MyPoint - Feedback system for Smart Cities

João Rocheteau, Instituto Superior Técnico, joao.rocheteau@tecnico.ulisboa.pt November, 2023

***Abstract*—In contemporary societies, people increasingly gravitate toward large urban metropolises. This ongoing urbanization exerts significant pressure on both the public transportation sector and local authorities. Consequently, public transportation service users often express dissatisfaction with the service's quality. Moreover, there is a pressing need to develop sustainable mobility solutions that can mitigate the environmental impact of the transportation and mobility sector.**

This thesis investigated the feasibility, creation, testing, and marketing of a full-stack real-time feedback and mobility information system for cities, with integration with a data analytics tool whose interface was made available to MobiCascais, a municipal company operating in the public transportation sector.

The results proved that the system was functioning well for the current production load. Load tests showed that over 50 clients can use the service simultaneously. The community adoption at this stage proved not enough to stress the system to its maximum.

***Index Terms*—mobility, feedback, feedback system, smart city**

I. Introduction

In modern societies, citizens tend to concentrate in large urban metropolis where they expect to find better jobs, better public infrastructures, services, and more diverse leisure activities. Therefore, to promote economic growth, governments and local authorities must offer their inhabitants a good quality of life to keep attracting people and business activities.

This sustained growth trend places significant strain on the public transportation sector and local authorities. This heightened pressure is particularly challenging, as public transportation service customers express higher levels of dissatisfaction with service quality when compared to other sectors, which leads to fewer people using them [1]. Consequently, citizens keep using their private cars, so greenhouse gas emissions continue to increase, especially in large and crowded cities that experience very high levels of pollution [2]. Therefore, it is essential to understand the factors that impact the citizens' perceptions of the public mobility service.

The expansion of technology allows authorities to be aware of citizens' perspectives by conducting large-scale surveys or implementing feedback systems where they can report their opinions. These tools have been essential to managing the cities, solving problems, and increasing the quality of the service provided by the local authorities. Moreover, these mobility surveys provide a snapshot of mobility and transportation services at a given time but fail to provide continuous and real-time feedback

The massive adoption of smartphones and wide coverage of LTE/5G cellular networks created new opportunities for citizens' participation, creating a user-centric network in which people, organizations and intelligent objects interact to pursue specific goals [3]. Within this scope, Mobile crowd-sensing applications have gained popularity in recent years, becoming an appealing paradigm for sensing and collecting data [4].

However, existing feedback systems are difficult to adapt to mobility and transportation domains due to the dynamics of these systems [5]. The problem of matching the transportation used by the customer with the location reported in his complaint is not trivial to solve: in some cases, there is no real-time information about the vehicle location, or if it is available, it may require significant processing to get the necessary association for a unique vehicle association. In addition, the range of elements that impact the perception of the quality of the service goes far behind the vehicle. Including different items and properties in such a system would provide a more complete and accurate perspective of the users' experience.

II. Problem Statement

A real-time feedback system for public transport and mobility can improve the quality of the service offered to the citizens and, at the same time, contribute to a sustainable mobility solution capable of reducing the environmental impact of the transport and mobility sector. Nevertheless, these goals can only be achieved if the system does not represent a burden of work for local authorities and if citizens adhere to its use in a responsible way. So, two important questions need to be raised:

- Is it possible to design a system that is easy to implement and to accommodate the specificities of each particular city?
- Can citizens engage in service improvement through extensive and responsible participation?

III. Mobility Support Applications

Two uni-modal MaaS (Mobility as a service) solutions were studied: Gira, a shared bike service offered by Lisbon's city council, and Eurail/Interrail Rail Planner, an app for planning and booking train travel in Europe using Eurail or Interrail pass. Gira allows users to find and rent bikes at designated stations using a city map in the app, with the option to pay for daily, monthly, or yearly subscriptions. Eurail/Interrail Rail Planner allows users to plan routes, view train schedules, make reservations, and view tickets in a paperless process. The app has four main sections: Planner, Stations, My Trip, and My Pass.

Four multi-modal MaaS solutions were also investigated: Uber, Mobi Cascais, Moovit, and UbiGo.

Uber is a ride-sharing and transport information application that allows users to see the best route to a destination using a combination of different transports.

Mobi Cascais is a mobility management system for Cascais, Portugal that offers a range of transportation services, including tracking user movements, seeing transport information,

TABLE I
MOBILITY APPLICATIONS COMPARISON

Company	RailP	Gira	Uber	Mobi	Moovit	UbiGo
Type of transports	1	1	9	7+PT	9	4+PT
Booking	Yes	No	Yes	No	No	Yes
Payment	No	Yes	Yes	Yes	No	Yes
Multi-PT Pay	No	No	No	No	No	Yes
Subscription	Yes	No	No	No	No	Yes
Societal goals	No	No	No	No	No	No
Feedback system	—	Yes	Yes	No	Yes	—
Detailed feedback	—	No	No	No	Yes	—

TABLE II
FEEDBACK APPLICATIONS COMPARISON

Characteristic	Booking	TripAdvisor	Airbnb	Waze
Photos	X	X		X
Text	X	X	X	X
Overall evaluation	X	X	X	
Infrastructure	X		X	X
Cleanliness	X	X	X	
Comfort	X			
Value for money	X	X	X	
Crowdedness				X
Traffic-related				X
Staff and service	X	X	X	
Location	X	X	X	

providing information about network disruptions, or buying tickets.

Moovit is a MaaS company and route planner that offers information about the best routes and current transit options using official and crowd-sourced data.

UbiGo was a multi-modal transportation service that offered six types of transportation. It conducted a trial in Sweden and found that it could reduce the need for private cars and encourage people to change their transportation habits, but it closed due to a lack of funding.

Table I summarizes the properties of the applications previously described. It contains the following characteristics: the number of transports included in the application (PT being public transports), the ability to book and pay while using the service, the integration of a subscription package to use various types of transportation, whether the application has integrated social goals or not, if it has a user feedback system implemented and finally if it provided more detailed information than a simple star rating.

Out of those applications, none of them, except UbiGo in the past, currently presents a flexible subscription that includes several transportation means.

After analyzing several applications regarding transports, it was possible to conclude that most of them had somewhat of a map interface, at some point, that made it possible for the end users to see where the stations and stops were (e.g. Gira, Moovit, MobiCascais, Uber). Another critical feature in MobiCascais and Moovit was the possibility of seeing the expected time for public transport to pass through the station or stop. Gira, Uber, and Moovit had a feedback system in place. However, Gira and Uber didn't present a detailed feedback system on their infrastructures. Moovit developed its strategy by providing detailed feedback given by users, for example, the state of a metro line or the cleanliness of a determined stop.

IV. Feedback applications

Feedback applications are projects that rely on user feedback to provide good service. These applications include Waze, a real-time traffic application that allows users to report road conditions and other issues; TripAdvisor, a platform that allows users to review their experiences with products and services during their travels; Airbnb, a peer-to-peer accommodation platform that allows hosts and users to review each other after a stay and finally Booking, a travelling platform that allows customers costumers to reserve properties, flights among other services. Table II summarizes each system's feedback characteristics.

After analyzing several solutions regarding feedback systems, it was possible to conclude that each solution had its version implemented with parameters that were found pertinent for a particular service. While travelling parameters like host friendliness, cleanness, or location were pivotal, for Waze, crowdedness, closures or the police location were valued.

All travel-related businesses employed star ratings, and it was discovered that some categories were repeated: staff-related, cleanliness, location, and infrastructure condition. Waze's feedback system was distinct from the others, with a three-level-based report mechanism based on the frequency or impact of a determined event. This method proved to be a quick review alternative that is adequate for the service it provides because it does not necessitate a lot of mental burdens, making it ideal for the road. It's possible to conclude that all the applications studied incorporated text input from users, and most of them also permitted photos to ensure that the information posted is reliable and helpful for other users.

V. Rewarding applications

Rewarding systems have been shown to effectively promote the use of collaborative carpooling systems and multi-modal mobility options. In a study published by *Vieira et al.*, [6], 63% of participants found rewards to be a motivating factor in using a carpooling app, with 47% stating that discounts were the most preferred form of incentive.

A more recent study in 2019, conducted by *Tsirimpa et al.* [7], found that rewards effectively promoted greener, multi-modal transportation options, with monetary rewards found to be more effective in promoting car-sharing and park-and-ride solutions, and credits and reserved seats more effective in motivating the use of public transportation. Another study in the Netherlands [8] showed that monetary rewards successfully increased the use of e-bikes, with utilization rising from 0% to 68% after half a year of participation.

Several applications have implemented rewarding mechanisms to motivate users to use their services. Moovit allows users to contribute and receive points that translate to levels, increasing motivation to use the app. Waze also includes gamification features such as points and levels based on

TABLE III
REWARDING APPLICATIONS COMPARISON

App	Points	Levels	Alerts	Badges	Daily prize	Ranking
Moovit	X	X	X			
Waze	X	X	X	X		X
League of L.	X	X		X	X	X
C.Royale	X	X	X	X	X	X

contributions, push notifications about relevant events and the ability to earn badges and skins by helping other users.

The multiplayer online battle arena game League of Legends includes a ranking system that tracks players' progress. It awards them for moving up the ladder and daily rewards to motivate users to keep playing. Clash Royale provides daily and weekly missions to its players and has a "trophy mechanism" similar to a leaderboard and push notifications to encourage progress in the game. It also includes the concept of points and levels within the game.

Overall, every system had one constant: they had points and levels. Waze was found to be, compared to Moovit, a complete experience specifically because of the skins that users unlock while ramping up points. Finally, both games that were presented had a better rewarding experience. This may be part of why they are among the most played games worldwide, having several ways to engage users.

VI. Architecture

A. Interactions

MyPoint aims to promote sustainable mobility by allowing mobility managers to improve the service offered to the citizens based on their feedback. In addition, local entities may also participate in the system by rewarding user's participation. The system provides specific interfaces and applications for each user type coordinated through a central server. Figure 1 overviews the system with its entities and their interactions.

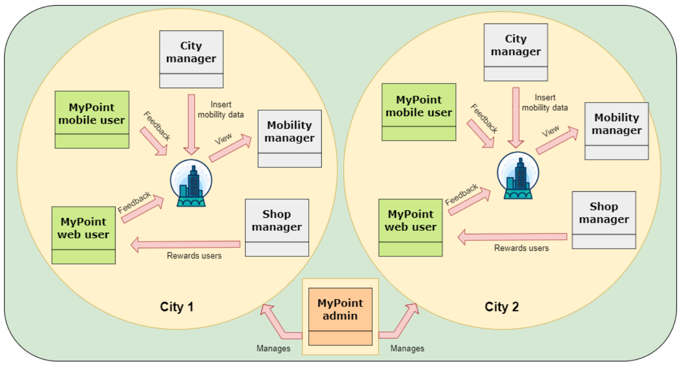

Fig. 1. System entities and interactions

The main entities that interact with the systems are:

- **myPoint admin**: a member of myPoint staff that creates and manages instances of the platform of each partner city or local authority.
- **Users**: the cities' inhabitants that access myPoint to retrieve mobility information, report feedback or redeem points at partner shops.
- **Shop manager**: entities that want to partner with myPoint, allowing users to redeem the items they make available in exchange for publicity.
- **City manager**: a member of the technical team of a partner responsible for maintaining up-to-date information on the city's mobility assets.
- **Mobility manager**: a member of the operational team of the city responsible for city mobility service. The team will access reports on users' feedback and use them to improve the service.

B. Components

The central component of MyPoint is the central server, which mediates all user interactions, offering endpoints for regular clients to create, read, update, and delete information, creating requests with specific information within its body. A complete list of these endpoints is described in appendix [9].

Users can access the server via their mobile application to view all system facilities. They can click on the map or scan a QR code to provide feedback or view information. Additionally, users can access features such as checking parking and routes, viewing timetables, creating accounts, logging in, and redeeming prizes. Users can also access MyPoint through a web application; however, the functionalities are limited to providing feedback on facilities and obtaining information about stop timetables using the website's link commonly incorporated in QR Codes present in facilities.

Prizes can be obtained in events or stores around the city using QR Codes. These transactions will occur if the user has enough points, ensures that item and store details are correct, and confirms the action.

To create items, stores and events, the stores' managers must contact the myPoint administrator to set up the partnership. The MyPoint administrator creates a QR code for each product the store associates with the system using a specific interface created for administrative duties. Additionally, designated City managers can also access certain parts of this interface to update the mobility information for the city by uploading GTFS files, a standard composed of a set of files that contain details about stops, schedules, and routes.

Mobility managers can also view the state of their city by accessing a dynamic report created by the system, which retrieves information from the Central Server database. It is possible to filter data to obtain the best insights dynamically, allowing cities to know parameters for which facilities have been performing below average and why or heat maps of positive or negative streams of opinions.

C. Architecture overview

This section introduces all the project's services, the group of components and how they interact with the entities.

- **Central Server**: This remote component makes data available to stakeholders daily.
  - **Entry point**: Forwards entity requests to the relevant endpoints that provide the requested services

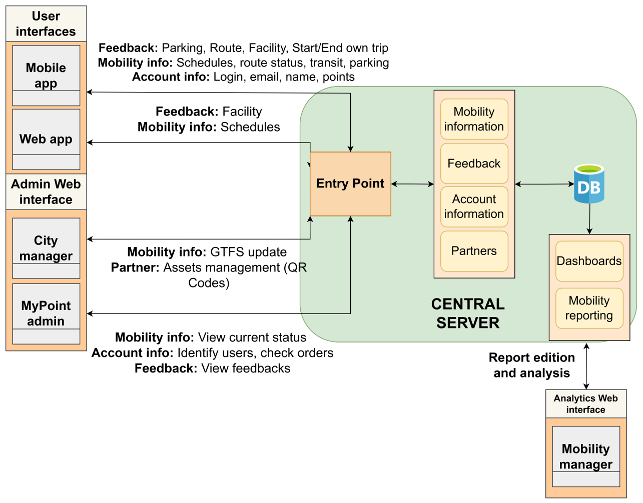
User interfaces
Mobile app
Web app
Admin Web interface
City manager
MyPoint admin
Feedback: Parking, Route, Facility, Start/End own trip
Mobility info: Schedules, route status, transit, parking
Account info: Login, email, name, points
Feedback: Facility
Mobility info: Schedules
Mobility info: GTFS update
Partner: Assets management (QR Codes)
Mobility info: View current status
Account info: Identify users, check orders
Feedback: View feedbacks
Entry Point
Mobility information
Feedback
Account information
Partners
DB
Dashboards
Mobility reporting
CENTRAL SERVER
Report edition and analysis
Analytics Web interface
Mobility manager

Fig. 2. System architecture

- **Data storage**: The data storage component manages customer information, transportation data, infrastructure details, and feedback. It accomplishes this by employing a relational database, allowing for structured and organized storage of the relevant data.
- **API**: Interface for third-party clients to conveniently retrieve and utilize valuable data derived from client feedback and information about the current transportation. Using the REST architectural style, the API excels in its universality and ease of use with HTTP requests, making it a good choice for this project.

- **Administrator Web Interface**: This section presents a vital functionality that increases the project's maintainability, allowing users with privileged roles to view, create, update and delete data. From this interface, all stakeholders' tickets can be answered, business opportunities can be addressed, and unusual occurrences can be monitored. The staff can access all the required information by accessing a set of available services.
- **User interfaces**: Composed of mobile and web user applications, these interfaces connect the end user and the services MyPoint provides through HTTP requests. Design considerations, functionalities, and navigation were addressed.
- **Analytics interface**: To show results to companies and stakeholders and monitor the development of the system, a PowerBI-based interface was created recurring to heat maps, graphics and lists that displayed, in a dynamic manner, all the information within the database of the system, in which a connection is directly made. A general description of the dashboard is provided.
- **Security architecture**: In each of the components referred, a security assessment is done to understand what measures must be taken when implementing a solution such as MyPoint. Mobile application authentication using JWT tokens, measures to mitigate common attacks such as SQL Injections, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF), encryption of the communication, and hashing of sensitive data at rest, among others, are discussed.

VII. Implementation

After deciding the architecture used to build this solution, it is now time to choose the appropriate technologies that will allow the creation of the project in conformity with the requirements.

A. Design options

During the development of the application, due to constraints and requirements, a few design options had to be taken in order to ensure the final product was delivered on time without compromising quality:

- Central System: Since this project only has one remote server available, all the components: User web app, API, Admin web page, User web page, and the database share the same physical infrastructure; this solution doesn't take advantage of redundancy or load balancing.
- Android app: Due to a lack of time and infrastructure, the mobile software front-end was only tested for the Android operating system, which is the most widely used in the phone market. However, it must be indicated that the framework possesses the same codebase for iOS and Android.
- Security-based approach: The creation of this system was based on a security-by-design approach, as it is an essential aspect of any system.
- Default visualization interface: At this moment, it was decided to create a default data visualization interface that companies could access to have real-time mobility data.
  Besides this, other aspects will be detailed during this chapter's implementation overview of the different services.

B. Deployment and Entry Point

Since the service must be open to all people who use it, whatever their location and whenever they want, these components must be hosted remotely on a physical server. With the help of the University of Lisbon's VMCloud initiative [10], the project received a Linux server with 2 VCPUs, 20 GB of Volume, and 4GB of RAM where the central server, with both the API and the database, is implemented.

In order to receive and process the requests that arrive at the central server, a web server is necessary. While Django, the web framework used to create the central server services, makes available a local web server, it is advised by their documentation [11] not to be used within production since it has not gone through security audits and performance tests and instead should be used only during testing and debugging.

To solve this, Nginx, a popular open-source web server and reverse proxy, handles the requests at the central server. Among others, this technology [12] offers security features such as the possibility of adding SSL termination, caching, and load balancing that are pivotal to creating a scalable and efficient system. Caching and load balancing were not implemented due to budget and time constraints, whereas SSL was.

Along with Nginx, Guicorn [13], a WSGI often used within Python services, is used to process the requests Nginx forwards, translate them to a format that the web application can read, and return the response.

For the server implementation, these general steps were done:

- In order to ensure good deployment practices and separation of concerns, it is vital to create a Linux user specifically for handling the deployment process rather than using the root user.
- It's necessary to set up appropriate permissions for the deploy user to access the remote code repository, in our case, GitLab, to facilitate code deployment. It allows this user to easily pull code developers created and deploy it to the server.
- PostgreSQL should be installed on the server, and a remote database should be created. A database user with the necessary permissions should also be added. This will enable the server to store and retrieve data as needed.
- Since Django is programmed in Python, a virtual environment is a standard approach that should be set up to ensure that the application runs smoothly and without conflicts between packages.
- Django should be configured to access the database by inserting data such as the IP, ports, and user details.
- Jenkins must be installed to provide the capability of deploying code automatically, as well as configured to have the necessary permissions in the code repository (explained in more detail in VII-E)
- For a secure and efficient communication setup, it is crucial to deploy Nginx and Gunicorn services, acquire a CA certificate, and configure all components to utilize this certificate for encryption.

After successfully deploying the service remotely for testing purposes, it must be prepared so people worldwide can use it.

SSL/TLS encryption is imperative to establish robust and secure communication across all service components. This can be achieved by acquiring a reputable Certificate Authority (CA) certificate, which guarantees that all data exchanged between the database and web services remains confidential and protected from unauthorized access.

Certbot [14] is a popular, easy-to-use [15], open-source tool for automatically configuring and managing SSL/TLS CA certificates issued by Let's Encrypt [16] for different services. This tool was implemented in the remote server and was used to make the connections to the database, API, Administrator web application, User Web application and the Pipeline Web service (explained in the next section) encrypted, trusted and safe.

Since Let's Encrypt certificates are only valid for 90 days, certbot addresses this by creating a timer that runs twice daily and automatically renews any certification within thirty days of expiration.

C. Web framework

As mentioned before, web services are used to orchestrate communication among the different components of the MyPoint system. They provide an API that mediates access to the central server and database from the different third-party platforms, such as the MyPoint user mobile and all web applications using HTTP messages.

With several Web frameworks in the market, with different languages and capabilities, Django, based on Python, was chosen to follow this project at this stage [17]. Django, currently employed in companies such as Spotify and Instagram [18], is one of the most popular solutions to create Web services. With over 18 years of active support and a broad community, this framework encourages rapid and pragmatic design due to its structure, well-written documentation, an extensive ecosystem of plugins and add-ons, and built-in security features.

Besides Django, the Django REST framework is installed to ensure the creation of the REST API previously mentioned [19]. A complete list of these endpoints is described in appendix [9].

For the API, JWT (JSON Web Tokens) [20] was employed to enable the system to function seamlessly across both web and mobile applications by utilizing SimpleJWT [21]. In order to restrict access to specific functionalities to only authorized clients or admins, it is possible to verify the access token present in the ”Authorization” header and check the user's permissions on different endpoints.

The API implements rate-limiting measures on all endpoints to prevent abusive activities, following Django's rate-limit package. This limits the number of requests that a particular person can make within a specific period.

Another security measure that is implemented by default by Django is the hashing of user passwords. Upon a new user creation, the Django framework applies SHA256 hashing to the password, ensuring that user passwords are not seen by either administrators or possible malicious stakeholders, ensuring that this data is secure at rest [22].

D. Data Storage

During the project's initial phase, the team decided to develop and evaluate the data structure to ensure that crucial data necessary for the proper functioning of services is accessible to front-end applications.

As referenced in the last chapter, the data storage solution that was found to be the best choice for the project's requirements was one to possess a relational structure, meaning that interactions with it would have to go through SQL. To create this feature, PostgreSQL, a known and widely used relational database management system, was used due to its open-source nature, robustness, extensibility and broad community support [23], as well as Pg4Admin, an open-source platform designed for the administration and development of the DBMS [24] that allowed the creation of its tables, rows and columns that adhere to standard language structure.

E. CI/CD Pipeline

After implementing the database and web services, it is crucial to prioritize the maintainability aspect of the project.

Since the code will be remotely deployed, it's essential to have a straightforward and accessible way of adding new features, fixing bugs, and removing functionalities not well-received by the public.

To ensure this, a CI/CD Pipeline was introduced, allowing developers to push code from local environments to the central server and enabling users to see real-time updates.

Jenkins, one of the most widely used automation tools worldwide, was utilized to create this Pipeline. The third-party plugin, Blue Ocean, provides a modern UI to display the Pipeline's state and identify any problems.

As part of the software development workflow automation process, it is crucial to establish a connection between the local developer and a remote code repository, such as GitLab [25], so one can update the code. Besides the local connection, it's also necessary to create a link between the remote server where Jenkins is installed and the same repository so it can have all the permissions to detect changes and actively trigger the pipeline process.

The process is comprised of three stages that must be completed without errors to ensure that new code is deployed:

- The Build stage installs recently added packages locally.
- The Test stage runs Django tests to ensure the newly added functionalities are bug-free before connecting to the remote server.
- Finally, the Deploy stage connects to the remote server and runs the deploy script created beforehand that is already installed in the server.

F. Administrator Web application

Besides creating the endpoints to serve a wide range of applications, the project decided to use the Django native functionality of the Administrator web page and implement it inside the central server. The framework presents a group of default "/admin" endpoints that render built-in web pages where an overview of the production database is given based on previously defined models. These endpoints are only allowed for administrators or super users and require log-in. This group of pre-defined URLs helped manage the service as it presented a group of simple and intuitive web pages to view data in the tables and accomplish CRUD operations while being easily configurable. While most of the implementation was covered by Django, a few functionalities were added in the admin.py file within the MyPoint business application:

- CSV Exports: The system allows for CSV export of data to improve the answering of tickets and sharing of information in a clear and organized manner. Administrators can select specific rows from any table and export them in .csv format. This feature is designed to streamline communication in a readable form.
- Import of GTFS: Since this information could be up to millions of lines, there is the possibility of uploading GTFS files to the administrator webpage, which will automatically review the files and put them in the database format, which is very similar.
- Deletion of all GTFS-related and feedback data: This button was added to the webpage for rebuilding the system.
- Deployment initial configuration: This button is useful when a system is just starting, for example, when it is first deployed, as it automatically creates all infrastructure types such as buses, rails and bus stops and their categories and subcategories.

Besides this, Django's default UI did not present a modern, user-friendly design. To accomplish this, jazzmin [26], a drop-in third-party package for the framework, improved user experience by redesigning the platform but maintaining the functionalities described before.

G. User Mobile application

The user mobile application plays a vital role in this project. Apart from enabling users to give feedback and check their schedules, it also provides parking information, rewards users with points that they can redeem, and gives real-time traffic updates with a dynamic map feature. Additionally, it allows users to track a specific transport route and view its upcoming stops. Flutter, a multi-platform open-source framework, was used to create this component because of its ability to develop for iOS, Android and Web, rapid widget-based development and performance [27]. While this project is currently only tested in Android, Flutter's single code base should provide a simple way to integrate the current application into iPhone users.

The application will function based on the creation of HTTP requests to the REST endpoints present in the API, which will allow actions such as the retrieval of facilities, creation of feedback, and management of log-ins, among other vital operations.

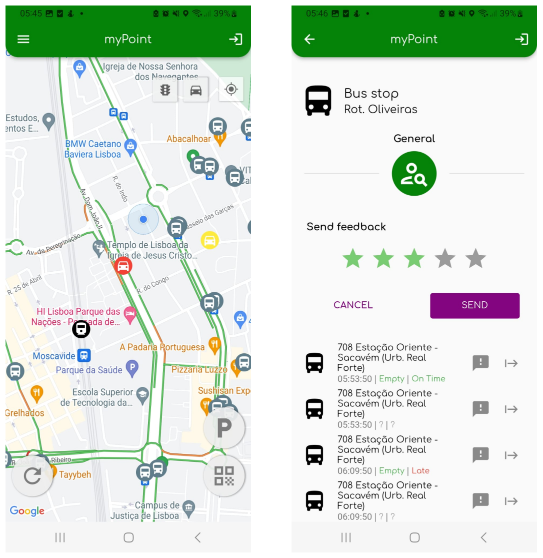
myPoint
myPoint
Bus stop
Rot. Oliveiras
General
Send feedback
CANCEL
SEND
708 Estação Oriente - Sacavém (Urb. Real Forte)
05:53:50 | Empty | On Time
708 Estação Oriente - Sacavém (Urb. Real Forte)
05:53:50 | ? | ?
708 Estação Oriente - Sacavém (Urb. Real Forte)
06:09:50 | Empty | Late
708 Estação Oriente - Sacavém (Urb. Real Forte)
06:09:50 | ? | ?

Fig. 3. User Mobile app snapshots

After successfully testing the application in a real-world environment and several different Android phones, bringing the product from local settings to the market was essential. In Android, the most famous application distribution channel is the Google Play Store [28], present in most devices by default, which connects clients to millions of applications worldwide.

This channel distributes the Google Play console, an interface that is open to all developers at a price of around 25€, that allows the management of releases, monetization, analytics, policies and performance of the project.

To comply with Google Play's regulations [29], MyPoint has implemented a privacy policy [30] that outlines the management of user data in addition to the app itself. The terms and conditions of use can also be found at [31].

On September 19th, the application passed Google Play's testing and was released [32] under the ”Open testing” [33] category in the Console. This means that the app is now available for all platform users in Portugal; however, this phase aims to identify and resolve any issues that may have been missed during development or testing. Notably, feedback during this phase is only visible to MyPoint's administrators and won't affect public ratings.

H. User Web application

At a certain point of the planning phase, where requirements of the mobile application were already defined, it was decided to create a new interface based on the web that would introduce users to the system in a prudent, more straightforward approach without the need to install or trusting an application that, at first, wouldn't have much users and ratings. The website is an important strategic point due to:

- It serves as a way to continue to engage in public conversation.
- It's an excellent way of advertising the mobile application by redirecting users to install it.
- While it allows clients to give feedback and view schedules, it only has some of the functionalities the mobile application presents.

The website's design adopts a ”mobile approach” that adheres to the standard Android design metrics. This is because the service aims to cater to users who access the interface by also scanning a QR code on a marketing poster without having the mobile version installed while on the street. This project is a fork of the initial Mobile application Flutter code base, having very similar design and interactions adapted to the web. Given that most of the code utilized in creating the mobile app pertains to this particular service, this segment will summarise the alterations made to guarantee that the website operates without issues while offering a seamless, positive user experience.

I. Analytics interface

As data is collected, it's necessary to have means to understand, visualize and analyze it. In compliance with the requirements of the MobiCascais team, the tool chosen for this effect was PowerBI. This interactive data visualization tool allows companies to make reports based on several data sources.

Within the context of this project, PowerBI directly connects to the central server database and retrieves its data, which, through filtering, creates its own schema with only the information MyPoint deems necessary for third parties to be able

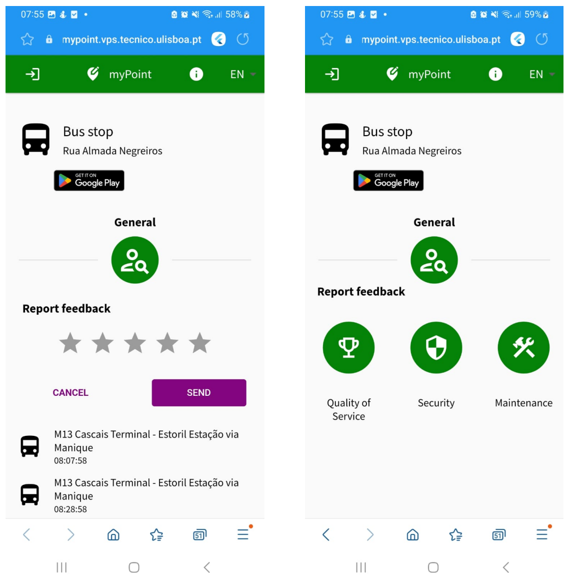

Fig. 4. User Web app snapshots

to use. The communication between the database and PowerBI is encrypted.

A Report with key performance indicators was created featuring important details as:

- The location and heat maps of feedbacks around the city
- The amount of authenticated and non-authenticated feedback
- The number of clients
- Number of feedback regarding each category and subcategory
- Score of feedback
- Feedbacks by facility type
- Record of the last opinions recorded in the database

VIII. Marketing and Posters

After the system was implemented and ready for use by the community, the project proceeded to its last stage: Marketing and poster glueing.

During a previous conversation with MobiCascais, certain infrastructures were provided to promote this project. As a result, several marketing posters were created with a unique QR code specific to each facility, using a Python script that utilized infrastructure data to generate PDFs of the design created.

During this phase, posters in all eight stops of MobiCascais M43 [34] routes were implemented, as well as two buses where users could scan the inherent QR Codes.

IX. Evaluation and Results

During development, several tests were made to ensure the correct functioning of the MyPoint services before going out in production. While testing could have been more extensive, the backend services were target of load, unit and integration testing.

- Integration testing: Integration tests were performed during the API development for most of the REST endpoints

created. These tests help understand whether a determined set of functionalities works as expected.
- Unit testing: Tests were made to check if a URL resolves correctly to a specific view class (http handler). Generally, unit testing refers to testing a specific functionality.
- Load testing: These tests were performed to understand the service capability regarding concurrent usage.

A. Unit testing

During the development phase, unit testing was conducted to ensure all the URLs were going to the desired view and no URL would route to the wrong functionalities. We used Django's REST Framework SimpleTest to ensure the URLs were correctly resolved and routed to the specific request handler.

B. Integration testing

Integration testing was performed using Django's REST Framework APITestCase module, allowing fake data, users, roles, and request creations to test endpoint responses and compare them to the desired output. Among others, this verifications were made:
- Confirm if an endpoint is only accessible by the roles permitted (e.g. admin-only endpoints)
- Confirm if clients can make the desired actions for what the endpoint was created
- Confirm if the response body data, if required, possesses the necessary information that the front end needs
- Confirm if the HTTP status code from the response is as expected

C. Load testing

These tests were created using JMeter, an open-source tool for analyzing and measuring the performance of various services. It is important to note that these tests are all conducted locally within the development machine and not in a production environment. As a result, the performance values in this section are expected to be better than they are due to the controlled nature of the testing environment, potential network transit times and other external factors that could directly impact performance.

The computer used to host the backend server is comprised of the following specifications:
- Processor: 11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz
- GPU: RTX 3050 Laptop version
- RAM: 16GB
- Disk: 500GB SSD
- OS: Windows 10 Home

Two tests are performed in order to evaluate the backend's performance:
- Simulating users: Request's response time when having multiple users loading and giving feedback within the application - following a storyline approach
- Endpoint stress test: Have several requests at the same time in a determined endpoint

This test comprises analyzing indicators regarding the system by simulating the process that a user would have to do to give feedback. The script created tried to emulate the user behaviour by creating HTTP requests necessary to create an action and adding random timers to simulate parts where users might think.

1) The user loads the application and waits for the loading screen: GET HTTP request
2) The user clicks on a facility somewhere in the map: Random timer (3500,8000) ms
3) Users get the schedule of a facility: GET HTTP request
4) The user views the menu and sees where to create feedback: Random timer (2500,5000) ms
5) User gives general feedback: POST HTTP request
6) Gets the categories and subcategories of a facility: GET HTTP request
7) The user chooses a category and a subcategory: Random timer (1500,5000)
8) User gives feedback: POST HTTP request

This script adds a new testing user each second, up to two hundred. When the set of actions of a single client is completed, it loops from step two to step eight, simulating a user that is still using the app to give feedback on other facilities; therefore, the results of this test comprise two hundred users actively using the service after three minutes and 20 seconds. Figure IX-C presents a chart that shows the profile of the average time of requests during each second.

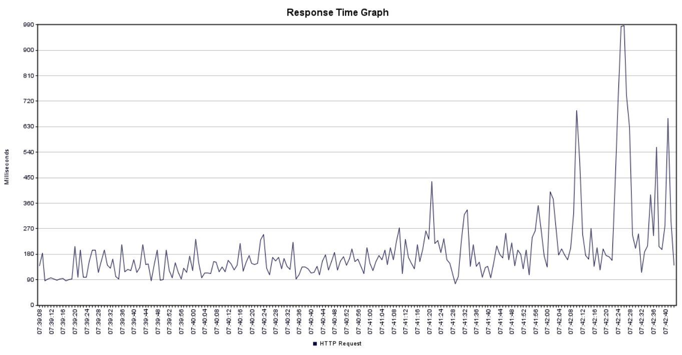

Fig. 5. Response Time vs. User Growth (one per second): Simulating User Behavior

TABLE IV
STATISTICS OF THE USER BEHAVIOR TEST

Average	Median	90%	95%	99%	Er%
235ms	164ms	428ms	603ms	1361 ms	0.05%

However, this analysis fails to show what would happen when the variability of users is consistent for a more extended period; therefore, four more tests were created based on the previously defined user behaviour but with groups of 50,100,150 users using the app simultaneously for 5 minutes.

Another essential test to understand the service's load capabilities is to test the request times for each endpoint. However, this assessment was performed for only some endpoints,

TABLE V
TEST 2 - 50/100/150 SIMULTANEOUS, CONSTANT, USERS TEST

Users	Average	Median	90%	95%	99%	Er%
50	251ms	142ms	405ms	1187ms	1545 ms	0.2
100	656ms	206ms	2395	3550ms	4102ms	2.41
150	1401ms	403ms	3581ms	8028ms	9138 ms	7.66

as clients do not utilize some and, therefore, do not hold significant relevance for performance evaluation.

The tested endpoints encompass all the business logic and data users request. They provide a solid foundation for assessing the performance of both the database and the code in the API, as the final time recorded indicates their overall efficiency. The tests performed are based on 50,100,150, and 200 requests per second for each functionality using the local and production setups. Since, at this stage, the project is already up and running, only endpoints that do not create new data are tested within the remote environment. The testing results are present in appendix A.

From all the test results, the following conclusions were taken:

- 150 and 200 request/s proved far too much for both the local and remote setup, going way beyond the thresholds defined (400ms for all endpoints [35], expect for loading - "/api/facilities/view/en" - which is higher).
- It was found that the Django local web server had processing limitations as several requests were dropped due to server overload. However, the production setup with Nginx and Guicorn, the entry point, proved to be more efficient in handling more requests at the cost of a higher time due to packet travel time.
- Four endpoints performed well above the threshold due to the time spent processing the data:
  - /api/busstop/information/asset - Get the schedules of a determined facility
  - /api/trip/trip value/stops/asset - From the stop a user is it, retrieve the following stops of a determined trip
  - /api/token/ - Get the refresh and access token
  - api/feedback/ - Provide feedback
- Both giving feedback and retrieving the map of the next stops are key functionalities that should be improved.
- As expected, the loading screen was the endpoint that took longer; however, the under-load stress achieved values of over 10 seconds.

D. Results

The project went on production officially on October 1st, 2023 and having now 28 days since the release, we highlight some of the following results:

- 32 Feedbacks were retrieved
- thirteen of whom were rated as five stars
- five of whom were rated as four stars
- five of whom were rated as three stars
- three of whom were rated as two stars
- six of whom were rated as one-star
- 50% of the feedback received was from the General category
- 15.63% of the feedback received was from the Security category
- 15.63% of the feedback received was from the Quality of Service category
- 18.75% of the feedback received was from the Maintenance category
- 26 out of the 32 feedbacks were about bus stops
- 6 client accounts were created
- 4 out of the total number of feedbacks were created by logged-in users

The results have shown that, at this point, the community's adoption of the service is not significantly pronounced. With only 32 feedback submissions, the data suggests that the marketing efforts have not been effective in persuading people to try any of the applications. Ineffective marketing could have several causes, such as the difficulty in scanning the available QR codes due to their placement within the buses and the fact that the M43 route only has one "traditional" stop, most of which were rudimentary with no seating, leading to inconspicuous placement, potentially making some feedback seem unreliable.

X. CONCLUSION

The MyPoint project was focused on studying the feasibility of a real-time mobility and feedback system for public transport and mobility that would be accessible to citizens. The aim was not only to provide a convenient solution for commuters and valuable information to mobility entities but also to contribute to sustainable mobility practices that could help reduce the environmental impact of the transport and mobility sector. The results obtained demonstrated that the service worked well, providing fascinating insights into the data obtained while also guaranteeing sufficient response times. The service managed to hold the current workload; however, retrieving only 32 feedbacks, the stress needed to be more meaningful to provide concrete conclusions.

Overall, user interaction could have been more significant, having only six registered accounts, proving some distrust at this project stage and ineffective marketing strategies.

REFERENCES

[1] V. I. Rosca, "The impact of public transportation investments made by the municipality of bucharest upon quality of life," *Journal of Community Positive Practices*, vol. 18, no. 3, pp. 39–49, Sep. 2018. [Online]. Available: http://www.jppc.ro/index.php/jppc/article/view/137

[2] P. Labee, S. Rasouli, and F. Liao, "The implications of mobility as a service for urban emissions," *Transportation Research Part D: Transport and Environment*, vol. 102, p. 103128, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361920921004235

[3] M. H. Miraz, M. Ali, P. S. Excell, and R. Picking, "A review on internet of things (iot), internet of everything (ioe) and internet of nano things (iont)," in *2015 Internet Technologies and Applications (ITA)*, 2015, pp. 219–224.

[4] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry, "A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities," *IEEE communications surveys & tutorials*, vol. 21, no. 3, pp. 2419–2465, 2019.

[5] R. Utriainen and M. Pöllänen, "Review on mobility as a service in scientific publications," *Research in Transportation Business & Management*, vol. 27, pp. 15–23, 2018, special Issue on Mobility as a Service. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210539518300336

[6] V. Vieira, A. Fialho, V. Martinez, J. Brito, L. Brito, and A. Duran, "An exploratory study on the use of collaborative riding based on gamification as a support to public transportation," in *2012 Brazilian Symposium on Collaborative Systems*, 2012, pp. 84–93, (Accessed [17-10-2023]).

[7] A. Tsirimpa, A. Polydoropoulou, I. Pagoni, and I. Tsouros, "A reward-based instrument for promoting multimodality," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 65, pp. 121–140, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1369847819300786

[8] J. de Kruijf, D. Ettema, C. B. Kamphuis, and M. Dijst, "Evaluation of an incentive program to stimulate the shift from car commuting to e-cycling in the netherlands," *Journal of Transport & Health*, vol. 10, pp. 74–83, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214140517306564

[9] MyPoint. (2023) Api docs. (Accessed [13-10-2023]). [Online]. Available: https://mypoint.vps.tecnico.ulisboa.pt/doc/

[10] Instituto Superior Técnico. (2023) Vm cloud - what is vmcloud. (Accessed [30-10-2023]). [Online]. Available: https://iaas.projects.dsi.tecnico.ulisboa.pt/index.html#what-is-vmcloud-

[11] Django. (2023) Django documentation - introduction to django tutorial: The development server. (Accessed [17-10-2023]). [Online]. Available: https://docs.djangoproject.com/en/4.2/intro/tutorial01/#the-development-server

[12] Nginx. (2023) Nginx - features. (Accessed [19-10-2023]). [Online]. Available: https://nginx.org/en/

[13] Gunicorn. (2023) Gunicorn - python web server gateway interface http server. (Accessed [19-10-2023]). [Online]. Available: https://gunicorn.org/

[14] E. F. F. (EFF). (2023) Certbot - eff's free, automated, and open certificate authority. (Accessed [19-10-2023]). [Online]. Available: https://certbot.eff.org/

[15] C. Tiefenau, E. von Zezschwitz, M. Häring, K. Krombholz, and M. Smith, "A usability evaluation of let's encrypt and certbot: Usable security done right," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1971–1988. [Online]. Available: https://doi.org/10.1145/3319535.3363220

[16] L. Encrypt. (2023) Let's encrypt - free ssl/tls certificates. (Accessed [18-10-2023]). [Online]. Available: https://letsencrypt.org/

[17] Django. (2023) Django official website - home page. (Accessed [13-10-2023]). [Online]. Available: https://www.djangoproject.com/

[18] I. A. Bairagi, A. Sharma, B. K. Rana, and A. Singh, "Uno: A web application using django," in *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 2021, pp. 1371–1374, (Accessed [12-10-2023]).

[19] Django REST Framework. (2023) Django rest framework - home page. (Accessed [13-10-2023]). [Online]. Available: https://www.django-rest-framework.org/

[20] M. Jones, J. Bradley, and N. Sakimura, "Json web token (jwt)," IETF, RFC 7519, 2015. [Online]. Available: https://tools.ietf.org/html/rfc7519

[21] Django REST framework SimpleJWT. (2023) Django simplejwt documentation. (Accessed [30-10-2023]). [Online]. Available: https://django-rest-framework-simplejwt.readthedocs.io/en/latest/

[22] Django. (2023) Django documentation - passwords. (Accessed [20-10-2023]). [Online]. Available: https://docs.djangoproject.com/en/4.2/topics/auth/passwords/

[23] PostgreSQL. (2023) Postgresql - official website. (Accessed [13-10-2023]). [Online]. Available: https://www.postgresql.org/

[24] PGAdmin. (2023) pgadmin - home page. (Accessed [30-10-2023]). [Online]. Available: https://www.openapis.org/

[25] GitLab. (2023) Gitlab documentation. (Accessed [19-10-2023]). [Online]. Available: https://docs.gitlab.com/

[26] Django. (2023) Django jazzmin documentation. (Accessed [13-10-2023]). [Online]. Available: https://django-jazzmin.readthedocs.io/

[27] K. Kishore, S. Khare, V. Uniyal, and S. Verma, "Performance and stability comparison of react and flutter: Cross-platform application development," in *2022 International Conference on Cyber Resilience (ICCR)*, 2022, pp. 1–4.

[28] Google. (2023) How google play works. (Accessed [30-10-2023]). [Online]. Available: https://play.google/howplayworks/

[29] Google. (2023) Google play store - developer policy center. (Accessed [30-10-2023]). [Online]. Available: https://play.google.com/about/developer-content-policy/

[30] MyPoint. (2023) Privacy policy. (Accessed [30-10-2023]). [Online]. Available: https://mypoint.vps.tecnico.ulisboa.pt/privacypolicy

[31] MyPoint. (2023) Terms of service. (Accessed [30-10-2023]). [Online]. Available: https://mypoint.vps.tecnico.ulisboa.pt/termsconditions

[32] MyPoint . (2023) Google play store - mypoint app. (Accessed [30-10-2023]). [Online]. Available: https://play.google.com/store/apps/details?id=com.mypoint.mypointfrontend

[33] Google. (2023) Google play store - open testing. (Accessed [30-10-2023]). [Online]. Available: https://play.google.com/console/about/opentesting/

[34] M. Cascais. (2023) Mobi cascais - autocarros municipais - horários e percursos. (Accessed [19-10-2023]). [Online]. Available: https://mobi.cascais.pt/geral/autocarros-municipais-horarios-percursos

[35] D. Green, C. Hargood, and F. Charles, "Contemporary issues in interactive storytelling authoring systems," in *Interactive Storytelling*, R. Rouse, H. Koenitz, and M. Haahr, Eds. Cham: Springer International Publishing, 2018, pp. 506–508.

APPENDIX

TABLE VI
LOCAL ENDPOINT STRESS TEST - 50, 100, 150 AND 200 USERS

Endpoint	HTTP method	50 requests/1000ms				100 requests/1000ms			
		Avg	**Med**	**99%**	**Er%**	**Avg**	**Med**	**99%**	**Er%**
1	GET	50	35	149	0	64	55	170	0
2	GET	49	36	151	0	66	55	168	0
3	GET	75	54	173	0	454	388	1085	0
4	GET	260	286	451	0	1106	1076	2019	0
5	GET	2479	2522	2855	0	4663	4878	5845	7
6	GET	56	42	152	0	103	84	267	0
7	GET	–	–	–	–	–	–	–	–
8	POST	63	47	161	0	110	112	224	0
9	POST	47	36	147	0	65	57	177	0
10	POST	519	556	792	0	1262	955	2588	0
11/	POST	3	3	39	0	2	2	3	0
12	POST	1740	1656	3279	0	2071	2030	3741	43
13	POST	61	53	167	0	213	207	562	0
14	POST	70	48	165	0	131	126	263	0

Endpoint	HTTP method	150 requests/1000ms				200 requests/1000ms			
		Avg	**Med**	**99%**	**Er%**	**Avg**	**Med**	**99%**	**Er%**
1	GET	396	267	1562	0	474	262	1578	0
2	GET	250	226	650	0	514	370	1321	0
3	GET	802	812	1881	0	1248	1378	2491	6.5
4	GET	1473	1510	2912	8.67	1632	2011	2805	28
5	GET	3941	4392	6236	36	3439	2041	6218	53.50
6	GET	340	267	1085	0	642	410	1717	0
7	GET	–	–	–	–	–	–	–	–
8	POST	374	272	1196	0	648	687	1697	0
9	POST	273	198	749	0	549	301	1575	0
10	POST	1707	1901	3191	17.33	1839	2036	3023	40.50
11	POST	2	2	3	0	2	3	3	0
12	POST	2091	2028	3615	64.67	2116	2032	3865	73.50
13	POST	538	306	1608	0	924	776	2141	0
14	POST	413	279	1183	0	688	693	2082	0

1) /api/client/account/
2) /api/facility/view/{asset}/{lng_code}
3) /api/categories/{asset_type}/{lng_code}
4) /api/trip/{trip_value}/stops/{asset}
5) /api/busstop/information/{asset}
6) /api/item/{item_id}/
7) /api/facilities/view/en
8) /api/item/{item_id}/
9) /api/excursion
10) /api/token/
11) /api/token/refresh/
12) /api/feedback/
13) /api/feedback/carpark
14) /api/feedback/carpark/validate