and improves developer productivity by allowing a live view of any change on the code.

4.1.1 General View

A good architecture is important for an effective development of any project. The two required components are now defined to be developed as a **microservice**. Since there were two components, there are now two microservices with two different functions.

The **first component** is in charge of the management of the information in the system. This microservice manages all the information within the system, and allows the modification of the principal data structure. From now on, this microservice is called **Management Microservice.**

All the information generated in this first microservice is the setup to be sent to the second component.

This **second component** is responsible for simulating a Contact Centre‘s real events with the information received. Therefore, this microservice is now called **Simulation Microservice.**

After this simulation ends, the result from the comparison of the expected results with the generated results is returned. Management Microservice will receive this result and store it.

4.2 Components and connector view

This architecture shows the main use of these technologies and their interaction with each component of the project.

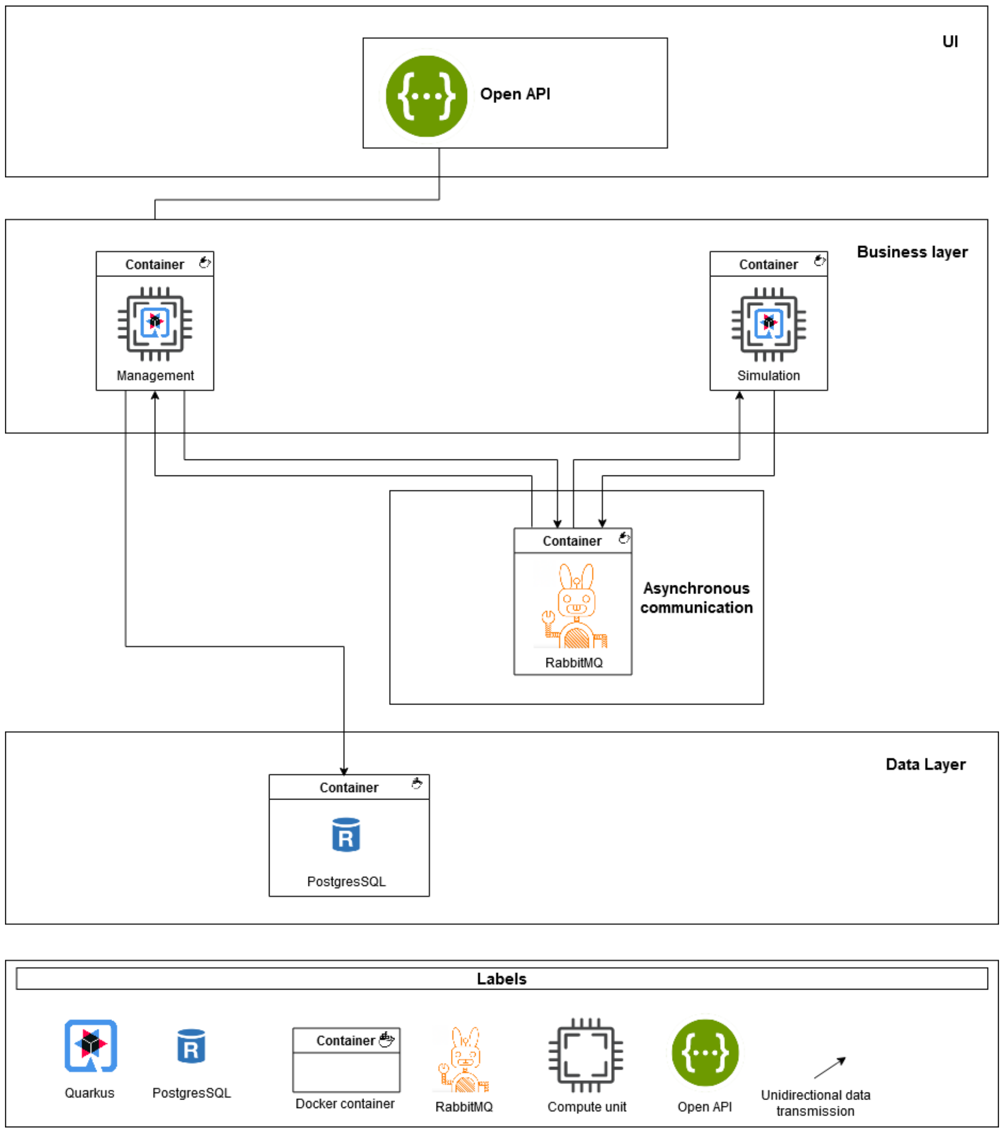


Figure 4.1: Connector and components view of the project

There are four layers:

- **User interface (UI) layer**: This layer is the start point of interaction with the system. It is through this layer that the communication is made.
- **Business layer**: Two microservices are located in this layer, Management Microservice and Simulation Microservice .
- **Asynchronous communication**: This layer is responsible for the communication between the two microservices. There is no direct connection between both microservices, all the interactions are required to pass through this layer.
- **Data layer**: This last layer, the database, stores every component that stores/produces data.

This page is intentionally left blank.

Chapter 5

Project Planning

This chapter will describe all the division of steps and how they were scheduled.

5.1 Methodology

In this section will be identify all frameworks and models that the project followed for a more complete and efficient project planning.

5.1.1 Software Development Life Cycle (SDLC)

The software management will follow the interactive model of Software development life cycle (SDLC). This model aims to construct a high quality software at a lower cost in a limited period of time. The beginning of this type of development is focused on the simplification of the implementation and as the cycle stage progresses the complexity increases, following the agile methodology of development. For best results, this cycle has eight stages:

1. **Initiation**: This stage identifies the problems existing with the current solution. It also establishes the strength of this project in comparison with the existing software and what is the main goal of improvement in comparison to them.

2. **Planification**: This stage defines the type of software/hardware that is being used in the development. Most of the documentation is built in this stage.

3. **Requirements**: The requirements of the project are defined. It defines what is sufficient in the early stage of the project. As the cycle continually repeats, the complexity of the requirements increase.

4. **Design**: All the technical demand like programming language and data layers are defined in this stage.

5. **Implementation**: The implementation of the code begins in this stage.

6. **Verification**: This is a testing stage; any bugs or issue are now identified. New corrections are set in order to be fixed in the next phase of this cycle.

7. **Evaluation**: This stage is identifies the progress made in the project. It also establishes what is needed to be improved and what changes are required to be adjusted.

8. **Deployment**: When the project is considered finished and complete, this stage is the last stage that the project will be in.

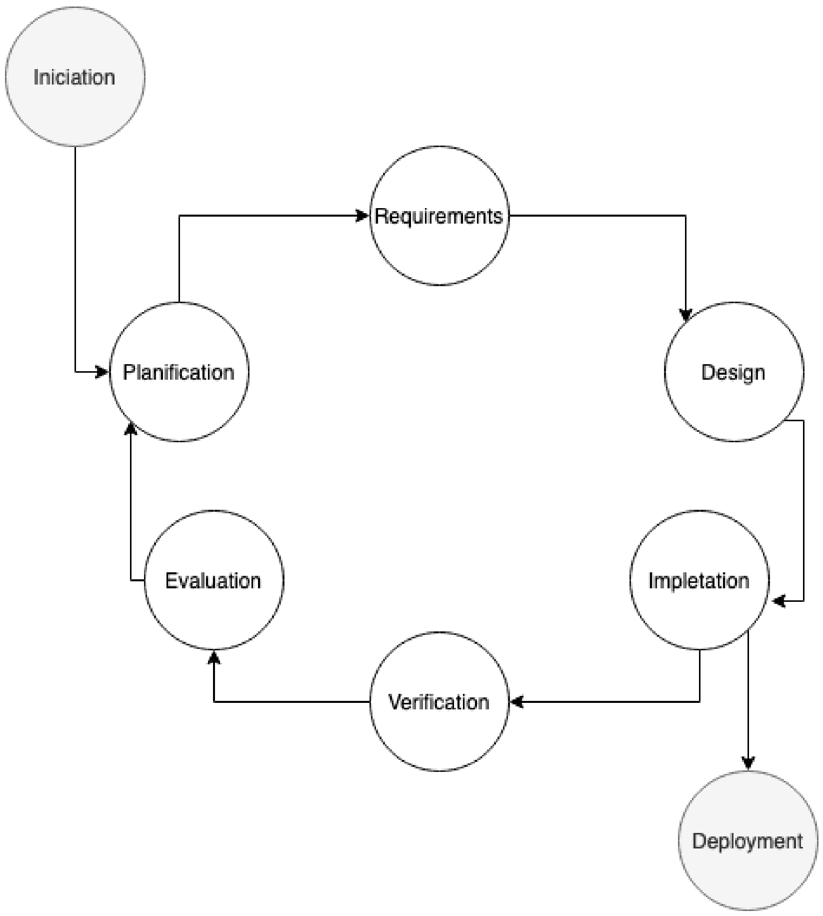


Figure 5.1: Software development life cycle

5.1.2 First Semester Planning

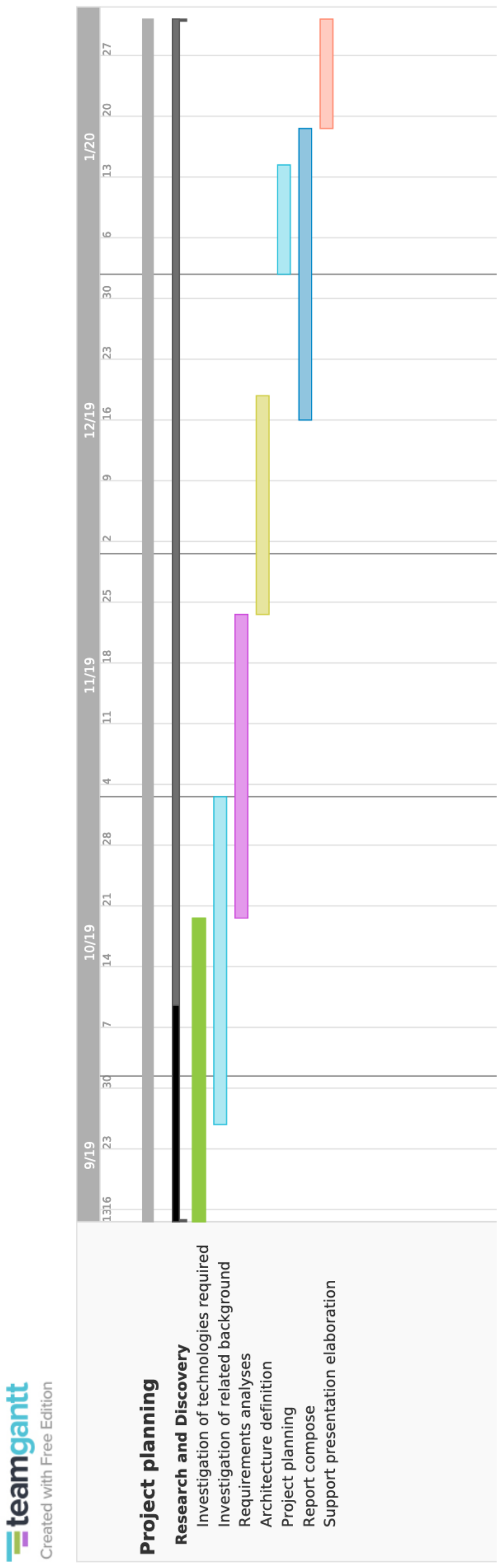


Figure 5.2: First semester's Gantt graphic

The focus of the first semester was document research and planing. All the research related with the discovery of the right technology to be used and how it is applied was done during this semester. The investigation of the related background and its main goal and the terminologies used were also covered.

The semester had the **Plan**, **Requirements** and **Design** stage has main goal.

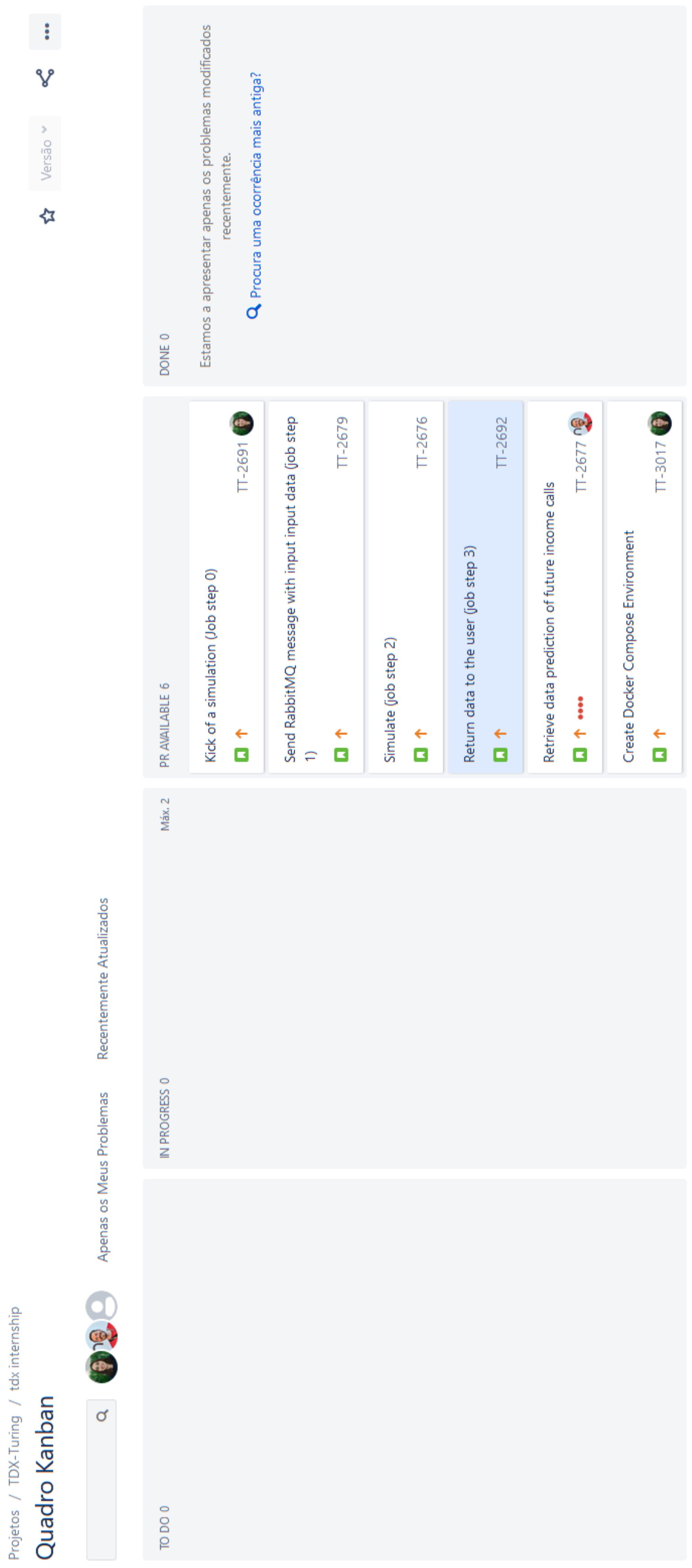
5.1.3 Second Semester Planning

The focus of the second semester was the **Implemention** and **Validation** of the project.

The methodology used to organise the tasks was the Kanban board. The software called Jira was used as auxiliary support to create this Kanban board.

This board has four columns:

- What is needed **to be done** in this semester.
- What is **in progress** at the time that the Kanban board is read.
- What is **ready to be reviewed** at the time that the Kanban board is read.
- What is **done** at the time that the Kanban board is read. A task that does not require future changes or testing is considered complete. A task is done when the implementation is finished and the unit tests are already completed.



Projetos / TDX-Turing / tdx internship

Quadro Kanban

Versão

Apenas os Meus Problemas

Recentemente Atualizados

TO DO 0

IN PROGRESS 0 Máx. 2

PR AVAILABLE 6

Kick of a simulation (Job step 0) TT-2691

Send RabbitMQ message with input input data (job step 1) TT-2679

Simulate (job step 2) TT-2676

Return data to the user (job step 3) TT-2692

Retrieve data prediction of future income calls TT-2677

Create Docker Compose Environment TT-3017

DONE 0

Estamos a apresentar apenas os problemas modificados recentemente.

Procura uma ocorrência mais antiga?

Figure 5.3: Kanban board on the second semester

This project was divided into two main components, because of that, its timeline was equally divided.

The first microservice was developed in the first place since the second component depends on the input sent by that microservice.

During the first stage of development, some research was required to apply the best code practice. In addition to this research, the use of Quarkus as framework also required an investigation. Besides these two occasions, the first part of the second semester went smoothly.

The second half of the semester focuses on the development of the second microservice. The different task was divided along the months. It is possible to see the division the task in the following image.

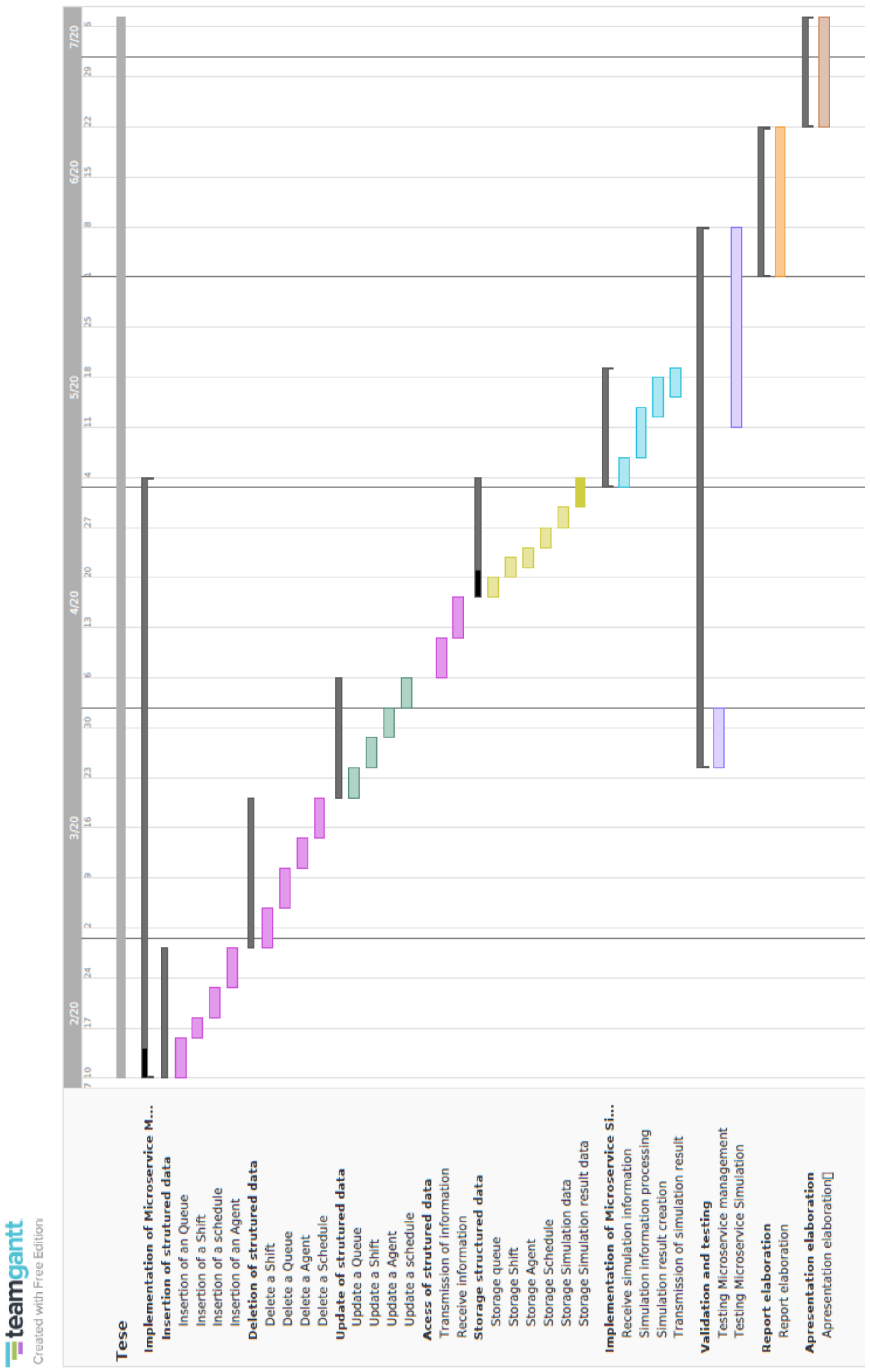


Figure 5.4: Second semester's Gantt graphic

The semester was divided into tasks. There were the only two tasks being done at the same time. The main goal of this second semester was, by the end of this internship, to complete all tasks. This development had a similar approach to the Agile methodology.

The intern was self-organised throughout the entire development.

Throughout the internship, the main goal of the project changed slightly. The modification was a strategic decision made by the company. The first delivery of this documentation had the future implementation described with the intent of the creation of two services, the first one to allow the user to insert data in the system and the second one to generate an ideal Contact Centre environment with specific information in consideration received from the first one component.

Talkdesk with this new change to the final project has now the goal of the validation of the information received by the first microservice. The main objective of the first microservice is still the same, allow the user to create data in the system.

Chapter 6

Risk Assessment

This chapter is about identifying possible issues. Being conscious of potential problems helps minimise them and allows the creation of a backup plan in case any problems occurring. This chapter is very important part of the project planning.

Each risk will be described using seven fields:

1. **ID**: This field identifies the risk with a unique code.
2. **Risk**: Every risk is described with a short description.
3. **Description**: The risk detail characterisation can be found in this field.
4. **Probability**: Each risk has a probability of occurring. This probability is divided into four different names:
   - **Minimum**: The probability of this risk occurring is less then 25%.
   - **Low**: Probability between 26% and 50%.
   - **Medium**: This probability occurs when the risk has 51% to 75% of probability of occurring.
   - **High**: A risk has high probability when it has more than 75% probability of occurring.
5. **Impact**: Each risk can have a serious impact in the project. This impact can affect the project in four different ways:
   - **Low**: A risk with low impact means the development/scheduling of the project may need a slightly change.
   - **Medium**: Medium impact can require a visible change on the development/scheduling of the project.
   - **High**: High impact risk requires mandatory restructuring of the system.
   - **Critical**: A critical risk that can compromise the whole project.
6. **Mitigation Plan**: A mitigation plan is a backup that when activated can minimise the risk.
7. **Time-frame**: To really understand how long a risk requires to be fixed/resolved, the time-frame is divided into four groups:

- **Immediate**: When a risk takes less than a week to be fixed.
- **Short**: A risk that takes between one to two weeks to be solved.
- **Medium**: When a risk takes between two weeks to a month to be resolved we call it a medium time-frame.
- **Long**: A risk that takes between one to two month to be fixed.

The table below is useful as a guide to better understand the danger that the risks are to the project. Each risk is mainly described with the combination of an impact and a probability. The table is divided into colours that until reaching each opposite points of the table the colour is less intensified, the more intense the colour represents its meaningfulness. The colour green represents small impacts in the project while red shows that the risk has a critical impact on the project and system. All risks represented by the colour level yellow or above have a pre prepared mitigation plan, this plan will minimise the change of failure or non success of the project.

High	🟨	R6;R7;R4	🟥	🟥
Medium	🟩	R3; R5;R8	🟧	🟥
Low	🟩	R1;R2;R9	🟨	🟧
Minimum	🟩	🟩	🟩	🟨
Probability Impact	Low	Medium	High	Critical

Table 6.1: Matrix of the risk identified

6.1 Early Stage Risks

The tables described above represent all the possible risk in the early stage of the project.

ID	1
Risk	**Data input per entities being wrongly written**
Description	For each simulation the Schedule Microservice requires external data insertion. It requires data that defines the specification of each agent/shift/queue/schedule. If any of this input data is corrupt or written incorrectly all the experience becomes compromised.
Probability	Low
Impact	Medium
Mitigation Plan	None
Time-frame	Immediate

Table 6.2: Risk 1: Data input per entities being wrongly written

ID	2
Risk	**Inaccurate estimation of effort**
Description	A good performance and organisation of this internship requires a good division of tasks. If one of these tasks takes more time, a delay to the other task will be shown.
Probability	Medium
Impact	Medium
Mitigation Plan	None
Time-frame	Short

Table 6.3: Risk 2: Inaccurate estimation of effort

ID	3
Risk	**Change of requirements in the middle of development**
Description	Good specification of requirements is fundamental for a better understanding of the project. If the requirements change, the predefined architecture of this project could be compromised. The technologies chosen may be submitted to change.
Probability	High
Impact	Medium
Mitigation Plan	The intern has to prioritise the requirements into better fulfilling the goal of this project.
Time-frame	Short

Table 6.4: Risk 3: Change of requirements in the middle of development.

ID	4
Risk	**Being part-time in the first semester**
Description	The internship involves 16h dedicated to the development of the project. The developer is dedicating time to other tasks that are not related to the project. This can lead to some delay in the scheduled tasks during this period of time.
Probability	Medium
Impact	Medium
Mitigation Plan	None
Time-frame	Medium

Table 6.5: Risk 4: Being at part-time in the first semester.

ID	5
Risk	**Inexperience in developing microservices**
Description	The developer has no experience in developing projects using the technique of microservices. This risk can lead to the same delay in the schedule task due to the required time needed to get familiar with this technique.
Probability	High
Impact	Medium
Mitigation Plan	The intern must search and get him self familiar with this type of technology.
Time-frame	Immediate

Table 6.6: Risk 5: Inexperience in developing microservices.

ID	6
Risk	**Inexperience in using Docker**
Description	The developer has no experience in developing projects using Docker. This risk can lead to the same delay in the schedule task due to required time associated with getting familiar to this technique.
Probability	High
Impact	Medium
Mitigation Plan	The intern must search and get him self familiar with this type of technology.
Time-frame	Immediate

Table 6.7: Risk 6: Inexperience in using Docker.

ID	7
Risk	**Since the project uses an agile approach, the requirements can change at any time.**
Description	An agile methodology defends the continuous improvement and flexibility within the development of the project. Changing requirements throughout the development process can happen depending on the needs of the client (Talkdesk).
Probability	Medium
Impact	Medium
Mitigation Plan	None
Time-frame	Short

Table 6.8: Risk 7: Since the project uses an agile approach, the requirements can change at any time.

6.1.1 Risk identified on the development stage

ID	8
Risk	**Project goals change**
Description	A change in the goal of the project can lead to extra work. A new set of requirements must be defined and sometimes research is necessary.
Probability	Low
Impact	Medium
Mitigation Plan	None
Time-frame	Short

Table 6.9: Risk 8: Project goals change.

ID	9
Risk	**Encountering unresolved errors with the framework used**
Description	When using an updated version of a framework with the use of combined technology sometimes incompatibilities can be found.
Probability	Low
Impact	Medium
Mitigation Plan	None
Time-frame	Short

Table 6.10: Risk 9: Encountering unresolved errors with the framework used

ID	10
Risk	**Inexperience using RabbitMQ**
Description	An excessive amount of requirements can not only lead to a confused architecture but also to an excessive amount of work and delay in important schedule tasks.
Probability	Low
Impact	Medium
Mitigation Plan	The intern must prioritise research and a well defined and organised plan that allows the extra time to be more informed in the technology, and that does not compromise the internship.
Time-frame	Short

Table 6.11: Risk 10: Inexperience using RabbitMQ.

ID	11
Risk	**Unfamiliarity with new interfaces on Java 11**
Description	Due to the lack of experience of the intern, he/she may not be familiar with some interfaces that are more appropriate.
Probability	Low
Impact	Medium
Mitigation Plan	The intern must research in case of doubt or ask the hosting company adviser for advice.
Time-frame	Short

Table 6.12: Risk 11: Unfamiliarity with new interfaces on Java 11.

6.2 Risks impact

Most of the risks identified in this chapter, after and before the development have not occurred during this internship. The lack of experience with RabbitMQ was defeated with the time spent learning this message-broker software. With this time and training exercise, this situation was overcome, and the project timeline was not affected.

An investigation of existing alternatives conquered the risk number 11. Same investigation time was necessary to be able to find a new solution, but the project was not affected by it.

Unfamiliarity with the different interfaces provided by version eleven of java was solved by the guidance from the hosting company adviser. The project was not affected since the risk was identified, and the developer investigates and learns the best interface to use to develop this project.

The overall experience and identification of risks have not affected this project negatively. The impact of these four risks was minimal to this project because of the fast response to them.

This page is intentionally left blank.

Chapter 7

Implementation Specification

The chapter of Implementation Specification describes the configuration necessary and implementation regarding the different components of this project.

This chapter includes three sections:

1. Management Microservice
2. Simulation Microservice
3. Enviroment

In the Management Microservice, the user inserts information for the creation of agents, queues, shifts and schedules. These four main entities generate a vision of essential information needed to manage a Contact Centre.

The second microservice, Simulation Microservice, validates if the information inserted in the system has a positive/negative impact compared to the expectations set for this collection of information. The aim of this internship project was to develop a system that can **evaluate** any schedule when compared with a given **expected service level**.

The system is divided into two microservices. Each service has different functionality. The first microservice, called **Management Microservice** has as main goal to manage all the information within the system.

The second microservice, called **Simulation Microservice** is responsible for processing the information provided from the first microservice. In return, this microservice reports the validation made with this information when compared to the expected results.

After this response, the first microservice has the **classification** from the service level expected of the schedule sent.

Since the Management Microservice has to represent a real schedule created within a contact centre, it is essential to take into account the **four primary entities** that any contact centre is dependent on:

- **Agent**
- **Queue**
- **Shift**

- **Schedule**

Considering that this project's goal is the representation of an ideal contact centre system, each entity only has the necessary components that better reproduce the rule within this system.

- **Queue**: A queue is an object developed by the intern that represents a stack of tasks to be completed. A queue is characterised by different fields
  - **Identification Code(ID)**: Each queue can be identified with a number. With this number every list (list of tasks that needs to be completed, list of tasks completed) that is generated after the end of each active hour, the queue can be identified with this number.
- **Agent**: An agent is responsible for dispatching a task. An agent has a code of representation. An agent also has a shift per working day. These entities can also be described by the following fields:
  - **Identification Code(ID)**: An agent is represented by this identification code. Every task completed by this agent can be identified by this code.
- **Shift**: The shift represents the active hours in the system that an agent has to have per day. This also has a specific representation of the following fields per agent.
  - **Identification Code(ID)**: An integer represents a unique code for each shift in the database. This integer is automatically increased when inserting in the database.
  - **Hour That An Agent Start To Work**: A shift starts at a specific hour.
  - **Hour That The Agents Ends The Shift**: A shift ends at a specific hour.
  - **List of Breaks (Non Active Working Hours) That An Agent Can Have In A Specific Shift**: An agent can have a specific amount of hours that is still on the system but not receiving any task. This action is called Break. The list of hours on break that an agent can have per day is defined in a shift.
- **Schedule**: A schedule is divided into a set of shifts and queues. A schedule with a period of time. This depends on the setup configurations of this entity.
  - **List Of The Shifts Per Each Agent Per Period Of Time**: A schedule has all the shifts that will occur during that period.
  - **Starting Instant**: This field represents the date and hour that the shifts represents in this scheduled start.
  - **End Instant**: This field represents the end date for the last shift represented in the entity.
  - **Identification Code(ID)**: An integer represents a unique code for each schedule in the database. This integer is automatically increased when inserted in the database.

7.1 Management Microservice

The implementation of this microservice allows easy management of all the data that is stored in the database.

There are three types of actions executed with this microservice:

1. Insertion of information relative to the four entities referenced.
2. Request for simulation with the information already existent in the database.
3. Request a new simulation with specific information given by the user.

Those request are made by RESTFull API[1] specified per each action and is required to submit in attachment JSON object.

When **insertion the information** in the system, validation is triggered. If the insertion data is validated, the method returns the code 200(ok)[9]. If the validation is not successful, a 400 error is returned.

When **requesting a simulation using data already insert in the system**, a JSON object is received with the specification of the expected result, from which the data of this period should be extracted from the database and the specific affluence that this information should be submitted.

With the last option, **every information needed to be tested out is inserted by the user.** The JSON object needs to have a schedule defined by the user, the expected result, and the specification of the affluence that this data needs to be submitted.

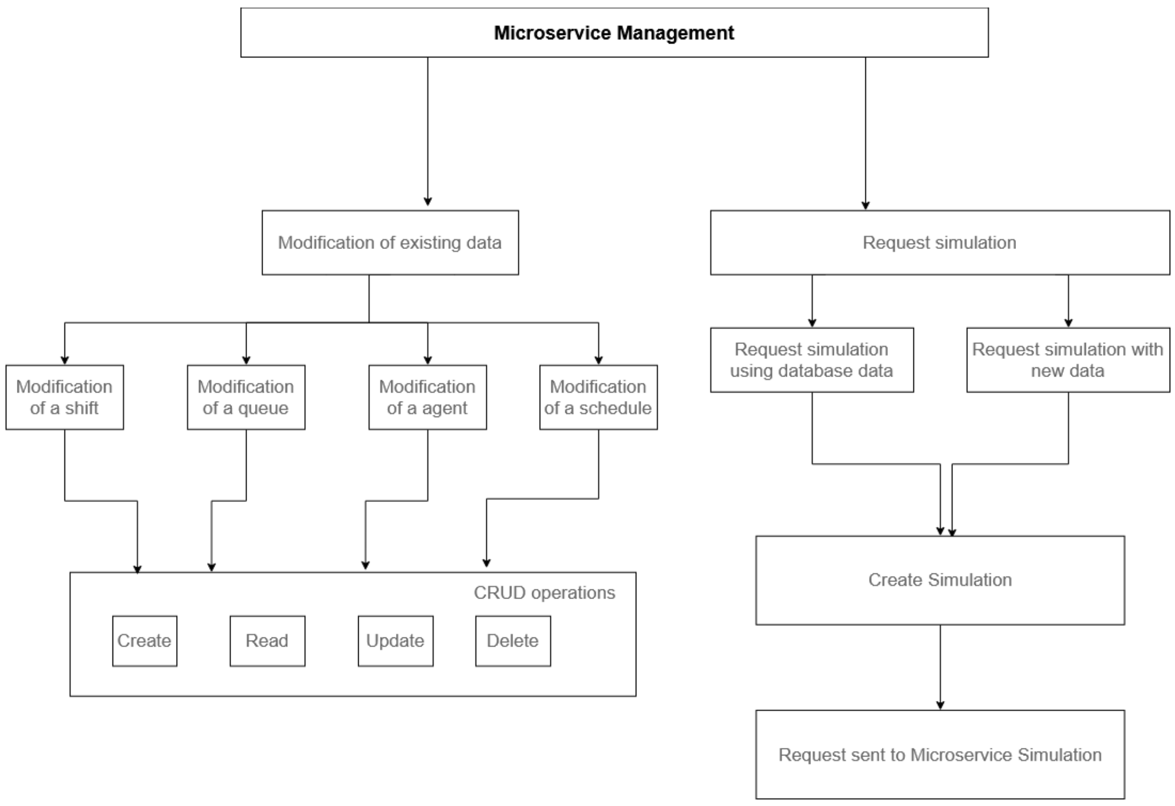

Microservice Management
Modification of existing data
Request simulation
Modification of a shift
Modification of a queue
Modification of a agent
Modification of a schedule
Request simulation using database data
Request simulation with new data
CRUD operations
Create
Read
Update
Delete
Create Simulation
Request sent to Microservice Simulation

Figure 7.1: Roadmap of the Management Microservice .

7.2 Simulation Microservice

This microservice has a fundamental rule in the project because it validates a schedule by simulating possible behaviours and compares it with the final result expected defined by the user.

The algorithm **"Discrete-event simulation"** is used to simulate an environment with the following events:

- Call arriving.
- Agents arriving at the queue.
- Agents terminating their shift.
- Agents are starting or finishing their breaks.

When using this algorithm, call abandonment was not considered. For the generation of a call arriving instantly, the arrival is evenly distributed within the duration of each simulation. The duration of each call is generated using an exponential distribution with an average value that is included in the user input. The simulation receives a time-series where every entry represents a spot in time. Each shift has its start and finish that can or not coincide with the time of each time-series entry, and the same goes for the generated calls. When simulating, this microservices generates each event represented above and creates new ones depending on the type of event handling. The final result is a list of events that were created and generated while the simulation passes each time-series entry received by an input. At this stage, the events that we need to analyse are the calls handling events. They will allow seeing the average of time that the calls were on queue and it is possible to compare it to the expected. The simulation result object has two components:

- A **list of integers** that each slot represents a classification (if a call has lower handle time compared to the average input given by the user, one will be representing the call) of a call.

- A **list of integers** that each slot represents the time of waiting for a call. Every slot has the time in seconds that each call had to be on queue.

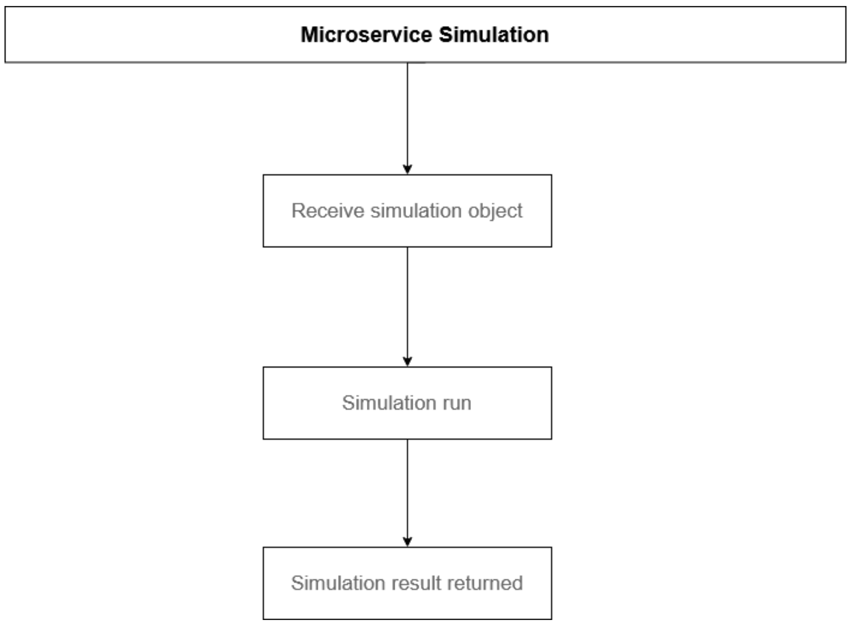


Figure 7.2: Roadmap of the Simulation Microservice

7.3 Environment

This third component of this project was created to help the deployment of the project and their depending part.

The environment is composed by a file called docker-compose.yml that when called the project becomes fully functional.

With the help of the framework called Docker, it is possible for this file to deploy an image of:

1. **rabbitmq3**: This container represents the image defined by the developer that configures RabbitMQ.
   rabbitmq3 represents the container name for this image of RabbitMQ.

2. **management**: This container represents the image of the microservice Management and the configuration. The ports that allow the access of this microservice are defined here. The environment variables to link the microservice to the database and

the RabbitMQ server.

3. **simulation**: This container represents the image of the microservice Simulation. Similar to the microservice above, it also defines:

    - The ports that allow access to this service.
    - The environment variables configuration.
    - The dependencies to other services.
    - To restart the service in case of failure.

4. **db**: This last container has the configuration of the database. It defines the image to use to create the database. It also describes the environment variables needed like the password and the username. In this container, the last section identifies the ports where the database is available.

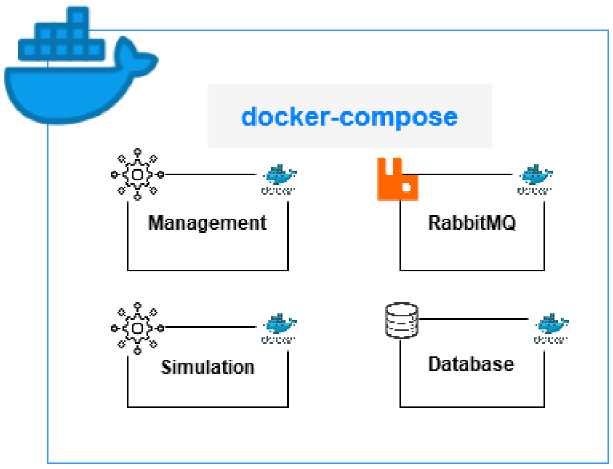


Figure 7.3: Docker-compose representation

Chapter 8

Validation

In the chapter is possible to find all the information related to the validation of the implementation.

8.1 Tests

Software unit testing helps to validate the logic of a particular piece of the program/functionality.

Black box testing was used to validate this project. It is crucial to this project the functional part works with no flaws. Since the primary goal of this project is to have a functional system that can validate data, it is crucial to be sure if the results can be trusted.

The project is tested with border not acceptable input values, the minimum accepted value and the maximum allowed value. Also, to see how the project behaves with unaccepted values and how it protects from failure, these values were used within the tests.

Both components have specific functionalities and a group of function and classes that constitutes each component.

In both components, unit tests and integration test were made to better search for any flaws. The framework used to create them is JUnit tests.

8.2 Test example

In this section is showed an example test made to validate a simulation request. The process of sending information from the microservice management to the second microservice return the results. To start a schedule need to be defined. To be able to establish a schedule, it is required to set shifts first.

In this case, the following shifts were defined:

- **First shift**: Starts at 14h and ends at 15h.
- **Second shift**: Starts at 14h30 and ends at 15h30.
- **Third shift**: Starts at 16h and ends at 17h.

The **sample interval** (the time that each simulation will take in consideration) fifteen minutes.

The last component to be defined is the Time Series. This component is a list of Time Series Entry (has called within the implementation). A time-series entry requires the following fields to be defined:

- **Interaction volume**: Each time-series entry has a specific number of calls to be tested.
- **Average handle time**: The average of time from the call is arriving and becoming solve.
- **ID**: Each time-series when first saved in the database, a particular ID is generated. Since this component does not affect, this will not be considered.
- **Timestamp**: Each time-series starts at a specific day and day time.
- **Queue name**: Since all data represents information from a queue work is essential to have a reference of wish information represents a queue.

The follows tables will represent the values that this test is will consider in the time-series. In the case three time-series entry will be consider.

Components	**Time-serie details specification**
ID	1
Interaction volume	5
Average handle time	4
Timestamp	2020-06-21 14h00:00
Queue name	Queue A

Table 8.1: Time-series entry 1.

Components	**Time-serie details specification**
ID	2
Interaction volume	0
Average handle time	4
Timestamp	2020-06-21 15h15:00
Queue name	Queue A

Table 8.2: Time-series entry 2.

Components	**Time-serie details specification**
ID	3
Interaction volume	2
Average handle time	4
Timestamp	2020-06-21 16h00:00
Queue name	Queue A

Table 8.3: Time-series entry 3.

This project uses an exponential distribution to generate a call duration. After more than twenty call duration created, a random group of seven calls was chosen. This simulation will have **seven call arrivals**. **Five** in the first time-series entry, **none** on the second and for last **two** on the third. The times will be the following set:

Time-series entry 1	**Call duration(in seconds)**
Call 1	202 seconds
Call 2	170 seconds
Call 3	105 seconds
Call 4	1 second
Call 5	871 seconds

Table 8.4: Table of call duration for the time-series entry 1.

Time-series entry 3	**Call duration(in seconds)**
Call 1	39 seconds
Call 2	2 seconds

Table 8.5: Table of call duration for the time-series entry 3.

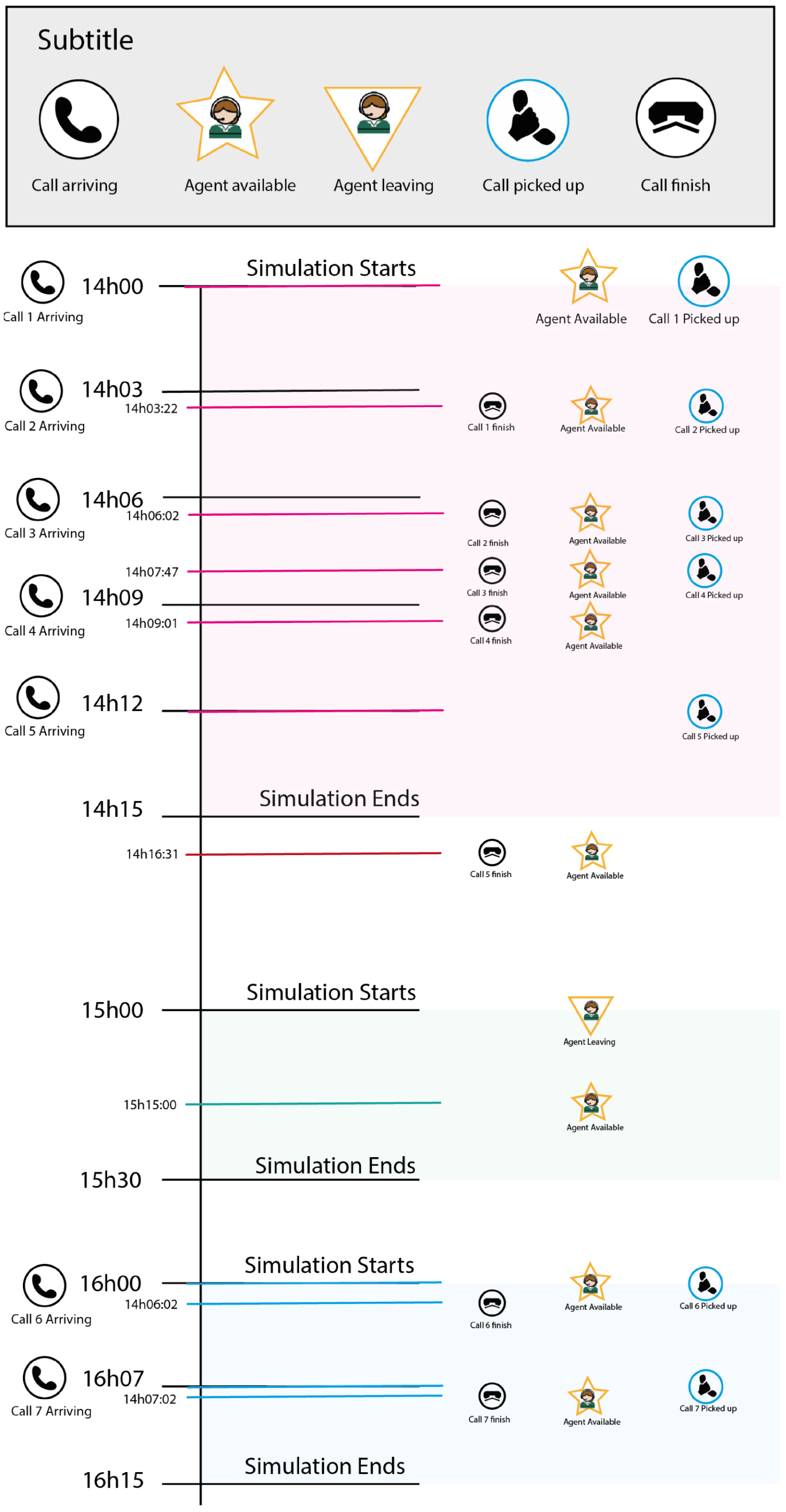


Figure 8.1: Representation of the timeline of the simulation.

The scheme represented in helps to have a visualise the timeline of the example simulation. There are three primary colours the divide the timeline into three parts. Each part represents a time-series entry per section, and there is a specific number of calls arriving and shifts available. In the first slot with the colour pink, we have five calls equally distributed.

- The first in at 14h00.
- The second is at 14h03.
- The third is at 14h06.
- The next one is at 14h09.
- The last one is at 14h12.

Per time-series entry there is a set number of volume, and there is also a stipulated duration for this simulation. To determine the time between calls, the following formula is applied:

- Simulation_time/calls_volume

In this case, would be 15min/5 calls with the final results as 3min. The list of calls arriving set above has 3min between each call.

Within the scheme represented, there is also a pink line with time and symbols. The line has the time attached of the end of the call picked before. The symbols at the front of the line represent the events that are generated with the line. The second line of the pink section is possible to find:

- A new agent is available.
- The call picked before being finished at 14h03:22.
- A new call being picked up, in this case, the second call that arrives at the system.

Same cases, the pink line only represents the call end because, at that time, the agent ends his shift. Other cases, an agent is waiting for a call and when it arrives the event of picked the call is generated.

With resemblance with the section pink, the section green and blue behaves in the same way, each of their lines represents a call being picked up of finished. In figure 8.2, the difference between the lines is just the colour. This choose was made to distinguish the sections better.

An important detail to notice is there is a red line between the sections. The algorithm required processing events within a duration time. This line represents an event created. Since it is outside of any of these sections, the event is not considered.

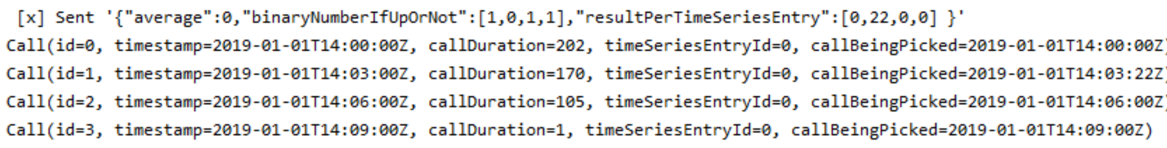
The expected result of the problem is the processing of four initial calls and the last two calls.

- The first call is picked right away and has no waiting time. Service level is within expected.
- The second call has to wait twenty-two seconds to be picked up. The service level is not what is expected. The service level was higher.
- The third call was picked right away. The service level was within the expected.
- The last call processed within the pink section was also pulled right away. The service level was within the expected.

- The last call was picked up, but the event of finishing the call was not processed. This finish call event was marked outside of the pink area and is not in any others sections.

The last two calls from the blue section:

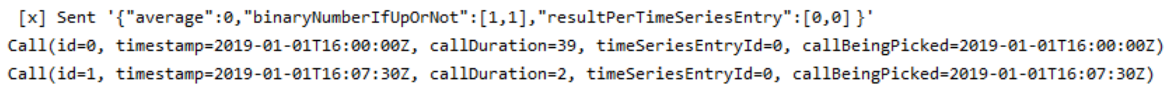
- The first call was picked right away. It had no waiting time. The service level is within the expected.
- The last call was also pulled immediately. The service level is with the accept one.



```
 [x] Sent '{"average":0,"binaryNumberIfUpOrNot":[1,0,1,1],"resultPerTimeSeriesEntry":[0,22,0,0] }'
Call(id=0, timestamp=2019-01-01T14:00:00Z, callDuration=202, timeSeriesEntryId=0, callBeingPicked=2019-01-01T14:00:00Z)
Call(id=1, timestamp=2019-01-01T14:03:00Z, callDuration=170, timeSeriesEntryId=0, callBeingPicked=2019-01-01T14:03:22Z)
Call(id=2, timestamp=2019-01-01T14:06:00Z, callDuration=105, timeSeriesEntryId=0, callBeingPicked=2019-01-01T14:06:00Z)
Call(id=3, timestamp=2019-01-01T14:09:00Z, callDuration=1, timeSeriesEntryId=0, callBeingPicked=2019-01-01T14:09:00Z)
```

Figure 8.2: Result from simulating with only the first time-series.

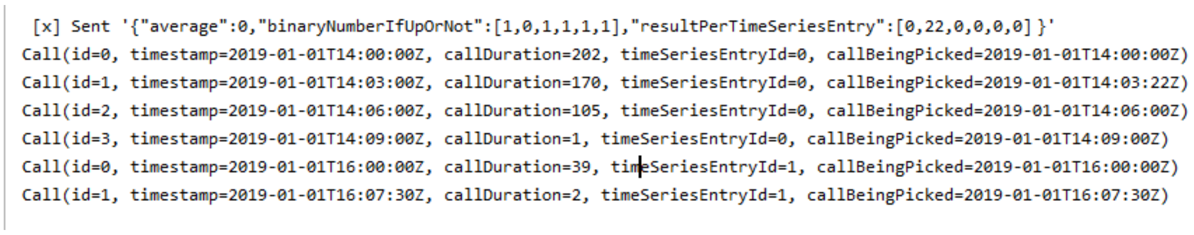
In the picture 8.33 is possible to see the result of the simulation. This simulation only had one time-series that represents the first time-series define.



```
 [x] Sent '{"average":0,"binaryNumberIfUpOrNot":[1,1],"resultPerTimeSeriesEntry":[0,0] }'
Call(id=0, timestamp=2019-01-01T16:00:00Z, callDuration=39, timeSeriesEntryId=0, callBeingPicked=2019-01-01T16:00:00Z)
Call(id=1, timestamp=2019-01-01T16:07:30Z, callDuration=2, timeSeriesEntryId=0, callBeingPicked=2019-01-01T16:07:30Z)
```

Figure 8.3: Result from simulating with only the third time-series.

In the picture 8.34 is present the result of the simulation using the third time-series.



```
 [x] Sent '{"average":0,"binaryNumberIfUpOrNot":[1,0,1,1,1,1],"resultPerTimeSeriesEntry":[0,22,0,0,0,0] }'
Call(id=0, timestamp=2019-01-01T14:00:00Z, callDuration=202, timeSeriesEntryId=0, callBeingPicked=2019-01-01T14:00:00Z)
Call(id=1, timestamp=2019-01-01T14:03:00Z, callDuration=170, timeSeriesEntryId=0, callBeingPicked=2019-01-01T14:03:22Z)
Call(id=2, timestamp=2019-01-01T14:06:00Z, callDuration=105, timeSeriesEntryId=0, callBeingPicked=2019-01-01T14:06:00Z)
Call(id=3, timestamp=2019-01-01T14:09:00Z, callDuration=1, timeSeriesEntryId=0, callBeingPicked=2019-01-01T14:09:00Z)
Call(id=0, timestamp=2019-01-01T16:00:00Z, callDuration=39, timeSeriesEntryId=1, callBeingPicked=2019-01-01T16:00:00Z)
Call(id=1, timestamp=2019-01-01T16:07:30Z, callDuration=2, timeSeriesEntryId=1, callBeingPicked=2019-01-01T16:07:30Z)
```

Figure 8.4: Result from simulating with the three time-series.

This last picture is the result of the project to the input defined above. It was to expected the processing of six calls with only one above the expected service level with 22 seconds.

This result is possible to see the time when the calls are picked up and their duration.

8.3 Overview

All the requirements set at the beginning of this document were successfully validated.

This table holds all the confirmation of all the requirements. The requirement mark, as completed, uses the symbol ✓.

Requirement	**Status**
Creation of information	✓
Insertion of information from input	✓
Modification of the information	✓
Creation of a Queue	✓
Modification of a Queue	✓
Delete of a Queue	✓
Read from the Queue	✓
Creation of a new Agent	✓
Modification of an Agent	✓
Delete of an Agent	✓
Read from an Agent	✓
Creation of a Shift	✓
Modification of a Shift	✓
Delete of a Shift	✓
Read of a Shift	✓
Creation of a Schedule.	✓
Modification of a Schedule	✓
Delete a Schedule	✓
Read from a Schedule	✓
Transmission of information	✓
Receive information from another component	✓
Storage of information	✓
Simulation of the information	✓
Receive information	✓
Data processing	✓
Transmission of the information	✓

This page is intentionally left blank.

Chapter 9

Conclusion

The last chapter has the reflection of the developers' point of view of the internship.

9.1 Internship feedback

With the conclusion of this project, it is possible to reflect on the negative and positive experiences encountered. All the established goals for this project were completed, the overall experience was positive.

The investigation of a Contact Centre management dependencies and concepts were necessary for a better understanding of the depending variables impact. This research led to the development of a solution that best fits the necessities of the company. The invested time in the study of the approaches used by Talkdesk resulted in the better development of the project.

The developer used technologies that had never used before. Therefore a significant percentage of this internship was influenced by the learning curve of said technologies. Besides the learning stage, the developer also found an interest in working with these technologies in the future.

Concerning the planning of the tasks, some of the tasks ended during the expected time. The project was not affected by any wrong estimation. The developer was able to balance the unfinished tasks, and the tasks affected with an inaccurate estimate.

This internship started with a software engineering problem. The balance between finding a solution and applying to a professional project was quite positive. The developer learned different methodologies to organise the development and the work as a team. The developer during the internship was placed in a development team where it was possible to observe the different meetings, from daily meetings to storyboards sessions.

This internship was an exhilarating experience, this area of work reveals appealing and professional, and personal growth culminated.

9.2 After the internship

The project developed with the hosting company offers a new service that helps validate and simulate with specified data.

When generating schedules, the company can now have feedback on the new data. A new asset to the company was now created, and the documentation required to continue developing this service is available to allow future development.

All assets developed in this internship were delivered to the company.

References

[1] RESTFull api. In *https://restfulapi.net/*, Last time accessed: 2020-06-23.

[2] AWS. In *Kanban: Successful Evolutionary Change for Your Technology Business*, Blue Hole Press; 3.8.2010 edition (April 7, 2010).

[3] Brad cleveland. In *Call Center Management*, ICMI Press; Third edition (May 8, 2012).

[4] Github. In *https://github.com/*, Last time accessed: 2019-12-14.

[5] Java. In *https://adoptopenjdk.net/*, Last time accessed: 2020-01-13.

[6] Henrik Kniberg. In *Scrum and XP from the trenches*, InfoQ Press; Second edition (November 24, 2015).

[7] Postgre. In *https://www.postgresql.org/o*, Last time accessed: 2019-11-25.

[8] Quarkus. In *https://quarkus.io*, Last time accessed: 2019-11-20.

[9] Http specifications. In *https://tools.ietf.org/html/rfc2616*, Last time accessed: 2020-06-23.

Appendices

Appendix A

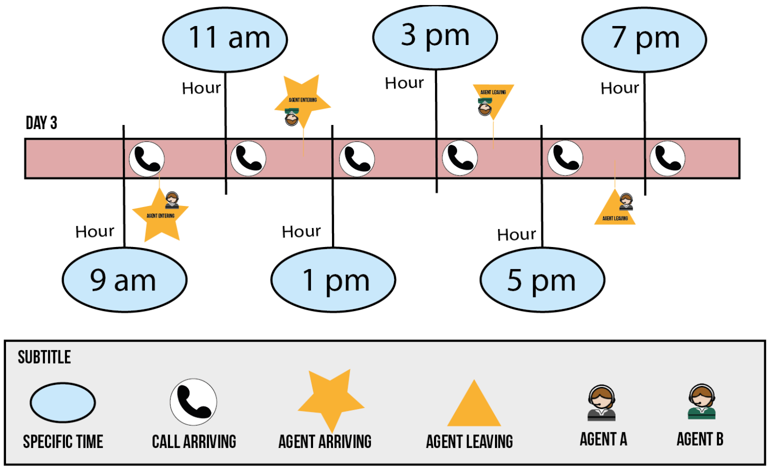


Figure 1: Representation of a time-series entry.