Melissa Gimenes Alvarenga Ferrari

INTELLIGENT INCIDENT MANAGEMENT

Ticketing Software API Integration for Efficient Visualization and Statistical Insights

School of Technology
2024

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology, Bachelor of Engineering

ABSTRACT

Author	Melissa Gimenes Alvarenga Ferrari
Title	Intelligent Incident Management Ticketing Software API Integration for Efficient Visualization and Statistical Insights
Year	2024
Language	English
Pages	43
Name of Supervisor	Anna-Kaisa Saari

This thesis focuses on creating an application based on the TOPdesk ticketing system, aiming to produce a more efficient and user-friendly software. By incorporating advanced filtering options, statistical insights, and intuitive design elements, the webpages created seek to simplify the ticket management process.

Implemented through the utilization of tools and technologies such as Visual Studio Code, Node.js, and SQLite, the project offers a comprehensive exploration of the iterative development process. HTML, CSS, and JavaScript facilitate the creation of an intuitive and visually appealing interface, while backend development ensures integration with the TOPdesk API for data retrieval. The resulting webpage enables users to navigate ticket categories efficiently, access detailed ticket information, and gain insights into ticket resolution metrics.

The thesis achieved its main goal of creating a filtered version of the TOPdesk ticketing system and revealing statistics of incident-solving processes.

Keywords Web Development, full-Stack, UI, ticketing system

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
CSS	Cascading Style Sheets
FIQL	Feed Item Query Language
HTML	HyperText Markup Language
IT	Information Technology
SQL	Structured Query Language
UI	User Interface
JSON	JavaScript Object Notation

GLOSSARY

First-line support: The first point of contact with IT support, normally contacted for simple problems.

Second-line support: Issues that cannot be solved by the first-line support and require the involvement of personnel with extended knowledge or access to solve the given problem.

Front-end: The part of a website that is visible to the users.

Back-end: The structure that supports the front-end and is not visible to users, such as servers and databases.

Bot: A computer program designed to act as a user or another program, mimicking human actions.

Runtime Environment: Environment in which a program is executed.

JSON: File and data format often used for data interchange.

FIQL: Query language mainly used for filtering and querying feed items in syndicated feeds. It enables users to specify queries in a human-readable format.

1 INTRODUCTION

In modern organizations, ticketing systems serve as tools that provide structured and efficient means of managing various requests and inquiries. They offer a centralized platform where employees can submit their issues, questions, or service requests that are transformed into tickets that are later answered by the designated department.

The efficiency and usability of user interfaces (UI) play an important role in improving user experience and productivity across several applications. This thesis is about the creation of a webpage aimed at refining the user interface of a ticket system known as TOPdesk, taking into consideration the specific needs of the client - CitizenM Hotels. The goal is to not only enhance the interface visually but also to add new functionalities and statistics. The inclusion of advanced filtering options and statistical insights allows users to make informed decisions and effectively address emerging challenges, thus maximizing the utility and effectiveness of the ticket system. /1/

This thesis includes the conceptualization, development, and implementation stages, providing an in-depth exploration of the iterative process of creating an intuitive and efficient webpage. From initial ideas and sketches to the practical application of codes, this thesis offers a comprehensive insight into the evolution of the project.

2 INCIDENT MANAGEMENT

Incidents within a company are nearly impossible to avoid entirely, even in the most meticulously organized and equipped systems. Despite the implementation of rigorous protocols, advanced technologies, and trained personnel, unpredicted circumstances and variables can still lead to disruptions in operations. Technical glitches, human errors, or external factors beyond the company's control - incidents have the potential to arise unexpectedly. For that reason, it is fundamental that a company develops protocols and designates employees to both troubleshoot and solve these problems.

Incident management includes detecting, investigating, and answering incidents as early as possible. Early detection and prompt resolution of incidents are crucial to mitigate their impact on the organization. By identifying and resolving issues at their early stages, companies can prevent minor disruptions from escalating into major crises. Moreover, proactive incident management demonstrates a commitment to maintaining operational continuity, protecting higher customer satisfaction, and preserving the company's reputation, while benefiting from increased efficiency and team productivity, also auxiliating in future incident mitigation. Therefore, investing in effective incident management strategies is an essential characteristic of an agile and organized company. /2/

2.1 Methods for Solving Incidents

Several different incident management tools can be implemented within a company. Chatbots, for example, can support users using artificial intelligence, while chat rooms allow users to receive help from specialists. Documentation tools can contain instructions on how to solve a problem, and ticketing systems store and display requests from users that can be replied by the system operators. Ideally, a combination of these methods is used so incidents are solved more quickly and efficiently – for example, a chatbot can be used as a first-line support, and the issue is escalated to the second-line only if necessary. /3; 4/

2.2 Ticketing Systems in Detail

Ticket systems are platforms that facilitate the management of incidents. It is a shared online space where the requester can send messages, also known as “tickets”, detailing the problem and sending it to the software operators who can solve the incident, give advice or re-assign the issue to others. The communication is consolidated in a single thread, facilitating easy reference to the ongoing dialogue for both parties. In a company, departments that are responsible for ticket solving can be either working resolving complaints and requests within the company, communicating exclusively to employees, and/or outside the company with customer support.

Upon the resolution of the issue, a ticket can be closed, only being re-opened in case further assistance or clarification is needed. This ensures continuity in communication, allowing the customer to resume interactions with the same operator, which is important to shorten the resolution time of problems, consequently improving customer experience. A typical and simplified ticket flow is illustrated in **Figure 1**. /5/

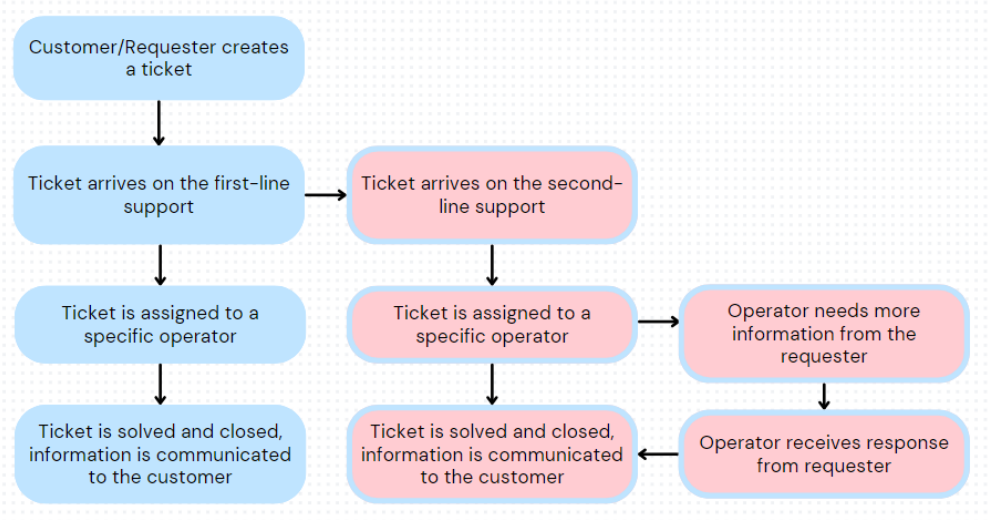


Figure 1. Typical flow of tickets

It is essential to note that ticket flow processes may vary. For instance, within a company, requesters might have the ability to directly submit tickets to second-

line support. Alternatively, tickets can be de-escalated from second-line to first-line support. In some cases, tickets may even be redirected to entirely different departments when necessary.

Ticketing systems offer several advantages, including personalized support, which can provide more accurate information to the requester. Additionally, they facilitate incident tracking, enabling operators to refer to past tickets, including the ones solved by other colleagues, to resolve new ones encountering similar issues, possibly also being combined with other documentation software. Besides, these systems possibilitate data collection, enabling the identification of trends and patterns over time.

3 CONTEXT OF THE PROJECT

3.1 Client's Background

CitizenM is a multinational hotel chain founded in 2008 in the Netherlands. After the COVID-19 pandemic, the company has experienced rapid growth, nine new hotels have been opened in the past two years. This expansion has led to the recruitment of new employees and the initiation of various projects. Given the dynamic nature of the company, daily operations often require access to a diverse range of applications. /6/

In the context of CitizenM, employees seeking tech support must utilize a ticketing system platform called TOPdesk to create tickets to get support. Another alternative is sending emails to the department, which are also connected to TOPdesk and automatically transformed into tickets. These tickets are answered by the Operators from the IT Support Office team.

3.2 CitizenM and the Author of the Thesis

In September of 2023, the thesis author was hired by CitizenM as an intern in the IT team. The duties of an Office IT Support Intern include helping to fix issues related to computers, answering questions, resolving computer problems for CitizenM employees, and providing technology assistance in general inside of the office.

The most important task of an Office IT Support Intern is to have active participation in answering and resolving the tickets on TOPdesk. This means, most of the workload is about incident management.

3.3 IT Support Office Ticketing System: TOPdesk

TOPdesk is a company that develops and supplies help desk software, founded in 1997 in Delft, Netherlands. TOPdesk's software is the ticketing system platform utilized by the IT department at CitizenM to address employee requests. Its user

interface resembles an email inbox, but it displays additional information that is useful for the operators, as seen in **Figure 2.** /7/

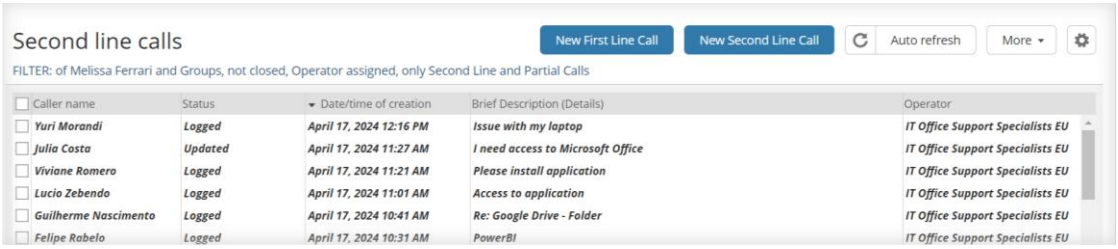


Figure 2. TOPdesk ticketing system interface.

Upon selecting an incident, a new page is opened, providing details about the caller and their request, as seen in **Figure 3**.

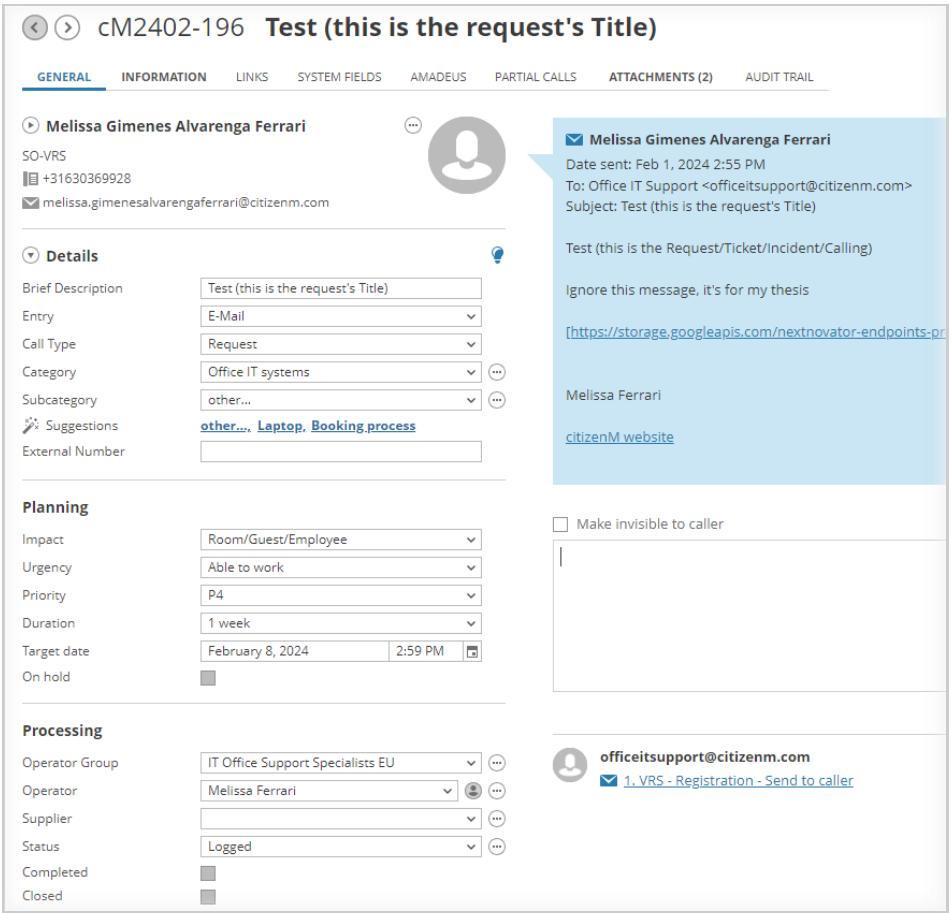


Figure 3. TOPdesk's individual ticket interface.

The "Processing" section enables operators to assign tickets to themselves or colleagues. Besides, operators can also set the ticket status based on the ongoing situation, as seen in **Figure 4.**

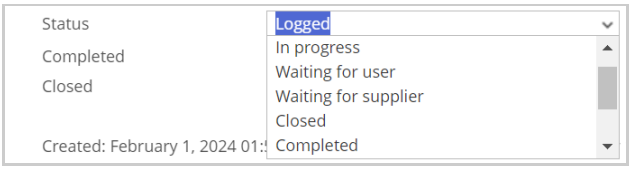


Figure 4. Options of statuses of a ticket.

Once a ticket appears on the inbox, it is initially labeled as "Logged." As it progresses, it remains in this state unless an operator intervenes. Once an operator takes ownership of the ticket or delegates it to colleagues, the status is typically switched manually to "In progress" to signal that work is underway. If additional information is required from the ticket requester, the status is changed to "Waiting for user" while awaiting their response. If the resolution depends on a supplier's input, the status becomes "Waiting for supplier." Otherwise, once all requirements are met, the ticket can be set as “Completed” and subsequently “Closed”, following the typical status flow outlined in **Figure 5**.

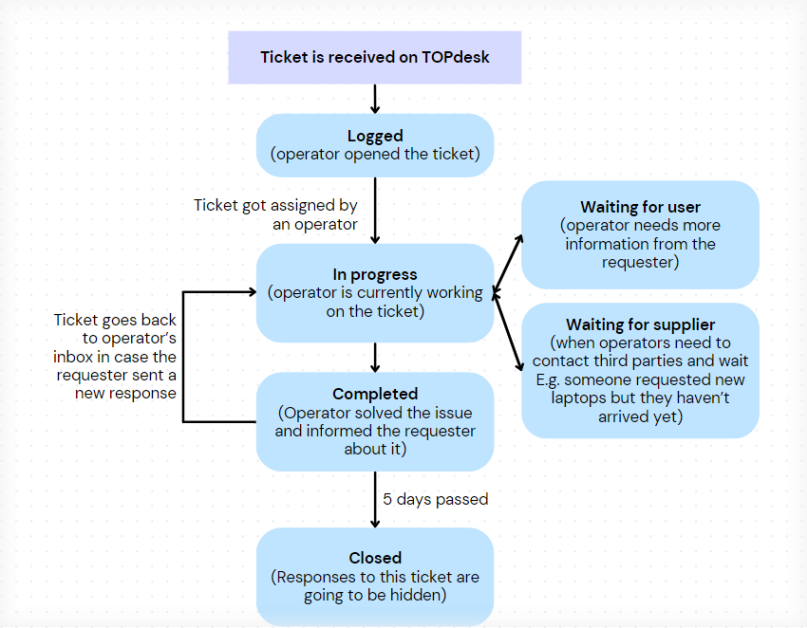


Figure 5. The status of a ticket

It is important to mention that when a case is marked as “Completed”, updates from users can still be received. On the other hand, a “Closed” status hides future replies from the main tickets page. After 5 days of inactivity on a ticket, a bot implemented by CitizenM causes TOPdesk to automatically transition the status of "Completed" tickets to "Closed," where Completed = True and Closed = True.

3.4 Issue Acknowledgement

The IT Support Office functions are often similar to a system administrator's work, and the role demands meticulous attention to detail. It is crucial to emphasize that possessing administrative access to critical applications requires a thorough understanding of the systems utilized. As the company grows, new team members are hired, making it necessary to mitigate potential risks associated with newcomers, so there is a careful approach to granting administrator rights. Instead of immediately providing such privileges, individuals are allowed to familiarize themselves with the work environment first. Consequently, each team member possesses distinct admin accesses tailored to their responsibilities.

TOPdesk’s interface includes all requests sent to the team and it lacks a variety of content filters. Given the collaborative nature of TOPdesk as a shared workspace, team members tend to address issues within their expertise and leave the remainder to their colleagues. Furthermore, the organization utilizes more than 80 applications, which makes it challenging to ascertain which team member has access to specific tools or platforms. Therefore, due to occasional ambiguity in role delineation, certain tickets may accumulate in the expansive mailbox, leading to delays in resolution.

3.5 General Idea

The general idea of this thesis emerged when the author identified a significant surge in tickets related to a common subject. Faced with a lack of necessary access

to address these tickets, the author meticulously compiled a categorized list built on Google Sheets, intending to seek assistance from colleagues at a later stage.

Categorizing and delegating incidents manually was helpful for the given occasion; however, it is not sustainable in the long run because the list has to be constantly maintained manually and data gets deprecated. The author brought the matter to the attention of the team leader and offered to initiate a project aimed at addressing the issue of unresolved tickets that often were forgotten and left in the queue. The team leader accepted it with one condition: in the project, the author should also create automatic reports about the number of tickets solved by each operator.

4 PROJECT DEFINITION

4.1 Theoretical Background

4.1.1 The Importance of Chunking Information

Numerous studies in the field of cognitive psychology and education explore the effectiveness of chunking information for learning. These studies often investigate how breaking down information into smaller, organized chunks can improve memory retention, comprehension, and problem-solving skills.

A study by George A. Miller, titled "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information", discusses the limitations of human working memory and suggests that people can generally retain around seven pieces of information at once. This study demonstrates the importance of chunking information to overcome these limitations. Coherent chunks facilitate learning compared to delivering it in a more overwhelming format. /8/

On TOPdesk, while the information is presented as a whole, instead of chunks, operators can feel overwhelmed and overlook tickets. Considering George A. Miller's study, it is valid to assume that dividing this information (requests) into chunks (categories) can help operators retain more attention and improve engagement regarding work. Even though Miller's project is mostly about learning, memory and attention, aspects can be taken into account during the development of this project.

4.1.2 Intuitive Design

Don Norman's book "The Design of Everyday Things" explores the principles of good design and usability in everyday objects. Objects should intuitively suggest their use, and/or have clear indicators of how they should be used. Users should receive immediate and clear feedback about their actions, systems should be

forgiving of user errors and allow for easy recovery, and limiting the possible actions can prevent errors. /9/

Implementing these principles enables the development of products that prioritize intuitiveness, user-friendliness, and the enhancement of user experience. Such principles apply not only to physical products but also to software design. Introducing software embodying these traits, particularly in professional contexts, plays a crucial role in ensuring a smooth transition when engaging with a new tool. Enhanced usability facilitates quicker user adaptation and proficiency, a key priority for companies seeking to optimize productivity and efficiency.

Within a company's Technology Support departments, efficiency holds particular significance, given the indispensable role of technology in the operations of numerous other departments. A malfunctioning software can rapidly disrupt the workflow of an entire company, underscoring the imperative for IT to swiftly and accurately address any issues that arise. Hence, it was decided that the project described in this thesis would possess an intuitive, self-explanatory, and simple user interface while ensuring it encompasses all essential tools to enhance the department's operations.

4.2 Conceptualization of the Project

The initial phase of the project involved the rudimentary categorization of applications used within the company. To accomplish this, a list featuring these applications was inserted into a new document and then categorized as follows:

- “Okta admin is enough”
- “Easy tickets”
- “Difficult tickets”
- “Others”
- “Send to other departments or request form”

- “Hardware, new equipment”

The categorization presented takes into consideration the process involved when onboarding new team members, where the initial administrative access provided is to Okta. Okta serves as the main software solution for IT Support to efficiently manage employee access to various applications. Within this category, access provisioning is automated between Okta and other software systems. Consequently, this category is regarded as foundational, including tickets that can often be promptly resolved by new hires. /10/

However, within the organization, certain applications lack automation, necessitating additional internal administrative access during the access provisioning process for employees. This delineates a distinction in the complexity of ticket resolution, with some applications posing greater challenges than others. While new operators can proficiently handle tickets related to easily accessible applications, those of more complex systems require the expertise of more experienced IT operators. Hence, it was necessary to establish two distinct categories: "Easy Tickets" and "Difficult Tickets."

The classification of "Send to other departments or request forms" includes tickets that are received by the team but do not fall within their direct responsibility. Tickets categorized under "Hardware, new equipment" necessitate either on-site resolution or remote control assistance, while the "Others" category comprises exceptional or miscellaneous requests. (See **Figure 6.**).

To maintain organization and enhance clarity, all potential categories were systematically listed in a spreadsheet, each identified with a distinct color code. Subsequently, a comprehensive list of applications utilized within CitizenM was incorporated into the spreadsheet, with each application color-coded according to the designated category scheme. This methodical approach facilitates the development of the website before the process of coding is started.

	A	B
19	Review Pro	
20	Sketchup	
21	Slack	
22	SmartSheet	
23	Tagetik	
24	TOPdesk	
25	Typeform	
26	Others	
27	Mpower	
28	Trello	
29	Business Central	
30	Retool	
31		
32	Okta admin is enough	
33	Easy tickets	
34	Difficult tickets Florbs or Drive (calendars, email, delegation)	
35	Others	
36	Send to other departments or request form	
37	Hardware, new equipment	
38	etc	

Figure 6. Initial design for application categories.

After sketching the initial idea, it was necessary to understand how to put it into practice. The most viable option was to create webpages from scratch, considering that then the software would be fully customizable according to the needs of the Office IT Support team, and would grant the client full ownership over its content and data.

4.3 Landing page: Ticket Categories and Count

The initial phase of the project involved translating the Google Sheets sketch into a webpage featuring categories, the count of open tickets, application names or descriptions, and interactive buttons redirecting users to the next page. Before starting the webpage programming, the concept was designed on paper, as seen in **Figure 7**.

With a sketch in hand, translating the concept into code becomes easier. Drawing before coding aids the programmer in understanding the progress of the program and visualizing what tasks remain. It serves as a visual roadmap, helping to bridge

the gap between design and implementation, as explained by Jonathan C. Roberts, the professor of visualisation at the School of Computer Science and Electronic Engineering of Bangor University. For that reason, all webpages were planned following this method. /11/

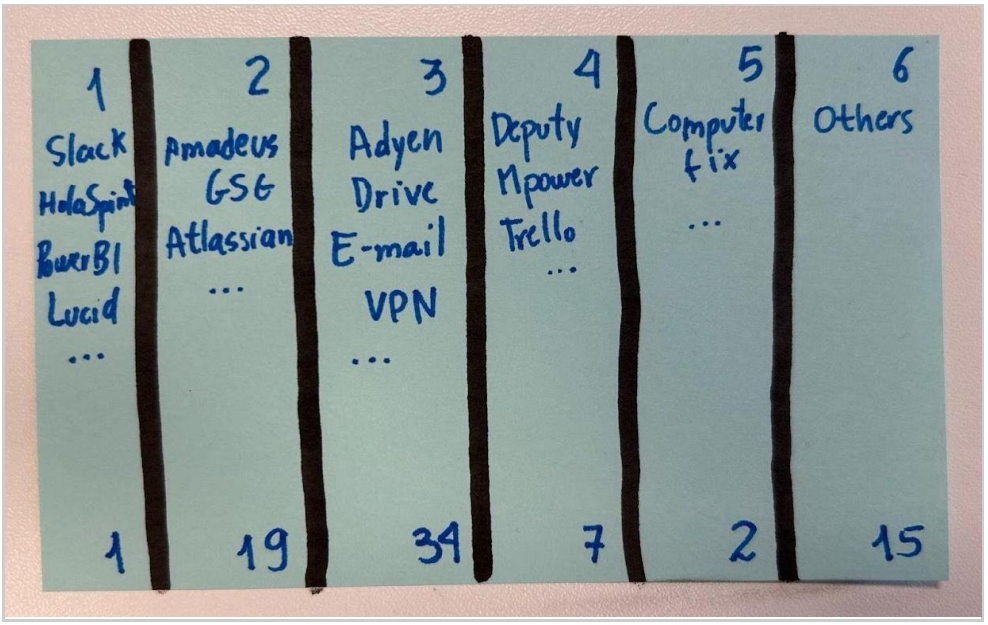


Figure 7. Landing page UI design

4.4 Unassigned Tickets Webpage: Tickets Information

After clicking the category buttons on the landing webpage, the concept involved redirecting the user to a second page that would display a table containing ticket information, such as the ticket ID, title, and a field not available in TOPdesk: subject. The design for the page is shown in **Figure 8**.

The program would categorize these tickets based on predefined groups using keyword search. IT Support team members could open the ticket in TOPdesk by clicking on the ticket ID which is linked to a URL. The displayed ticket table on the unassigned tickets webpage would only include requests that are yet to be resolved and are not assigned to any operator.

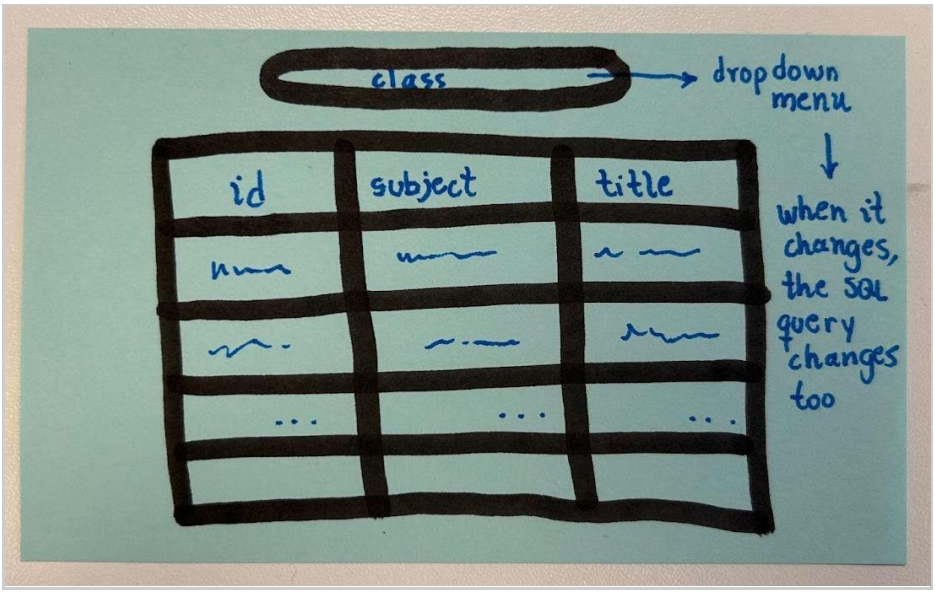


Figure 8. Ticket information UI design

4.5 Statistics Page: Performance Graph

On the landing webpage, a new button had to be created to redirect the user to the statistics page. This new webpage concept involved incorporating a dropdown menu containing operators' names, along with an option for "IT Support Office Specialists", which would represent the sum of every operator's ticket.

Upon selecting a specific name from the dropdown menu, on the left-hand side, a bar chart is generated. This chart features a bar for each month on the x-axis, depicting the number of tickets on the y-axis. On the right-hand side, pizza graphs are displayed, one for each month, representing the percentage of tickets responded to by each operator compared to the amount solved by the team as a whole (see **Figure 9**).

Integrating functionality that allows users to adjust the displayed time frame, such as selecting specific months or custom date ranges, promotes flexibility and usability. These features contribute to a more complete dashboard, catering to diverse user preferences and requirements.

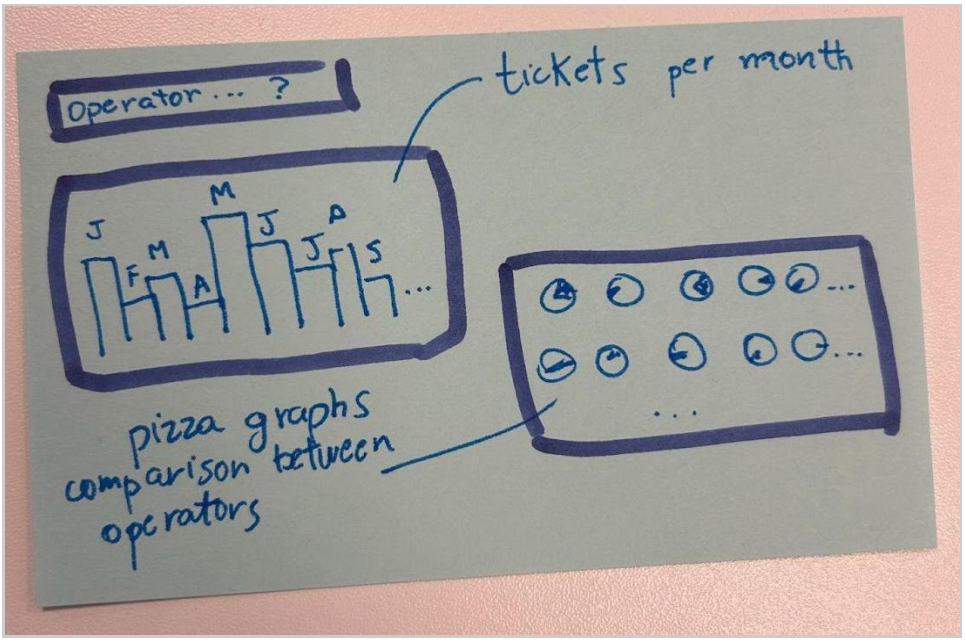


Figure 9. Statistics UI design

5 TECHNOLOGIES USED

5.1 Languages and Programming Tools

Hypertext Markup Language (HTML) serves as the standardized language for building webpages, and is essential for web development, as it composes the main structure for webpages. Cascading Style Sheets (CSS) defines the visual style of the webpages, such as layout, colors, and fonts. HTML and CSS are indispensable for developing a webpage project. /12, 13/

For the addition of working buttons and other complex features, it is necessary to add scripts; that means instructions that the computer will follow to guarantee a dynamic and interactive webpage. This is possible with the use of JavaScript, which is one of the most common script languages. Later, during the code development, JavaScript was again utilized due to its versatility. The creation of servers for both front and back ends was achievable by combining this language with Node.js, a JavaScript runtime environment. /14/

In order to store ticket information, a database table is required. SQL serves as the industry-standard language for creating and managing databases, with SQLite functioning as a library that implements SQL database engines. Thus, these tools constitute the technological framework of the project. /15/

5.2 Visual Studio Code

The execution of the project requires a text editor, or preferably, a code editor. In that case, the chosen tool was Visual Studio Code - one of the most used code editors in the present day - due to its easy-to-use nature and appealing interface. /16/

Visual Studio Code supports various programming languages, including HTML, CSS, JavaScript, and SQL if accompanied with the aid of appropriate extensions and programs. This facilitated the consolidation of all coding tasks within a single

software platform, organized in separate files, enabling a compact and organized coding environment.

5.3 Google Chrome

Google Chrome is a web browser developed by Google, being also the only browser supported and used at CitizenM. For that reason, it was the one chosen to test and foster the web application.

Chrome has a console tab that displays scripts on the current webpage. It was useful for observing information flow, allowing the identification of errors in the code script.

5.4 Node.js

To execute scripts on Visual Studio Code, a runtime environment is needed. Even though a web browser was being used, Node.js was also an important environment for building both the front and back-end servers with JavaScript.

A back-end server was necessary to store data on the database and make it always available, while the front-end server was responsible for serving the webpage. Node.js is efficient and versatile to accomplish these tasks, making up the author's runtime environment choice. /17/

5.5 Sqlite

A database to store ticket information was needed, and, for that reason, SQL was used. SQLite is a C-language library that implements SQL and is the most used database engine in the world. /18/

Due to the author's familiarity with this library, it was the option chosen over others. SQLite is easy to use and free source, guaranteeing that the project would not have extra costs.

5.6 Postman

API testing is crucial for ensuring the reliability and functionality of an application's backend services, helping to detect and prevent issues before they impact end-users. For that reason, choosing the right API testing software was critical. For example, Postman, Insomnia, and Newman. were viable options, but Postman was the preferred choice due to colleagues' recommendations.

Postman is one of the most popular software for API testing, allowing users to visualize data retrieved from the API to be later implemented on codes. The vast support available online for this software was what made it the top choice for this project. /19/

5.7 TOPdesk API

An API, or application programming interface, serves as a software bridge facilitating communication between two applications. APIs provide a convenient means to retrieve and exchange data within and between organizations. /20/

This project was fully conceptualized around the TOPdesk Application Programming Interface (API). In this case, the API is a bridge between TOPdesk, and the webpage being developed in this project, and it is where information about tickets, senders, receivers, and such can be found, all in the context of CitizenM.

The TOPdesk API provides information in JSON format. That means once the FIQL query is properly set up on an API testing Software such as Postman, the user should be able to visualize the data in JSON format. Even though the information is readable, it is still not ideally displayed for end-user viewers. When creating a webpage, it is important to know which information to extract from the API, if that is the case, and how to display it. In this project, some JSON values could be used without alterations, but some information had to be modified after being

gathered, including extraction of keywords from string values or modification of data format, which will be further explained.

6 IMPLEMENTATION

6.1 Information Flow

The information flow of the project is based on the following schema:

Step 1: Basic authentication is done - variables containing username and password to access the API are stored in a file that is separated from the code, making this information hidden from users; these variables are read by the API, guaranteeing that the program can access it with safety.

Step 2: Subsequently, the program dispatches a request to the API, triggering the execution of a FIQL (Feed Item Query Language) query.

Step 3: The system then proceeds to retrieve tickets specifically designated for the "IT Support Office Specialists EU".

Step 4: A comparison is made between the newly retrieved results and the existing entries in the database. Initially, the database starts empty, but over time it accumulates ticket data. Each ticket is uniquely identified by its ID, which is the database key, ensuring no duplication. New tickets are added to the database while existing tickets are updated with any changes.

Step 5: Continuing the process, tickets with new IDs are integrated into the database. Conversely, tickets with IDs that have been previously encountered undergo thorough updates, ensuring the database remains current and accurate.

Step 6: Once the database is up-to-date, the program utilizes queries to select only the necessary information for the correct functioning of each webpage.

Step 7: The values meant to be displayed are properly transferred to the HTML code.

Step 8: The values are displayed accordingly.

This schema represents a simplified version of the information flow, being detailed further in the following paragraphs.

6.2 Database

After the creation of sketches, it was possible to predict what information from TOPdesk would be crucial for this project. With that in mind, the next step was finding this data and identifying these properties in the API. With the API URL, it was possible to visualize data in JSON format, but an API testing software would facilitate the process.

The API testing software Postman was chosen to make the visualization of ticket data easier. After setting up a basic authentication to access the information, it was possible to see requests from every department of CitizenM. It is important to note that TOPdesk is utilized by several teams, but for this project, only the tickets from the operator group of IT Office Support EU were intended to be retrieved, so the FIQL query had to be adapted for this purpose. FIQL operates by allowing users to construct queries using a set of parameters, and these queries are inserted into URLs. The initial URL utilized was:

https://citizenm.topdesk.net/tas/api/incidents

After adding a query, the URL changed to:

https://citizenm.topdesk.net/tas/api/incidents?query=operatorGroup.name==\'IT%20Office%20Support%20Specialists%20EU\'&start=${offset}

where
“'operatorGroup.name==\’IT%20Office%20Support%20Specialists%20EU’\”
represents a command that guarantees only tickets from IT Office Support EU are being retrieved, while

'&start=${offset}' determines the page of tickets being retrieved from the TOPdesk API, considering its limitation of fetching only 10 tickets per request. By default,

the variable start, even if hidden, is set to 'start=0', the retrieval begins with the newest tickets, numbered 0 through 9. Altering this parameter to 'start=10' would result in retrieving the subsequent set of tickets, from 10 to 19, and so forth.

To retrieve a larger number of tickets, the variable 'offset' was introduced. This variable increments by 10 with each retrieval until a total of 200 tickets is obtained. Consequently, with each iteration, the offset value increases, facilitating the retrieval of tickets in batches. **Figure 10** provides a simplified representation of this iterative logic.

```
let offset = 0;
do {
    {
        // Code, code, code...
    } offset += 10;
} while (offset <= 190)
```

Figure 10. The variable "offset" as a query parameter

Knowing which FIQL query parameters to use allows the visualization of the intended incident information on Postman, as seen in **Figure 11**.

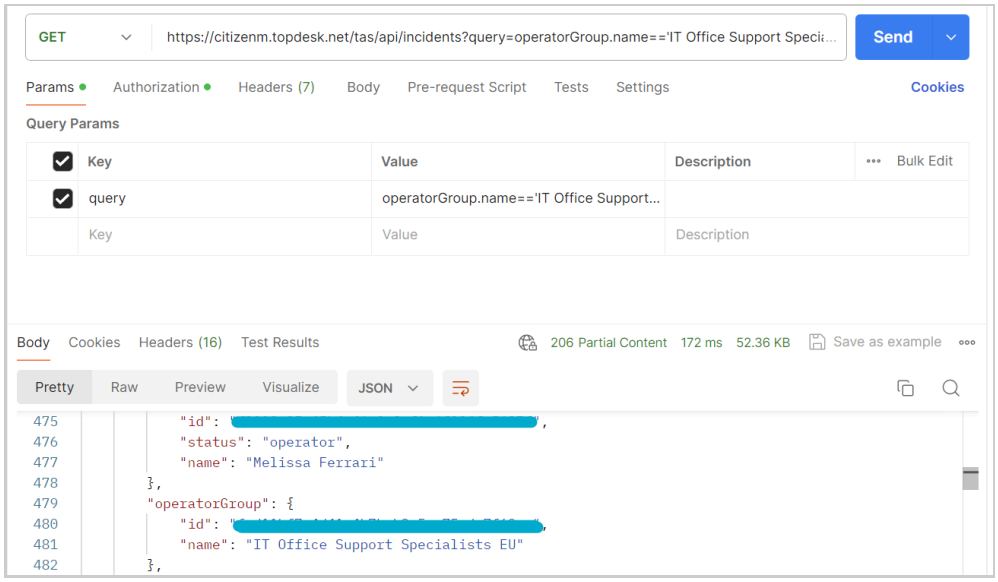
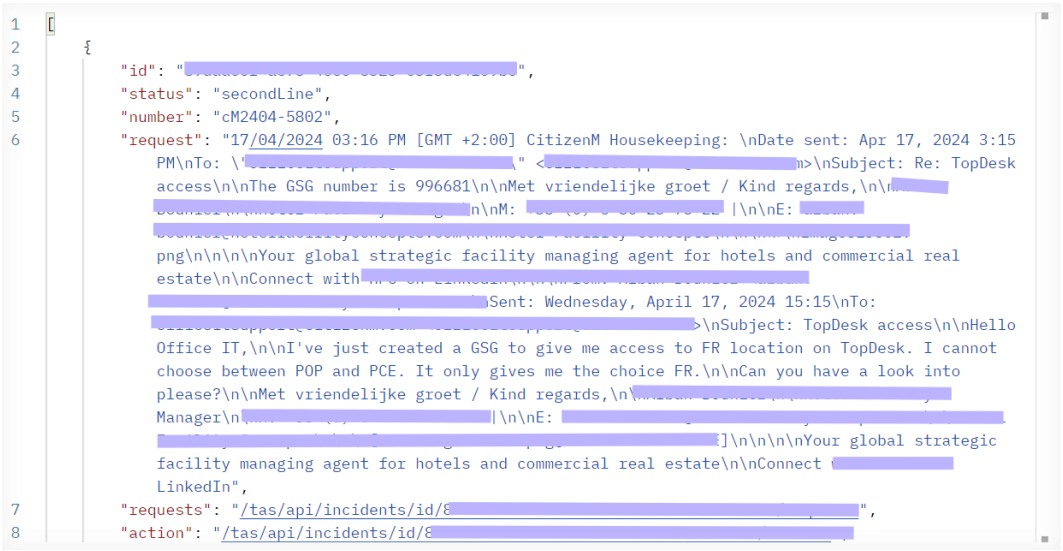


Figure 11. API test using Postman

The API responses are given in JSON format, and it is possible to verify all the available information about tickets received on TOPdesk**. Figure 12** is an example of response, having undisclosed information for privacy reasons.



```
[
    {
        "id": "[redacted]",
        "status": "secondLine",
        "number": "cM2404-5802",
        "request": "17/04/2024 03:16 PM [GMT +2:00] CitizenM Housekeeping: \nDate sent: Apr 17, 2024 3:15
            PM\nTo: \"[redacted]\" <[redacted]>\nSubject: Re: TopDesk
            access\n\nThe GSG number is 996681\n\nMet vriendelijke groet / Kind regards,\n\n[redacted]
            [redacted]\n\nM: [redacted] |\n\nE: [redacted]
            [redacted]
            png\n\n\n\nYour global strategic facility managing agent for hotels and commercial real
            estate\n\nConnect with [redacted]
            [redacted]Sent: Wednesday, April 17, 2024 15:15\nTo:
            [redacted]>\nSubject: TopDesk access\n\nHello
            Office IT,\n\nI've just created a GSG to give me access to FR location on TopDesk. I cannot
            choose between POP and PCE. It only gives me the choice FR.\n\nCan you have a look into
            please?\n\nMet vriendelijke groet / Kind regards,\n[redacted]
            Manager\n[redacted]|\n\nE: [redacted]
            [redacted]]\n\n\n\nYour global strategic
            facility managing agent for hotels and commercial real estate\n\nConnect [redacted]
            LinkedIn",
        "requests": "/tas/api/incidents/id/[redacted]",
        "action": "/tas/api/incidents/id/[redacted]
```


Figure 12. API response example (Postman screen capture)

Moreover, it is important to consider the architectural implications of this data retrieval process. By using tools like Postman, not only was data visualization made easier, but it also set the stage for a scalable and modular system architecture. This means that the project architecture prioritized flexibility, scalability, and maintainability, crucial factors in the long-term viability of the solution.

Once only the tickets from IT Office Support were listed, the process of finding the necessary API values (properties in JSON response) was easier. These values included Status, ID, Title, Operator Name, Closed date, and Creation Date. **Figure 13** showcases a mental map of what information is required to build which webpage, without considering duplicated values. However, the values Subject and Class (highlighted in green on the figure) are non-existent on TOPdesk and would later be artificially set up on the code.

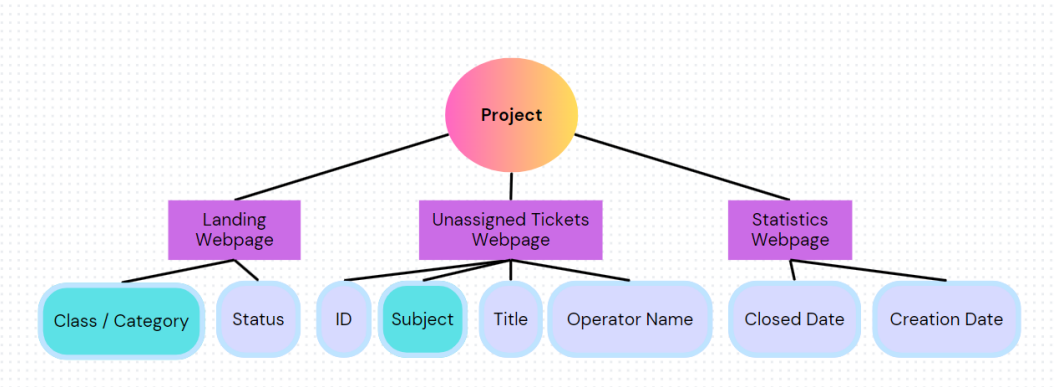


Figure 13. Data needed to compose the project

After analyzing the values individually, a few obstacles were noticed, and it was necessary to evaluate and modify them further:

Closed Date and Creation Date were both shown in the format yyyy-mm-ddThh:mm:ss.000+0000. To facilitate the creation of graphs, it was decided that transforming this data into a different format would be a better option to later store it in the database. So, from that, it was possible to get Closed Month, Closed Year, Creation Month, and Creation Year instead. That means only the month names and years would be stored. A simplified example of how this worked for Closed Date is shown in **Figure 14**. The same logic applies to the Creation Date.

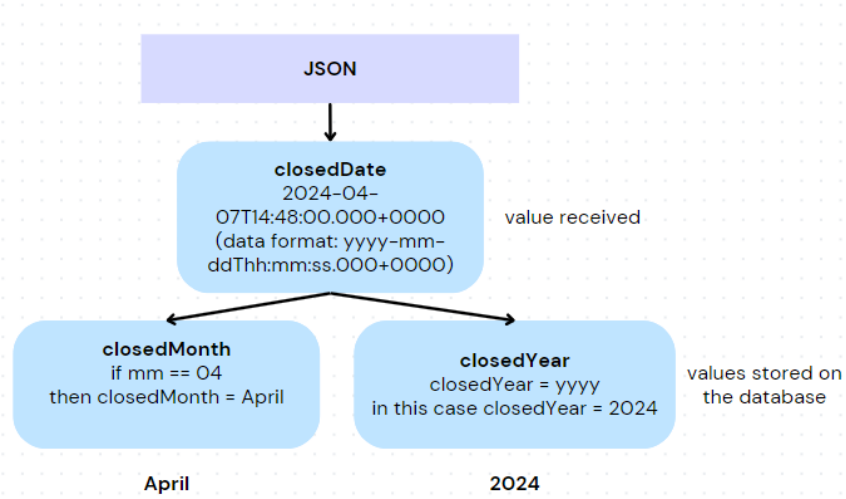


Figure 14. Example of extraction of data for month and year

Subject and Class values were idealized by the thesis author, in order to set up filters on the TOPdesk ticketing system. This information was not directly receivable via the API as a property in the JSON response. To get the values, it was necessary to develop a functionality that identifies keywords from the “request” string (which represents the body of the text inserted by the requester), and, from the first keyword identified, the request is categorized.

The “Subject” value then can only be based on the keywords that were chosen manually by the author. Upon encountering the first keyword, the program stops reading the request, and the logic that applies is “Subject = the first keyword found”. That means, each ticket has only one subject and it is classified into only one category, even if there are several keywords from different classes into that request. That was set up to avoid having duplicates on the table.

Depending on the identified keyword, the ticket is classified into categories ranging from 1 to 5. If no categories are identified, the ticket is assigned to category 6, labeled as "Others"(see **Figure 15**).

```
const keywordClasses = {
    1: ['Contentful', 'Holaspirit', 'LogicGate', 'Lucid', 'Microsoft Office', 'New Relic', 'Sl
    2: ['Amadeus', 'Apicbase', 'Atlassian', 'Florbs', 'GSG', 'iReckonU', 'Nexudus', 'Retool',
    3: ['Adyen', 'Auth0', 'BeyondTrust', 'Beyond Trust', 'Drive', 'E-Mail', 'Google Calendar',
    4: ['Business Central', 'BC365', 'Deputy', 'Duetto', 'Externals access', 'Mpower', 'Okta n
    5: ['Update', 'Camera', 'Webcam', 'Microphone', 'Fix', 'Install', 'Installing', 'Download'
  };

let incidentClass = null;
for (const [classNumber, keywords] of Object.entries(keywordClasses)) {
    if (keywords.some(keyword => request.toLowerCase().includes(keyword.toLowerCase()))) {
        incidentClass = classNumber;
        break;
    }
}

// Assign class 6 if none of the specified keywords were found
if (incidentClass === null) {
    incidentClass = 6;
}
```

Figure 15. Code snippet of Class categorization based on keywords.

For a better understanding of this process, examples were set up as shown:

Example 1: Request – *“Hello, IT! Could you please provide me access to Amadeus, Business Central, and update my computer?”.* In this case, the Subject is “Amadeus”, because this is the first keyword identified. Even though “Business Central” and “Update” are also classified as keywords in this code, these are discarded. Only the first one is considered, and, because of that, the ticket belongs exclusively to Class 1.

Example 2: Request – *“Hi! Can you give me the wifi password?”*. In this request, no keyword is found, so it goes to Class 6 “Others”.

To store all of the necessary information, the creation of a SQL database was crucial. A server for this database was created with Node.js, so the data can be accessed from the webpage.

Once the data has been retrieved and edited, it is stored in the newly created 'ticketsIT' table in the database. Initially, the program was executed to collect all tickets processed by the department. However, given the substantial volume of tickets solved over the years, this operation may endure for hours. Recognizing the inefficiency of repeating this process each time the program is executed, an optimization was needed. So, the program retrieved all the tickets only one time, and then it was modified to exclusively retrieve the most recent 200 tickets daily at 8 am. This adjustment offers a significant margin to ensure data integrity. It is important to mention that the TOPdesk API does not offer methods for retrieving specifically new or modified tickets from the previous hours; it is only possible to retrieve 10 tickets per query, and for that reason, the FIQL had to be updated every 10 tickets and all of them needed to be analyzed, not only the modified ones.

Consequently, during program execution, new tickets are inserted into the table, while any modifications to existing tickets — such as transitioning from "Closed = False" to "Closed = True" — are updated. This functionality is made possible through the utilization of the SQL query commands 'INSERT OR REPLACE INTO ticketsIT'.

6.3 Implementation of the Landing Webpage

In order to have an organized and easy-to-manage workspace, three different files were created to compose the landing webpage, excluding the back and front-end servers. These files contained the HTML, CSS, and script (JavaScript) parts of the code. The HTML and CSS were the first part developed, with the creation of divs, styled and containing the necessary information of each category. With the script file, buttons were created to redirect the user to a second webpage, later mentioned in this thesis. The numerical data showcases the amount of non-completed tickets (see **Figure 16**), which, in database terms, means the count of tickets (rows) where the column “completed” is 0 (false).

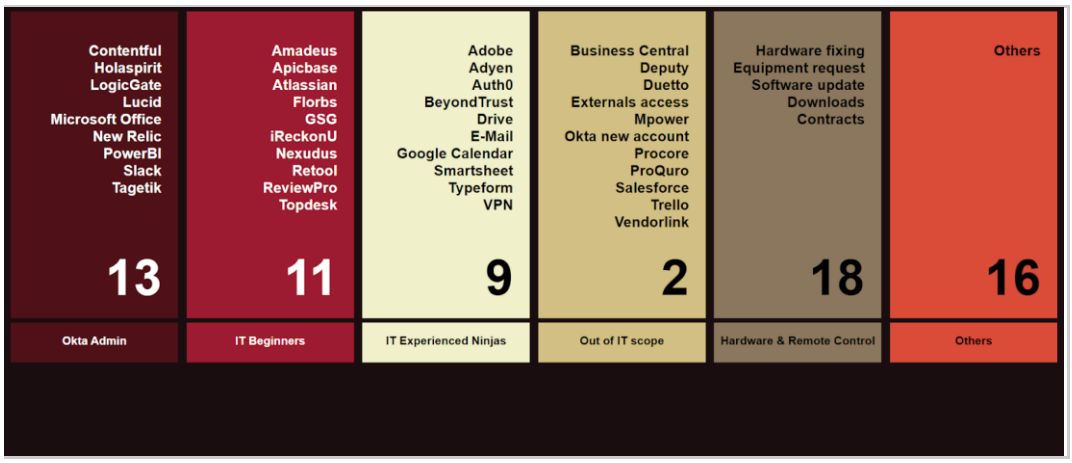


Figure 16. Screen capture of the landing webpage

6.4 Implementation of the Unassigned Tickets Webpage

The unassigned tickets webpage went through alterations from its initial paper and pen sketch (**Figure 8**) to accommodate a bar chart on the right-hand side, which is the number of open tickets (even the ones with operators assigned). JavaScript was utilized to introduce a dropdown menu, enabling users to select a category of tickets and check on the table the preview, containing the ID, subject, and title, of non-completed tickets with no assigned operator. On the graph, the total amount of open tickets in each category was displayed. The Chart.js library

was utilized to facilitate the creation of these graphs and provided a visually appealing aspect. As explained in more detail later in this thesis, with the TOPdesk API it is not possible to generate URLs that redirect the user directly to each corresponding ticket on TOPdesk, but the presence of the IDs is enough to indicate the exact ticket that has to be accessed.

It is also important to mention that the buttons created on the landing webpage redirect the user to the unassigned tickets webpage showcasing the preselected category. For example, if a user clicks on “IT Experienced Ninjas” on the landing webpage, the unassigned tickets webpage is opened with the table displaying the “IT Experienced Ninjas” category. After that, the category can also be changed in the dropdown menu, as seen in **Figure 17**.

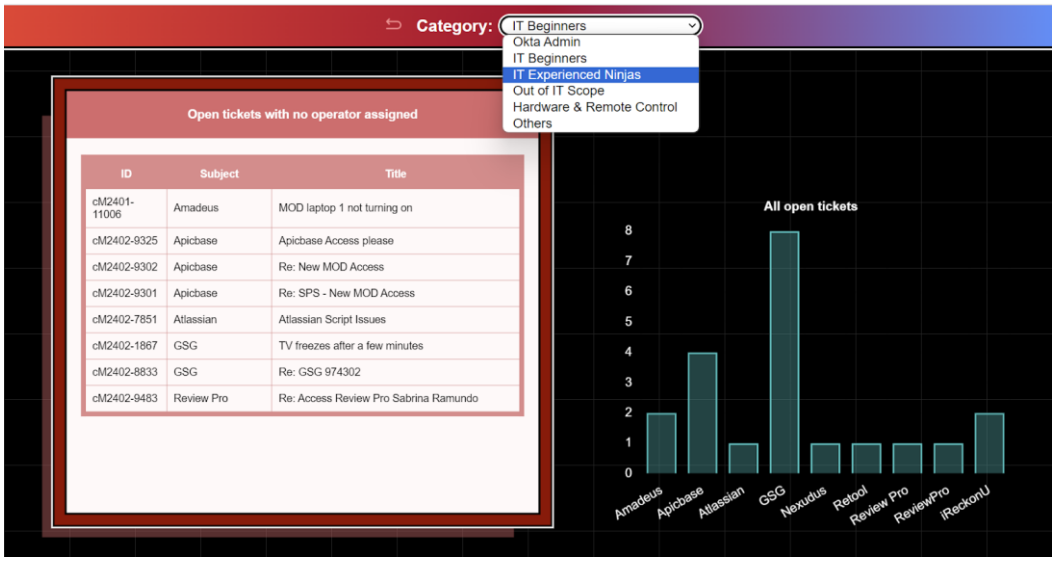


Figure 17. Screen capture of the unassigned tickets webpage

6.5 Implementation of the Statistics Webpage

The execution of the statistics webpage followed similar steps as the previous two pages; however, it deviated from the initial idea to present more relevant information. On the left-hand side block, the number of completed tickets is displayed, while the right-hand side block showcases the number of received requests. At the top left, there is a dropdown list with the names of operators, and

it is also possible to observe buttons to navigate through the years. On this same list, an option “IT Office Support Specialists” was added, to represent the sum of all operators. (**Figure 18**, **Figure 19**).

Figure 18. Statistics webpage with no operator specified

Figure 19. Statistics webpage with specified operator

The dropdown menu, in this case, interacts exclusively with the left-hand side block. Once an operator is chosen, the purple graph changes accordingly, displaying the number of completed tickets on the y-axis, and the names of the months on the x-axis, always taking into consideration the year chosen. This

mechanism, on the back end, works by using different SQL queries. For example, when the operator “Melissa Ferrari” is chosen in the dropdown menu, the query utilized contains “WHERE operator == “Melissa Ferrari”. This same logic is applied to every operator, except “IT Office Support Specialists EU”, which has no operator specified in the query, to get a sum of the tickets of all the operators.

The right-hand side block, representing the amount of received tickets, only changes when the button to change the year is clicked. This graph operates independently of the dropdown menu, as its value remains unaffected by the operator resolving specific tickets.

7 TESTING

Following the completion of the software development phase, it was necessary to test it. To facilitate this, the author of the thesis and IT employees utilized the software as intended.

7.1 TOPdesk Tickets URLs

Initially, the author intended to have in the unassigned tickets webpage embedded links on each ID that would redirect the user to the corresponding ticket in TOPdesk. It was noted that this information was not available on the API, however, after receiving feedback from the users, further research was made by opening different tickets directly on TOPdesk and observing their corresponding URLs. The URL of each ticket was not unique, making it unfeasible to redirect the employees to specific incidents directly from the unassigned tickets webpage. The workaround for this problem is copying the ID from the table and pasting it on the TOPdesk search engine, which still proved to be a simple and fast process, even though it involves a few more steps. For simplicity, a clipboard icon was also added to each ticket so the ID can be copied with only one click, as shown in **Figure 20**.

Figure 20. Unassigned tickets webpage updated with "Copy ID" buttons

7.2 Updating Times

The database of the project is configured for daily updates at 8 am, but this approach has proven to be inefficient. Users have observed that when x tickets are resolved in a day, an additional x tickets are erroneously displayed in the table. Consequently, operators frequently encounter already resolved tickets upon opening them, leading to delays in processing requests. To solve the problem, it was necessary to configure database updates every hour, reducing the number of closed tickets being displayed.

7.3 Lack of Keywords to be Identified

A large number of tickets was also noted and categorized as “Others” (category 6), which defeats the purpose of the program and showcases an inefficient application of filters. The solution was investigating the tickets falling under this category and identifying words that could be utilized as keywords for the categories from 1 to 5. Words such as “update” and its variations (“updating”, “updates”, etc.) are examples of keywords that were added after the deployment. This change cooperated, so the number of tickets in the category “Others” was the closest to zero, as this class is only meant to be used in exceptional requests. However, it was also noted that misspelling of application names would be categorized as “Others”, and this issue could not be resolved due to the unpredictability of the requester’s input.

8 DEPLOYMENT

After finalizing the application for its official launch, it was necessary to decide how to deploy it. With careful consideration, the team leader recommended setting up a physical server. The option chosen was a compact Dell R210 server that was available to be used in the company, popularly known as a “pizza box server”, due to its compact format. That guaranteed adequate performance and integration into the company’s infrastructure. The server runs Ubuntu Linux and uses Apache web server, SQLite, and all needed libraries. Due to security concerns, the application was only reachable from the internal CitizenM network, and not from the public internet. The server had internet access to use the TOPdesk API to retrieve data, and finally, it was available to users. A scheme of how the server hosts the program is shown in **Figure 21**. /21/

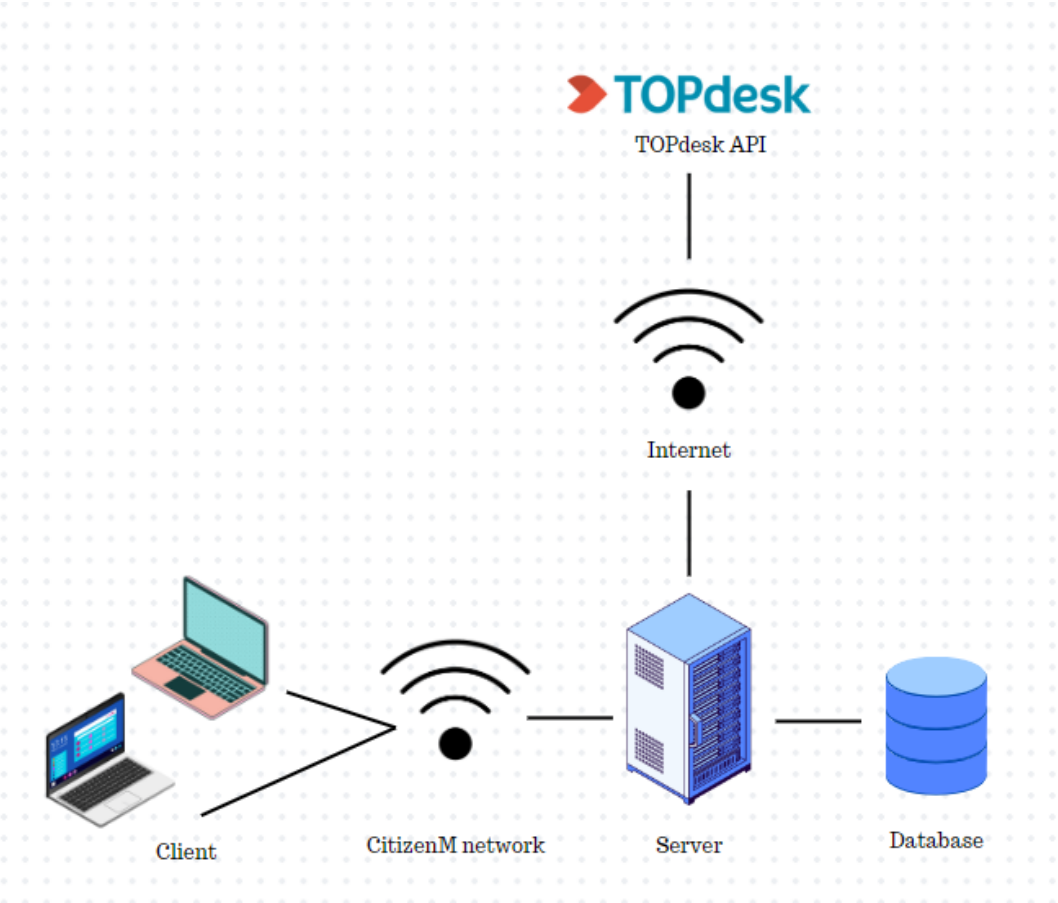


Figure 21. Deployment of the application

In other words, the server connects to the internet so the software can retrieve tickets from the TOPdesk API, this information is stored on the database, and once a user is connected to the CitizenM network, it is possible to access the application.

9 CONCLUSIONS

The project described in this thesis has successfully achieved its objectives of improving the user interface and functionality of the Topdesk ticketing system for CitizenM Hotels. By integrating advanced filtering options and statistical insights, the project has significantly improved the usability of the ticket management process.

The thesis author, working within the IT Support Office team, ensured a deep understanding of user needs and operational challenges. The identification of key issues, such as ticket categorization and statistics tracking, and the development of solutions showcase the practical significance and impact of the project.

The development process, documented throughout the thesis, highlights the implementation of various technologies such as Node.js, SQLite, HTML, CSS, and JavaScript. Through careful planning and execution, the project has integrated frontend and backend components to create a functional web application.

Finally, this thesis is a resource for understanding the process of conceptualizing, developing, implementing, and deploying intelligent incident management solutions within a real-world organizational context. The successful outcome of this project not only impacts IT operations at CitizenM Hotels but also demonstrates the potential of technology to optimize business processes and enhance user experiences in diverse industries.

REFERENCES

/1/ Hakuna. 2023. How Can UX/UI Improve Your Business? Accessed 15.02.2024. https://www.linkedin.com/pulse/how-can-uxui-improve-your-business-studiohakuna#:~:text=Greater%20efficiency%3A%20A%20well%2Ddesigned,knows%20the%20value%20of%20that.

/2/ Team Asana. 2024. What is incident management? Steps, tips, and best practices. Accessed 13.04.2024. https://asana.com/resources/incident-management

/3/ Atlassian. What is incident management? Accessed 13.04.2024. https://www.atlassian.com/incident-management#incident-management-tools

/4/ WhosOn. The difference between first and second line support. Accessed 13.04.2024. https://www.whoson.com/customer-service/the-difference-between-first-and-second-line-support/#:~:text=First%20line%20support%20is%20for,day%20as%20soon%20as%20possible.

/5/ Fontanella, Clint. 2022. What's a ticketing system? Accessed 13.04.2024. https://blog.hubspot.com/service/ticketing-system

/6/ CitizenM. A New Breed of Hotel. Accessed 19.03.2024. https://www.citizenm.com/company/about-citizenm

/7/ TOPdesk. 2022. 10 Things You Don't Know About TOPdesk. Accessed 15.02.2024. https://www.topdesk.com/en/blog/esm/service-culture/10-things-you-dont-know-about-topdesk/

/8/ Miller, George. Psychological Review, 1956. Massachusetts. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information.

/9/ Norman, Don. MIT Press, 1988. Cambridge, Mass. The Design of Everyday Things.

/10/ Okta Help Center. What is Okta and What Does Okta Do?. Accessed 04.01.2024. https://support.okta.com/help/s/article/what-is-okta?language=en_US

/11/ Roberts, Jonathan. 2020. Make a quick sketch before you code. Accessed 03/03/2024.https://csee.bangor.ac.uk/project-rainbow/using-the-critical-thinking-sheet/#:~:text=A%20vision%20of%20what%20your,to%20check%20the%20remaining%20tasks.

/12/ DeClute, Darcy. 2022. Is HTML a Programming Language?. Accessed 10.03.2024. https://www.theserverside.com/feature/Is-HTML-a-programming-language#:~:text=a%20markup%20language-,HTML%20is%20not%20a%20programming%20language.,technology%27s%20name%3A%20HyperText%20Markup%20Language

/13/ Logic Sphere. 2023. What is HTML, and What Is Its Role in Web Development. Accessed 13.02.2024. https://www.linkedin.com/pulse/what-html-its-role-web-development-logicssphere/

/14/ Tech Target Contributor. 2021. What is a Script?. Accessed 04.03.2024. https://www.techtarget.com/whatis/definition/script#:~:text=1)%20In%20computer%20programming%2C%20a,as%20a%20compiled%20program%20is).

/15/ W3 Schools. Introduction to SQL. Accessed 12.02.2024. https://www.w3schools.com/sql/sql_intro.asp

/16/ Uspenski, Anastasija. 2023. Why VS Code remains a developer favorite, year after year. Accessed 04.01.2024. https://shiftmag.dev/vs-code-171/

/17/ Semah, Benjamin. 2022. What Exactly is Node.js? Explained For Beginners. Accessed 10.03.2024. https://www.freecodecamp.org/news/what-is-node-js/#:~:text=front%2Dend%20applications.-,Node.,you%20may%20be%20familiar%20with.

/18/ Sqlite. What is SQLite? Accessed 12.02.2024. https://www.sqlite.org/index.html

/19/ Java T Point. Postman Tutorial. Accessed 12.02.2024. https://www.javatpoint.com/postman

/20/ Frye, Ma-Keba. What is an API? Accessed 19.03.2024. https://elfsight.com/blog/how-to-work-with-developer-console/#:~:text=The%20Console%20tab%20in%20Chrome,commands%20to%20perform%20the%20scripts