Université de Lille
IUT de Lille
Département informatique

Advanced Maintenance for the Railroad using Augmented Reality - Front-end part

Internship Report

by

Julien BOUIN

Internship supervisor : Marie Deletombe

University supervisor: Mateus Mendes

ISEC Coimbra, June 2024

Instituto Superior de Engenharia
Politécnico de Coimbra

Acknowledgments

I would like to thank the University of Lille and Mrs Deletombe, teacher at the IUT A, for giving me the chance to be part of the Erasmus program this year. I also would like to thank Mr Mendes and Mrs Malta, for being welcoming and for their help during this internship.

Table of Contents

List of Figures

Acronyms

API Application Programming Interface

AR Augmented Reality

BEM Block Element Modifier

CMMS Computerized Maintenance Management System

CSS Cascading Style Sheets

HTML Hypertext Markup Language

ISEC Instituto Superior de Engenharia de Coimbra

IUT University Institute of Technology

MVT Model-View-Template

ORM Object-Relational Mapping

REST Representational State Transfer

Abstract

Completing a three-month internship at the Instituto Superior de Engenharia de Coimbra (ISEC) in Portugal, as part of the Erasmus program, has been a rewarding and formative experience for me. Collaborating with my colleague Rayane Belguebli and under the guidance of our supervisors, Mateus Mendes and Rita Malta, our mission was to develop a full-stack website with tight integration between the backend and frontend. This project, part of a larger research endeavor titled "Advanced Maintenance for the Railroad using Augmented Reality," aimed to connect a Computerized Maintenance Management System (CMMS) to an Augmented Reality (AR) interface, thereby facilitating maintenance operations management.

My primary role was focused on frontend development of the application. Additionally, I also contributed to the development of mobile applications enabling maintenance technicians to update their interventions directly from their phones, even in the absence of Wi-Fi connectivity.

This internship represents a unique opportunity to explore the synergies between CMMS and AR technologies while deepening my understanding of frontend development. By concentrating my efforts on the frontend, I aimed to bridge the gap between backend infrastructure and user interfaces.

In the end, our goals were successfully fulfilled, as we developed the website and mobile app, making tangible contributions to the research project. This experience has bolstered my motivation by emphasizing the relevance and significance of our work.

Chapter 1

Introduction

At the end of my two years at IUT A of the University of Lille, I undertook a three-month internship to validate my diploma. I chose to do this internship abroad through the Erasmus program, which led me to Instituto Superior de Engenharia de Coimbra (ISEC) in Portugal. I completed this project with my colleague Rayane Belguebli, and we were supervised by our tutor, Mateus Mendes.

We were tasked with developing a full-stack website requiring close integration between the backend and frontend. Our application, developed with Django, will be connected to a Computer-Aided Maintenance Management System (CMMS) to effectively manage maintenance operations. The frontend, also built with Django, will serve augmented reality (AR), an immersive technology that enhances user experience by providing contextual information in real time. Our project is directly related to the work presented in the reference article [1]. Our goal is to connect the CMMS to AR, so that technicians can directly see what they need to do on our site and then visualize it through AR ; therefore, we will be responsible for the server and the database. My role primarily focuses on frontend development.

Additionally, I also worked on the development of mobile applications allowing maintenance technicians to update their interventions directly from their phones, even without Wi-Fi connection. My goal here is to create functional applications for technicians to use on a daily basis.

This internship offers a unique opportunity to explore the synergies between CMMS and AR technologies, and to deepen my understanding of frontend development. By primarily focusing on frontend development, this internship aims to bridge the gap between backend infrastructure and user interfaces. Furthermore, contributing to a project with real-world applications, as demonstrated in [1], is a great source of motivation, highlighting the relevance and importance

of our work.

In the second chapter, we will present my research on what a CMMS, Django, virtual reality, and the interest of augmented reality in maintenance are. Then, in Chapter 3, we will detail the development of the website using the Django framework. Finally, in Chapter 4, we will address the development of a mobile application using React Native, thus providing technicians with the possibility to use it without a connection.

Chapter 2

2.1 What is Django ?

Django is an open-source web framework written in Python, designed to simplify the development of complex and scalable web applications. Launched in 2005, Django is maintained by an active community of developers and is widely used in the industry to create a variety of web applications, ranging from simple websites to complex web platforms.

Key features of Django [2] :

— **Full-stack Web Framework** : Django is a full-stack web framework, providing all the necessary tools to develop both the frontend and backend of a web application, including database management, business logic, URL routing, and form handling.

— **MVT Architecture (Model-View-Template)** : Unlike the MVC model, Django follows the MVT architecture where models represent application data, templates are responsible for presentation, and views contain the processing logic.

— **ORM (Object-Relational Mapping)** : An ORM is a programming technique that allows mapping objects from an object-oriented programming language (such as Python, Java, etc.) to data stored in a relational database. Rather than writing SQL queries directly, the ORM allows developers to interact with the database using familiar objects and methods, thus facilitating application development and maintenance. In summary, an ORM abstracts the complexity of the relational database, allowing developers to work with data in a more object-oriented way.

— **URL Routing System** : Django uses a URL routing system that maps URLs to corresponding views, providing clean and efficient route management in the application.

— **Built-in Security** : Django comes with many built-in security features, such as protection against common security vulnerabilities like SQL injection attacks, cross-site scripting (XSS) attacks, and bypassing form validation.

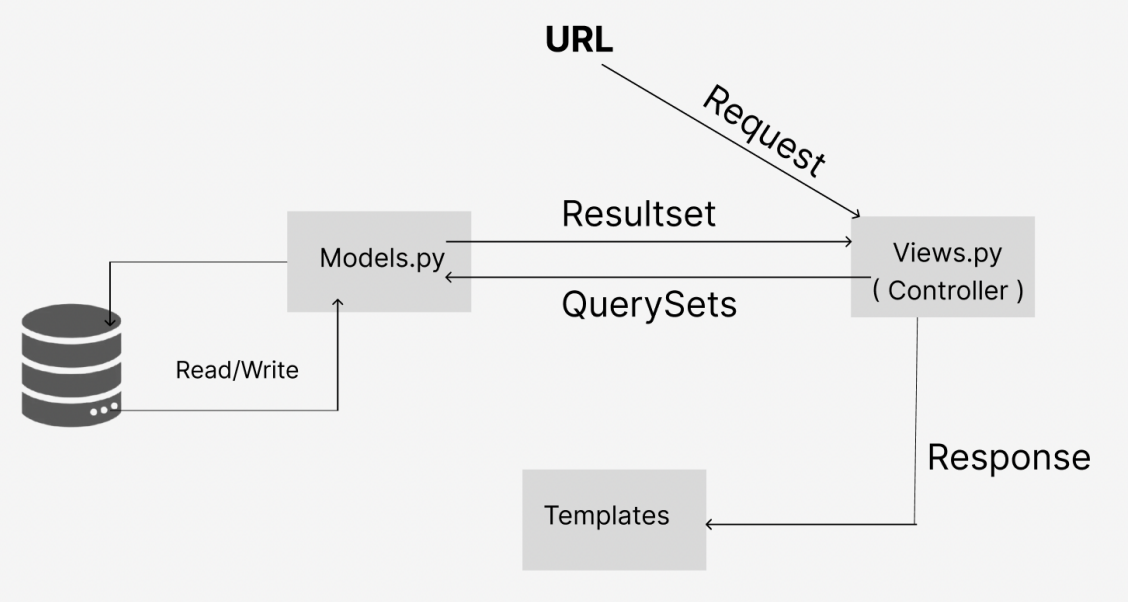

Figure 2.1 – Representation of MVT

— **Automatic Administration** : Django provides an automatically generated admin interface from the data models, allowing administrators to manage site content without needing to write specific code for it.
— **Batteries Included** : Django follows the "batteries included" principle, meaning it provides many ready-to-use features like user authentication, session management, pagination, static file management, and more.
— **Extensibility** : Django is highly extensible, allowing developers to add custom features to their application by using third-party packages or writing their own custom code.

2.2 What is a CMMS (Computerized Maintenance Management Systems) ?

Computerized maintenance management system (CMMS) are sophisticated software solutions deployed by organizations to effectively streamline their maintenance operations. These systems form the backbone of maintenance management, orchestrating various tasks such as planning, organizing, tracking and managing maintenance activities across the organization.

At the heart of a CMMS is a robust and comprehensive database, which acts as a centralized repository of vital information relating to the organization's maintenance operations. This database stores a wealth of data, including detailed equipment specifications, maintenance

schedules, historical maintenance records, spare parts inventory and work order histories.

One of the key benefits of a CMMS is its ability to empower maintenance technicians by providing them with instant access to relevant information needed to perform their tasks accurately. Using the CMMS interface, technicians can quickly identify equipment requiring maintenance, view detailed maintenance procedures, and determine the availability of required spare parts. This streamlined access to information not only improves the efficiency of maintenance activities, but also helps minimize downtime by ensuring timely maintenance interventions.

In addition, a CMMS plays a crucial role in facilitating informed decision-making at the managerial level. By analyzing the data stored in the CMMS database, managers can gain valuable insights into the performance of their maintenance operations. They can track key performance indicators, identify trends and forecast maintenance needs, enabling them to make data-driven decisions regarding resource allocation, budgeting and strategic planning. For example, managers can calculate the financial implications of repairing machine breakdowns versus preventative maintenance measures, thereby optimizing maintenance budgets and improving operational efficiency.

In summary, a CMMS is a powerful tool for effectively managing a company's maintenance resources. By centralizing maintenance-related information, streamlining processes and providing valuable insights, a CMMS enables organizations to optimize their maintenance operations, minimize downtime, reduce maintenance costs and, in turn, ultimately drive overall organizational performance and market competitiveness. But although it is very effective and useful for businesses it nevertheless remains very complex, as this article explains [3] .

2.3 What is Augmented Reality ?

Augmented Reality (AR) is a revolutionary technology that overlays virtual elements such as images, videos, or 3D models onto the real world, typically through an electronic device such as a smartphone, tablet, or smart glasses. Unlike Virtual Reality (VR), which creates an entirely virtual environment, AR enhances and improves the real environment by adding digital information in real time.

Augmented reality applications can take various forms, offering a wide range of uses. They can serve as product visualization tools, allowing consumers to virtually try out products before purchasing them. Additionally, they can be used in interactive games that blend the real world and the virtual world, providing immersive and innovative gaming experiences. In the field of

education, augmented reality can be used as an interactive learning tool, allowing learners to explore subjects in a more engaging and practical manner. Similarly, in the field of assistance, it can provide step-by-step guides to help users perform complex tasks, such as repairing electronic devices or navigating unfamiliar environments.

In summary, augmented reality offers significant potential to transform our interaction with the world around us, providing rich and immersive experiences in many areas, ranging from entertainment to education to commerce and industry.

Figure 2.2 – AR Illustration[1]

2.4 Integration of Augmented Reality into Maintenance

The integration of Augmented Reality (AR) into maintenance operations signifies a remarkable advancement in optimizing industrial processes. By harmonizing the sophisticated functionalities of Computerized Maintenance Management Systems (CMMS) with the immersive capabilities of augmented reality, companies can curtail unplanned downtime and enhance overall operational efficiency. Through AR, maintenance technicians gain access to real-time contextual information seamlessly integrated into their field of vision, empowering them to conduct inspections and repairs with unprecedented precision. For instance, utilizing augmented reality glasses or headsets, a technician can effortlessly visualize critical data such as electrical schematics, technical specifications, or repair instructions overlaid onto the actual equipment, thereby diminishing the reliance on traditional paper manuals or digital documents on laptops and enabling faster and more precise interventions. Additionally, AR facilitates interactive step-by-step guides for intricate maintenance procedures, mitigating the risk of human errors and elevating the quality of repairs. By seamlessly integrating augmented reality into

1. Source : gettyimages.fi/ZGl1bV9WMl8xOTgw/JUUyJTgwJThBJ Last search : 2024-04-24

the existing framework of maintenance management systems, companies not only modernize their maintenance processes but also ignite innovation and fortify their competitive stance in the market.

A tangible application of AR in maintenance is illustrated in the article by Henderson and Feiner (2010) [4]. They designed a prototype aimed at assisting military mechanics in executing routine maintenance tasks within an armored vehicle turret. The prototype utilizes a head-mounted display to augment the mechanic's natural vision with text, labels, arrows, and animated sequences, tailored to enhance comprehension, localization, and task execution. A qualitative survey revealed that mechanics found the augmented reality conditions intuitive and satisfactory for the tested task sequence.

Building on this foundation, a recent empirical study by Fiorentino et al[5] was undertaken to evaluate the efficacy of technical maintenance aided by interactive instructions in augmented reality. This study introduces an innovative methodology integrating augmented visualization on a large screen in conjunction with a fusion of multiple fixed and mobile cameras, leveraging commercially available solutions. During the experiment, 14 participants undertook a series of four maintenance tasks centered on manual inspections of a motorcycle engine. With augmented instructions providing support through visual labels, 3D virtual models, and animations, participants executed tasks such as tool selection, bolt removal, and part disassembly. Comparisons were made between traditional paper manuals and augmented instructions. Rigorous statistical analyses unequivocally demonstrated that augmented instructions significantly reduced overall execution time and participant error rates, thereby underscoring the transformative potential of AR in optimizing maintenance processes.

To delve further, let's explore a practical method of implementing augmented reality using Python. We will examine the Python OpenCV library [2], originally developed by Intel, specializing in real-time image processing. Supported by Willow Garage and ItSeez, this library offers a myriad of functions for developing programs, from raw data to basic graphical interfaces. Additionally, it can be used to develop facial recognition methods.

Furthermore, there are other notable frameworks for augmented reality development, such as ARToolkit and A-Frame. ARToolkit is an open-source tracking library for creating augmented reality applications. It provides a set of tools and utilities for tracking natural feature markers in real-time, enabling developers to overlay digital content on physical objects. A-Frame, on the other hand, is a web framework for creating virtual reality experiences using HTML and JavaScript. It simplifies VR content creation by providing declarative HTML syntax

2. Website of OpenCV : `https://opencv.org/`

and components for rendering 3D scenes. These frameworks offer versatile solutions for AR development across different platforms and technologies.

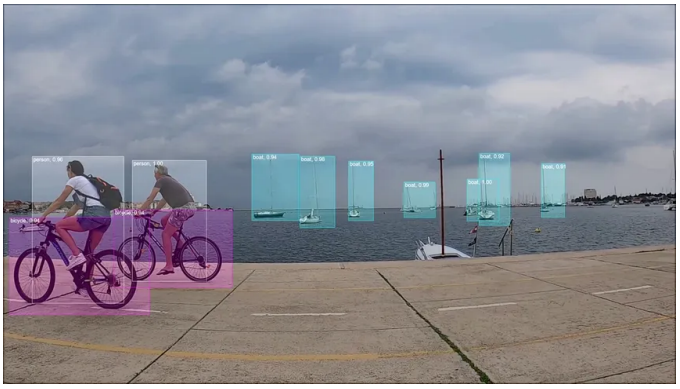

Figure 2.3 – Example of OpenCV[3]

3. Source : viso.ai/ZGl1bV9WMl8xOTgw/JUUyJTgwJThBJ Last accessed :2024-04-26

Chapter 3

Web Application

This chapter delves into the detailed process of developing our web application, from defining the requirements to practical implementation. It highlights the key stages of development, including requirement analysis, user interface design, and feature implementation.

3.1 Requirement Definition with Use Cases

The first step in development involved defining the functional requirements of the system using use cases. These allowed us to model the interactions between users and the system, identifying the main features and associated workflows.

3.1.1 Use Cases

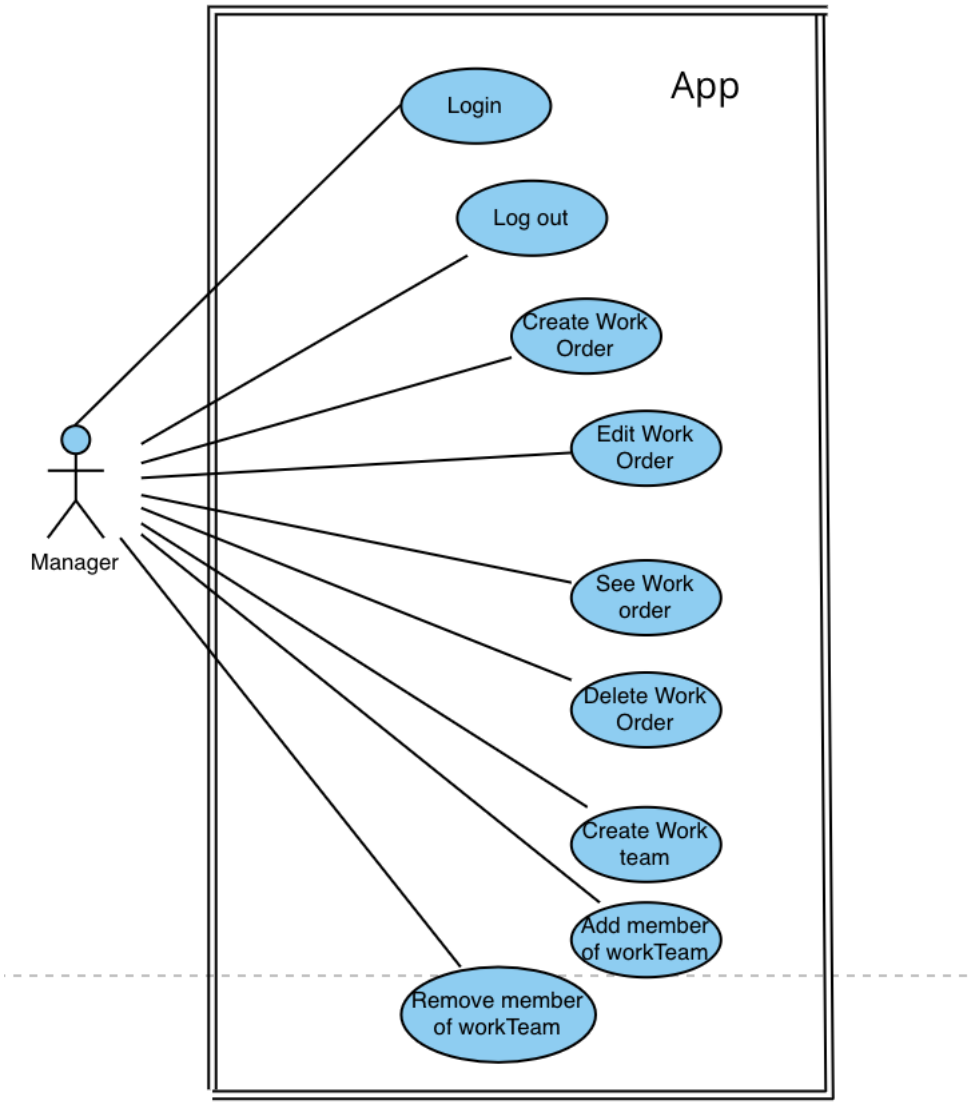

Figure 3.1 – Use cases for Technician

3.1.2 Description of Use Cases

Login :

Actor : Technician

Objective : Log in to access features.

Preconditions : On-site, connected to the Internet, and have an account.

Postconditions : Logged in and can access features.

Steps :

1. Click on login.
2. Enter required information.
3. Click login.
4. See success message.
5. Logged in.

Logout :

Actor : Technician

Objective : Log out.

Preconditions : On-site and logged in.

Postconditions : Logged out.

Steps :

1. Click logout.
2. Confirm logout.
3. See success message.
4. Logged out.

View Work Orders :

Actor : Technician

Objective : View work orders.

Preconditions : On-site and logged in.

Postconditions : View available work orders.

Steps :

1. Click work orders link.
2. See list of work orders.
3. View work orders.

Select Work Order :

Actor : Technician
Objective : Choose a work order.
Preconditions : On-site, logged in, and on work orders list page.
Postconditions : Selected work order, removed from list.
Steps :

1. Click desired work order.
2. Click "Choose this work order".
3. Work order associated.

Validate Work Order :

Actor : Technician
Objective : Validate a work order.
Preconditions : On-site, logged in, and selected a work order.
Postconditions : Validated work order, no longer in list.
Steps :

1. Click "View my work orders".
2. See list of work orders.
3. Click work order to validate.
4. Click "Validate work order".
5. See success message.

Add Media to Work Order :

Actor : Technician
Objective : Add media to a work order.
Preconditions : On-site, logged in, and selected a work order not yet validated.
Postconditions : Media added to work order, viewable by others.
Steps :

1. Click "View my work orders".
2. See list of work orders.
3. Select work order.

4. Click "Add Media".
5. Upload media.
6. Submit.
7. See success message.

Alternative Paths : Display error message if uploaded media is incorrect.

3.2 User Interface Design with Mockups

Next, I created mockups to visually represent the user interface. These mockups were invaluable in clarifying design expectations and facilitating faster, more organized graphical interface development.

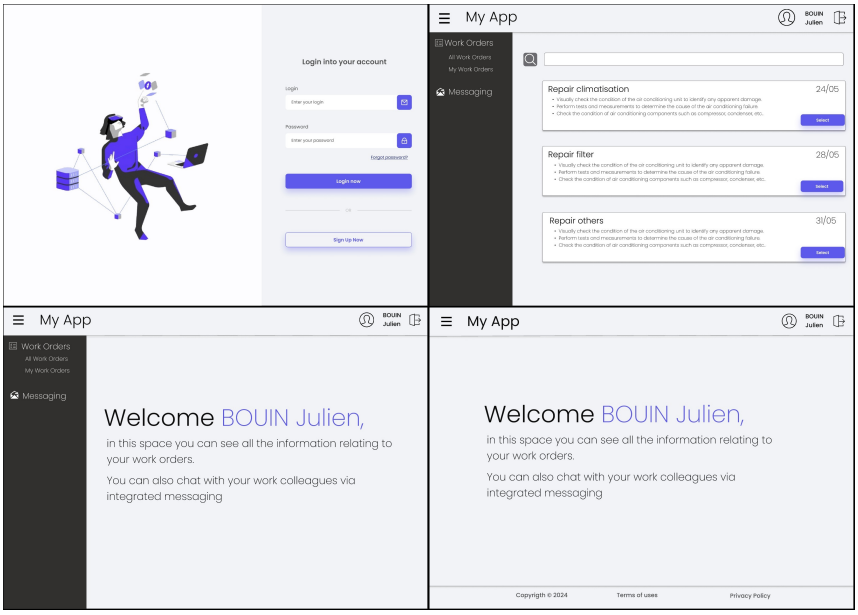

Figure 3.2 – Mockup

3.3 Implementation

3.3.1 Frontend Development Using Django Templates

When I started working on the application, I knew that the presentation and organization of the CSS code would be crucial for maintaining a clean and scalable design. Therefore, I learned and applied the BEM methodology (Block, Element, Modifier). This naming convention

for HTML and CSS classes allows the interface to be divided into independent blocks, with elements as child components and modifiers to indicate variations. This made the CSS more readable and easier to debug.

My first challenge was to create a simple yet effective login page. I used Django templates to structure the page, ensuring that the form fields were correctly connected to the backend views. By using BEM, I named the classes clearly, which facilitated CSS management.

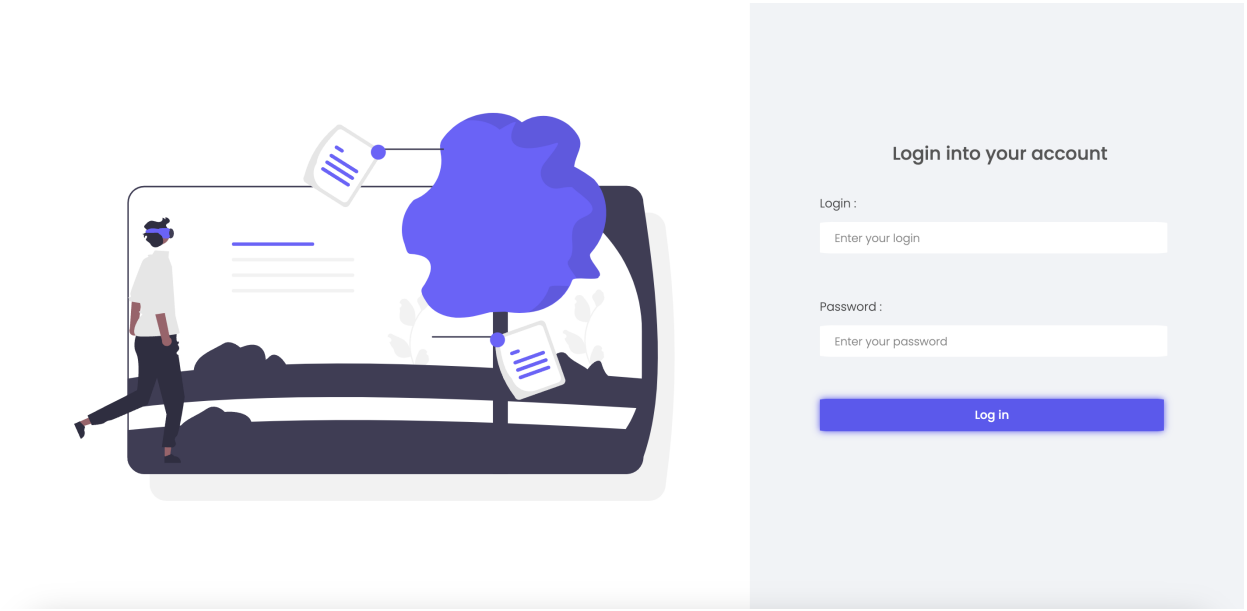

Figure 3.3 – Login Page

One of the main goals was to make the application responsive, meaning it had to function smoothly across various devices and screen sizes. To achieve this, I used a combination of CSS Grid and Flexbox to create flexible layouts. Media queries were essential for adjusting the design for mobile, tablet, and desktop views, thus ensuring a consistent user experience.

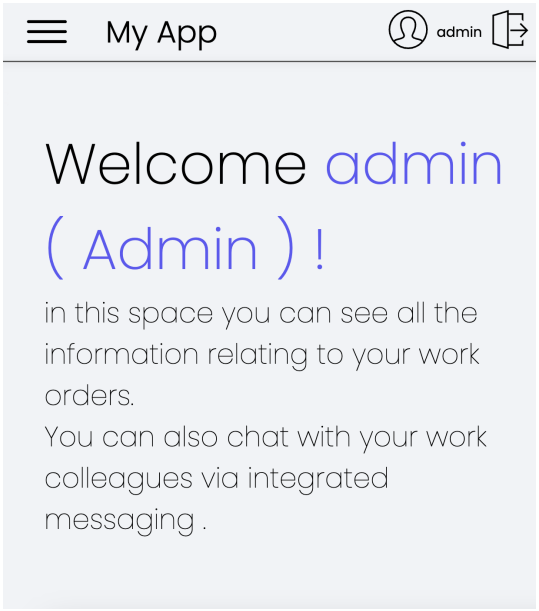

Figure 3.4 – Responsive Main Page

Next, I moved on to creating the main page of the application. I leveraged Django's template system to ensure consistent layout across different pages through template inheritance. By creating a base template containing common elements such as the header, footer, and navigation menu, I could reuse these components across multiple pages, thus reducing redundancy and simplifying maintenance.

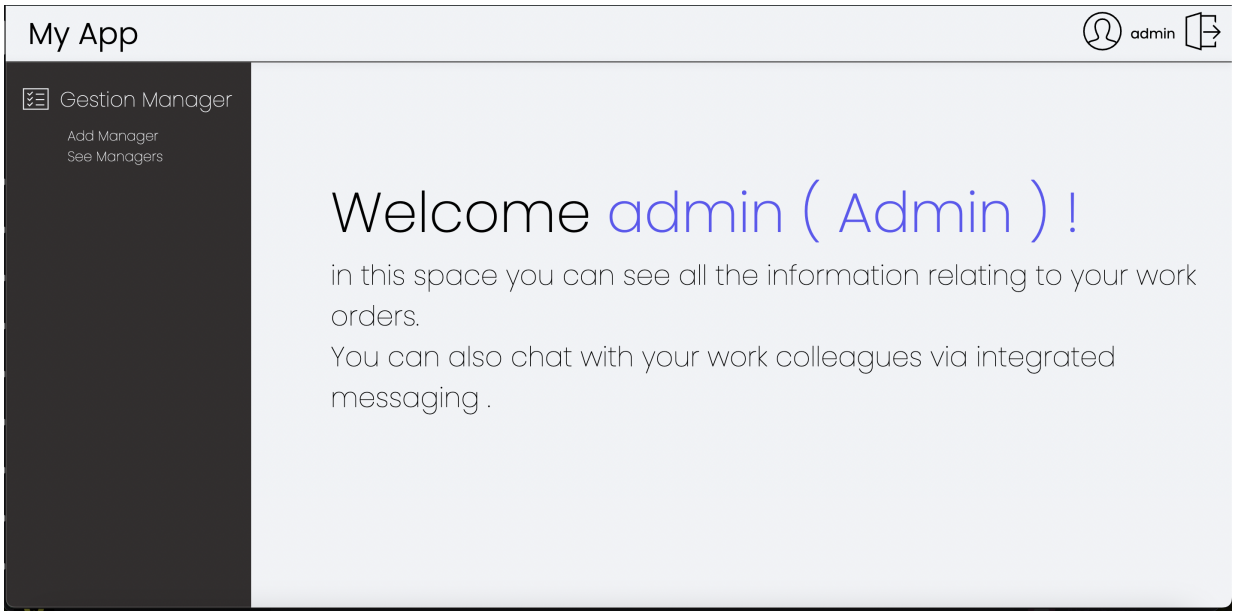

Figure 3.5 – Main Page

The next step was to ensure that the frontend displayed dynamic data retrieved from the backend. Using Django's template language, I integrated the data into the templates by passing context variables from the views. This included displaying lists of work orders, user information, and other dynamic content. Django's template tags and filters allowed me to manipulate and display the data within the templates.

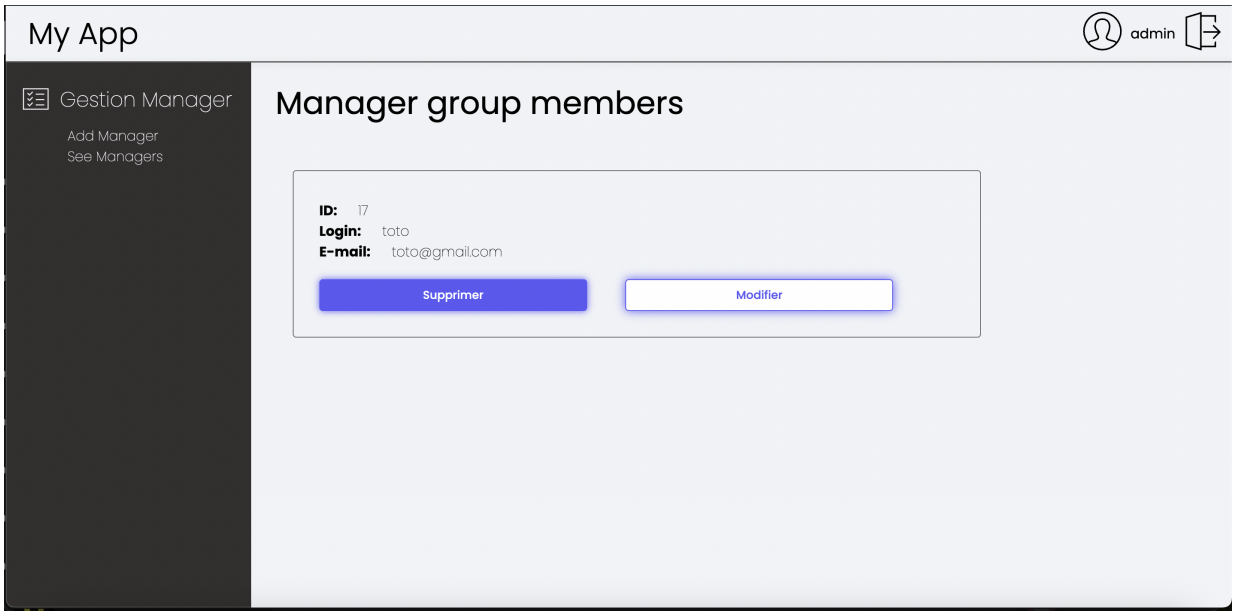

Figure 3.6 – List of Managers

3.3.2 API Development with Django REST Framework

The second part of my internship focused on creating a RESTful API using Django REST Framework (DRF). This API aimed to manage data exchanges between the frontend and backend, as well as facilitate integrations with other systems. This step was crucial for the continuation of my work.

Firstly, it was crucial to understand what a REST API is and how to develop one. A RESTful API (Representational State Transfer) is an architectural style for designing networked applications. It relies on a stateless client-server communication protocol, typically HTTP. RESTful APIs allow interacting with application data via standard HTTP methods such as GET, POST, PUT, and DELETE.

To set up this REST API using Django, I chose to use a Django framework called Django REST Framework, which allows for quicker and easier creation of REST APIs. I downloaded this framework.

Next, I created several API endpoints to handle different aspects of the application, such as user authentication and work order management. Each endpoint was defined in a dedicated view, using DRF's class-based views for better organization and scalability. These views were then associated with URL patterns in our Django app's urls.py file.

4. Source : `https://www.astera.com/fr/type/blog/rest-api-definition/`, Last accessed : 2024-05-21

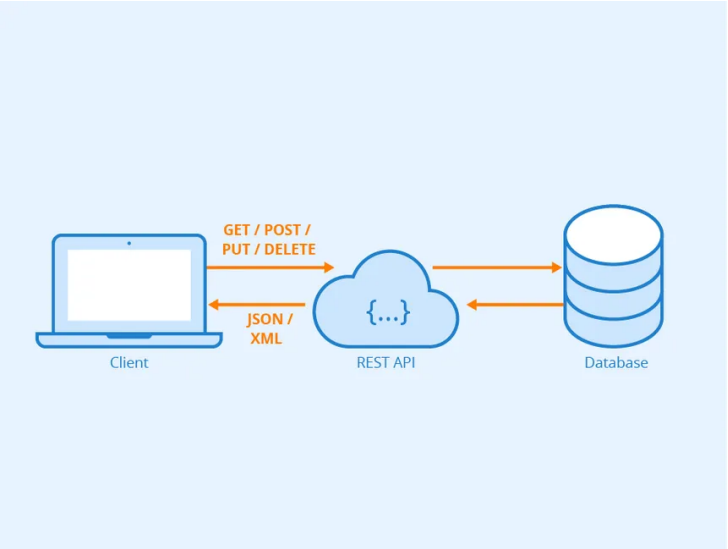

Figure 3.7 – API of a REST application[4]

For data serialization, I used DRF serializers to convert model instances into JSON data and vice versa. This was crucial for data exchange between the frontend and backend. ModelSerializers simplified the conversion of model instances into JSON, and in some cases, custom serializers were created to handle more complex data transformations or validations.

Finally, to secure access to the API, I implemented token-based authentication using DRF's built-in token authentication system. Several token systems are available via DRF, but I decided to use Token BASIC as it was the most suitable in this situation. Each user received a unique token after successful login, which had to be included in the headers of subsequent API requests. This ensured that only authenticated users could access protected endpoints.

In summary, this part of my internship allowed me to develop a comprehensive RESTful API via Django, from the initial configuration of DRF to the creation of secure endpoints and data serialization management. This strengthened my backend development skills using Django.

3.3.3 Test Cases

After finishing the development of our website, we performed test cases to summarize what we have done and what is possible to do in our application. Here, we focused only on the use cases that directly change the data of our website.

Test Scenarios - Function : `Take task`

Scenario 1 : Successful Task Takeover **Description :** Verify that a technician can successfully take a task with valid details.
Expected Result : The task is updated, added to the technician's list, and a success message is displayed.
Actual Result : The task is updated, added to the technician's list, and a success message is displayed.
Status : Passed

Scenario 2 : Task Takeover with Non-existent User **Description :** Verify that an error is handled when the user ID does not exist.
Expected Result : An error message "The user or task does not exist." is displayed.
Actual Result : An error message "The user or task does not exist." is displayed.
Status : Passed

Scenario 3 : Task Takeover with Non-existent Task **Description :** Verify that an error is handled when the task ID does not exist.
Expected Result : An error message "The user or task does not exist." is displayed.
Actual Result : An error message "The user or task does not exist." is displayed.
Status : Passed

Scenario 4 : GET Request to Take Task Endpoint **Description :** Verify that the take task endpoint does not allow GET requests.
Expected Result : The request is ignored or an appropriate response is given.
Actual Result : The request is ignored or an appropriate response is given.
Status : Passed

Scenario 5 : Authorization Check **Description :** Verify that only users with technician rights can access the take task functionality.
Expected Result : Access is denied, and the user is redirected or shown an error message.
Actual Result : Access is denied, and the user is redirected or shown an error message.
Status : Passed

Scenario 6 : Odoo Integration Error Handling **Description :** Verify that an error message is shown if there is an issue with Odoo integration.

Expected Result : An error message detailing the Odoo error is displayed.
Actual Result : An error message detailing the error

3.4 Summary

To summarize, I have designed a user-friendly interface for our web application, focusing on an intuitive and seamless user experience. This interface enables users to interact easily with the application's features, facilitating navigation and usage of various functionalities.

Additionally, I have successfully developed a robust RESTful API for our application, enabling efficient management of data exchanges between the frontend and backend. This API provides secure and well-defined access points for the different functionalities of the application, ensuring reliable and secure communication between the various components of the application.

Chapter 4

Mobile Application

4.1 Why a mobile application ?

Following the development of a website to address the aforementioned needs, my supervisor proposed creating a mobile application mirroring the website's functionalities. This would enable technicians to directly access it via their phones. Technicians can log in, view their work orders, select them, and add notes similar to the website. However, the significant advantage of this application is its offline functionality. Technicians can update their work orders even without an internet connection. They can add notes, change work order states, and upon reconnecting, all changes will synchronize with our application.

4.2 Choice of Language

Initially, with little knowledge about mobile applications, I conducted thorough research upon realizing the need to develop one. Two standout technologies emerged : React Native and Flutter.

React Native, an open-source framework by Facebook, and Flutter, by Google, support cross-platform development for both Android and iOS. Further research was conducted to determine the most suitable choice.

After evaluation, React Native emerged as the preferred option. Its utilization of JavaScript, a familiar language, was advantageous for me.

4.3 Use cases

For this application, the use cases remained consistent with our web application. The primary goal was to provide technicians with identical functionalities on their mobile devices as on the website.

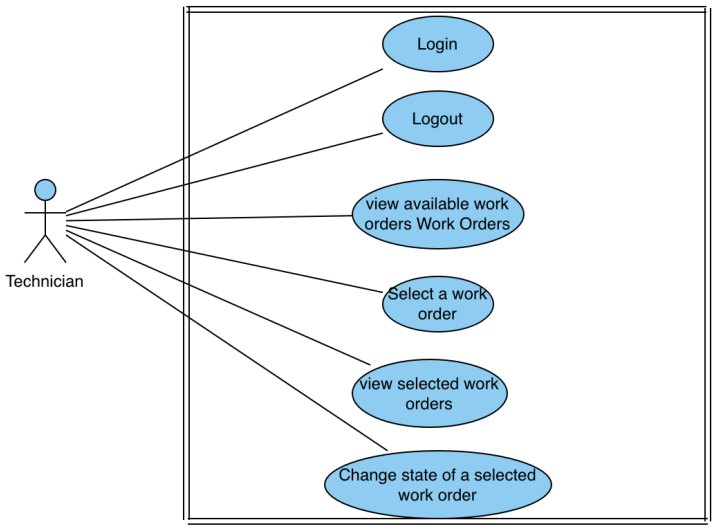

Figure 4.1 – Use Cases

4.4 Development with React Native

The development of the mobile application was an enriching journey, where I delved into React Native's intricacies while addressing project-specific challenges. Here's an account of my experiences throughout the development process :

4.4.1 Login Page

The first step in my mobile application was creating the login page. I started with the visual aspect of this page.

I focused on making this login page simple and aesthetically pleasing, adhering to the design guidelines of our web application. Next, I tackled the backend of my application, implementing the same authentication system as our web application. This allows users to log in with the same credentials on both the mobile app and the website. To achieve this, I used the REST

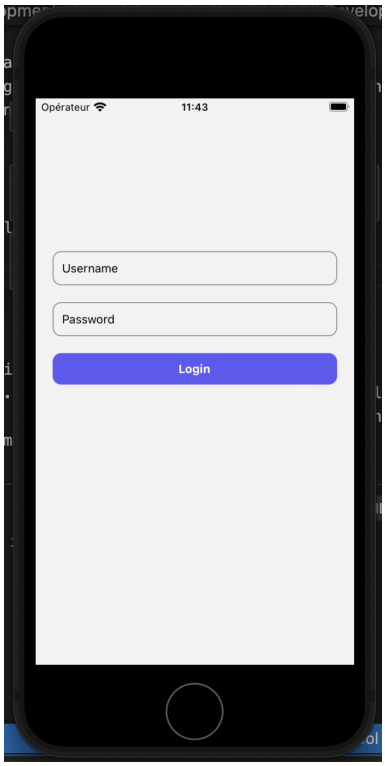

Figure 4.2 – Login page mobile application

API previously described, making a POST request to the authentication endpoint via the token system.

— **Frontend** : Utilized React Native to create a smooth and responsive user interface.

— **Backend** : Integrated the REST API to handle authentication, ensuring users can securely log in with their existing credentials.

4.4.2 Overall Structure of the Application

After the login page, I worked on the overall structure of the application, paying special attention to navigation.

When the user is logged in and navigates to the home screen, a horizontal bar at the top of the screen indicates their location within the application and includes a button to log out at any time. To implement this, I used a framework integrated with React Native and Expo called Expo Router, which greatly simplifies this functionality.

I then added a bottom navigation bar to allow users to switch pages easily at any time. I also used Expo Router for this navigation bar.

— **Navigation** : Utilized Expo Router for intuitive and responsive navigation.

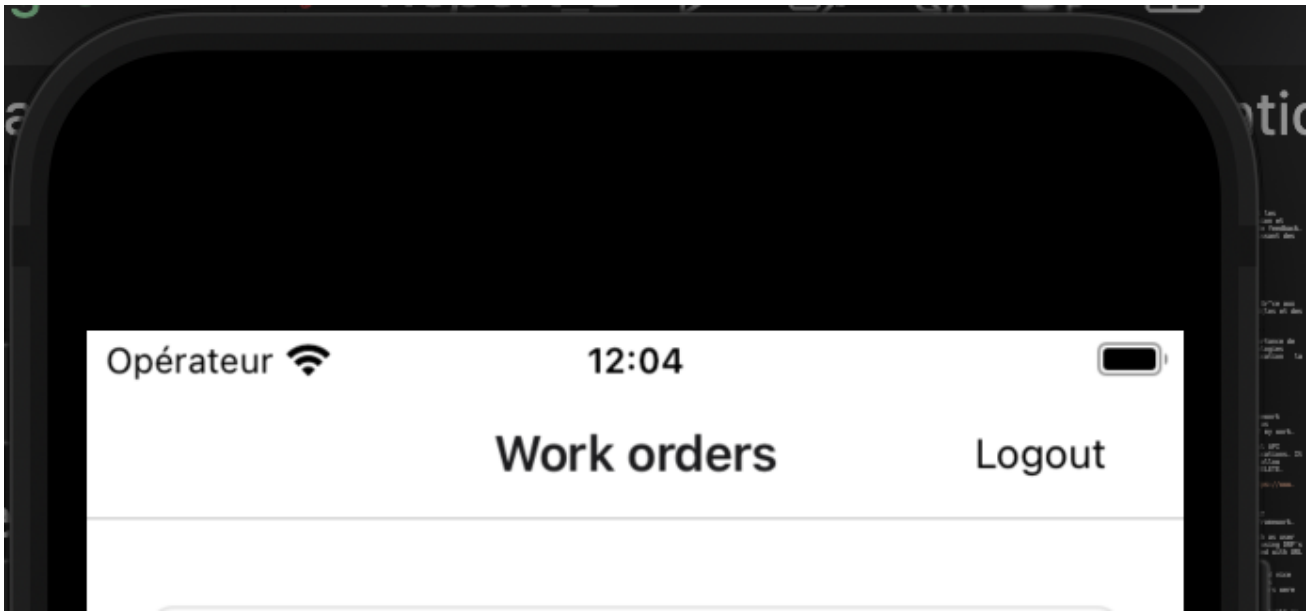

Figure 4.3 – Header navigation

— **UX** : User-centered design with clear visual indicators for location and easily accessible logout options.

4.4.3 Displaying and Processing Data

After completing these crucial steps, I focused on the main functionality : displaying data in the application via the REST API.

I started by displaying all available work orders, adding a button for technicians to select them. For this, I used a key component of React Native, FlatList, which optimizes the display of a list of data—in this case, the work orders returned by the API.

I combined this with a custom component that displays the main information of a work order and includes a button to select it. This component allows for quick and intuitive interaction for technicians.

— **FlatList** : Used FlatList for efficient and optimized rendering of work orders.

— **Custom Component** : Created a component to display work order details with facilitated user interaction.

I repeated the same process for the page listing all work orders selected by the technician, this time removing the selection button.

The main objective was to enable the modification of a work order's status. I added the ability for technicians to click on a work order to display additional information. Four buttons

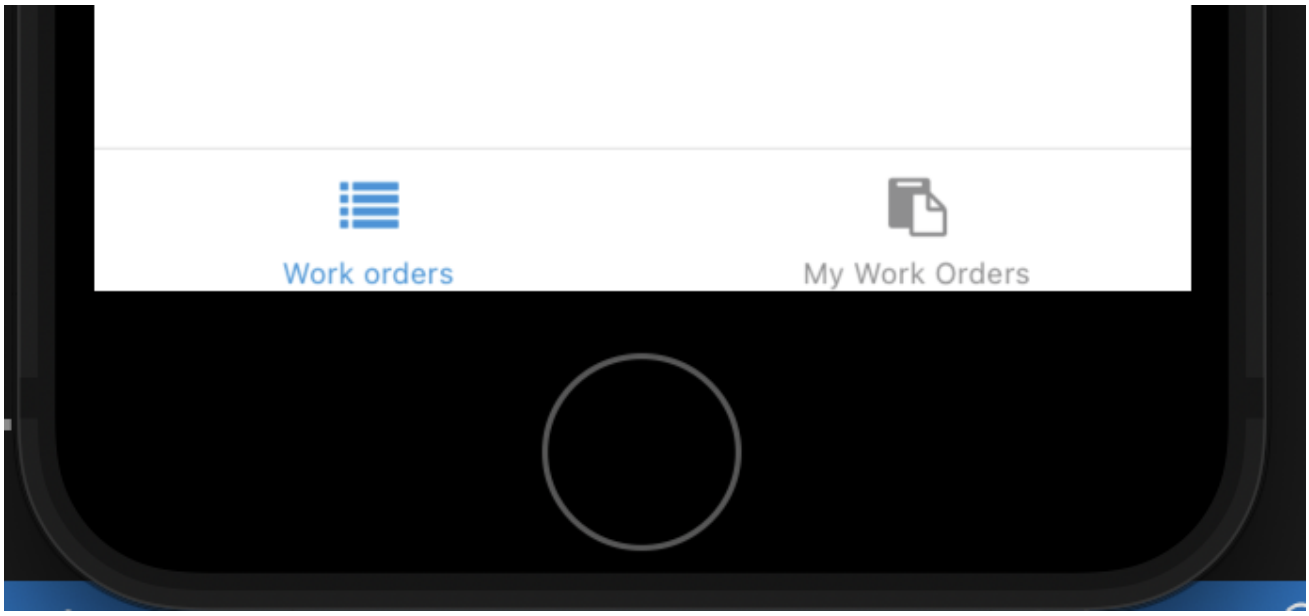

Figure 4.4 – Footer navigation

then allow the technician to change the status of the selected work order.

‘

— **Status Management** : Added buttons to modify the status of work orders directly from the app.
— **Additional Details** : Displayed detailed information for each work order for effective management by technicians.

4.4.4 Offline Data Management

At this point, I had a functional application, but it did not yet meet the primary requirement : allowing its use without an Internet connection.

To meet this requirement, I implemented an offline data management system using AsyncStorage, an essential feature of React Native.

How AsyncStorage Works AsyncStorage allows for persistent key-value pair storage on the user's device. Here is how I used it to manage offline data :

1. **Local Data Storage** : When a user interacts with the application without an Internet connection, data is temporarily stored locally using AsyncStorage. This includes actions such as creating, modifying, or deleting work orders.

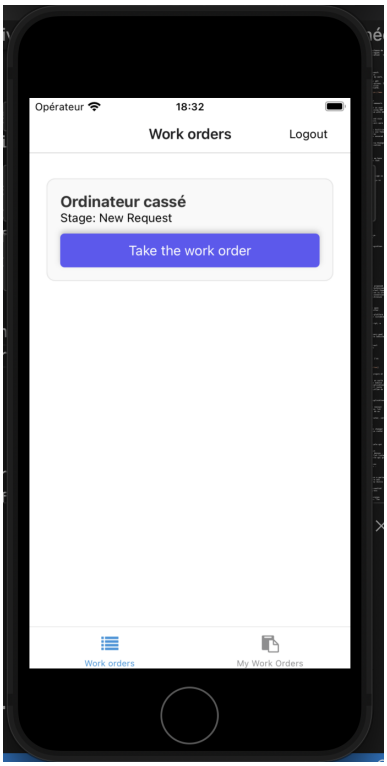

Figure 4.5 – Page "Work Orders"

2. **Automatic Synchronization** : Once the connection is restored, the application detects the presence of Internet and triggers a synchronization process. All locally stored modifications are sent to the REST API to update the central database.
3. **User Interface Update** : During offline use, the user interface remains smooth and responsive, using local data to display the necessary information. As soon as synchronization is complete, the interface updates with the latest data.

Benefits of This Approach

— **Resilience** : Users can continue to use the application even without an Internet connection, which is crucial for technicians working in areas without network coverage.

— **Improved User Experience** : The transition between offline mode and synchronization is seamless, providing a smooth user experience.

— **Data Integrity** : Synchronization ensures that all offline-modified data is correctly updated on the server as soon as possible, preventing any loss of information.

Figure 4.6 – Page "My Work Orders"

4.5 Test cases

After finishing the development of our website, we performed test cases to summarize what we have done and what is possible to do in our application.

Login : **Description :** Verify that a technician can successfully login
Expected Result : The technician was login .
Actual Result : The technician is login
Status : Passed

Logout : **Description :** Verify that a technician can successfully logout **Expected Result :** The technician logged out successfully. **Actual Result :** The technician is logged out successfully. **Status :** Passed

View Available Work Order : **Description :** Verify that a technician can view available work orders **Expected Result :** The technician can see the list of available work orders. **Actual Result :** The technician sees the list of available work orders. **Status :** Passed

Figure 4.7 – Page Details

Select a Work Order : Description : Verify that a technician can select a work order **Expected Result :** The technician successfully selects a work order. **Actual Result :** The technician has selected a work order. **Status :** Passed

View Selected Work Order : Description : Verify that a technician can view the details of a selected work order **Expected Result :** The technician can view the details of the selected work order. **Actual Result :** The technician views the details of the selected work order. **Status :** Passed

Change the State of a Selected Work Order : Description : Verify that a technician can change the state of a selected work order **Expected Result :** The technician successfully changes the state of the selected work order. **Actual Result :** The technician has changed the state of the selected work order. **Status :** Passed

4.6 Summary

In summary, using React Native, I developed the mobile application, ensuring seamless integration with our existing backend through a robust RESTful API. The application's structure and navigation were carefully designed to provide a smooth user experience, featuring clear visual indicators and intuitive controls.

Data management was a priority, with the application effectively displaying and processing work order data using components such as FlatList. Additionally, I implemented an offline feature using AsyncStorage, allowing technicians to continue using the application even without an internet connection.

Conclusion

My internship at Instituto Superior de Engenharia de Coimbra (ISEC) was a rewarding experience filled with discovery and learning. Working on the development of a full-stack website and mobile applications integrating CMMS and augmented reality technologies allowed me to enhance my skills in both frontend and backend development.

Collaborating with my colleague Rayane Belguebli and our supervisor, Mateus Mendes, was instrumental in the success of this project. Together, we overcame technical challenges and made significant progress towards achieving our goals.

During this internship, I successfully developed a user-friendly website that meets the needs of our target users. The application also works seamlessly, providing an intuitive interface for maintenance technicians to efficiently manage their tasks.

Additionally, I gained a deeper understanding of the importance of synergies between different technologies and their impact on the effectiveness of maintenance processes. The integration of CMMS and augmented reality offers promising prospects for the future of maintenance operations management.

In conclusion, this experience not only allowed me to acquire new technical skills but also provided valuable insights into the challenges and opportunities in the field of software development applied to industrial maintenance. I am grateful for this opportunity, and I am confident that the knowledge gained during this internship will be invaluable for my future professional journey.

Bibliography

[1] Malta Ana, Mendes Mateus, and Farinha Torres. Augmented reality maintenance assistant using yolov5 [j]. *Applied Sciences*, 11(11) :4758–4758, 2021.

[2] Django Software Foundation. Django official website. Online, 2024. Retrieved from `https://www.djangoproject.com/`, last accessed on 2024-04-26.

[3] Daryl Mather. *CMMS : A Timesaving Implementation Process*. CRC Press, 2002.

[4] Steven Henderson and Steven Feiner. Exploring the benefits of augmented reality documentation for maintenance and repair. *IEEE Transactions on Visualization and Computer Graphics*, 17(10) :1355–1368, 2010.

[5] Michele Fiorentino, Antonio E Uva, Michele Gattullo, Saverio Debernardis, and Giuseppe Monno. Augmented reality on large screen for interactive maintenance instructions. *Computers in Industry*, 65(2) :270–278, 2014.