IST-Scope: Talkdesk

Miguel Dias do Rosário Pinto
miguel.dias.rosario.pinto@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2019

Abstract

The evolution of engineering education has sought to bring students' experiences closer to the needs and challenges of the labor market. Two major trends have gained particular relevance in recent years: the interdisciplinary approach to solve even more complex problems and the approach to highly competitive global markets. The SCOPE project aims to introduce an innovative model of interdisciplinary collaboration across the different engineering specialties at IST, allowing dissertation/project students to collaborate in teams to solve real problems posed by companies. With this in mind I opted to choose Talkdesk as the company to do my work because I made an internship that involved me in the company and led me to help them (and me) to grow. The main goal of this project is to build a simulator tool that is able to integrate the Talkdesk system components. This work will facilitate the possibility to simulate the entire platform for newer versions of *Callbar*. The part that I developed during this project was the backend of the application where it was necessary to make the connections from our applications to Talkdesk staging environment and the core logic necessary to produce the responses to the requests from the frontend.
Keywords: SCOPE project; Talkdesk; Simulator; Callbar; Scalability

1. Introduction

The Capstone project is an innovative idea brought to Instituto Superior Técnico (IST) with the purpose to give students a culminating experience to end the academic school and final thesis. This kind of project is a multifaceted assignment because it allows the students to work with colleagues from other areas and universities.

Capstone as the name implies as a core idea that will be the main goal that the students will aim to achieve or solve. For that, it is necessary to adopt the right strategies and think critically to the faced problems and for the ones that arise while solving the current ones. Since this work is done in a team there are lots of soft skills that are developed during the project phase, such as cooperation, oral communication, time management, planning and goal definition.

1.1. Goals of the Project

This project started with the goal to build a test harness that could provide information and metric analysis for the new version of the *Callbar*. The provided description for the problem was:

"The Capstone Test Harness for Internet of Things (IoT) Messaging System pretends to operate an IoT system for thousands of devices with strict requirements of latency. The main factors in this system are reliability and scalability to accommodate the production loads. This project wants to build a Test Harness that is capable of reproducing realistic loads in the system. To this end the system must be able to simulate thousands of devices in specific network conditions, with degradation, loss of signal, etc."

As it's possible to observe in the description the proposed goal was to achieve a system that could recreate realistic loads to demonstrate scenarios of possible failures in the future with the release of the new version.

Since the new version wasn't ready when we started to work on this project, they only had a mock application, we (our team and Talkdesk) adapted the goal to fit with what they had provided and what we could accomplish in the remaining time.

So this project gained a new form and our main purpose is to build a simulator which is capable of representing the loads that the real system will have in the future with the components that they had to date

1.2. Talkdesk

Talkdesk [1] is a company founded in 2011 by Tiago Paiva (who also is the current CEO) and Cristina Fonseca both former students of IST. They

realized that the paradigm of contact centers was outdated and so they entered in a contest created by Twilio to develop their own ideas in 10 days using the provided API.

It's a cloud-based contact center software provider in the digital industry of SaaS. So the main product that Talkdesk sells is precisely an online call center aiming for increased customer experience, reliability, data security, quality and therefore increasing clients' satisfaction, productivity and profit.

They have several offices in Porto, Coimbra, Lisboa, Salt Lake. Its headquarters are based in San Francisco, in United States of America.

The IST-SCOPE project is closely in contact with Lisbon's office.

This company will be the focus of our project where we will target their product - Callbar (detailed in Section 3.2) - that will need to be studied, known, addressed and treated as a mix of client and partner.

1.3. SCOPE Team

The multidisciplinary team that worked with Talkdesk is composed by the following four people:

Carlos Silva Student from Faculdade de Belas Artes in Master Degree in Communication Design and New Media - *UX/UI Designer*

Diogo Lopes Student from Instituto Superior Técnico in Master Degree in Information Systems and Computer Engineering - *Software Developer (frontend)*

Miguel Pinto Student from Instituto Superior Técnico in Master Degree in Information Systems and Computer Engineering - *Software Developer (backend)*

Ricardo Monteiro Student from Instituto Superior Técnico in Master Degree in Industrial Engineering and Management - *Project Manager*

Each member of the team will contribute to all phases of the project but different members will adjust their efforts according to their expertise and academic field. The software developers will be responsible for the development of *frontend* (Diogo Lopes) and *backend* (Miguel Pinto) of the application; the designer (Carlos Silva) will be responsible for the creation and visualization aspects of the user interface concerning the results of the tests; and the project manager (Ricardo Monteiro) will be responsible for the team's planning and results analysis resulting from the application project.

1.4. Division of work

In Table 1 it is described what was the work that each one of the elements of the team mainly did through the SCOPE project. We all helped each other in our tasks but we had our major part attributed to a specific element in the team as it is possible to see in the table.

2. State of the Art and Technology

Contact Centers [2] are a central point from which all customer contacts are managed. From there it is possible to have multiple calls centers do diversify the channels used to enter in contact with the clients creating an omni-channel base for the enterprise's overall CRM. Contact centers are able to have inbound and outbound calls (or an hybrid version).

The traditional Contact Centers had their infrastructure on premises which required to have specialized staff and have additional costs maintaining the infrastructure. With the appearance of the Cloud it is possible to outsource these tasks and have the ability to have elastic computing, which reduces the costs comparing to the traditional way. Another advantage of Cloud infrastructure is that it allows to access anywhere with a single restriction on Internet access. So, for example an agent wouldn't necessarily need to be in the facilities to answer and make calls.

CCaaS are built natively in the cloud so the providers maintain and develop the software which gives more flexibility and better customer experiences. The main purpose of having the infrastructure on the cloud besides the previous advantages is that it allows to have intelligent routing, this means that it is possible to have the calls assigned to the most suitable agent, which is frequently denominated *Skills-Based Routing* [3].

2.1. Chosen Technologies for the Project

Through the amount of programming languages that are available today we opted to go with:

- **Python** - to implement the backend of the application;
- **Javascript, HTML and CSS** - for the frontend of the application as well as the bridge from the backend to frontend.

We chose Javascript, HTML and CSS since we were most familiarized with them and they are the most common languages today. We can see in Figure 1 that these 3 languages are in the top choice of most users in the development world.

For the backend the main reason we chose Python was because of our familiarity with this language. We had one better choice which would be **GoLang** by Google. This language is more recent than Python and have many advantages over it, such as more *performance*, *scalability* (Concurrency and Parallelism) and it's a language de-

Table 1: Comparative analysis of drug administration routes

	Work Dvision
User Research	Each element of the team contributed equally to this task.
Project Management	We all had part on this and deciding which tasks we should prioritize but it was **Ricardo Monteiro** who managed Jira Platform and created epic, stories and the rest of the elements to reflect the work that we were doing.
Frontend Design	**Carlos Silva** main task was to design the wireframes and pick the right elements that matched closely with Talkdesk Cobalt components
Frontend Implementation	**Diogo Lopes** did his major work in this part, where he translated the concepts and ideas that Carlos Silva had to actual code.
Backend Implementation	My main focus on the project was to design, create and implement the backend of our application (Simulator)

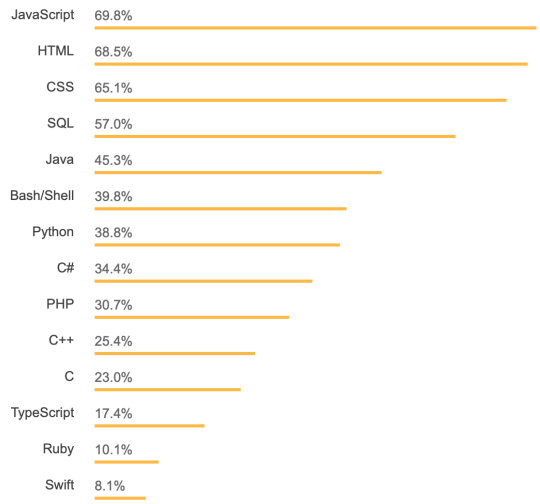

Figure 1: Insight took from Stack Overlfow Developer Suvey 2018 [4]

signed to produce systems at a large scale. Since this is a statically typed language it gives more control to the programmer building and developing the systems. But since we had limited time to achieve the goals for this project and none of us had the experience on this language we opted to go to a known language and open the opportunity for the system to be implemented in GoLang in a future iteration for better performance.

2.2. Related Applications

There are several applications in the market that simulate and create the load to an application to test, categorize and describe its behaviour. In the next subsections it will be described briefly some of the applications that were researched through the process of creating our own and influenced us to create and develop some features such as the ability to load test the scenarios (custom scenarios or pre-defined) and to customize the parameters of each scenario through a web GUI. The distinct part is that the simulator has integrated Talkdesk components and it's adjusted to their needs to assess the potential problems that might arise in future versions.

2.2.1 WebLOAD

WebLOAD [5] is a load and performance testing tool specifically designed for web applications. It accommodates the enterprise-grade segment in the load performance tools since it's very robust. It's main features are ease of use and flexibility, so it allows to be used with several protocols and technologies which facilitates the work when doing large load assessments to an application.

The dashboards are divided in three sections:

1. Main Dashboard - The Main Dashboard provides a high-level view of test results, including failure counters, failures and errors over time, and other useful measurements.

2. Transactions details Dashboard - A Transactions details dashboard lets you view the performance of transactions throughout the session as well as success/failure over time.

3. Log dashboard - The Log dashboard lets you correlate log events with the timeline progress of the session.

2.2.2 LoadNinja

LoadNinja [6] is a tool that lets the user to create scriptless load tests that improves the time spent in generating them. Its best advantage is that it can reduce test script creation time by 60% using artificial intelligence in recognizing objects in applications and by using their component *InstaPlay Recorder* which eases the process of loading the tests. It also allows to analyze in real time to tackle performance and bottlenecks faster. A quick summary of their features is: Record and Playback Scripts; Load tests on real browsers; Analyze the performance results; Full coverage with the load tests; Possibility to test in the local network; And it automates within the pipeline CI/CD.

2.2.3 Apache JMeter

Apache JMeter [7] is an open source application developed in Java that can be loaded in the server

and in the network to analyze and check the performance under different scenarios. It supports a wide range of protocols and supports variable parameterization, assertions (response validation), per-thread cookies, configuration variables and a variety of reports. Apache JMeter works by adding plugins to increment its functionalities.

2.2.4 LoadView

LoadView [8] is a tool similar to LoadNinja in the way that it runs the tests in a real browser instead of simulated ones. It's a tool that needs to be payed, but their price model goes in the way of pay for what you use, since it is 100% cloud based. The advantage of being based in the cloud is that it can be deployed in minutes and it can be fully configured from there.

2.2.5 Locust

Locust [9] is also an open source tool for load testing. This framework allows to load test an application using a well known code language - Python. It's all customizable through code, scalable and distributed, and the great benefit is the parameters can be changed through a web GUI.

3. Research and Knowledge about Talkdesk

During the first semester our work involved getting to know Talkdesk internal teams as well as their product - *Callbar*. In the next subsection it will be described the research that the team did and what is *Callbar*.

3.1. User Research

In the research phase we had the possibility to conduct shadowing and small interviews to Talkdesk members which contributed later on to build the profiles that simulated agent behaviour. The shadowing was guided by Prof. Pedro Oliveira and was ruled by the following 10 principles [10]:

1. Proactively engage issues with shadowees ahead of time;
2. Prepare for embodied shadowing;
3. Take classes or hold discussion on the emotional side of qualitative methods;
4. Pack a "shadow kit";
5. Plan to follow the rules, at first;
6. Play around with strategies for notetaking;
7. Dance in the doldrums - this means to take rests during the work;;
8. Locate or create social support;
9. Mitigate the anticipation of shadower-as-betrayer;
10. Exit the field mindfully;

During the same time that we conducted the shadowing and the interviews after the shadowing process we had meetings with Talkdesk internal teams that helped us understand better what was their work and how could it help us develop our project. We had meetings with: *Callbar Team*, *Sales Team*, *Product Team* and *Cobalt Team*.

3.2. Callbar

Callbar [11] (Figure 2, 3 and 4), Talkdesk's product, is a software that enables users to make and receive calls from anywhere in the world with internet connection on their desktop. This application was done to be unobtrusive and always accessible in case there is work to do with the application. The main benefits of using *Callbar* are:

- Always accessible being normally in the bottom of the screen to be easier to make actions within.
- Give context every time there is an incoming call to reduce the time spent by the agents getting the context of the call. It allows to view the previous history that the client had as well as detailed information about him to ease the process of the current call.
- This is a lightweight application because it wants to be unobtrusive and enables the agent to do other work without being worried about the response time of the applications to answer to the customer needs.
- One of the great benefits is to fully integrate with known CRM like Zendesk and Salesforce which gives the agent lots of detailed information about the client.
- It supports Click-to-Call [12] (a feature that enables the users to call directly numbers in webpages without having to copy-paste to the application. The required software is an extension that intelligently scans the sites searching for numbers), screen pops, call logging.

3.3. Callbar Features

These are the features that are currently available to anyone that uses *Callbar*:

1. Make and receive calls - This is the core of the application, it works as a normal phone (or soft-phone) where you type the number you want to call and where you can see who is calling to you;

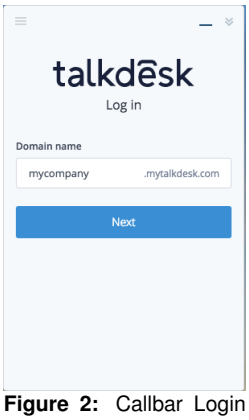

Figure 2: Callbar Login Screen

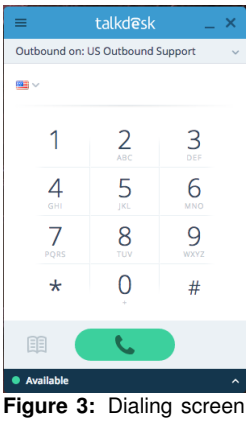

Figure 3: Dialing screen in the Callbar

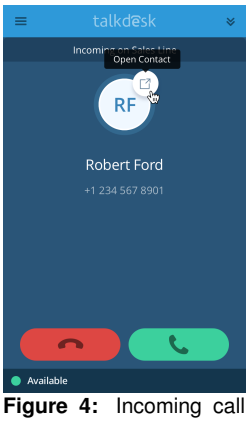

Figure 4: Incoming call screen

2. Set agent availability - There are at least 4 states in which an agent can be: Available, Away, After Call Work, Offline. Swapping between the states is done automatically by the application when a user is logged in or logged off; when he receives a call and answer it; and when he terminates the call and is registering the work that he did. It is also possible to define custom states to better translate the agent availability to the company that he is in;

3. Mute or hold a call;

4. Transfer a call - There are two types of transfers that can happen while in a call with the *Callbar* application: *Warm Transfer* and the *Blind Transfer*. The difference lies that in *Warm Transfers* there is a conversation previous to the transfer to give context to the next agent that will pick the call;

5. Initiate a call conference;

6. Merge or create new contacts;

7. Pause the call recording;

8. Add notes to a caller's activity history - It allows to keep track of what is said during the call to facilitate the work of the agent when he is in the state - *After Call Work*;

9. Access the contact's profile page in the integration of your choice - One of the greatest selling points of *Callbar*, is that it allows to integrate seamlessly with CRM to make the job of the agent when searching for information about certain clients easier.

4. System Design and Implementation

The main goal of the project, as stated in Section 1.1, is to create a simulator that is capable of representing the loads that the real system will have in the future with the components that they had to date.

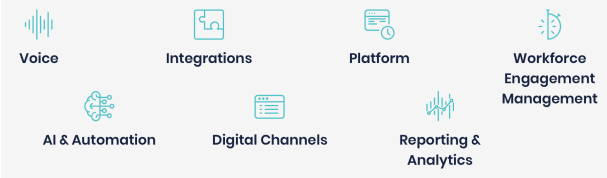

Figure 5: Contact Center Software Features with *Callbar*

Despite the changes that we had in our scope project we adapted well and created a product that can be used by Talkdesk to include the future components when they get released.

This simulator allows the users to define or to chose a predefined scenario to run and at the end retrieve results and analyze them if they deem so.

The following subsections will describe in detail the *What*, *How* and *Why* of the Simulator.

4.1. Simulator Features

Taking into account what we learned in the sessions with the teams and the knowledge that we got from the shadowing we defined some features that we must have to create the initial form of the application. We wanted to have the possibility to create scenarios to give freedom to the users to explore the metrics that we selected to parameterize the scene. We gave the possibility to change the following parameters:

- Number of Devices;
- Number of Calls;
- Call Duration (ms);
- Call Pickup Delay (ms);
- Call Standby (ms);
- Profiles.

The **Number of Calls** and **Number of Devices** is the core of the simulator, with them we can have

more or less calls and devices being that the number of calls is the number of total incoming calls and which any device that is in the state waiting for a call can answer. The **Call Duration** is the time that each call will have in the simulator with a corresponding agent; The **Call Pickup delay** is how much time the agent takes to pick the phone; And the **Call Standby** is the time between incoming calls. In Figure 6 is an example explaining what is each one of the custom parameters that we made available.

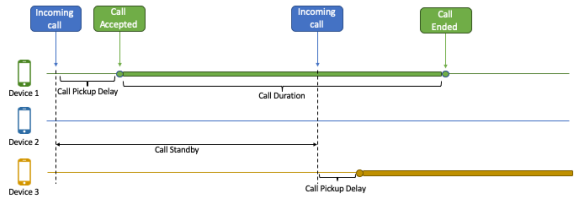

Figure 6: Example of 2 incoming calls with 3 devices and the time labeled

The **Profile** item that we made available to customize has 4 options: **Shy**, **Champion**, **Normal** and **Sleepy**. To each of the options we associated a call answer percentage which is the percentage at which the agent answers a call.

The values are:

- Shy - 10% rate - corresponds to a *shy* agent which doesn't answer to many calls;
- Champion - 90% rate - corresponds to an extroverted agent and very talkative;
- Normal - Value retrieved from the Normal Distribution - from the name this is the average of the agents, it has an approximate 50% chance of accepting the call;
- Sleepy - Value retrieved from the Exponential Distribution - this agent has a low rate at the beginning of the day and that rate increases as the day goes on.

4.2. System Architecture

In the following subsections it'll be described how the system was built.

4.2.1 RabbitMQ

RabbitMQ is a messaging broker that gives applications a common platform to send, receive and store messages while in the life cycle of an application. It provides: *Reliability*, *Flexible Routing*, *Clustering*, *Highly Available Queues*, and many more features that were necessary to use through our development.

The reasons that made us use RabbitMQ as our platform for messaging is because Talkdesk already use it as a core component, it is Open Source with a big and pro-active community and it is widely adopted from small start-ups to large enterprises. Since the message exchange was already deployed in an AWS by Talkdesk we didn't have much space to do tests and evaluate different configurations since this was the scenario they gave us.

The way that the messaging system is setup is with a direct type exchange. This means that every message that is published to the exchange with a certain routing key, in our case is **#**, will be directed to the queue that has that binding key that matches.

In this way every time an interaction is created the process will be this:

1. POST the interaction file that has the properties to create one incoming call in the system to the Orchestrator (this component is outside of the SCOPE project but it's in charge of creating the interactions and distributing them hover the other components that we abstracted here in the project);
2. Exchange routes the message (direct type) to the key that is binded with the **#**;
3. The consumer that is a thread that was created in the beginning will listen and consume the messages as they arrive to the queue;
4. The messages are then processed by the backend and the response to the request is created;

4.2.2 Communication between Frontend and Backend

As we are developing a Simulator, we need information in real time to be passed between the backend and frontend. For this purpose we opted to go with *Flask-SocketIO* which is a library compatible with the language that we chose (Python 3) and allows for low latency bi-directional communications between them (frontend and backend). With this library we can transmit events between them as soon as there are any new activity to be presented in the interface of the Simulator.

The first request will be handled by the function *handle_my_custom_event()* which is binded to receive events that have the tag *devices*. The request will have a *JSON* object in the body with the parameters defined for the scenario to be run in

the Simulator. This method will have the following order of operations:

1. Clear the structures that store the devices object;
2. Create the devices that were defined by the user;
3. Send an update to the frontend to display the list of the devices;

After this first three steps it's when the interactions start to be created with the parameters defined and the amount desired. The messages will be retrieved from the RabbitMQ queue by the consumer thread that was created in the beginning and than they will be parsed and transmitted to the frontend through the*SocketIO* library. So the following steps are:

4. Enter the loop that will produce the calls;
5. Create the interaction;
6. Retrieve the output from the queue - which is a message in JSON format that gives endpoints to accept or reject the call (comes from the Talkdesk Orchestrator);
7. Update the structures that contain the devices to *PENDING*;
8. Send the information to the frontend with the updated status;
9. Use the delay to pick up a call and define a timer to better simulate the environment.

4.3. Device states

I've mentioned about the state of the devices before and now I'll explain them. There are three states that the devices can be (Figure 7):

- Waiting Call (Starting State);
- Pending;
- On Call.

As we can see in Figure 7 the transition between them depends on the incoming calls and the actions that were taken for that call. Every device is initialized with the state of *Waiting Call*, this means that the device is ready to receive calls when they are assigned to it. After a call is assigned the device will transit to the state *Pending*. In this state the device will have two possibilities: accept or reject the call. If the call is rejected the device will return to the *Waiting Call* state, if it accepts the call it will transition to the *On Call* which represents when a device is with a call. After the defined period for the duration of the call it will transition back to the *Waiting Call*.

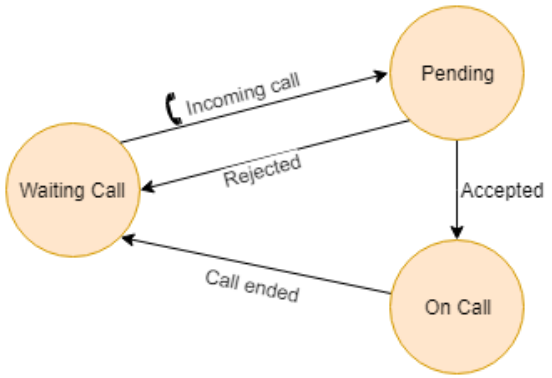

Figure 7: Devices states and their possible transitions to the other states when a call arrives

5. Conclusions

This project aims to bring students closer to the actual market and the state of the art. It introduces a new model of interdisciplinary collaboration to solve problems posed by companies. With this in mind we had to answer to a problem where Talkdesk needed a platform that could test their newer product without being dependent on providers such as Twilio. This means that when it was necessary to make a call it wouldn't be needed to make an actual call but encapsulate this behaviour in such a way that the system would keep functioning and its behaviour would keep intact. This problem was also raised because every time Talkdesk wanted to test their product the costs were significantly high because Twilio charges for each call made.

So our answer for this problem was to encapsulate and mock the calls in our simulator that could show information in real time about the events that occurred in the system and if possible give some feedback of what happened in each scenario defined by the user.

The Scope of the project suffered some changes along the way because of time constraints. Our simulator targeted the new version of the Talkdesk, which isn't publicly available and that's one of the reasons why our scope suffered changes because some components that were to be included weren't ready at the time of the development. It started to include the integration with the newly Presence API which could manage the status and define a occupancy level for devices. It got later reformulated to create a simulator in which we would manage the status of the devices internally when the calls were received and the appropriate action was taken. The research that we did in the first part of the project was expecting different requirements than in the later phase. We firstly accounted to include the behaviour of the agents in the applica-

tion such as their reactions in short time and long time while answering the calls. We worked on the research that we made and turned the information into four different profiles that could fit the behaviour of the agents (shy, champion, normal and sleepy).

After this research we than started to divide the work between us where I took the development of the backend of the application. The backend of the application was responsible to create the interactions of the devices with the given input parameters that were inserted by the user. It collected the messages from RabbitMQ that were put by Talkdesk's components, namely the Orchestrator, and created the events in real time to be passed to the frontend where they were displayed.

Since this project is an innovative model we were the first ones to get in touch with it and it was a great experience to have contact with Talkdesk and professors from a different university - Faculdade de Belas-Artes. Aside from the fine tuning that this project needs, such has defined objectives and materials to be delivered it went very well. The experience that the students gain from entering in contact with the exposed market and from the workers in the company is very valuable and it's something that all students should have before getting a job.

We could reach the goals that we defined in the beginning of the project such as the development of an application that could be used by Talkdesk in their tests of the Callbar. We also gained soft-skills during this project, mainly, coordination, team work and critical thinking.

5.1. Future Work

It is noteworthy that this simulator is built to work with Talkdesk core components, so any use outside of this scope to test other applications won't work. For the future of our simulator we defined some features that could be added:

- Possibility to hover on D3 elements to give more information about them;
- After the scenario has just run it would be presented a report with the metrics and a analysis of the results that were collected during the execution. This could identify possible problems and solutions. A further analysis by the user would be necessary on this document;
- When the Presence API is in a stable state where it can be used, integrate with this application and remove the internal management of the status. This would also give to the application more parameters for customization since it would be possible to change the occupancy of each device;
- Add a custom profile where it would be possible to define the peaks of the function. This peaks would define the percentage in which an agent with this profile would accept a call. This is similar to the normal distribution but defined by the user.

References

[1] https://www.talkdesk.com/resources/guides/a-guide-to-the-talkdesk-and-contactually-integration/. Talkdesk, July. Accessed 09-July-2019.

[2] https://searchcustomerexperience.techtarget.com/definition/contact-center. Guide to customer experience management best practices, technologies, margaret rouse, 2017.

[3] https://en.wikipedia.org/wiki/Skills-based_routing. Skills based routing, December 2018.

[4] https://insights.stackoverflow.com/survey/2018most-popular-technologies. Most popular technologies, March 2018.

[5] https://www.radview.com. Webload.

[6] https://loadninja.com. Loadninja.

[7] https://jmeter.apache.org. Apache jmeter.

[8] https://www.loadview-testing.com. Loadview.

[9] https://locust.io. Locust.

[10] Joshua Barbour Rebecca Gill and Marleah Dean. Shadowing in/as work: ten recommendations for shadowing fieldwork practice. *Qualitative Research in Organizations and Management: An International Journal*, 9(1):69–89, March 2014.

[11] https://support.talkdesk.com/hc/en-us/articles/218195763-Talkdesk-Callbar. Talkdesk, July. Accessed 24-July-2019.

[12] https://support.talkdesk.com/hc/en-us/articles/200370369. Talkdesk, July. Accessed 24-July-2019.