Instituto Politécnico de Viana do Castelo

ipvc estg

Degree in Informatics Engineering

Project III Curricular Unit

GESFaturacao

Alexandre Santos, n.º 24585

António Gomes, n.º 26780

2023 - 2024

Supervised by: Prof. Doutor. Ricardo Freitas | rifreitas@estg.ipvc.pt

Supervised by: Eng. Miguel Guerra | geral@ftkode.com

Index

Illustration List

Introduction and objectives

This project was developed associated with the curricular unit of Project III of the 3rd year of the Graduation Degree in Informatics' Engineering (Computer Science) at the School of Technology and Management of the Polytechnique institute of Viana do Castelo.

The **main objective** of this project is the development of a mobile application using the React Native framework based in Javascript and utilizing Node.JS.

The **scope of this project** is the development of a mobile application at IPVC and was proposed by Professor Ricardo Freitas and the company FTKode, lda.

We built a business management application with an authentication system, where the user can log-in.

After logging into the application, the user can manage 4 categories, invoices, budgets, clients and products. In the management process, the user can create, list, edit, remove and send documents via e-mail.

In order to make the development of this project possible, our **methodology** was to search for the most amount of information on various platforms to learn the new technologies like React Native, the usage of APIs and Javascript. To make it possible we searched for similar projects on Github, watched tutorials and step-by-step guides on YouTube about the subject and used a cloud-based artificial intelligence tool developed by GitHub and OpenAI called Github Copilot capable of assisting in the development of the project.

In the following report is possible to find an introduction to this project, the explanation of the project architecture and information about the flow we used to develop the project since the beginning of the semester. After that, we have the section which has a description of the main technologies, tools, and libraries used. In the next title we show a sequence diagram representing the flow of the code.

2. Technologies, tools, libraries, methodology and project management

2.1 Programming Environment

The environment used for the development of the project consisted in a collection of technologies (**React Native**, **Node.JS** and **Rest API**) associated with JavaScript, which is used to build web-based applications. This environment is responsible for the development of each component of the APP development from client-side/server-side.

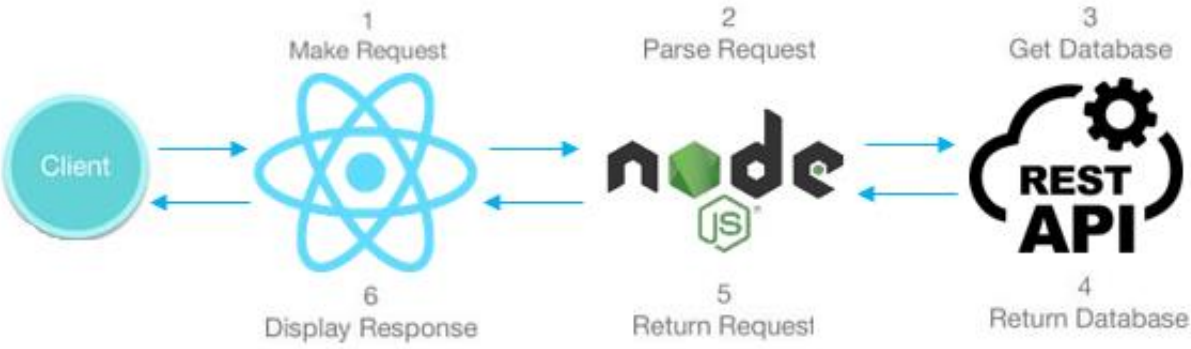


Figure 1 - Illustration of the development environment.

Used Technologies:

- Rest API: API developed by the company to make all the requests available.
- React Native: It is an open-source UI software framework created by Meta Platforms, Inc.
- Node: It is used to handle the server-side logic.

2.2 Methodology and Project Management

In this project we used Skype to communicate with the supervising engineer, it also was used to set deadlines and keep track of what needed to be done. We held weekly meetings with the engineer to maintain some consistency in the developed functionalities.

The communication between the work group was done via Discord, with this tool we made calls every time we would make changes to the project together. We also got together in the school library to work on the project every week.

To work together in this project, we used Github to share the code between us.

2.3 Technologies and Architecture

The architecture of the project is composed by a set of frameworks and technologies. For the project modelling and development, we used React Native, Skype and Discord for communication and Git for the project versioning.

2.3.1 Javascript

JavaScript is a versatile programming language primarily used for adding interactivity to web pages. It allows developers to create dynamic content, manipulate HTML and CSS, and respond to user actions in the browser. JavaScript is widely supported by all modern web browsers and is essential for web development.

Some key features of JavaScript are:

- Client-Side Scripting: JavaScript runs on the client-side (i.e., in the user's web browser), enabling dynamic interactions without requiring server interaction for every action.

- Event-Driven Programming: JavaScript is event-driven, meaning it can respond to user actions such as clicks, mouse movements, and keyboard inputs.

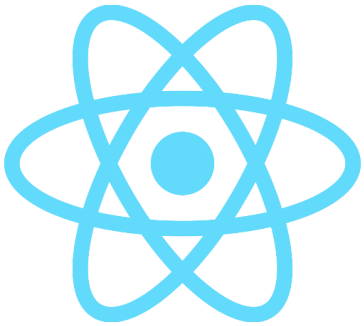
- DOM Manipulation: It allows manipulation of the Document Object Model (DOM), enabling developers to dynamically change the structure, content, and style of web pages.

- Asynchronous Programming: JavaScript supports asynchronous programming through features like callbacks, promises, and async/await, allowing for non-blocking code execution.

- Cross-platform Compatibility: JavaScript is supported by all major web browsers and can also be used for server-side development (Node.js), desktop applications (Electron), and mobile app development (React Native, NativeScript).

- W3 Schools: https://www.w3schools.com/js/
- Javascript: https://www.javascript.com/

2.3.2 React Native



React Native is a popular framework for building mobile applications using JavaScript and React. It allows developers to create cross-platform apps that run natively on both iOS and Android devices, sharing a significant portion of the codebase between platforms.

Key features of React Native:

- Reusable Components: React Native allows developers to build reusable UI components using JavaScript and React, which can be shared between different parts of the application and across platforms.
- Native Performance: React Native apps leverage the native components and APIs of iOS and Android platforms, providing a smooth and responsive user experience comparable to that of native apps.

- Hot Reloading: Developers can see the changes they make to the code instantly reflected in the running app, speeding up the development process and enabling rapid iteration.
- Third-Party Libraries: React Native has a vibrant ecosystem of third-party libraries and tools that extend its capabilities, allowing developers to integrate features like navigation, animations, and data management into their apps easily.
- React Ecosystem Integration: React Native is built on top of React, so developers familiar with React can quickly get started with mobile app development using React Native.
- React Native: https://reactnative.dev/
- Meta: https://github.com/facebook/react-native

2.3.3 Visual Studio Code

Visual Studio Code (VS Code) is a popular open-source code editor developed by Microsoft. It is widely used by developers for various programming tasks, including web development, mobile app development, and more.

Some key features of Visual Studio Code:

- Cross-Platform: Visual Studio Code is available for Windows, macOS, and Linux, making it accessible to a wide range of developers regardless of their operating system.
- Extensible: VS Code has a rich ecosystem of extensions that enhance its functionality. These extensions cover a wide range of functionalities such as language support, debugging, source control integration, and more.
- Integrated Terminal: It comes with an integrated terminal that allows developers to run commands, scripts, and terminal-based tools directly within the editor, eliminating the need to switch between different applications.

- Debugging Support: VS Code offers built-in support for debugging various programming languages and frameworks. Developers can set breakpoints, inspect variables, and step through code to diagnose and fix issues efficiently.

2.3.4 GitHub

GitHub is a web-based platform that serves as a hub for software development collaboration, version control, and project management. It utilizes Git, a distributed version control system, to track changes in codebases, manage branches, and facilitate collaboration among developers.

2.3.5 NodeJS

The Node.js run-time environment includes everything needed to execute a program written in JavaScript. Node.js came into existence when the original developers of JavaScript extended it from something you could only run in the browser to something you could run on your machine as a standalone application. Now you can do much more with JavaScript than just making websites interactive. JavaScript now has the capability to do things that other scripting languages like Python can do. Both your browser JavaScript and Node.js run on the V8 JavaScript runtime engine. This engine takes your JavaScript code and converts it into a faster machine code. Machine code is low-level code which the computer can run without needing to first interpret it.

2.3.6 Discord

Discord is a popular communication platform designed primarily for gamers, although it is used by a wide range of communities and organizations for various purposes. It allows users to communicate with each other via text, voice, and video chat in real-time.

Some key features and aspects of Discord:

- Text Channels: Discord allows users to create and join text channels organized into servers (also known as guilds or communities). Text channels can be used for discussions, announcements, sharing links, and more.
- Voice Channels: Users can also create and join voice channels within servers, enabling real-time voice communication like a conference call. Voice channels support features like push-to-talk, voice activity detection, and customizable voice settings.
- Video Calls: Discord supports video calls, allowing users to engage in face-to-face conversations within servers or direct messages. Video calls can be initiated from text or voice channels.
- Roles and Permissions: Discord servers can have roles assigned to users, granting them specific permissions within the server. Roles can be used to manage access to channels, moderate discussions, and distinguish users based on their roles or privileges.
- Bots and Integrations: Discord offers a robust API that allows developers to create bots and integrate Discord with other services and platforms. Bots can automate tasks, provide information, and enhance the functionality of Discord servers.

2.3.7 Skype

Skype is a telecommunications application that provides video chat, voice call, and instant messaging services. It was one of the earliest platforms to offer these features and has remained popular for personal and business communication.

Some key features and aspects of Skype:

- Video Calls: Skype allows users to make video calls with one or multiple participants. Video calls support features like screen sharing, background blur, and virtual backgrounds.
- Voice Calls: Users can make voice calls to other Skype users or mobile/landline numbers using Skype credits or subscriptions. Skype offers high-quality voice calling over the internet.
- Instant Messaging: Skype supports instant messaging, allowing users to send text messages, emojis, files, and multimedia content to their contacts or groups.
- Group Chats: Users can create group chats with multiple participants, making it easy to communicate with friends, family, or colleagues.
- Contacts and Address Book Integration: Skype integrates with users' address books, making it easy to find and connect with contacts. Users can import contacts from other services or add them manually.

3. Project Sequence

3.1 Sequence of actions

The sequence of actions when a request is initiated begins with a client selecting a category from the application menu, such as creating an invoice. Upon selection, the user proceeds to populate the requisite form with necessary inputs. Once the form is completed, the user confirms the submission. At this juncture, a request is triggered, initiating the parsing process within Node.js.

Upon parsing, the request is then transmitted to the designated API endpoint. If the request is successfully transmitted and processed by the API, it returns a response devoid of errors. Conversely, if any discrepancies or issues arise during processing, the API will return an error response, signalling the need for further investigation or corrective action.

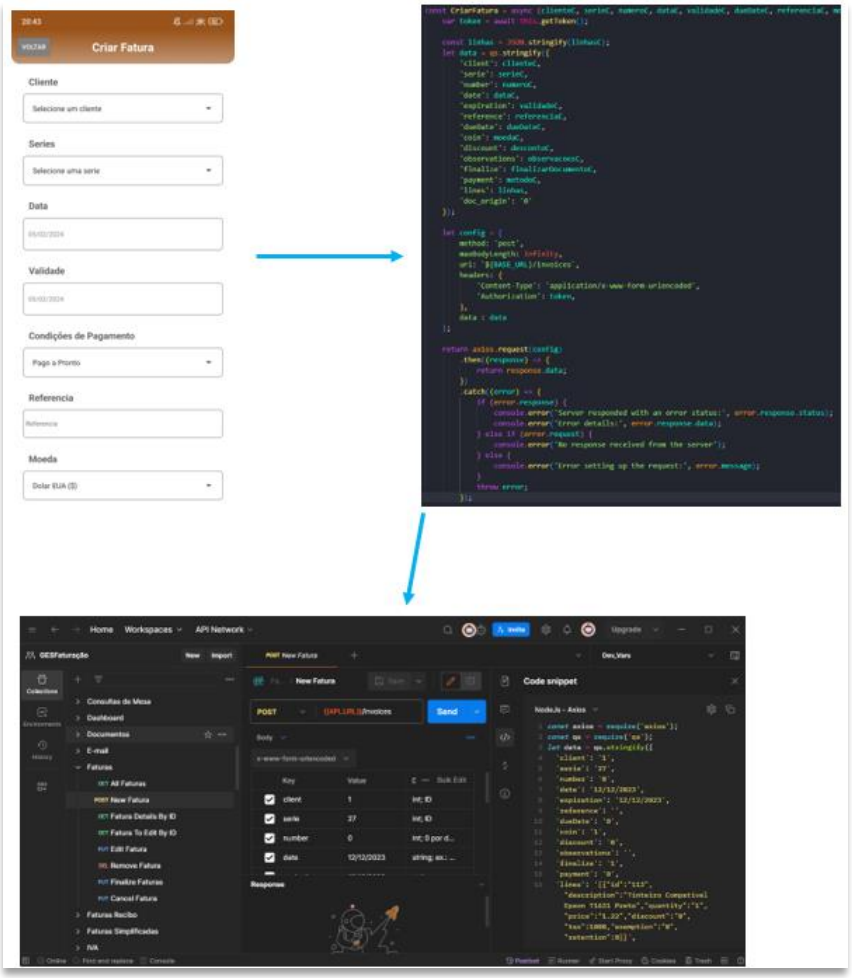


Figure 2 - Example of the sequence of actions.

4. Developed features

At the start of the project, the supervising engineer outlined a set of features for our development roadmap. Among these features were the core functionalities of creating, editing, listing, and removing invoices, budgets, and products. These key operations formed the foundation of our project objectives, serving as the primary focus areas for our development efforts.

Figure 3 - Illustration of the Authentication methods.

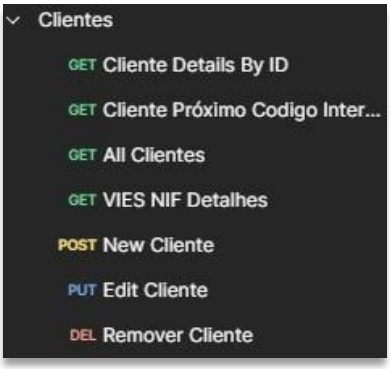


Figure 4 - Illustration of the client management methods.

Figure 5 - Illustration of the invoice management methods.

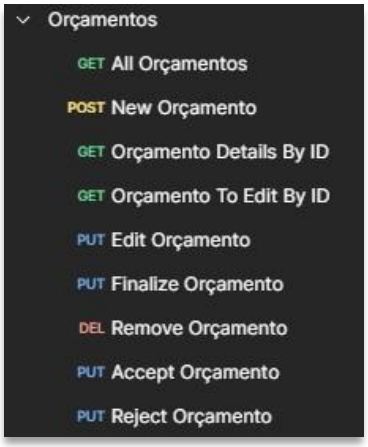


Figure 6 - Illustration of the budget management methods.

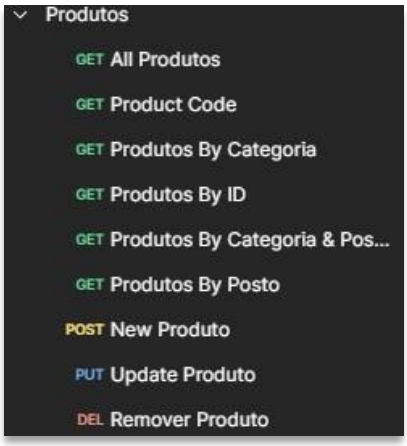


Figure 7 - Illustration of the article management methods.

All the features we developed, depend entirely on the companies API. Using postman, we were able to get the code necessary to the implementation of those features.

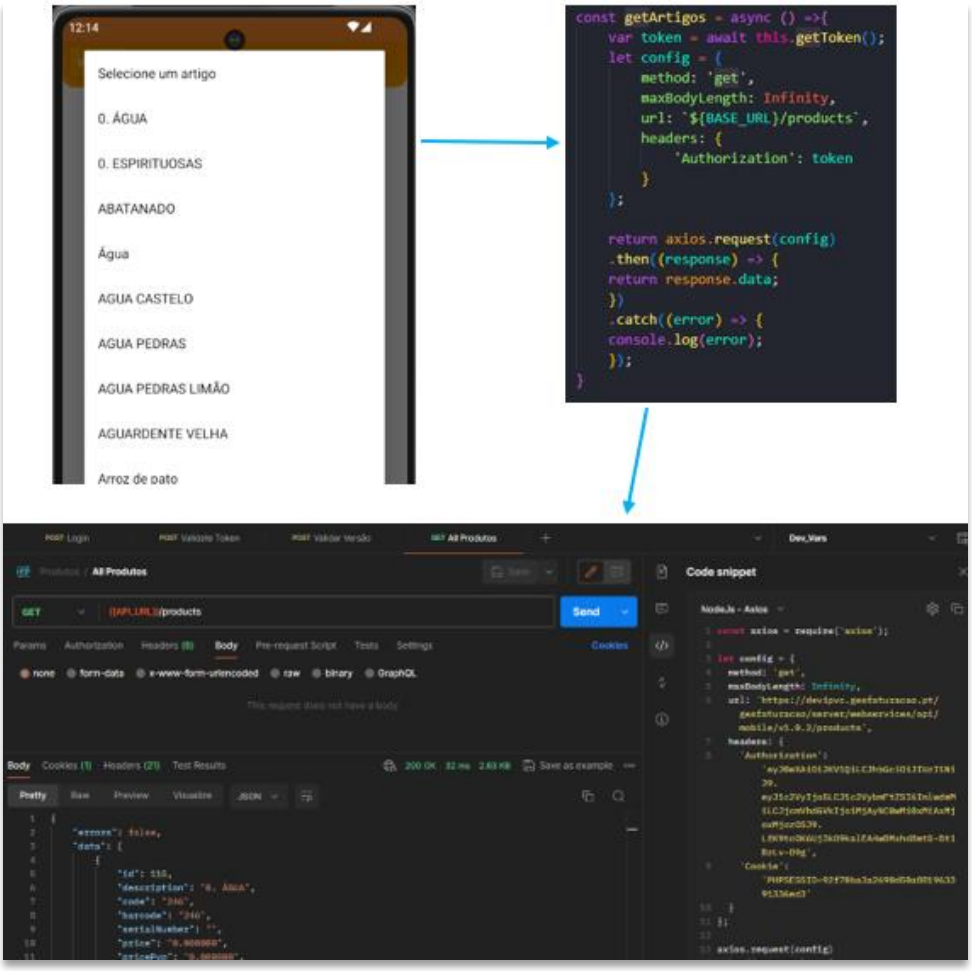


Figure 8 - API GET request example.

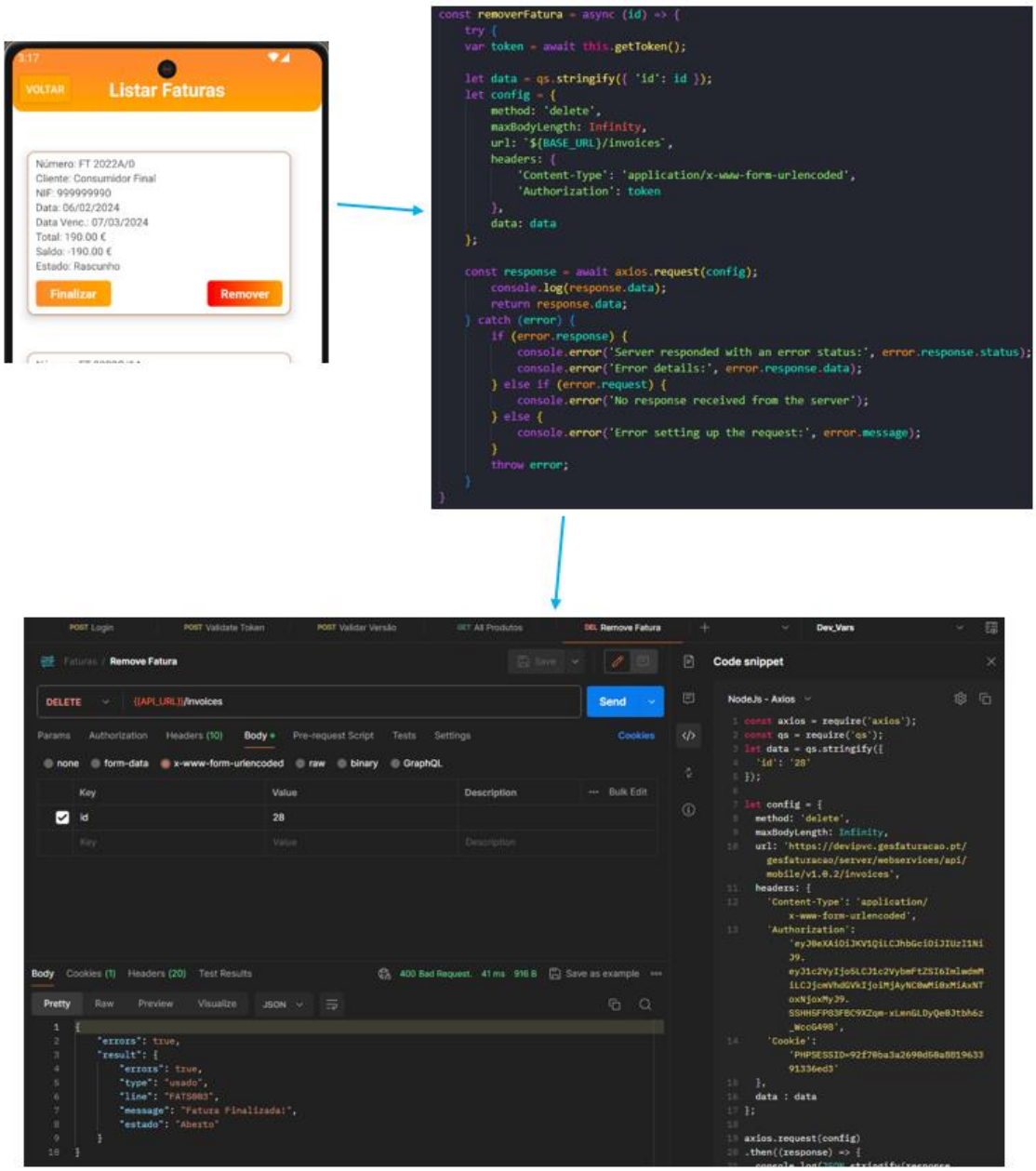


Figure 9 - API DELETE request example.

5. Practical Case/Project Developed

In our project, we initiated development by creating interface prototypes using Figma, providing a visual blueprint for the application's design. We then proceeded to implement a range of features, including user authentication, a dynamic dashboard interface, CRUD operations for data management, API integration, and UI enhancements.

Figure 10 - UI prototypes developed in figma.

Following the creation of illustrative examples, our focus shifted towards implementing the Login functionality, which necessitated the integration of authentication logic. This pivotal step marks the commencement of user authentication within the application, ensuring secure access to authorized users while safeguarding sensitive information.

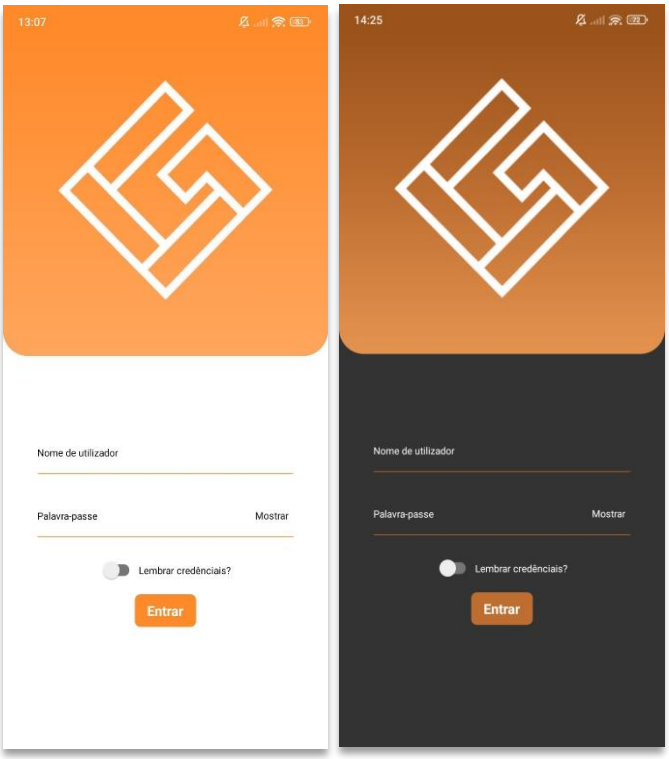


Figure 11 - Login screen (Light/Dark mode).

Due to constraints in time and familiarity with the tools, we opted to implement a category-based menu on the dashboard instead of tables representing data from invoices and budgets as originally planned. This decision was made to streamline development and prioritize functionality within the given timeframe. While tables would have provided a more detailed and structured display of data, the category-based menu offers a simplified navigation approach, allowing users to access specific categories efficiently.

Figure 12 - Dashboard/Menu screen.

In our project, we utilized React Native's "react-native" library components to create uniform and intuitive forms for both creation and editing pages. This approach ensured consistency in design and functionality, streamlining development efforts and enhancing user experience.

In the creation the user only needs to fill out required fields to create a document or article. In the Details page, if the selected article or document allows editing, the user can click the “Editar” button to make the fields editable, after editing, at the bottom the page the user only needs to click the “Confirmar” button to confirm the changes.

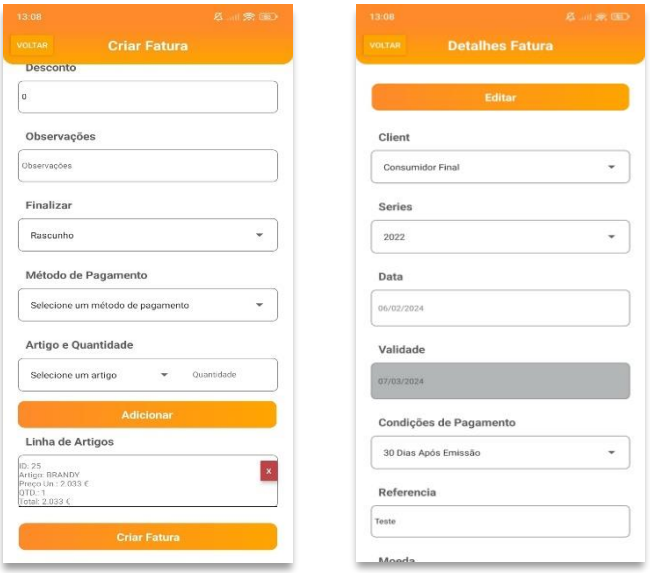


Figure 13 – Creation and Editing/Details screen.

To facilitate user access to all data items, we implemented the API GET requests. These requests enabled seamless retrieval of information, empowering users with complete visibility and accessibility to each item within the application. This approach ensured that users could effortlessly browse and interact with the entirety of the available data, enhancing their overall experience and productivity within the system.

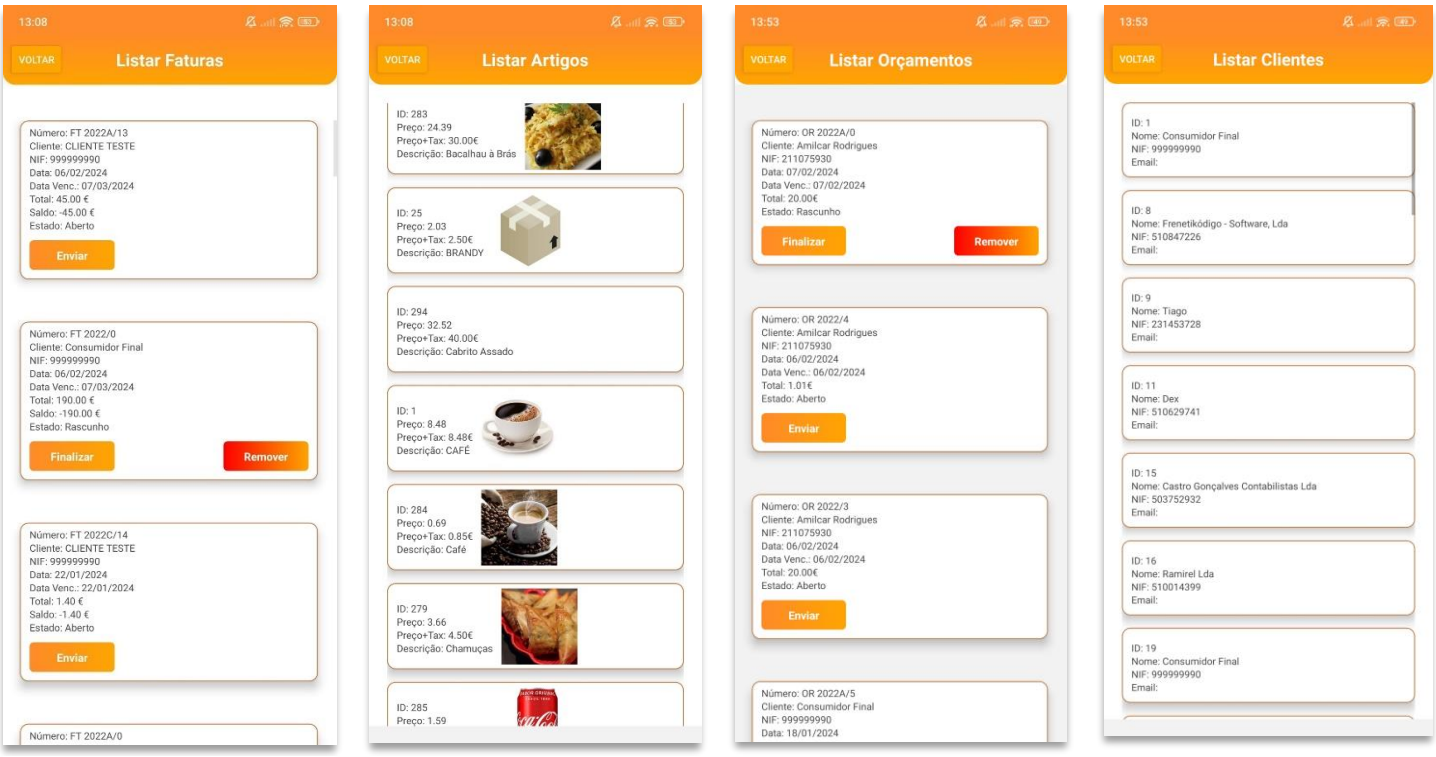


Figure 14 – Listing screens.

6. Difficulties & future features

6.1 Difficulties

Navigating the initial encounter with new technology posed a formidable challenge, requiring us to grasp its underlying mechanics and functionalities. The pressure to meet deadlines in the early stages was exacerbated by the learning curve associated with this unfamiliar technology, necessitating concerted efforts to adapt and progress efficiently.

Another significant hurdle we encountered was in the implementation of API access functions, where intricacies in integration and communication protocols demanded meticulous attention and troubleshooting. Furthermore, transferring user-inputted data to the server proved to be a complex endeavour, primarily due to discrepancies in field naming conventions and data types between client and server environments.

Despite these obstacles, our perseverance and collaborative spirit enabled us to navigate through these difficulties, leveraging each setback as an opportunity for growth and refinement in our project development process.

6.2 Future Features

In considering potential enhancements, integrating tables for data representation directly onto the dashboard, instead of buttons, emerges as a promising avenue. Additionally, implementing tabs to facilitate streamlined access to various functionalities could further optimize user experience and navigation within the system. These features hold the potential to enhance usability and efficiency, laying the groundwork for a more intuitive and robust platform.

6.3 Final thoughts

In reflection, while grappling with the unfamiliarity of the tools we employed presented its fair share of challenges, the inherent intrigue and significance of the project fuelled our collective determination. This motivation spurred us to push past obstacles and strive for excellence, ultimately culminating in the realization of our best possible outcome.

7. Conclusions and Future Work

In summary, this project holds significant importance not only within our professional development in the competitive job market. Engaging in this project has provided us with valuable exposure to corporate development environments, equipping us with practical skills and insights.

Moving forward, there are opportunities for further research into industry landscapes, refining our methodologies, fostering collaboration, and embracing emerging technologies. By staying proactive and committed to continuous improvement, we can ensure ongoing success and relevance in our endeavours.

8. Web References

- Course to learn Javascript: https://www.codecademy.com/learn/introduction-to-javascript l
- Course to learn the basics of React Native: https://www.codecademy.com/learn/learn-react-native
- React Native documentation: https://reactnative.dev/
- NPM documentation: https://www.npmjs.com/
- React Native navigation documentation: https://reactnavigation.org/
- Axios documentation to execute API calls: https://axios-http.com/docs/intro
- React native development methodology: https://markovate.com/blog/react-native-development/
- Figma tool for the UI prototypes: https://www.figma.com/

9. Appendix

9.1 Examples of Code

- Login and logout methods.

```
const login = async (username, password) => {
    setIsLoading(true);
    try {
        const res = await axios.post(`${BASE_URL}/authentication`, { username, password });
        let userInfo = res.data;
        setUserInfo(userInfo);
        setUserToken(userInfo._token);
        setNome(username);
        await AsyncStorage.setItem('@userInfo', JSON.stringify(userInfo));
        await AsyncStorage.setItem('@userToken', userInfo._token);


        ToastAndroid.show("Bem-vindo, " + username, ToastAndroid.SHORT);
    } catch(e) {
        ToastAndroid.show('Nome de utilizador ou palavra-passe incorretos.', ToastAndroid.SHORT);
    } finally {
        setIsLoading(false);
    }
    setIsLoggedIn(true);
}


const logout = async () => {
    setIsLoading(true);
    setUserToken(null);
    await AsyncStorage.removeItem('@userInfo');
    await AsyncStorage.removeItem('@userToken');
    setIsLoading(false);
    setIsLoggedIn(false);
}
```

- GET request example.

```
const getArtigos = async () => {
    var token = await this.getToken();
    let config = {
        method: 'get',
        maxBodyLength: Infinity,
        url: `${BASE_URL}/products`,
        headers: {
            'Authorization': token
        }
    };

    return axios.request(config)
    .then((response) => {
    return response.data;
    })
    .catch((error) => {
    console.log(error);
    });
}
```

- POST request example.

```
    const CriarArtigo = async (code, name, type, unit, qtdStock, qtdStockMin, stockMin, pvp, precoUnit, precoIni, iva, category) => {
    var token = await this.getToken();
    let data = qs.stringify({
      'code': code,
      'name': name,
      'category': category,
      'type': type,
      'stock': qtdStock,
      'minStock': qtdStockMin,
      'stockAlert': stockMin,
      'unity': unit,
      'pvp': pvp,
      'tax': iva,
      'price': precoUnit,
      'initialPrice': precoIni
    });
    let config = {
      method: 'post',
      maxBodyLength: Infinity,
      url: `${BASE_URL}/products`,
      headers: {
        'Content-Type': 'application/x-www-form-urlencoded',
        'Authorization': token,
      },
      data : data
    };
    return axios.request(config)
      .then((response) => {
        console.log(JSON.stringify(response.data));
      })
      .catch((error) => {
        console.log(error + ' Erro Faturas');
      });
  }
```

- PUT request example.

```
const EditarFatura = async (id, clienteC, serieC, numeroC, dataC, validadeC, dueDateC, referenciaC, moedaC, descontoC, observacoesC, metodoC, linhasC, finalizarDocumentoC) => {
    var token = await this.getToken();
    const linhas = JSON.stringify(linhasC);
    let data = qs.stringify({
      'id': id, 'client': clienteC, 'serie': serieC, 'number': numeroC, 'date': dataC, 'expiration': validadeC 'reference': referenciaC, 'dueDate': dueDateC, 'coin': moedaC, 'discount': descontoC, 'observations': observacoesC, 'finalize': finalizarDocumentoC, 'lines': linhas, 'payment': metodoC,
    });
    let config = {
      method: 'put',
      maxBodyLength: Infinity,
      url: `${BASE_URL}/invoices`,
      headers: {
        'Content-Type': 'application/x-www-form-urlencoded',
        'Authorization': token,
      },
      data : data
    };
    return axios.request(config)
      .then((response) => {
        return response.data;
      })
      .catch((error) => {
        if (error.response) {
          console.error('Server responded with an error status:', error.response.status);
          console.error('Error details:', error.response.data);
          console.error('Error details:', error.message);
        } else if (error.request) {
          console.error('No response received from the server');
        } else {
          console.error('Error setting up the request:', error.message);
        }
        throw error;
      });
  }
```

- DELETE request example.

```
const removerFatura = async (id) => {
    try {
    var token = await this.getToken();
    let data = qs.stringify({ 'id': id });
    let config = {
        method: 'delete',
        maxBodyLength: Infinity,
        url: `${BASE_URL}/invoices`,
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
            'Authorization': token
        },
        data: data
    };
    const response = await axios.request(config);
        console.log(response.data);
        return response.data;
    } catch (error) {
        if (error.response) {
            console.error('Server responded with an error status:', error.response.status);
            console.error('Error details:', error.response.data);
        } else if (error.request) {
            console.error('No response received from the server');
        } else {
            console.error('Error setting up the request:', error.message);
        }
        throw error;
    }
}
```

9.2 WebServices API Schema

In order to access the data and execute requests, our team leveraged an API supplied by the company. To interact with and evaluate the functionality of the API, we opted for Postman, a tool renowned for its user-friendly interface and versatile data representation capabilities.

Utilizing Postman, we could easily formulate and test requests, aided by its intuitive interface and support for diverse data formats. This facilitated thorough testing and validation of API endpoints across various scenarios.

The API itself was accessed via the URL format:

https://devipvc.gesfaturacao.pt/gesfaturacao/server/webservices/api/mobile/{{API_VERSION}}/endpoint

Here, endpoint corresponds to the specific category or functionality we sought to access within the API. By using the variable {{API_VERSION}} with the relevant version identifier and specifying the desired endpoint, we could seamlessly interact with the API and retrieve the requisite data or perform desired actions.

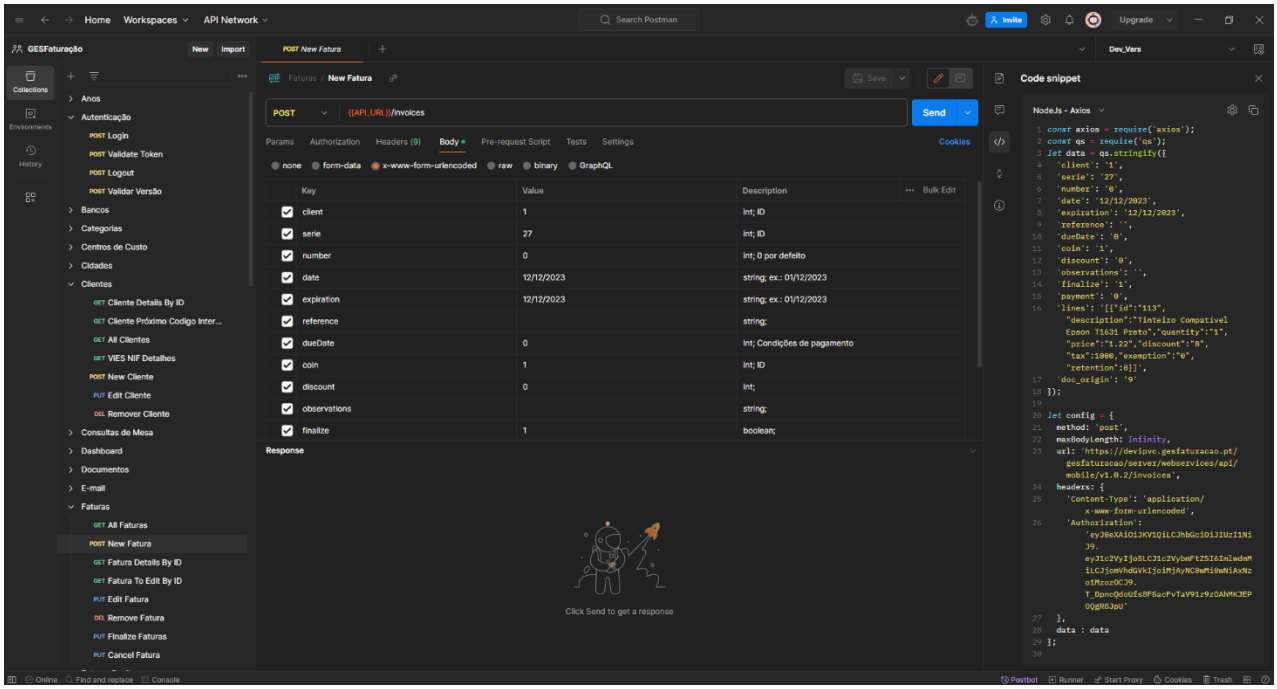


Figure 15 – Example of a request in the API.

9.3 Installation Manual

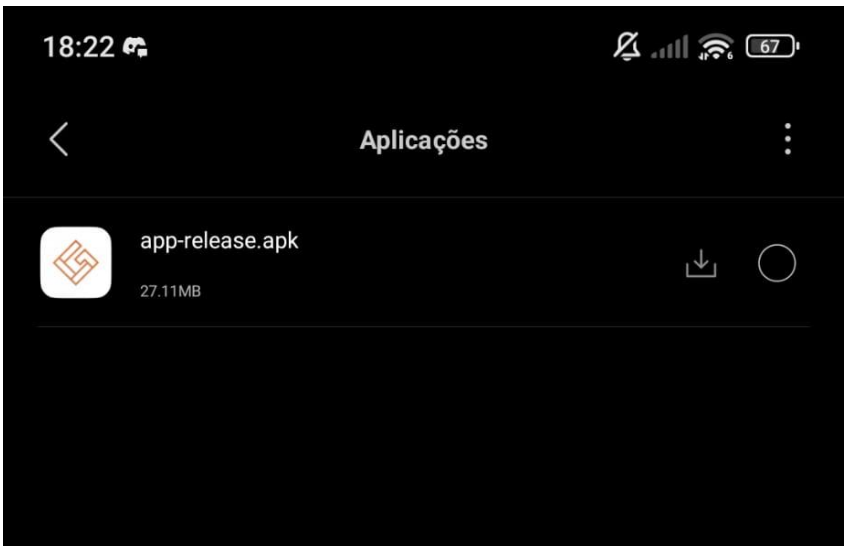


Figure 16 – APK selection for installation.

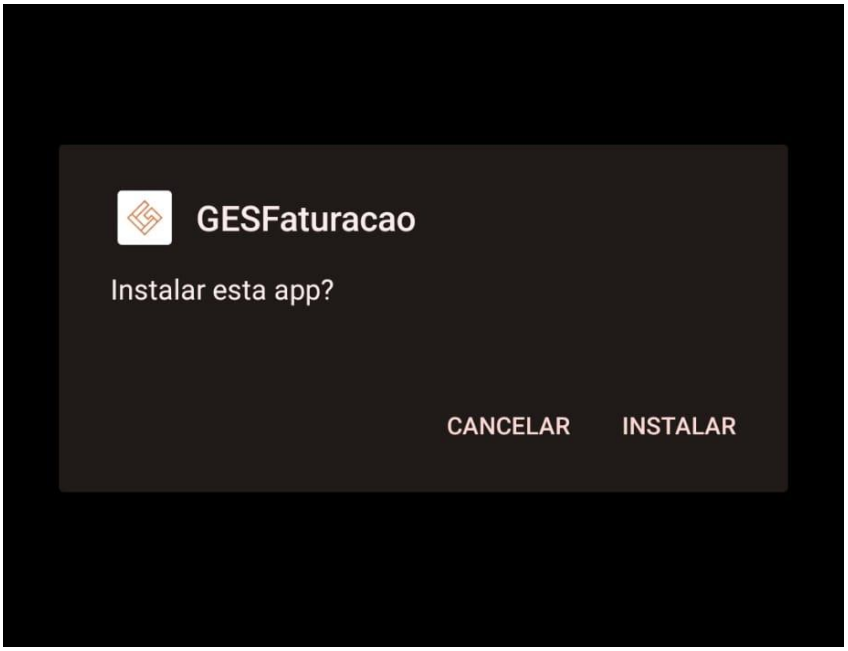


Figure 17 – Installation screen for the APK.

Figure 18 – APP installed and ready to be used.