UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO, DE CIÊNCIAS EXATAS E EDUCAÇÃO
DEPARTAMENTO DE ENG. DE CONTROLE, AUTOMAÇÃO E COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Guilherme Schlindwein Matiola

Aplicativo móvel para controle do processamento de dados de amostras de solo com integração da tecnologia RFId

Blumenau
2023

Guilherme Schlindwein Matiola

Aplicativo móvel para controle do processamento de dados de amostras de solo com integração da tecnologia RFId

Trabalho de Conclusão de Curso de Graduação em Engenharia de Controle e Automação do Centro Tecnológico, de Ciências Exatas e Educação da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Engenheiro de Controle e Automação.
Orientador: Prof. Maiquel de Brito, Dr.

Blumenau
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Matiola, Guilherme Schlindwein
Aplicativo móvel para controle do processamento de dados de amostras de solo com integração da tecnologia RFId / Guilherme Schlindwein Matiola ; orientador, Maiquel de Brito, 2023.
63 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Blumenau, Graduação em Engenharia de Controle e Automação, Blumenau, 2023.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Engenharia de Controle e Automação. 3. Automação de processos. 4. Integração de sistemas. 5. Aplicativo móvel. I. Brito, Maiquel de. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Guilherme Schlindwein Matiola

Aplicativo móvel para controle do processamento de dados de amostras de solo com integração com RFId

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenheiro de Controle e Automação" e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação.

Blumenau, 7 de julho de 2023.

Banca Examinadora:

Documento assinado digitalmente
Maiquel de Brito
Data: 12/07/2023 23:08:44-0300
CPF: ***.657.150-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Maiquel de Brito, Dr.
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Carlos Roberto Moratelli
Data: 12/07/2023 16:05:05-0300
CPF: ***.999.369-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Carlos Roberto Moratelli, Dr.
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Fabio Rafael Segundo
Data: 12/07/2023 22:41:07-0300
CPF: ***.593.909-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Fábio Rafael Segundo, Dr.
Universidade Federal de Santa Catarina

Este trabalho é dedicado aos meus pais, que batalharam muito pela minha formação não só como cidadão, mas como estudante e futuramente profissional, à minha namorada Lais, meus colegas de curso e aos meus professores.

AGRADECIMENTOS

Presto meus agradecimentos à empresa Hasar Brasil e todos seus colaboradores, pois através dela conheci parte das tecnologias que usei para o desenvolvimento deste trabalho. Não posso esquecer de mencionar os professores Dr. Maiquel de Brito e Carlos Roberto Moratelli pelos seus conhecimentos transmitidos nas áreas de computação durante a graduação, que foram essenciais para meu aprendizado na área.

“A tecnologia vai reinventar o negócio, mas as relações humanas continuarão a ser a chave do sucesso“. (Stephen Covey)

RESUMO

Este trabalho contempla o desenvolvimento de um aplicativo mobile, desenvolvido em *React Native*, que automatiza o processo da empresa contratante proporcionando segurança e assertividade no controle de dados do processo de amostras de solo. Este processo, quando feito manualmente possibilitava fraudes e avarias nos documentos preenchidos. O aplicativo obtém dados a partir da inserção de uma planilha que contém informaçoes sobre furos em minas de exploração. Os dados são interpretados pelo software, onde cada linha da planilha representa uma amostra do furo. Os dados são inseridos em um banco de dados NoSQL (*Firebase Firestore*). Para controle de estoque as caixas que contém as amostras são vinculadas a tags RFID e a um QrCode específico da caixa. Os usuários podem, então, usar o aplicativo para executar os processos necessários sobre as amostras. Os dados de cada etapa de processamento podem ser exportados, incluindo informações dos usuários que as executaram, bem como sobre o tempo de execução do processo.

Palavras-chave: Aplicativo mobile; Automação de processos; RFId.

ABSTRACT

This work reports the development of a mobile application, built with React Native, that automates the process for the contracting company, providing security and accuracy in the control of soil sample data. Previously, manual handling of this process led to potential fraud and damage to filled documents. The application retrieves data by inputting a spreadsheet containing information about boreholes in mining exploration. The software interprets the data, with each row in the spreadsheet representing a sample from a borehole. The data is then stored in a NoSQL database (Firebase Firestore). To track inventory, the boxes containing the samples are associated with RFID tags and a specific QR code for each box. Users can utilize the application to perform necessary processes on the samples. The data from each processing step can be exported, including information about the users who performed them, as well as the process execution time.

Keywords: Mobile application; Process automation; RFId.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CSS	Cascading Style Sheet
DOM	Document Object Model
EPC	Electronic Product Code
HF	High Frequency
HTML	HyperText Markup Language
IP	Internet Protocol
JSON	Javascript Object Notation
LF	Low Frequency
MHz	Megahertz
NoSQL	Not Only Structured Query Language
SQL	Structured Query Language
TCP	Transmission Control Protocol
UHF	Ultra High Frequency
XML	eXtensible Markup Language
ZPL	Zebra Printer Language

SUMÁRIO

1 INTRODUÇÃO

Automatizar processos é uma tendência crescente nas empresas, especialmente naquelas que lidam com grandes quantidades de dados. As empresas que analisam dados manualmente podem enfrentar problemas de produtividade, eficiência e precisão. A automação pode trazer muitos benefícios, como a redução de erros, aumento da velocidade de processamento e análise, e a liberação de tempo e recursos para atividades mais estratégicas. Neste contexto, é importante entender as opções disponíveis de automação para empresas que desejam melhorar seus processos de análise de dados e aumentar sua eficiência e precisão.

Geralmente, cada empresa possui um padrão de coleta de dados e armazenamento, tornando complexo o desenvolvimento de uma aplicação generalista que resolva o mesmo problema para todas. O comum é que cada uma requeira uma solução específica. Existem muitos aplicativos de análise de dados disponíveis no mercado que podem ajudar as empresas a automatizar seus processos de análise de dados. Esses aplicativos variam de soluções simples baseadas em planilhas eletrônicas a plataformas sofisticadas específicas para cada tipo de planilha e diferentes dados inseridos nela.

Quando dados representam elementos físicos, além do tratamento das informações sistemas computacionais, têm-se o trabalho de controlar o inventário físico. Ou seja, além de algumas empresas controlarem a geração de dados manualmente, executam verificações de forma manual em controle de estoques, o que promove um possível erro em cascata no processo. Uma solução para tratar deste problema é a identificação digital dos elementos físicos que são controlados, usando tecnologias como, por exemplo, o RFID.

A tecnologia *Radio Frequency Identification* (RFID) é um método de identificação automática de objetos e pessoas, que utiliza ondas de rádio para transmitir informações de um pequeno chip presente em um objeto ou pessoa para um leitor eletrônico. Atualmente, é uma das tecnologias mais utilizadas no mundo para a identificação automática de objetos e pessoas. O RFID é utilizado em diversas áreas, como controle de estoque, logística, transporte, segurança, saúde e entretenimento. Ela permite a automação de processos, redução de erros, aumento da eficiência, rastreamento de produtos e melhoria da segurança.

A união de um controle de dados completo e bem executado, unido da tecnologia RFID pode trazer ganhos para a empresa que aderir a esse modelo de gestão de dados com RFID. Além da precisão do método, ganha em velocidade de execução e praticidade. Através de um aplicativo Android desenvolvido no estágio na Hasar Brasil pelo autor, será feito o tratamento dos dados de amostras de solo garantindo segurança e confiabilidade.

A integração de regras de negócio do sistema, aliado ao controle no processamento de dados deve permitir organizar cada amostra em caixas (com 2 a 25 caixas com amostras de solo) e paletes (com até 45 amostras de solo) em informações dentro de um banco de dados em nuvem. Além disso, é necessário que registre-se o tempo de execução e o usuário

que realizou cada um dos processos automatizados pelo aplicativo.

1.1 OBJETIVOS

O objetivo geral deste trabalho é automatizar o processamento das informações sobre amostras de solo coletados por uma empresa do ramo de mineração. Para automatizar esse processo, que atualmente é realizado de forma manual, será desenvolvido um aplicativo para ser executado em dispositivos móveis.

Os objetivos específicos deste trabalho são:

- Garantir consistência nos valores inseridos no banco de dados, sem divergências de valores;
- Simplificar o processo de detalhamento e inspeção de cada amostra de solo;
- Permitir a associação de cada amostra de solo a uma tag RFID única, para que cada uma delas possa ser tratada adequadamente pelos sistemas computacionais da empresa;
- Organizar as amostras de solo em três níveis distintos: amostras (chamadas de "chip box", caixas com amostras (chamadas de "white box") e paletes de amostras.
- Realizar, no processamento de cada amostra (ou cada "chip box"), o controle de tempo de cada processo (serão dez, ao todo) realizado sobre ela, bem como o usuário que a processou;
- Exportar as informações pertinentes do processamento das amostras de furo em formato de planilha eletrônica.

1.2 ESTRUTURA DO TRABALHO

Após definidos os objetivos gerais e específicos, o trabalho é organizado em uma sequência lógica pra tentar integrar todas as ideias do sistema para que seja o mais simples possível a compreensão do mesmo. O restante deste trabalho está estruturado nos seguintes capítulos:

- Capítulo 2 - Fundamentação teórica: Explicação, com base na literatura, das tecnologias utilizadas no desenvolvimento deste trabalho de conclusão de curso. Inclui referências das linguagens de programação utilizadas, banco de dados, tecnologia RFID e protocolos de comunicação.
- Capítulo 3 - Desenvolvimento e resultados: Capítulo destinado a apresentar todas as saídas físicas e digitais do sistema, bem como todo o processo de desenvolvimento da solução visual, regras de negócio impostas no tratamento de dados, e por final, a explicação do propósito final da aplicação, que já foi

comentada nos objetivos gerais e específicos. Também consta a justificativa da escolha das tecnologias e solução proposta, destinado a mostrar a valência de cada tecnologia utilizada na aplicação e argumentos que sustentam a escolha destas para o desenvolvimento e integração.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo descreve os conceitos e tecnologias utilizadas neste trabalho, por meio de definições técnicas obtidas a partir da literatura na área.

2.1 ARQUITETURA CLIENTE-SERVIDOR E CONCORRÊNCIA

A arquitetura cliente-servidor é uma abordagem de software que divide a funcionalidade de um sistema em duas partes distintas: o cliente e o servidor. O cliente é quem solicita recursos ou serviços que são fornecidos pelo servidor. Essa arquitetura tem sido amplamente utilizada em sistemas distribuídos, em que o cliente e o servidor podem estar localizados em diferentes computadores. É um modelo de arquitetura de sistemas muito utilizado em diversos tipos de aplicações, como gerenciadores de banco de dados, sistemas de correio eletrônico e serviços de rede (TANENBAUM; STEEN, 2007).

Um dos principais desafios na arquitetura cliente-servidor é a concorrência, que ocorre quando múltiplos clientes enviam solicitações simultâneas ao servidor. A concorrência pode levar a problemas de desempenho, como a diminuição da velocidade de resposta do servidor e a sobrecarga da rede. Para lidar com esse problema, várias técnicas de programação concorrente têm sido utilizadas na arquitetura cliente-servidor, como o uso de threads e processamento de dados em fila. De acordo com Silberschatz, Galvin e Gagne (2013), a concorrência é um dos principais desafios enfrentados pelos programadores em sistemas distribuídos (SILBERSCHATZ; GALVIN; GAGNE, 2013).

Além disso, a concorrência também pode levar a problemas de consistência de dados em sistemas cliente-servidor. Quando vários clientes tentam acessar e modificar os mesmos dados ao mesmo tempo, pode haver conflitos que resultam em dados inconsistentes. Para evitar esses problemas, as técnicas de controle de concorrência são frequentemente utilizadas no processo de desenvolvimento de um sistema ou aplicação, como o bloqueio de dados e a serialização de transações. Segundo Korth, Silberschatz e Sudarshan (2005), essas técnicas ajudam a garantir a consistência dos dados em sistemas cliente-servidor (KORTH; SILBERSCHATZ; SUDARSHAN, 2005).

Por fim, a arquitetura cliente-servidor também pode ser escalada horizontalmente para lidar com a concorrência. Isso envolve a adição de mais servidores para distribuir a carga de solicitações entre eles. A escalabilidade é um dos principais desafios enfrentados pelos arquitetos de sistemas cliente-servidor. A migração de sistemas SQL para NoSQL é uma tendência nesse quesito por possuir mais facilidade de implementação de concorrência e escalabilidade com grandes quantidades de dados (COULOURIS; DOLLIMORE; KINDBERG, 2011). Neste trabalho, optou-se pela utilização do Firebase como banco de dados e serviço de *backend* por possuir internamente soluções para sincronismo e concorrência de manipulação de dados.

2.2 JSON

O JSON (JavaScript Object Notation), criado no ano de 2000, é um formato para troca de informações bem definida e estruturada (SRIPARASA, 2013). Sua forma de representar os dados se baseia em elementos compostos por pares chave-valor, como pode ser observado na Figura 5 deste documento. Ao longo do tempo de uso, substituiu quase que 100% das outras alternativas à esse tipo de dado.

Esta forma de representação se popularizou por possuir um menor volume de texto para a representação das informações, tornando-se mais legível para humanos e de fácil interpretação por máquinas. Este tipo de arquivo é bastante utilizado por aplicações *web* que necessitam enviar e receber informações ao longo de sua execução. Em arquivos na linguagem JavaScript, o JSON é a representação de objetos criados e suas respectivas estruturas (SRIPARASA, 2013).

2.3 HTML E CSS

Desde início do uso dos computadores com a Internet, os *websites* são construídos com a linguagem de marcação HTML. O HTML (HyperText Markup Language) é fundamental para a criação e estruturação de páginas da web. Criada por Tim Berners-Lee e sua equipe no CERN (Organização Europeia para a Pesquisa Nuclear) na década de 1990, o HTML desempenhou um papel essencial no desenvolvimento da World Wide Web. Inicialmente, o HTML permitia apenas a marcação de texto simples, mas, à medida que a web evoluía, novos recursos foram adicionados para permitir a formatação de texto, inserção de imagens e criação de links hipertexto. Com o HTML, os desenvolvedores podem definir os elementos estruturais de uma página, como cabeçalhos, parágrafos, listas e tabelas, fornecendo a base para o conteúdo visualmente atraente e navegável (ROBBINS, 2018).

Já o CSS (Cascading Style Sheets) é uma linguagem de estilo poderosa que trabalha em conjunto com o HTML para controlar a aparência e o layout de documentos da web, onde o HTML expressa a estrutura do documento e o CSS define a aparência dos elementos da estrutura. Proposta pela primeira vez por Håkon Wium Lie em 1994 e posteriormente desenvolvida em colaboração com Bert Bos, o CSS permite que os desenvolvedores definam estilos consistentes e reutilizáveis para múltiplas páginas da web. Ao separar a estrutura e o conteúdo do design visual, o CSS proporciona uma abordagem modular para o desenvolvimento web. Com o CSS, é possível definir a cor, fonte, tamanho e posicionamento dos elementos HTML, além de aplicar efeitos visuais, como sombras, transições e animações. Essa separação de responsabilidades entre o HTML e o CSS simplifica a manutenção e atualização de um site, permitindo que as alterações no estilo sejam aplicadas de forma consistente em toda o *website*.

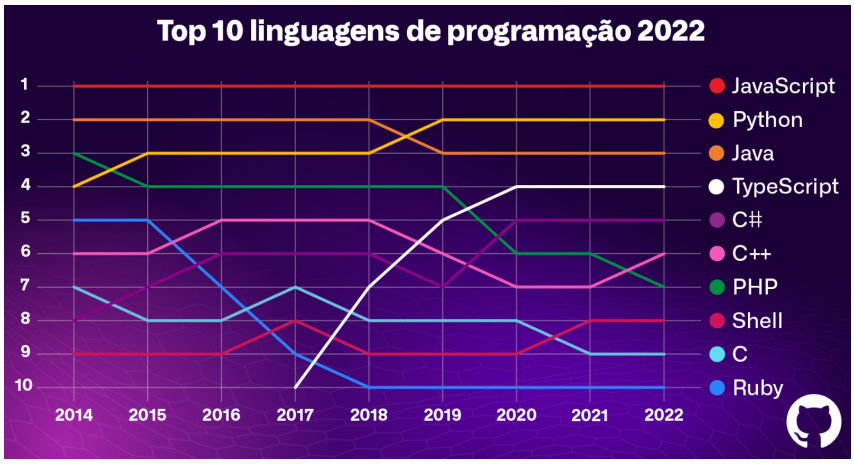
Um arquivo HTML é formado por tags, que são utilizadas para criar elementos como títulos, parágrafos, links e imagens. Isso permite que os desenvolvedores criem

interfaces de usuário interativas e dinâmicas que respondam a eventos e interações do usuário (FLANAGAN, 2011). A lógica de interação (chamadas de APIs ou conexão com banco de dados por exemplo) em um *website* desenvolvido em HTML e CSS é realizada pelo JavaScript, uma linguagem de programação de alto nível que foi criada com o objetivo de tornar as páginas da web interativas e dinâmicas.

2.4 LINGUAGEM JAVASCRIPT

Desenvolvida por Brendan Eich na Netscape Communications Corporation em 1995, o JavaScript rapidamente se tornou uma das linguagens de programação mais amplamente utilizadas no desenvolvimento web. Com o JavaScript, os desenvolvedores podem manipular o conteúdo de uma página da web em tempo real, responder a eventos, validar formulários, criar animações e interagir de forma dinâmica com os usuários. Desde 2014, é a linguagem mais utilizada segundo o estudo do Github Octoverse:

Figura 1 – Linguagens mais utilizadas desde 2014



Fonte: Github Octoverse, (GITHUB, 2022).

JavaScript é uma linguagem de programação interpretada, de alto nível e orientada a objetos. Ela é uma das principais linguagens usadas para desenvolver aplicativos *web* e móveis, além de ser amplamente utilizada em outras áreas, como internet das coisas e desenvolvimento de jogos. Sua ampla adesão tem relação com a expansão da internet, visto que foi desenvolvida primeiramente para uso em *browsers* (FLANAGAN, 2011).

Um dos recursos mais poderosos do JavaScript é a capacidade de interagir dinamicamente com elementos HTML e CSS de uma página *web*. Ao adicionar comportamentos e funcionalidades interativas às páginas da web, o JavaScript tornou possível criar experiências envolventes e interativas, como galerias de imagens com slides, formulários de validação em tempo real e carregamento de conteúdo assíncrono. O JavaScript é suportado

pelos principais navegadores da web e é uma peça fundamental no desenvolvimento de aplicativos web modernos (ROBBINS, 2018).

Outra característica importante do JavaScript é a sua ampla variedade de bibliotecas e frameworks disponíveis para os desenvolvedores. Alguns dos *frameworks* mais populares incluem React, React-Native, Nextjs, entre outros. Cada uma dessas bibliotecas oferece uma abordagem diferente para a criação de aplicativos *web* ou para dispositivos móveis. Além disso, o JavaScript também é amplamente utilizado para a criação de APIs RESTful, que permitem que aplicativos se comuniquem com servidores *web* e realizem operações complexas.

O JavaScript possui implementação de assincronismo, ou seja, antes que requisições sejam atendidas, a execução continua mantendo renderizações em um *websit*e. Pode-se citar um exemplo desse método ao abrir o site *YouTube*, que antes de exibir os vídeos da página inicial, é carregado uma pré-visualização. As aplicações modernas desenvolvidas usam, muitas vezes, programação assíncrona, mantendo a execução de códigos e outras chamadas de função em segundo plano. Em relação à programação assíncrona, o JavaScript usa a técnica de callbacks para lidar com operações assíncronas, como a leitura de dados de um servidor web. No entanto, essa abordagem pode levar a um código confuso e difícil de manter. Para resolver esse problema, foram introduzidas as *Promises*, que oferecem uma abordagem mais limpa e concisa para lidar com operações assíncronas (CROCKFORD, 2008). As promessas (chamadas de *promises*) são recursos que permitem lidar com operações assíncronas de forma organizada. Elas possuem três estados: pendente, resolvida e rejeitada. E enquanto a requisição é pendente, a aplicação não precisa ser travada.

Em resumo, o JavaScript é uma linguagem de programação versátil que é amplamente utilizada no desenvolvimento de aplicativos web e móveis, bem como em outras áreas. Com suas bibliotecas e frameworks, é possível criar aplicativos web interativos e dinâmicos, enquanto as Promises e outras técnicas ajudam a lidar com operações assíncronas de maneira limpa e concisa. Recentemente, o JavaScript também tem sido usado cada vez mais no desenvolvimento de aplicativos móveis híbridos, que são aplicativos que são escritos em JavaScript e depois compilados em código nativo para iOS e Android.

2.5 REACT

O React é um framework Javascript, desenvolvido por engenheiros de software do Facebook. Criada em 2015, é uma das formas mais utilizadas na atualidade para construir interfaces de usuário em versões *web*, aplicativos móveis ou programas de computador em geral.

As principais características do React incluem o uso de JSX, que é uma sintaxe que permite a mistura de código JavaScript e HTML para criar componentes; a renderização declarativa, que permite ao React atualizar automaticamente a interface do usuário quando o estado dos dados é alterado; e o uso do Virtual DOM, que é uma representação em

memória da estrutura da interface do usuário que permite ao React atualizar somente as partes necessárias da interface do usuário quando há mudanças no estado dos dados.

O DOM (Document Object Model) é uma interface de programação de aplicações (API) que representa um documento HTML ou XML como uma árvore de objetos, onde cada objeto representa um elemento do documento. O DOM permite que os desenvolvedores possam interagir com o conteúdo de uma página web de forma dinâmica, alterando sua estrutura, estilo e conteúdo através de scripts. Quando uma página web é carregada em um navegador, o DOM é criado automaticamente a partir do código HTML e CSS da página. Cada elemento HTML é representado como um nó na árvore do DOM, que pode ser acessado e manipulado por meio de scripts em linguagens como JavaScript (DUCKETT, 2014). Através do DOM, é possível modificar o conteúdo de uma página, adicionar ou remover elementos, alterar estilos e atributos, criar animações, entre outras coisas. O DOM é essencial para a criação de aplicações web. Nesse contexto, o React manipula no DOM um componente denominado "*raiz*" dentro do documento HTML, que é alterado a cada interação do usuário na página (BANKS; PORCELLO, 2021).

Outra característica do framework é o uso de um modelo de programação baseado em componentes. Geralmente, itens componentizados são rodapés, cabeçalhos, menus e todo item que possa reaparecer em mais de uma tela, que permite que os desenvolvedores dividam a interface do usuário em partes independentes e reutilizáveis. Cada componente pode ter seu próprio estado e comportamento, e pode ser usado para construir interfaces complexas e dinâmicas (REACT DOCUMENTATION, 2023).

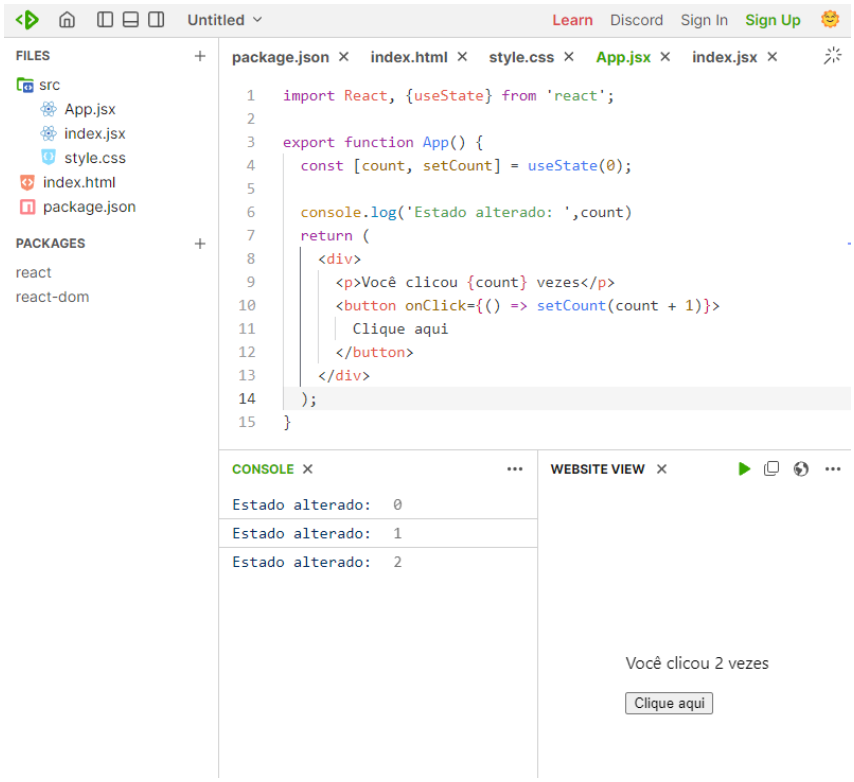
Além da organização e componentização do projeto, pode-se citar algumas funcionalidades específicas do *React* que, ao serem compreendidas, auxiliam na produção de eventos na tela. Por essas características da linguagem, provavelmente deu-se origem ao nome "React", em português, "reagir". Cada ação produz uma alteração na manipulação do DOM. Entre as funcionalidades específicas do React, pode-se destacar as seguintes:

- **Estados:** Os estados, no React, pertencem a sua biblioteca padrão de funções, a "React Hooks", que será explicado a seguir. Os estados servem, literalmente, para guardar um estado durante uma ação (como abrir a página, por exemplo), e alterar caso seja necessário, para outro estado caso o usuário realize outra interação.
- **React Hooks:** Mais uma ferramenta importante oferecida pelo React são os hooks (em portugues "ganchos"). A analogia com ganchos faz sentido, pois sua função é executar assim que alguma interação do usuário acontece, porém, de uma forma mais abstraída de sua percepção. Por exemplo, com o hook *"useState"*, pode-se definir um estado dentro de um componente, e atualizá-lo com o método *"setState"*. Com o hook *useEffect*, pode-se executar efeitos colaterais, como atualizar o título da página ou carregar dados de uma API, quando o estado de um componente é alterado. Os Hooks são uma maneira

mais moderna e concisa de escrever componentes em React, tornando o código mais fácil de ler e manter. Além disso, os Hooks permitem a reutilização de lógicas de estado e efeitos em diferentes componentes, tornando seu código mais modular e escalável.

Um exemplo de manipulação de estados é dado a seguir:

Figura 2 – Manipulação de variável com estados



Fonte: Autor, 2023.

Ao pressionar o botão "Clique aqui", o estado é atualizado a cada interação, como se pode ver no console na parte inferior esquerda da Figura 2.

2.6 REACT NATIVE

React Native é um framework para desenvolvimento de aplicativos móveis, também criado pelo Facebook em 2015, que permite criar aplicativos para iOS e Android com uma única base de código em JavaScript. Ele utiliza uma abordagem de desenvolvimento de interface do usuário baseada em componentes, o que significa que os elementos de interface do usuário são criados como componentes reutilizáveis que podem ser combinados para formar a interface do usuário do aplicativo.

Por ser mais dedicado para o desenvolvimento mobile, embora possa ser usada na web, possui ferramentas de comunicação com sensores próprios do dispositivo, como

bluetooth, verificação de conexão à rede, GPS, leitores NFC, entre outros.

Uma das principais vantagens do React Native é a sua capacidade de fornecer um desempenho nativo, uma vez que os aplicativos criados com ele utilizam os componentes nativos da plataforma subjacente. Além disso, o React Native possui uma grande comunidade de desenvolvedores e bibliotecas de terceiros que podem ajudar a acelerar o processo de desenvolvimento (LIKOVYI, 2018).

De maneira geral, a forma de desenvolver uma aplicação no *React Native* é similar na construção do código do *React*, por meio de divisões (chamadas de tags), não necessariamente iguais ao HTML padrão, mas muito similares. O *React Native* é uma tecnologia amplamente usada para desenvolvimento de aplicativos móveis, que pode oferecer uma experiência nativa e uma maneira eficiente de desenvolver aplicativos para iOS e Android com uma única base de código.

2.7 TECNOLOGIA RFID

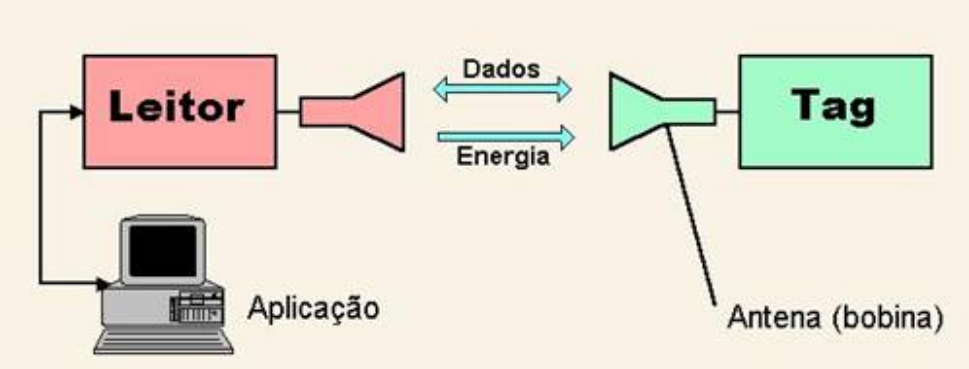
RFID significa Identificação por Radiofrequência (em inglês, Radio Frequency Identification). É um sistema que permite a identificação e rastreamento de objetos através do uso de ondas de rádio. Esta tecnologia traz soluções com etiquetas chamadas de passivas ou ativas, podendo ser utilizada em diversas aplicações, desde o controle de acesso de pessoas e veículos em estacionamentos até a identificação de animais em fazendas, controle de estoque em empresas e logística de transporte. O sistema com RFID é capaz de capturar dados automaticamente, sem a necessidade de contato físico entre o leitor e o objeto, o que traz uma série de benefícios em termos de eficiência e segurança, além de praticidade (SANTOS, 2022).

Tanto o sistema ativo quanto o passivo consistem em três componentes principais: um transmissor de rádio, um receptor de rádio e um microchip com antena que é incorporado ao objeto que se deseja rastrear. O microchip contém um código exclusivo que é transmitido através de ondas de rádio para o receptor, que captura e interpreta o código e o envia para um sistema de processamento de dados. O sistema usa ondas de rádio para transferir dados sem fio entre um leitor RFID e uma etiqueta RFID (ALI; KHAN; ISLAM, 2015).

Um sistema RFID passivo é composto por uma etiqueta (tag) e um leitor. A etiqueta é alimentada por energia eletromagnética do leitor, que envia um sinal na radiofrequência específica, enquanto a tag possui um leitor que envia de volta a informação armazenada nela. Essas etiquetas não possuem bateria, o que as torna mais baratas e duráveis em relação aos sistemas RFID ativos. No entanto, a distância de leitura dessas etiquetas é menor do que a dos sistemas ativos e não possuem capacidade para serem utilizadas em aplicações que exijam transmissão de dados em tempo real.

Já um sistema RFID ativo, além da etiqueta e do leitor, possui também uma bateria na etiqueta. Dessa forma, a etiqueta pode enviar um sinal de rádio por conta própria,

Figura 3 – Esquema de funcionamento do RFID



Fonte: Grupo de Teleinformática e Automação (UFRJ, [s.d.])

sem depender do sinal do leitor. Isso permite que esses sistemas tenham uma distância de leitura maior do que os sistemas passivos e sejam utilizados em aplicações que exijam transmissão de dados em tempo real. No entanto, essas etiquetas são mais caras e menos duráveis do que as etiquetas passivas. (WANT; PERING, 2006)

Figura 4 – Leitor RFID da marca ZEBRA

Fonte: Zebra Technologies Corporation, 2020

O processo de recebimento das frequências no RFID começa quando um leitor envia um sinal de rádio para uma etiqueta RFID. A etiqueta responde ao sinal de rádio do leitor RFID enviando uma transmissão de volta ao leitor. Esta transmissão contém informações sobre a etiqueta RFID, como o número de identificação exclusivo da etiqueta, também chamado de EPC (Electronic Product Code). O leitor RFID recebe a transmissão do EPC específico e decodifica as informações contidas nele. O leitor pode então usar essas informações para executar uma determinada ação, como identificar um item específico

em um estoque ou permitir que um funcionário acesse uma área restrita, dependendo da aplicação (ZEBRA (ZEBRA TECHNOLOGIES CORPORATION), s.d.).

As frequências utilizadas no sistema RFID podem variar de acordo com o tipo de sistema RFID utilizado. Por exemplo, as frequências usadas em sistemas de baixa frequência (LF) geralmente variam de 125 kHz a 134 kHz, enquanto as frequências usadas em sistemas de alta frequência (HF) geralmente variam de 13,56 MHz a 27,25 MHz. Já as frequências utilizadas em sistemas de ultra-alta frequência (UHF) podem variar de 433 MHz a 928 MHz (FINKENZELLER, KLAUS, 2010) .

2.8 FIREBASE

O Firebase é uma ferramenta desenvolvida pelo Google, e funciona como uma plataforma chamada de *Backend as a Service*. Possui uma considerável facilidade de uso em aplicações em que o volume de dados será grande, visto que armazena seus dados em um formato NoSQL (FIREBASE, 2018).

Além disso, a plataforma do *Google* oferece uma base de funcionalidades comuns em aplicativos já pronta, como por exemplo, de autenticação de usuário, *hosting* e armazenamento de dados em um banco de dados. A ferramenta oferece opções de integração de serviços que compartilham dados e *informações* entre si, além de integração com o *Google Analytics*, que permite verificar informações estatísticas sobre a aplicação através de gráficos e dashboards.

Os projetos e aplicações podem ser gerenciados através do Firebase CLI, que disponibiliza diversas ferramentas para visualização e implantação, ou pela própria página do Firebase, onde pode-se gerenciar de uma maneira mais visual e intuitiva as regras da aplicação e a estrutura do banco de dados (chamadas de coleções, termo comum em bancos NoSQL) (FOWLER; SADALAGE, 2013).

Como citado anteriormente, o Firebase fornece serviços implementados na base da própria ferramenta que permite desenvolvimento de uma aplicação de modo mais ágil, até de uma certa forma, pulando etapas necessárias se fosse utilizado outro serviço. Após a conexão da aplicação ao Firebase (etapa que envolve configuração do ambiente de desenvolvimento, *apiKey*, URL do banco de dados, entre outros), é possível dar início ao uso dessas ferramentas "prontas". As principais são detalhadas nas subseções 2.8.1 a 2.8.3.

2.8.1 Autenticação de usuários

O serviço de autenticação de usuários permite cadastro, login, gerenciar senhas e ou até logar com contas vinculadas ao Google, Facebook, entre outros. Possui um método já implementado para otimização de uso, como mensagens de erros ("senha incorreta", "usuário não existente", por exemplo) e processo de alterações de credenciais prontos. Os usuários cadastrados são armazenados no banco de dados fornecido, as credenciais são

transmitidas ao SDK do Firebase *Authentication*, o backend verifica e envia uma resposta ao cliente, e cabe ao criador da aplicação decidir o que fazer para cada uma das respostas.

2.8.2 Banco de dados

Provavelmente o serviço mais utilizado, oferece dois tipos de banco de dados hospedados em nuvem, o *Realtime Database*, que armazena dados em forma de árvore JSON, e o *Firestore Database*, um banco de dados NoSQL que armazena dados em coleções. Ambos podem ser integrados com o serviço de autenticação, e possuem recursos de implementação poderosos, que o próprio SDK oferece, como os descritos abaixo:

- Persistencia de dados offline: de modo geral, o firebase possui uma configuração para aplicações desenvolvidas para dispositivos móveis (objeto de estudo do autor deste trabalho) que mantém dados da aplicação salvos em cache. Isso permite que funcionalidades importantes do aplicativo não sejam prejudicadas, e caso haja uma desconexão, o serviço pode continuar sendo usado. Posteriormente, o próprio Firebase gerencia e sincroniza o banco de dados resultante das ações realizadas em modo offline;
- Banco de dados *Real Time*: como mencionado anteriormente, o banco de dados do Firebase possui uma taxa de resposta muito alta se comparado a outros serviços gratuitos. O nome *"Real-Time"* refere-se ao bom tempo de resposta à alterações e consultas. Ele armazena seus dados em forma de árvore JSON, tornando a consulta mais veloz;
- Cloud Firestore Database ou Firestore Database: é um banco de dados NoSQL que permite aos usuários armazenar e sincronizar dados para desenvolvimento do lado do servidor e do cliente. Os usuários podem utilizar o desenvolvimento na web, móvel e servidor e sincronização entre aplicativos. Essa opção armazena dados em coleções de documentos. Tem possibilidade de realizar consultas mais complexas e escalabilidade horizontal, o que o torna uma opção melhor para aplicativos.

2.8.3 Firebase Storage

O *Firebase Storage* é um serviço para armazenar arquivos provenientes de aplicações no *Google Cloud*. Uma das principais motivações de uso é a disponibilizade de ferramentas que auxiliam no desenvolvimento de aplicações que necessitam de desempenho e segurança, como na área de engenharia.

A integração com *frameworks* e aplicações foi aprimorada ao longo dos anos pelo Google, e permite a utilização em escala considerável na versão gratuita. Tanto com o *Real Time Database* quanto no *Firestore Database*, é fornecido possibilidade de sincronização *offline*, facilitando a implementação em ambientes de campo ou locais com acesso à rede

com instabilidade (FIREBASE, 2018). A ferramenta oferece vários planos para atender às necessidades de diferentes tipos de aplicativos e empresas.

2.9 NOSQL

NoSQL, que significa "Not Only SQL"(não apenas SQL), é um termo genérico que se refere a um conjunto de tecnologias de banco de dados que não utilizam o modelo relacional tradicional baseado em tabelas, colunas e linhas. Em vez disso, as soluções NoSQL usam modelos de dados diferentes, como documentos, gráficos, pares chave-valor ou colunas (FOWLER; SADALAGE, 2013). Existem várias opções de bancos de dados não relacionais disponíveis, como Riak, MongoDB, Cassandra, Neo4j e Firebase.

Diferente dos bancos de dados relacionais, que armazenam seus dados em tabelas de estrutura definida na própria criação, bancos de dados NoSQL não possuem estrutura de armazenamento de informações fixas. Dos exemplos de bancos NoSQL citados acima, MongoDB e *Firebase Firestore* armazenam dados em coleções, que possuem documentos (itens). Geralmente, elementos de entidades (como documentos e colunas) de um banco de dados NoSQL possuem formas mais voláteis, ou seja, a estrutura de diferentes instâncias de uma mesma entidade pode variar (FOWLER; SADALAGE, 2013). Para exemplificar, pode-se considerar o seguinte documento da Figura 5. A entidade representa um país, cujas camadas possuem principais cidades, e dentro de cada cidade, possuem dados referentes a população e industrialização. Parece, a primeira vista uma estrutura bem definida.

Figura 5 – Exemplo de estrutura NoSQL no formato JSON

Fonte: Fowler, 2013, adaptado

Contudo, no banco de dados NoSQL fornecido pelo Firebase e MongoDB, por exemplo, caso a consulta ao banco de dados seja feita de maneira errônea, podem-se criar campos indesejados extras sem perceber, justamente por não possuir uma regra definida de padrão de armazenamento dos dados.

As soluções NoSQL foram criadas para lidar com as limitações do modelo relacional, que não é escalável horizontalmente e não é adequado para lidar com dados não estruturados ou semiestruturados. As soluções NoSQL permitem a escalabilidade horizontal, são mais flexíveis e oferecem melhor desempenho em cenários de alta concorrência e grandes volumes de dados. Porém, como pode-se perceber no exemplo anterior a este parágrafo, possui regras de uso bastante sólidas, tão quanto um banco de dados do modelo relacional (FOWLER; SADALAGE, 2013).

As soluções NoSQL são usadas em muitas aplicações modernas, como armazenamento e processamento de Big Data, aplicações Web, IoT e jogos. Geralmente bancos NoSQL fornecem sua própria linguagem de consulta, reduzindo o tempo de resposta. Entre os mais utilizados, pode-se citar Firebase, MongoDB, Cassandra, Couchbase, HBase, Neo4j e Redis.

2.10 COMUNICAÇÃO TCP E SOCKETS

Sockets são uma abstração fundamental para a comunicação entre processos em sistemas distribuídos através de redes de computadores. Eles permitem que os processos estabeleçam conexões e troquem dados de forma bidirecional. Uma definição comum de socket é que ele é um ponto de extremidade para a comunicação, representado pela concatenação de um endereço IP e um número de porta. Basicamente o requisito é que os dois dispositivos estejam conectados à mesma rede.

No livro “Operating System Concepts“ (Silberschatz, Galvin e Gagne, 2013) os autores enfatizam que a comunicação ocorre entre um par de sockets. Por exemplo, pode-se realizar a conexão com um dispositivo na rede através do socket com o endereço IP 161.25.19.8 e porta 1625, podendo ser reescrito no formato 161.25.19.8:1625

Os sockets TCP são amplamente utilizados para comunicação em redes de computadores. Eles são identificados por um endereço IP e um número de porta específico, permitindo que os programas estabeleçam conexões e troquem dados de forma confiável utilizando o protocolo TCP (Transmission Control Protocol).

3 DESENVOLVIMENTO E RESULTADOS

Neste capítulo, serão apresentadas as ferramentas utilizadas para desenvolvimento do aplicativo móvel, bem como a solução proposta pelo autor para atender aos objetivos gerais e específicos e o processo detalhado de cada etapa.

3.1 DEFINIÇÃO DO PROBLEMA

O aplicativo desenvolvido atende aos requisitos definidos por uma empresa mineradora. A empresa mineradora coleta amostras de solo em suas minas. Cada perfuração do solo é registrada como um furo que tem diferentes metragens para contabilização. Para cada metragem, é extraída uma amostra de solo, que será processada para extração de informações de sua estrutura. Esse processamento é o objeto de estudo neste trabalho.

O processo de detalhamento de informações para controle sobre as amostras de solo atual da empresa mineradora é feito de maneira manual em uma planilha física (em papel). Esse método de preenchimento manual pode apresentar falhas de segurança sobre a informação. Nada impede a fraude na descrição de horas de início e fim da conferência da amostra. Além disso, o ambiente de chão de fábrica ou de campo apresenta um maior risco para a integridade dos documentos em formato físico.

Além do preenchimento das planilhas, o processo contempla a identificação das amostras de solo através de uma espécie de etiqueta metalizada. Contudo, a informação é inserida na etiqueta manualmente por marcação de metal, e que pode novamente causar confusão e erros no processo. Essa será outra correção que a aplicação desenvolvida deve fazer.

3.2 ESCOLHA DAS TECNOLOGIAS

O tratamento de dados e informações, como citado na seção 1.1, é o foco principal deste trabalho. Os dados tratados e manipulados na aplicação, que representam amostras de solo extraídas pela empresa mineradora, possuem características específicas, e a exigência da segurança e assertividade dos dados é vital para que o valor sobre elas seja garantido.

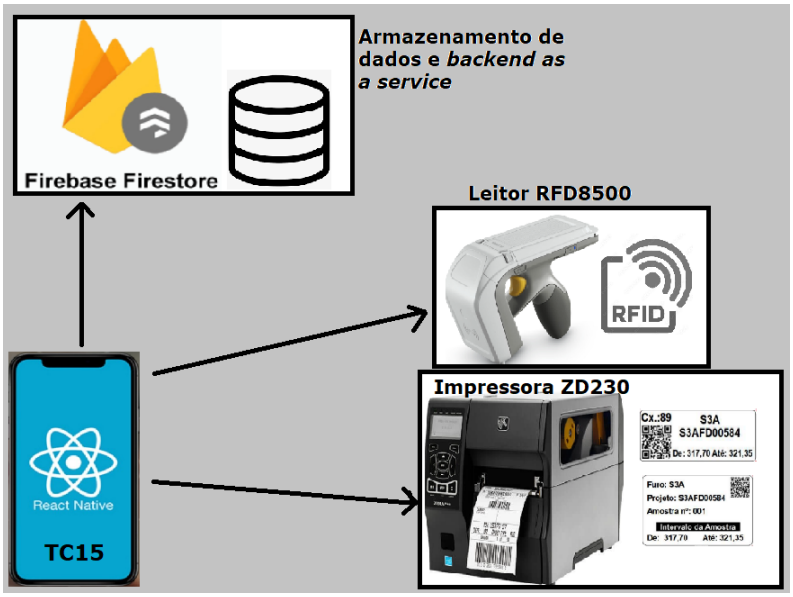
A aplicação a ser desenvolvida requer que os dados sejam armazenados de forma segura mesmo em situações em que a conexão de rede com o servidor seja perdida. Uma das motivações da opção pelo banco de dados do *Google Firebase Firestore* foi a possibilidade de operar requisições nele sem a necessidade de conexão com a internet e sem necessidade de se preocupar com o salvamento de dados localmente. Sua própria configuração no ambiente do *React Native* permite que os dados modificados permaneçam na memória, e ao obter conexão com a rede sejam sincronizados sem que qualquer informação seja perdida.

O *Firebase Firestore* armazena seus dados em um formato criptografado e em pastas próprias da aplicação, às quais o usuário não tem acesso, fazendo com que a proteção dos dados seja mantida e garantindo a sincronização ao conectar-se à internet.

O uso do *React Native* para desenvolvimento da aplicação justifica-se porque ele atende aos requisitos do projeto, além de possuir uma vasta comunidade de desenvolvedores que pode auxiliar em dúvidas pontuais. Vale citar também a grande quantidade de bibliotecas que abstraem certas funcionalidades desejadas dentro da aplicação. Dentre as utilizadas nesse projeto, pode-se citar bibliotecas de manipulação de arquivos do formato Excel (*input* ou ponto de entrada dos dados manipulados no sistema desenvolvido deste trabalho), até bibliotecas de manipulaçao de criação e leitura de arquivos dentro do dispositivo móvel, entre outras.

Foram utilizados ainda aparelhos disponibilizados pela empresa em que o autor realizou o estágio: Um dispositivo Zebra TC15, uma impressora Zebra ZD230 e um coletor RFD8500. O TC15 é um dispositivo móvel similar a um celular comum, porém com leitor de código de barras e QrCode embutido, que é acionado por um botão lateral extra, enquanto o RFD8500 é um leitor de tags RFID que dispara, através de um gatilho, ondas de radiofrequência que alimentam as tags passivas usadas no desenvolvimento da aplicação. A impressora é capaz de ler arquivos de texto com a linguagem "*zpl*" (Zebra Printer Language), que é a linguagem de instruções intrepretadas por impressoras Zebra. Nela, é possível ajustar um layout para etiquetas de referência para as amostras da aplicação com informações sobre elas, além de QrCodes únicos. A Figura 6 representa de maneira simplificada a integração dos elementos.

Figura 6 – Simplificação da arquitetura da aplicação.



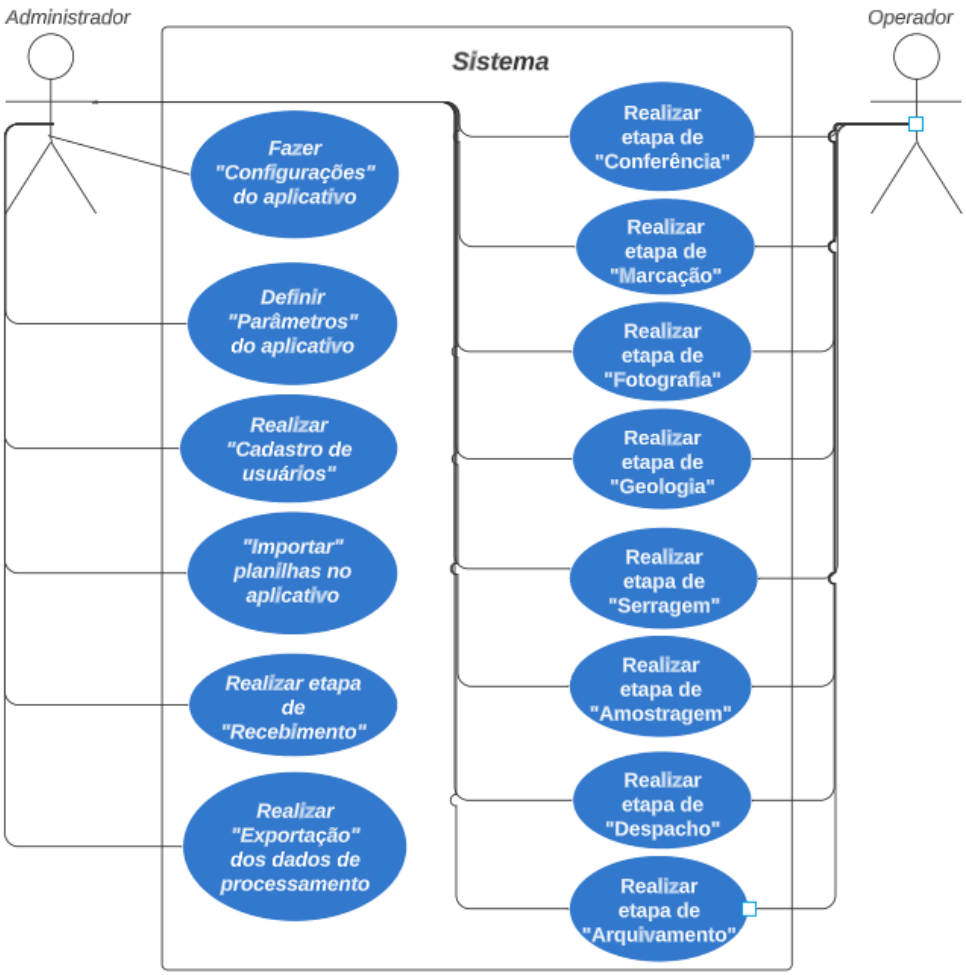
Fonte: Autor, 2023

A combinação das duas tecnologias citadas acima (*Firebase Firestore* e *React Native*) com os dispositivos da empresa Zebra para leitura de QrCodes, códigos de barras e tags RFID fez com que fosse possível o controle total dos dados das amostras de furo com segurança no tratamento de informações.

3.3 SOLUÇÃO PROPOSTA

A aplicação desenvolvida admite dois tipos de usuários: administradores e operadores. Os administradores possuem acesso em todas as funcionalidades da aplicação, com ênfase nas configurações gerias do aplicativo para utilização do usuário operador. O operador pode realizar apenas as etapas de processamento das caixas de amostras de furo, como mostra o diagrama de caso de uso na Figura 7.

Figura 7 – Diagrama de casos de uso da aplicação.



Fonte: Autor, 2023

O fluxo da aplicação inicia com o cadastro de usuários administradores ou usuários de operação por meio de uma conta *master* pré-cadastrada no banco. Usuários admi-

nistradores possuem acesso a todas as funcionalidades do aplicativo. As funcionalidades diferenciais que os administradores possuem incluem, além do cadastro de outros usuários, a configuração dos leitores Zebra (tanto coletor RFD8500, como impressora).

A entrada dos itens no banco de dados é feito através da importação de uma planilha eletrônica já presente no dispositivo. Depois de ser importada, a planilha é lida e tratada linha a linha. Suas informações são extraídas e salvas no banco de dados *Firebase Firestore*. Vale salientar que a planilha possui um formato padronizado pela empresa. Isso elimina a possibilidade de inserção errônea de informações.

Na Figura 8, têm-se o formato da planilha importada. A partir da linha 9, são coletadas as informações inseridas no banco de dados *Firebase Firestore*, onde cada linha representa uma amostra, e a planilha como um todo representa um furo realizado no solo. A coluna "A" é usado para coletar o nome do furo inserido, e o projeto, através dos três primeiros caracteres. A coluna "B" informa a sonda utilizada no processo de extração de cada amostra, a coluna "C" a data da extração, enquanto as colunas "D" e "E" informam a metragem da escavação.

Figura 8 – Modelo da planilha importada.

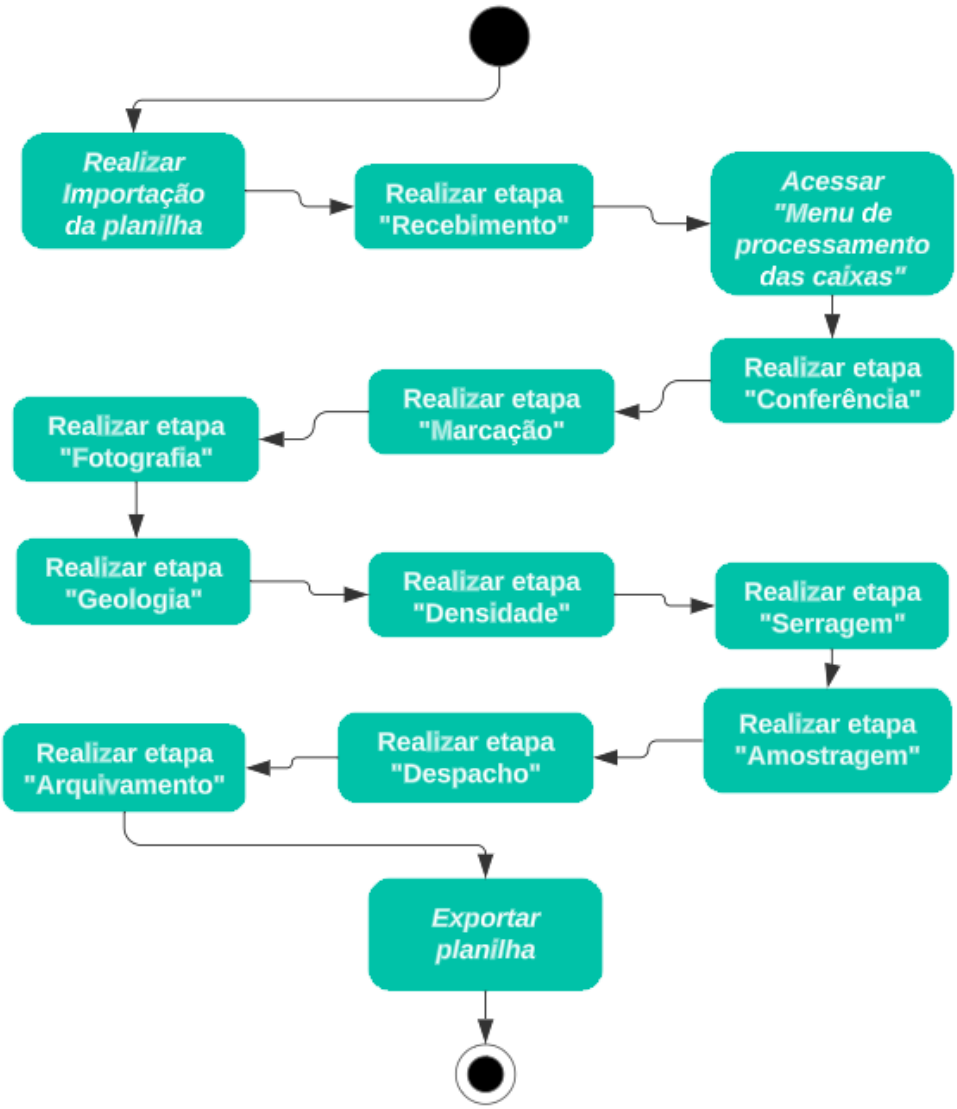
	A	B	C	D	E	F	G	H
1	Logo da empresa (não permitido a divulgação)	A12-NOME-FURO						Logo do projeto (não permitido divulgação)
2		Profundidade do Furo (m):						
3		Empresa:						
4	DIP PERFILAGEM INICIAL:							
5	DIP PERFILAGEM FINAL:						Informações extras (não permitido divulgação)	
6	Informações extras (não permitido divulgação)				0			
7					0			
8	FURO	SONDA	DATA	DE	ATÉ	Informações extras (não permitido divulgação)		
9	A12-NomeFuro	Sonda	26/02/2022	0,00	1,20			
10	A12-NomeFuro	Sonda	26/02/2022	1,20	2,40			
11	A12-NomeFuro	Sonda	26/02/2022	2,40	3,60			
12	A12-NomeFuro	Sonda	26/02/2022	3,60	5,10			
13	A12-NomeFuro	Sonda	26/02/2022	5,10	6,90			
14	A12-NomeFuro	Sonda	26/02/2022	6,90	8,40			

Fonte: Autor, 2023

O processamento de informações de cada amostra de furo é feito nas etapas de recebimento, conferência, marcação, fotografia, geologia, densidade, serragem, amostragem, despacho e arquivamento, conforme é demonstrado na Figura 9. Primeiramente, a etapa denominada de "recebimento" é efetuada, realizando a associação de cada amostra a sua etiqueta de informação. A etiqueta possui informações do número do furo realizado, informações da metragem da escavação que a amostra está contida, além de um QrCode que carrega essas informações no formato JSON.

Após isso, pode-se liberar as amostras conferidas para as demais etapas de processamento, realizadas por usuários de operação (normalmente, mas pode ser feito por contas adminstradoras). O primeiro de todos é a associação das etiquetas específicas com tags RFID passivas, fazendo com que ambas as tecnologias possam ser usadas em conjunto. O processamento é feito de maneira sequencial, ou seja, a segunda etapa precisa ser feita antes da terceira, e assim sucessivamente. O fluxo de processamento de uma amostra pode ser visto no diagrama de atividades da Figura 9, onde os atores de cada atividade são indicados no diagrama de casos de uso (Figura 7).

Figura 9 – Diagrama de atividades do processamento de amostras



Fonte: Autor, 2023

Ao final do processamento (terminando na etapa de arquivamento, com a paletização das amostras), um usuário administrador pode realizar a exportação dos dados

obtidos no formato de planilha novamente, contendo agora as informações de qual usuário realizou cada etapa, o tempo que cada uma levou e observação opcional.

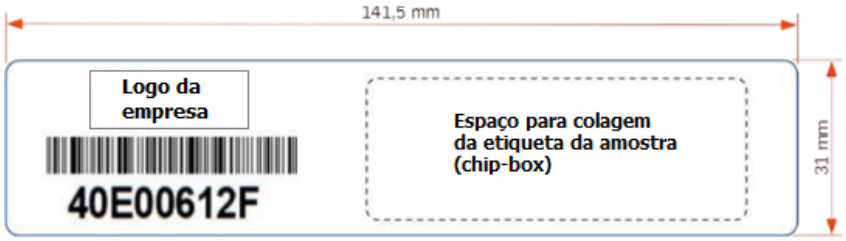
A seção 3.4 apresenta os dispositivos, hardwares e saídas físicas do sistema desenvolvido, para que fique claro as especificações técnicas da etiqueta RFID, etiqueta informativa (contendo os dados da amostra de solo) e disposivos utilizados.

3.4 DESCRIÇÃO TÉCNICA DOS HARDWARES E SAÍDAS FÍSICAS DO SISTEMA

Abaixo, são descritas de maneira mais técnica, a função e especificação dos harwares utilizados, bem como as saídas físicas do sistema através de impressoras.

- Dispositivo mobile Zebra TC15: dispositivo móvel com capacidade de acessar e coletar informações com facilidade por possuir um leitor integrado de QrCode através de um botão extra na lateral. Possui um *design* robusto com desempenho confiável para trabalhos nas instalações e em campo, com sistema operacional *Android* 11;
- Coletor de mão Zebra RFD8500: permite a leitura de tags RFID de baixas até altas frequências. Os leitores da série 8500 podem se conectar a dispositivos móveis, tablets e smartphones selecionados através de *bluetooth*;
- Impressora Térmica Zebra ZD230: permite a leitura de arquivos que possuam a linguagem de programação de impressoras Zebra (ZPL), facilitando manipulação e criação de arquivos para impressão. Possui opções de conectividade por *bluetooth*, *socket* ou por cabo;
- Tag RFID passiva: possui dimensão de 141,50 milímetros de largura, 31 milímetros de altura com 2,8 milímetros de espessura. Produzida com substrato de PVC rígido grau, possui durabilidade de até 10 anos ou mais, se acondicionada corretamente. É aplicada por fita dupla-face VHB. A retenção das informações no chip (EPC) será no formato de 24 caracteres com sequencial hexadecimal (exemplo: EAAC0020230023040E00612F), ilustrada na Figura 10;

Figura 10 – Modelo da tag RFID



Fonte: Autor, 2023

- Etiqueta das caixas de amostras (denominada também de "*chip-boxes")*: possui dimensão de 65 milímetros de largura e 25 milímetros de altura, sendo impressa na conferência das caixas do furo logo que chegam no local do processo. Deve ser colada sobre a tag RFID, e pode ser visualisada na Figura 11;

Figura 11 – Modelo da etiqueta das caixas do furo ("*chip-box*")

Fonte: Autor, 2023

- Etiqueta das amostras de uma caixa de furo (denominada também de "*sample-bag")*: possui dimensão de 80 milímetros de largura e 50 milímetros de altura, conforme mostra a Figura 12. São impressas na etapa de amostragem das caixas na fase de processamento, e devem ser coladas sobre a amostra retirada da caixa;

Figura 12 – Modelo da etiqueta da amostra de uma caixa ("*sample-bag*")

Fonte: Autor, 2023

- Etiqueta da caixa de transporte de amostras (denominada também de "*white-box")*: possui dimensão de 80 milímetros de largura e 50 milímetros de altura, sendo impressa na etapa de despacho das amostras de cada caixa na fase de

processamento. Representada na Figura 13, deve ser colada sobre a caixa que contém as amostras retiradas;

Figura 13 – Modelo da etiqueta da amostra de uma caixa ("*white-box*")

Fonte: Autor, 2023

- Etiqueta do palete gerado: possui dimensão de 80 milímetros de largura e 50 milímetros de altura, conforme mostra a Figura 14, sendo impressa na etapa de arquivamento das caixas na fase de processamento. Deve ser colada sobre os paletes, que contém 45 caixas.

Figura 14 – Modelo da etiqueta da amostra do palete

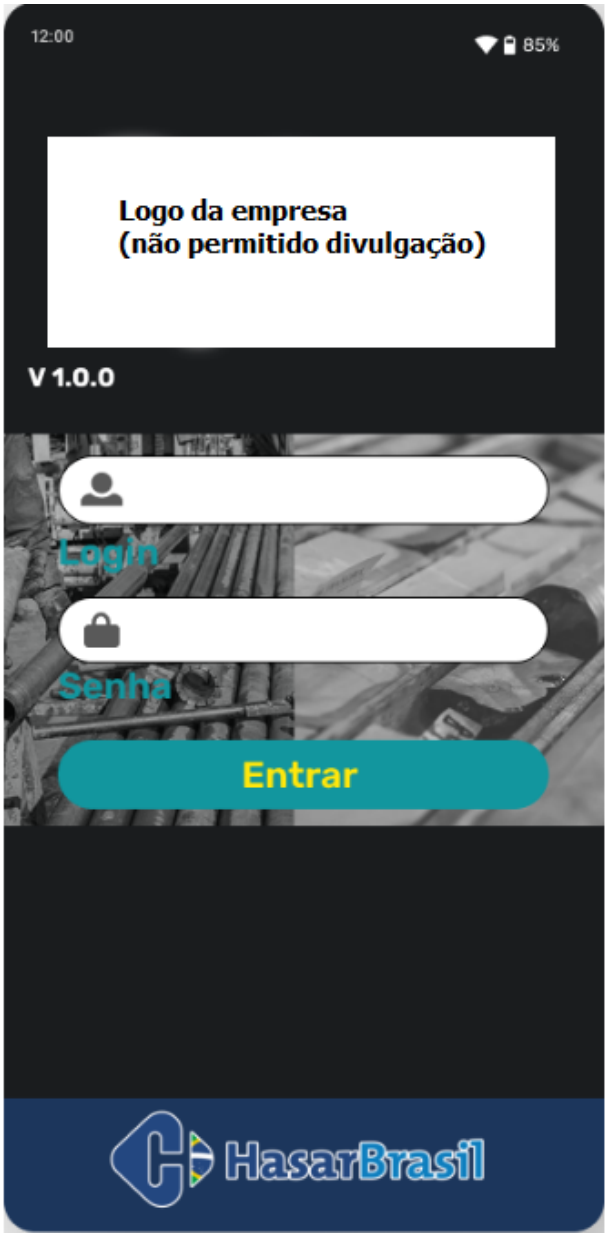
Fonte: Autor, 2023

3.5 DESENVOLVIMENTO

O sistema de cadastro de usuários tem como ponto de partida uma conta administradora pré-cadastrada no banco de dados, que possui acesso completo à solução, sendo capaz de cadastrar os demais usuários que a utilizarão. Os usuários administradores possuem acesso total ao sistema, enquanto que as permissões dos demais usuários são definidas quando estes são cadastrados pelo usuário administrador.

Ao abrir o aplicativo, o usuário acessa a tela representada pela Figura 15, em que deve fornecer senha e *email*:

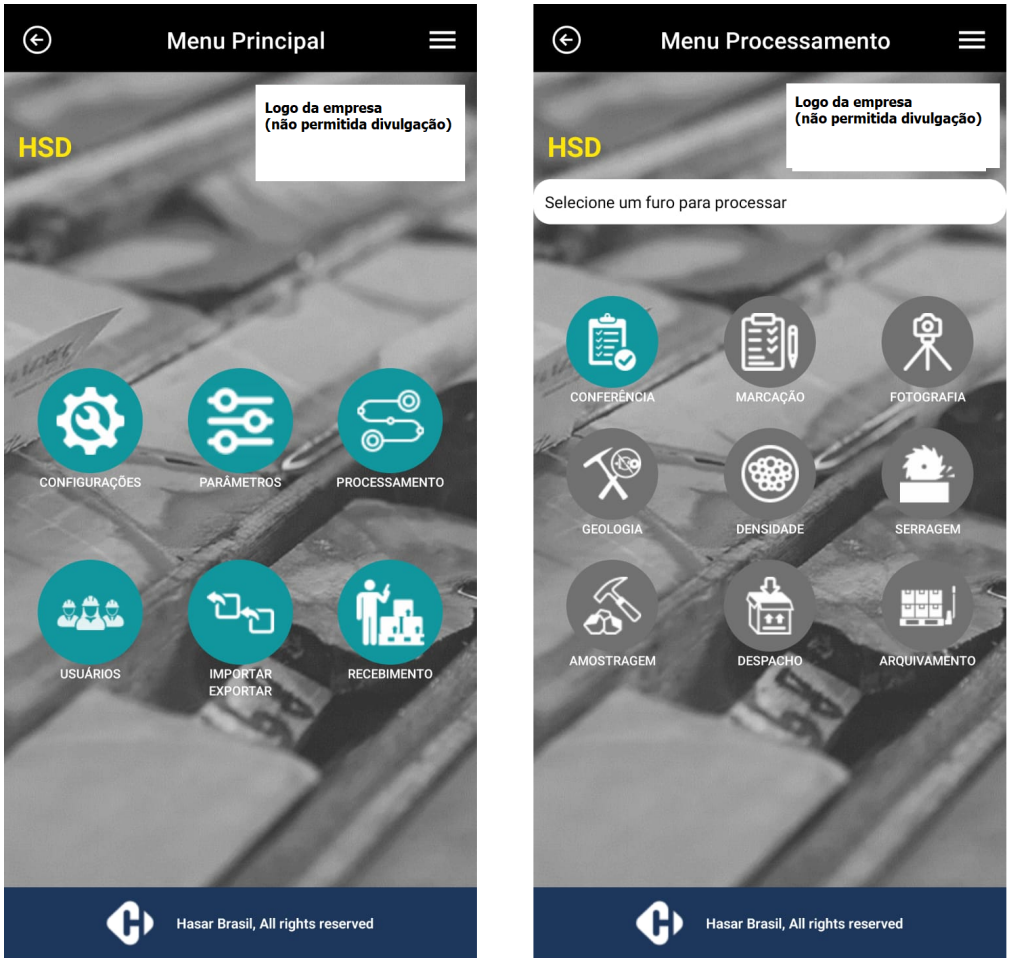
Figura 15 – Tela de login do sistema.



Fonte: Autor, 2023

Após o login, o sistema verifica se o usuário é administrador ou operador. Caso seja um usuário administrador, ele deverá cair no menu da esquerda da Figura 16, e caso seja um usuário operador, deve ser redirecionado para o menu da direita da Figura 16.

Figura 16 – Menu de administrador (esquerda) e menu de operadores (direita).



Fonte: Autor, 2023

3.5.1 Ajustes do administrador

Essa seção descreve as funções do menu de administrador. No botão “configurações“, é possível escolher e conectar o leitor RFD8500 pareado (por *bluetooth*) com o dispositivo que roda o aplicativo, bem como especificar o endereço IP e a porta da impressora ZD230 que está na rede. As demais informações são referentes ao IP da rede e verificação se o dispositivo está conectado ao WiFi, como pode ser observado na Figura 17.

Figura 17 – Tela de configuração.

Fonte: Autor, 2023

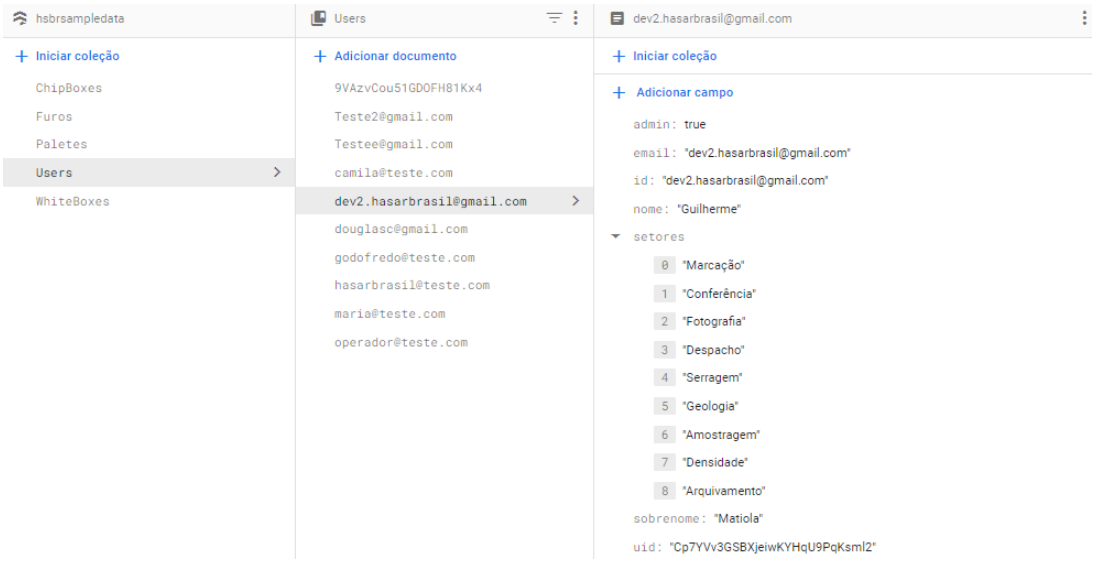
Na tela de parâmetros, o usuário administrador tem a opção de configurar a potência do leitor RFD8500, variando valores de 300 a 50 bdm, alterar o volume emitido pelo leitor ao realizar leituras e verificar a bateria. A opção de código de barras (*barcode* não foi implementada nessa versão do software, pois será utilizado o leitor sem código de barras). Tanto as configurações de dispositivos feitas na tela de configuração quanto na tela de parâmetros ficam armazenadas na memória do dispositivo. A configuração da potência do leitor é mais relevante para a tela "Procurar", que será apresentada posteriormente, em que é realizado uma busca de todas as tags cadastradas nas proximidades. Na Figura 18, têm-se a tela de parâmetros desenvolvida, assim como a de usuários, que já teve seu funcionamento explicado.

Figura 18 – Tela de definição de parâmetros e cadastro de usuários.

Fonte: Autor, 2023

Os usuários podem estar vinculados a um ou mais setores da estrutura organizacional da empresa, e cada setor possui acesso a uma etapa do processo. Usuários administradores possuem todos os setores marcados automaticamente, enquanto usuários de processamento poderão ter algumas funcionalidades apenas, ou todas. Na Figura 19, pode-se ver a estrutura dos usuários cadastrados no banco de dados.

Figura 19 – Usuários cadastrados no banco de dados.



Fonte: Autor, 2023

Na tela de importação e exportação, acontece boa parte da lógica inicial da entrada de dados da aplicação. Como comentado nos objetivos gerais e específicos deste documento, a solução desenvolvida possui como ponto inicial de entrada de dados uma planilha com formato pré-estabelecido. Esta padronização fez com que a consistência na entrada de dados fosse sempre respeitada. O modelo da planilha pode ser observado na Figura 20.

Figura 20 – Planilha de importação de dados.

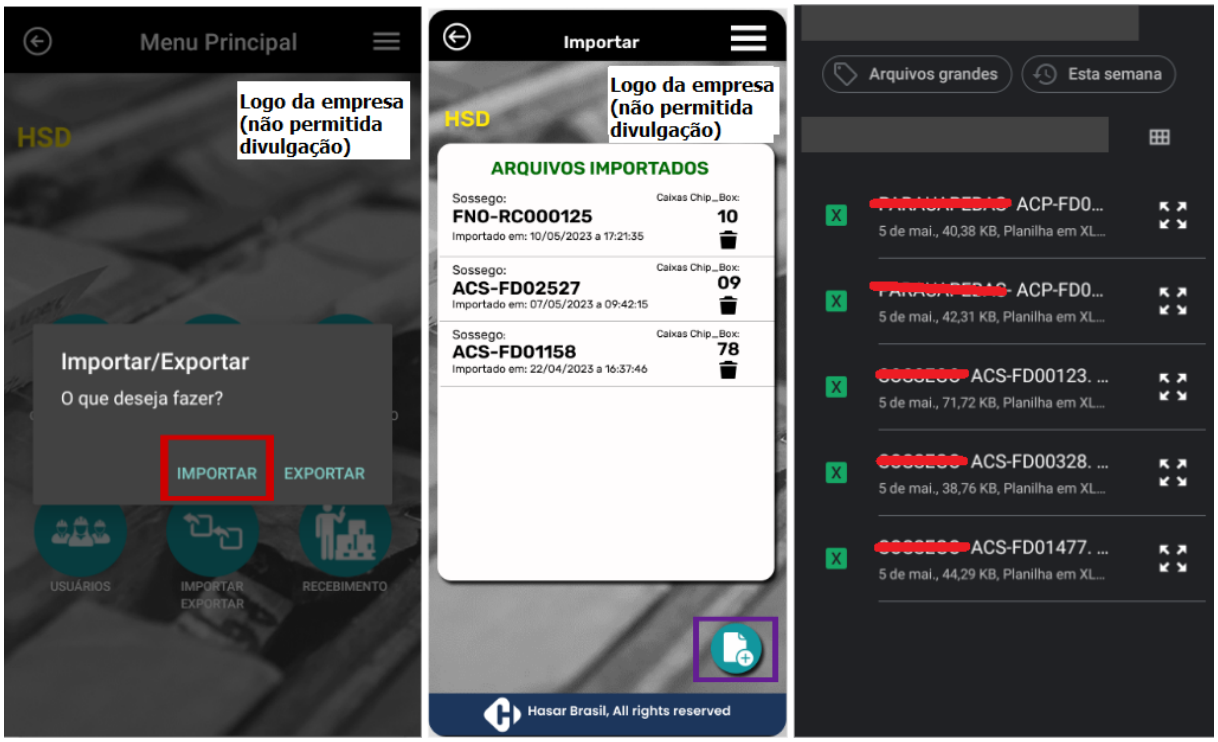
	A	B	C	D	E	F	G	H
1	Logo da empresa (não permitido a divulgação)	A12-NOME-FURO						Logo do projeto (não permitido divulgação)
2		Profundidade do Furo (m):						
3		Empresa:						
4	DIP PERFILAGEM INICIAL:							
5	DIP PERFILAGEM FINAL:						Informações extras (não permitido divulgação)	
6	Informações extras (não permitido divulgação)				0			
7					0			
8	FURO	SONDA	DATA	DE	ATÉ	Informações extras (não permitido divulgação)		
9	A12-NomeFuro	Sonda	26/02/2022	0,00	1,20			
10	A12-NomeFuro	Sonda	26/02/2022	1,20	2,40			
11	A12-NomeFuro	Sonda	26/02/2022	2,40	3,60			
12	A12-NomeFuro	Sonda	26/02/2022	3,60	5,10			
13	A12-NomeFuro	Sonda	26/02/2022	5,10	6,90			
14	A12-NomeFuro	Sonda	26/02/2022	6,90	8,40			

Fonte: Autor, 2023

Ao selecionar a opção de “Importar“, o administrador insere no sistema uma

planilha previamente armazenada no dispositivo no formato mostrado na Figura 20, e com o auxílio de bibliotecas do *React Native*, entre elas, a de selecionar qualquer documento do dispositivo (*Document Picker*). Após isso, o conteúdo da planilha é convertido para formato de arquivo JSON. Por fim, os dados podem ser tratados no código da aplicação. Tudo possível a biblioteca do *javascript* chamada "XLSX".

Figura 21 – Inserção de planilha na tela de importar.

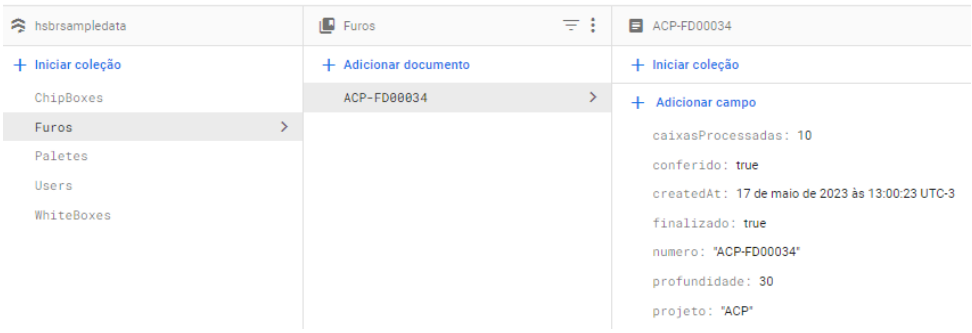


Fonte: Autor, 2023

Após a conversão dos dados do arquivo em um formato que pode ser inserido no banco de dados, é possível extrair as informações pertinentes para a aplicação. Observando a Figura 20, na célula "B2" (em amarelo), extrai-se o identificador do projeto (através dos 3 primeiros caracteres, na forma abreviada), e a partir do primeiro hífen, têm-se o nome do furo. Na linha 8, têm-se o cabeçalho dos dados das amostras do furo, e a partir da linha 9, os dados de cada amostra de furo, cujas informações contém o próprio furo, a sonda utilizada para escavação do solo, a data de extração, e a metragem do furo a partir da superfície (0 metros) informados com "de" e "até". Isso foi feito através de um *hook* personalizado no *React*.

As informações inseridas no banco de dados NoSQL *Firebase Firestore* são retiradas da planilha e geram duas coleções, a coleção "furos" e "$chip - boxes$". Cada documento da coleção furo possui os dados demonstrados na Figura 22, após exportação da planilha.

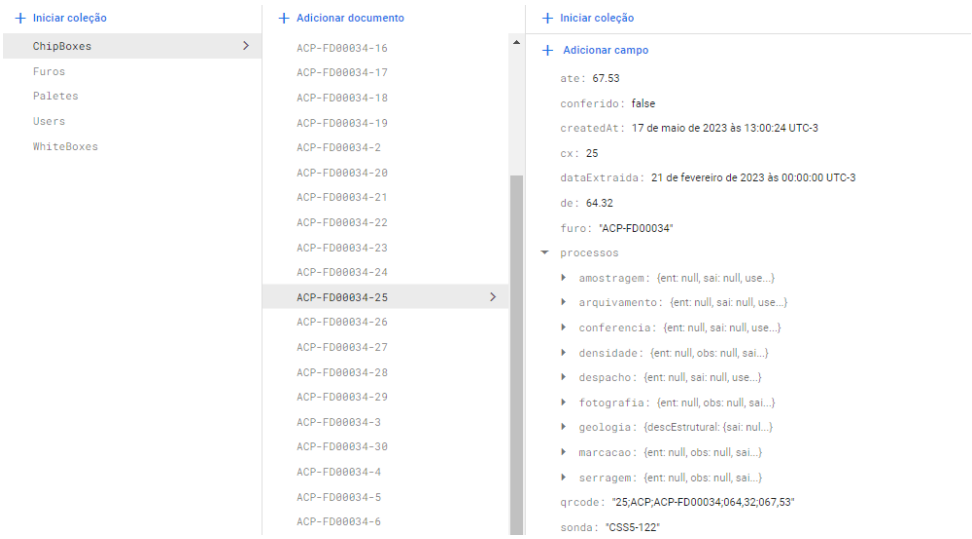
Figura 22 – Documento do furo ACP na coleção “Furos“ no banco de dados Firestore.



Fonte: Autor, 2023

Após inserir o documento referente ao furo na coleção “Furo“, o código segue iterando por todas as linhas que possuem preenchimento a partir da linha 9 da planilha, buscando as informações de cada célula, e adicionando-as como um documento da coleção “ChipBox“. Cada iteração do código adiciona um item no banco de dados. As caixas de amostra (“chip-boxes“) do furo possuem o formato no banco de dados conforme a Figura 23.

Figura 23 – Documento de caixas de amostra na coleção ChipBoxes no banco de dados Firestore.



Fonte: Autor, 2023

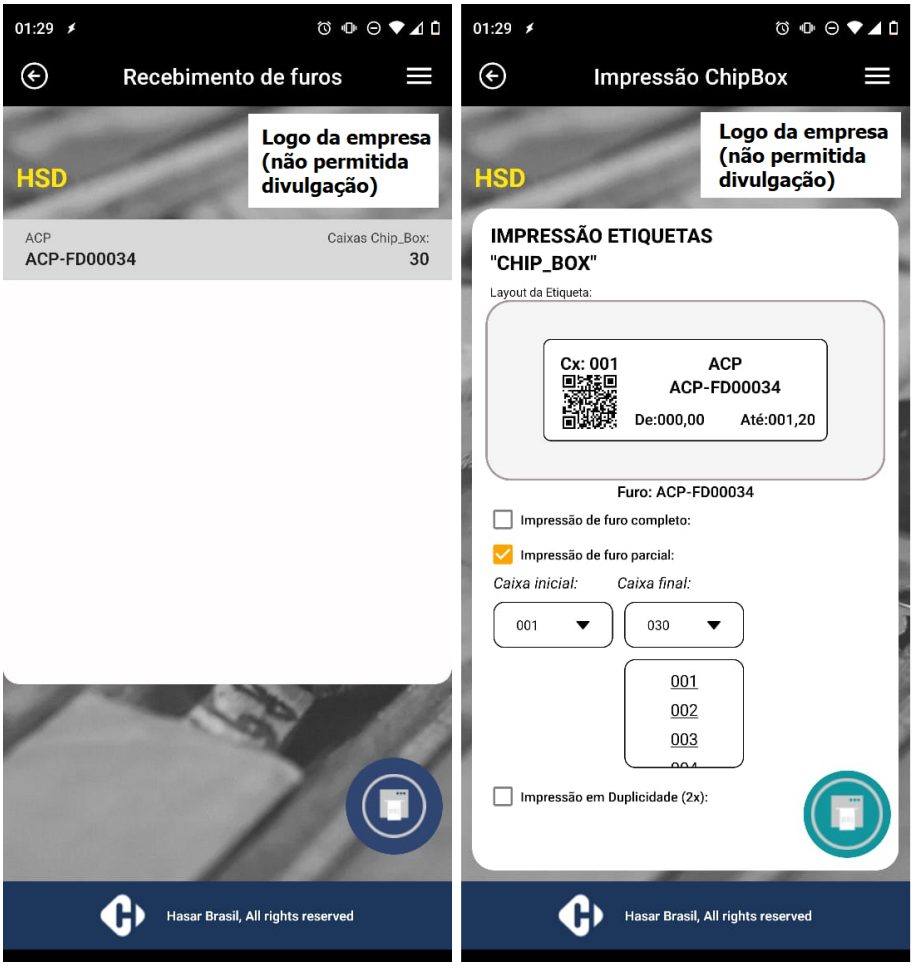
Percebe-se que todas as ações possuem registro de data e hora, assim como o usuário que a realizou. Esse é um dos principais requisitos do projeto, visto que será de grande utilidade para a empresa acompanhar o rendimento dos processos. Até esse momento, a aplicação oferece uma opção segura de registro dos dados, onde certamente todos os dados da planilha estarão corretamente inseridos no banco de dados, sem que haja erros

de digitação. Esse processo era feito manualmente após preenchimento da planilha em campo. Ainda assim, o principal ganho da aplicação é na fase de processamento, que será descrita adiante.

A exportação dos dados gerados sobre todas as amostras (principal saída do sistema) é permitida apenas após todas as caixas que passaram pelo processo de conferência terminem todas suas nove etapas de processamento. A planilha exportada contém todas as informações das etapas, porém será melhor explicada na sequência deste documento.

A outra etapa permitida pelo administrador é o recebimento. Depois de importar a planilha, o usuário aguarda o caminhão entregar as caixas com amostras do furo vindas da mina. Após descarregamento delas no galpão de processamento (local onde se utiliza a aplicação desenvolvida neste trabalho), o usuário seleciona o furo compatível com essas caixas, e realiza a impressão das etiquetas que serão anexadas em cada uma delas. A tela de impressão desenvolvida para este processo pode ser verificada a seguir.

Figura 24 – Recebimento das caixas do furo e impressão de etiquetas.



Fonte: Autor, 2023

As opções fornecidas são de imprimir todas as etiquetas do furo, ao selecionar a

opção de “Impressão de furo completo“, impressão de etiquetas individuais ou de forma ajustada na opção de “Impressão de furo parcial“, e pode-se ainda imprimir duas tags de cada com a opção de “imprimir em duplicidade“.

Com essa etapa de recebimento dos furos, finaliza-se o processo dos usuários administradores, em que pode ser feita toda a configuração do ambiente para o processamento feito nas etapas posteriores, que serão descritas na seção 3.5.2.

3.5.2 Processamento das caixas

O menu de processamento, tela em que o usuário operador é direcionado ao realizar autenticação, possui acesso também pelo administrador, conforme visto na Figura 16. O menu de processamento possui nove funcionalidades que correspondem às etapas em que as caixas geradas no processo de conferência são obrigadas a passar (para que seja possível gerar a planilha final de exportação). Essa é uma das regras de negócio do sistema para que o ciclo seja executado de forma completa, além de que a sequência precisa ser respeitada, ou seja, uma caixa não pode passar pelo processo de fotografia sem ter passado pelo processo de marcação, e assim sucessivamente. A sequência deve ser seguida conforme a Figura 25:

Figura 25 – Ordem dos processos das caixas.

Fonte: Autor, 2023

Vale ressaltar que, nesse menu de processamento, o operador deve selecionar, através de um menu *dropdown* qual furo deseja processar. Após isso, ao selecionar uma das opções do menu de processamento, é possível realizar a leitura do QrCode da etiqueta vinculada à caixa na etapa de recebimento. Caso o usuário tente ler uma caixa em um processo à frente do que a caixa realmente deveria estar, o sistema bloqueia o processamento da caixa com um alerta sonoro de erro, além de informar visualmente o com um círculo vermelho,

mostrando que a caixa ou não pertence ao furo selecionado para processamento, ou que está em um processo adiantado. Existem três tipos de alertas visuais e dois tipos de alertas sonoros que são emitidos ao ler o QrCode da caixa em cada etapa de processamento:

Figura 26 – Tipos de status na leitura do QrCode da caixa nas etapas de processamento.

Fonte: Autor, 2023

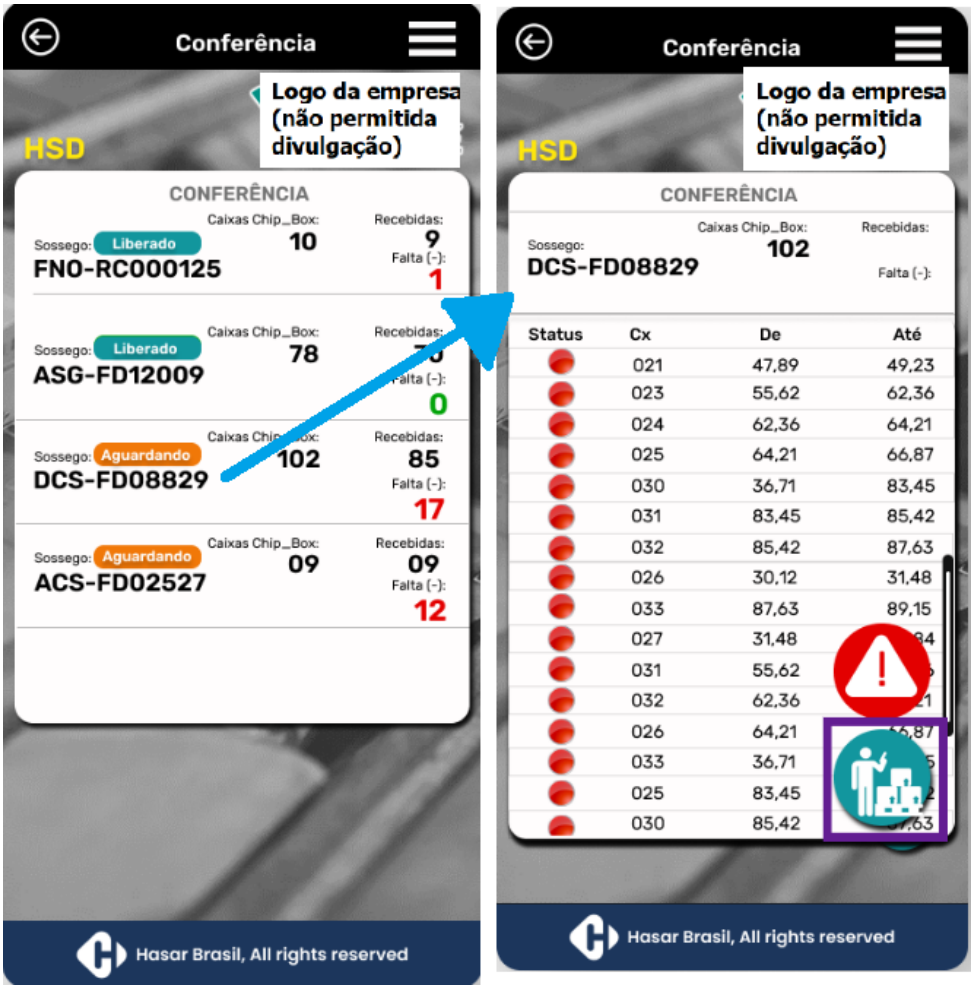
1. Alerta visual verde: através de um círculo verde e um efeito sonoro que remete acerto, mostra que a caixa já foi processada naquela etapa e já foi finalizada, e o sistema não reinicia seu processamento, pois uma vez finalizada, não deve-se sobreescrever os dados salvos.
2. Alerta visual amarelo: através de um círculo amarelo e um efeito sonoro que remete acerto, mostra que a caixa está pronta para realizar o processo específico em que ela foi lida, aparecendo sua opção de ação (seja fazendo uma observação ou liberando para impressão da etiqueta da etapa, se for o caso). O sistema dá início a este processo com um dado do tipo *timestamp* que informa a data e hora, além de salvar o usuário que iniciou o processo. Ao reler a mesma caixa no mesmo processo, a data de início não é alterada no banco de dados.
3. Alerta visual vermelho: através de um círculo vermelho e um efeito sonoro que remete erro, informa que a caixa está no processo errado, ou que o usuário selecionou o furo que desejava processar de forma incorreta. Uma informação aparece na tela no estilo *pop up* mostrando qual erro foi cometido, no caso de processo avançado ou de furo selecionado errado.

Cada etapa de processamento será apresentada nas seções a seguir, seguindo a ordem em que devem acontecer, começando por "Conferência".

3.5.2.1 Conferência

Na etapa de conferência, após a tag RFID e a etiqueta específica da caixa da amostra serem coladas, será feito a associaçao da tag com a etiqueta (casamento das informações, para que se torne basicamente um único elemento). O usuário seleciona o furo que deseja conferir, e é informado à ele quantas caixas o furo possui e quantas já foram conferidas. O botão em vermelho, visto na Figura 27, faz a liberação do furo mesmo que nem todas as caixas estejam conferidas para que o processo comece. O botão abaixo dele direciona para a tela de vinculação (denominada "Vincular Caixa") demonstrada pela Figura 27.

Figura 27 – Tela inicial de conferência.

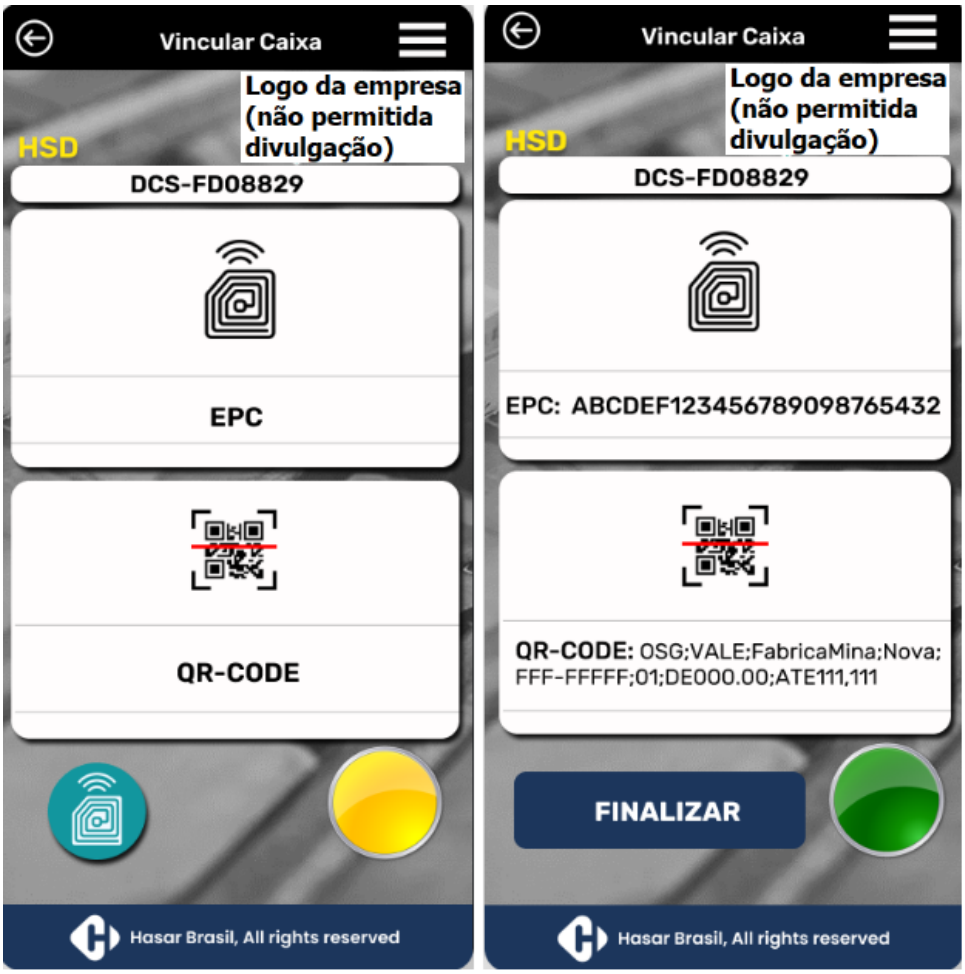


Fonte: Autor, 2023

Na tela "Vincular Caixa", onde o leitor RFID (conectado pelo menu de configuração)

estabelece a conexão e permite a leitura de tags. Ao acionar o botão do dispositivo TC15, é possível ler o QrCode da etiqueta colada (Figura 28).

Figura 28 – Tela de associação da tag RFId com o QrCode.



Fonte: Autor, 2023

Ao pressionar o botão “Finalizar“, a informação é consolidada no banco de dados.

3.5.2.2 Marcação

Ao selecionar a opção “marcação” destacada em vermelho na Figura 29, o usuário é direcionado para a tela onde deverá realizar a leitura (com o coletor de dados TC15) do QRCode da caixa. Caso o círculo fique amarelo, a marcação da caixa pode ser realizada, clicando no botão de observação (destacado em roxo). Se julgar necessário, pode fazer alguma observação, escrevendo no campo de escrita (*input*), e ao fim, salvar essa marcação (destacada em verde), conforme demonstrado na Figura 29.

Figura 29 – Processo de marcação.

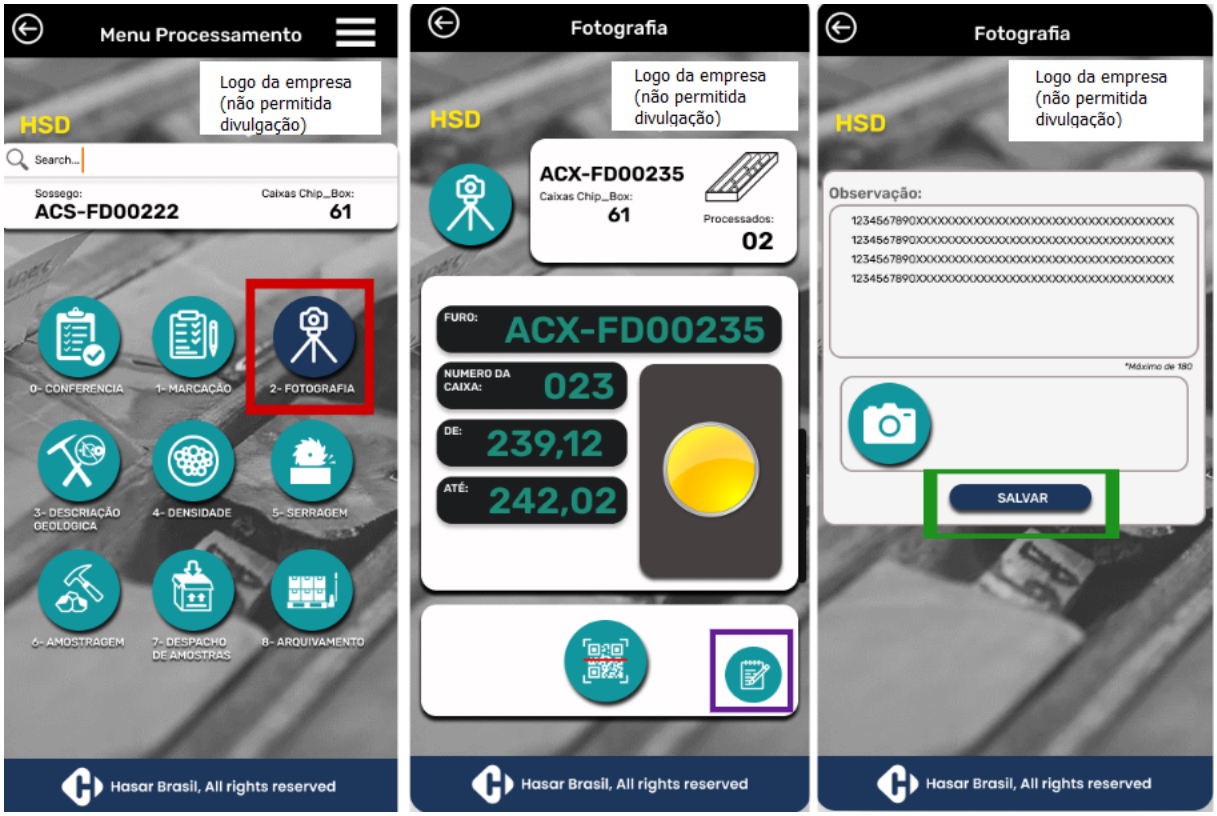
Fonte: Autor, 2023

Ao final da marcação, o sistema encaminha o usuário novamente para a tela de leitura do QrCode do processo de marcação. A informação de caixas processadas é atualizada, e mostra ao usuário a última caixa marcada através de uma busca sobre o campo de data e hora das "ChipBoxes" no banco de dados. A inserção de fotos através do ícone, presente na Figura 29, não foi implementada nessa fase do desenvolvimento da aplicação por opção da empresa.

3.5.2.3 Fotografia

O processo de fotografia é semelhante ao processo de marcação. Ao selecionar a opção de fotografia no menu, destacada em vermelho na Figura 30, o usuário é direcionado para a tela onde deverá realizar a leitura (com o coletor de dados TC15) do QRCode da caixa. Caso o círculo fique amarelo, a etapa de fotografia da caixa pode ser realizada, clicando no botão de observação (destacado em roxo). Se julgar necessário, pode fazer alguma observação, escrevendo no campo de escrita (*input*). Novamente, destaca-se que anexar uma foto ao processo não é necessário nos requisitos do projeto. O processo é finalizado com a opção "Salvar" (em verde, destacado na Figura 30).

Figura 30 – Processo de fotografia.



Fonte: Autor, 2023

3.5.2.4 Geologia

A etapa de descrição Geológica, na prática, é mais demorada. O processo deve ser feito por três geólogos, pois existem três etapas diferentes. As descrições geológicas, geotécnicas e estruturais deverão, obrigatoriamente serem preenchidas, selecionando o *checkbox* de concluído ao lado de cada uma delas.

A regra de negócio desenvolvida, seguindo a exigência da empresa, foi que, para realizar a descrição seguinte, é necessário ter realizado a descrição anterior. Elas não precisam ser preenchidas no mesmo momento, mas só serão liberadas dessa fase após sua total conclusão. Ao finalizar uma das etapas do processo de Geologia, já é salvo no banco a data e hora da realização, bem como o usuário que a fez. A regra para leitura da caixa segue o padrão dos processos anteriores, lendo o QrCode pelo TC15 (conforme demonstrado na Figura 31):

Figura 31 – Processo de geologia

Fonte: Autor, 2023

Como cada uma das três etapas de descrição geológica podem ser feitas por geólogos diferentes. A informação do usuário que a realizou é armazenada de forma individual, como demonstrado na Figura 32:

Figura 32 – Captação dos dados na processo de geologia.


```
geologia
  descEstrutural
    ent: 25 de maio de 2023 às 07:38:14 UTC-3
    obs: "Tudo ok"
    sai: 25 de maio de 2023 às 07:44:49 UTC-3
    user: "geologo1@gmail.com"
  descGeologica
    ent: 25 de maio de 2023 às 11:28:43 UTC-3
    obs: "descricao: metragem correta e formato condizente"
    sai: 25 de maio de 2023 às 13:40:35 UTC-3
    user: "geologo2@gmail.com"
  descGeotecnica
    ent: 26 de maio de 2023 às 10:41:02 UTC-3
    obs: "descricao geotecnica: condiz com o esperado "
    sai: 26 de maio de 2023 às 15:41:41 UTC-3
    user: "geologo3@gmail.com"
```


Fonte: Autor, 2023

3.5.2.5 Densidade

Para este processo, basta selecionar a opção “densidade“ (destacada em vermelho na Figura 33) no menu de processamento, direcionando para a tela onde deve-se realizar a leitura (com o coletor de dados TC15) do QrCode da caixa (o círculo ficará amarelo), em seguida clicar no botão de observação (destacada em roxo), se julgar necessário pode-se fazer alguma observação, e então salvar (destacada em verde).

Figura 33 – Processo de densidade.

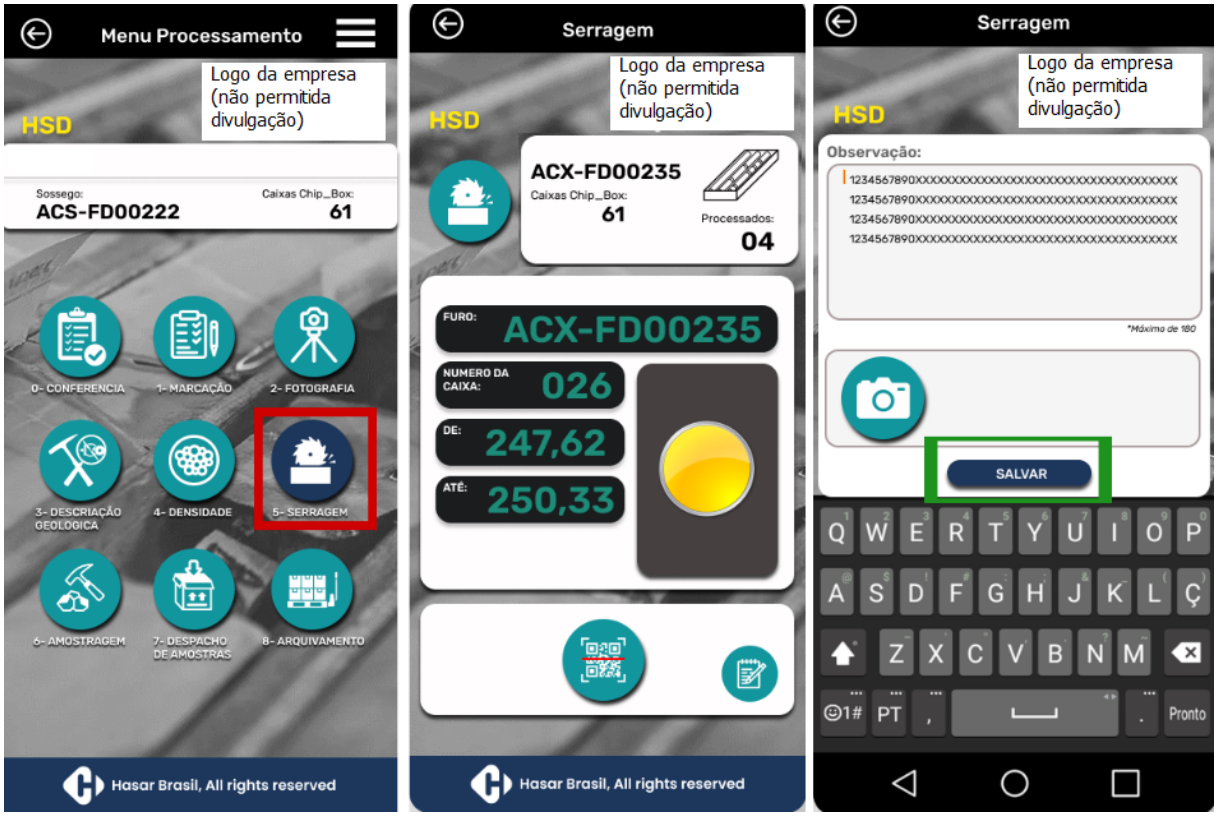
Fonte: Autor, 2023

3.5.2.6 Serragem

O processo de serragem também é demorado, pois nele é feito literalmente um corte no minério presente na caixa, para que posteriormente seja separada uma amostra para despacho, enquanto a maior parte do minério (restante da amostra) seja paletizada e guardada.

Para este processo, basta selecionar a opção “serragem“ (destacada em vermelho na Figura 34) no menu de processamento, direcionando para a tela onde deverá realizar a leitura do QrCode da caixa (o círculo ficará amarelo), em seguida clicar no botão de observação (destacada em roxo), se julgar necessário pode-se fazer alguma observação, e então salvar.

Figura 34 – Processo de serragem.



Fonte: Autor, 2023

3.5.2.7 Amostragem

Na etapa de amostragem, a parte de minério obtida no processo de serragem será colocada em uma saca (denominada de "*sample bag*") para controle da empresa contratante da solução. A parte maior do minério da caixa será colocada em uma caixa maior, denominada de "*white box*" no processo de despacho.

Neste processo de amostragem, o operador terá em sua frente a caixa e a amostra serrada no processo de serragem, e pode-se realizar a leitura de diversos códigos de caixas em sequência, ou seja, podem ser processadas várias caixas ao mesmo tempo, pois é um processo simples de ensacamento.

Após a leitura dos QrCodes, a opção de impressão é liberada na parte de baixo da aplicação, direcionando para a tela de impressão da(s) etiquetas da(s)"*sample bag*", conforme mostra a Figura 35, lembrando que cada saca receberá uma amostra e uma etiqueta.

Figura 35 – Processo de amostragem

Fonte: Autor, 2023

3.5.2.8 Despacho

Conforme comentado no processo acima, o despacho é um processo em que coloca-se a amostra do minério extraída em uma caixa maior ("*white box*"), no qual o operador deve decidir quantas sacas (*sample bags*) serão colocadas em cada uma através de um *pop up*, onde podem ser selecionados valores de 5 em 5 unidades, ou digitar um valor específico através do teclado.

A possibilidade da impressão da etiqueta da *white box* é liberada somente quando o operador realizar a leitura da quantidade estipulada por ele no *pop up*, e para isso, um contador é gerado na tela. Após a leitura do(s) QrCode(s), a opção de impressão é liberada na parte de baixo da aplicação, direcionando para a tela de impressão das etiquetas das"*sample bag*", conforme mostra a Figura 36.

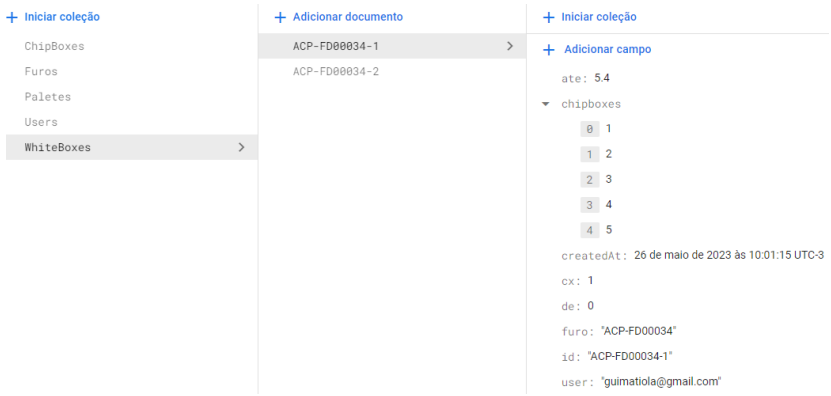
Figura 36 – Processo de despacho

Fonte: Autor, 2023

Após a impressão da etiqueta e consolidação das informações da caixa, uma coleção é gerada no banco de dados, conforme a Figura 37, com as seguintes informações:

- de: indica a informação de início da profundidade de extração (“de”) da caixa de menor valor (no exemplo abaixo, o da caixa 1);
- ate: indica a informação do fim da profundidade de extração (“até”) da caixa de maior valor (no exemplo abaixo, o da caixa 5);
- furo: indica a qual furo pertencem as caixas presentes na whitebox.;
- user: indica o usuário que efetuou o processo de despacho;
- createdAt: indica a data e horário que foi efetuado o processo de despacho;
- chipBoxes: Possui uma lista das caixas presentes na whitebox;
- cx: informação do número da whitebox (podem ter várias do mesmo furo);
- id: informação única do documento, formado pelo número do furo concatenado com o número da *white box* gerada.

Figura 37 – Coleção das *WhiteBoxes* no banco de dados Firebase



Fonte: Autor, 2023

3.5.2.9 Arquivamento

O processo final das caixas do furo é o arquivamento, que nada mais é que a paletização da porção da caixa de minério que não foi removida na etapa de amostragem. O palete pode ter até 45 caixas de minério dentro. O operador deve ler uma quantidade de caixas de no mínimo 1 a, no máximo, 45. Para auxiliá-lo, um contador foi adicionado na tela, conforme demonstrado na Figura 38.

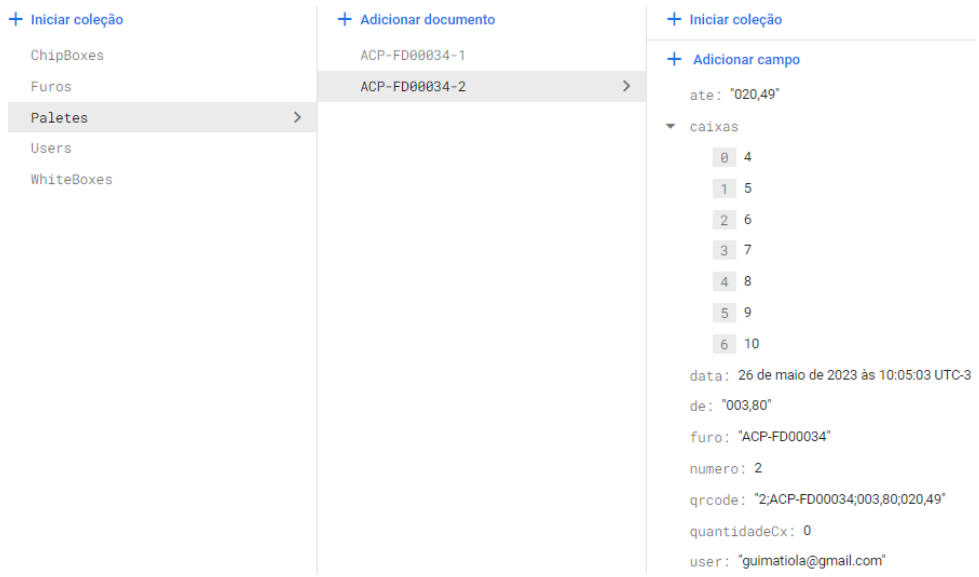
Figura 38 – Processo de arquivamento (paletização)

Fonte: Autor, 2023

Após a impressão da etiqueta e consolidação das informações do palete, a seguinte coleção é gerada no banco (Figura 39), as informações do mesmo:

- de: indica a informação de início da profundidade de extração (“de”) da caixa de menor valor (no exemplo abaixo, o da caixa 4);
- ate: indica a informação do fim da profundidade de extração (“até”) da caixa de maior valor (no exemplo abaixo, o da caixa 10);
- furo: indica a qual furo pertencem as caixas presentes na whitebox.;
- user: indica o usuário que efetuou o processo de despacho;
- createdAt: indica a data e horário que foi efetuado o processo de despacho;
- caixas: Possui uma lista das caixas presentes no palete;
- qrcode: código QR que possui todas as informações presentes no documento do palete;
- número: informação do número do palete gerado.

Figura 39 – Coleção do palete gerado no banco de dados Firebase



Fonte: Autor, 2023

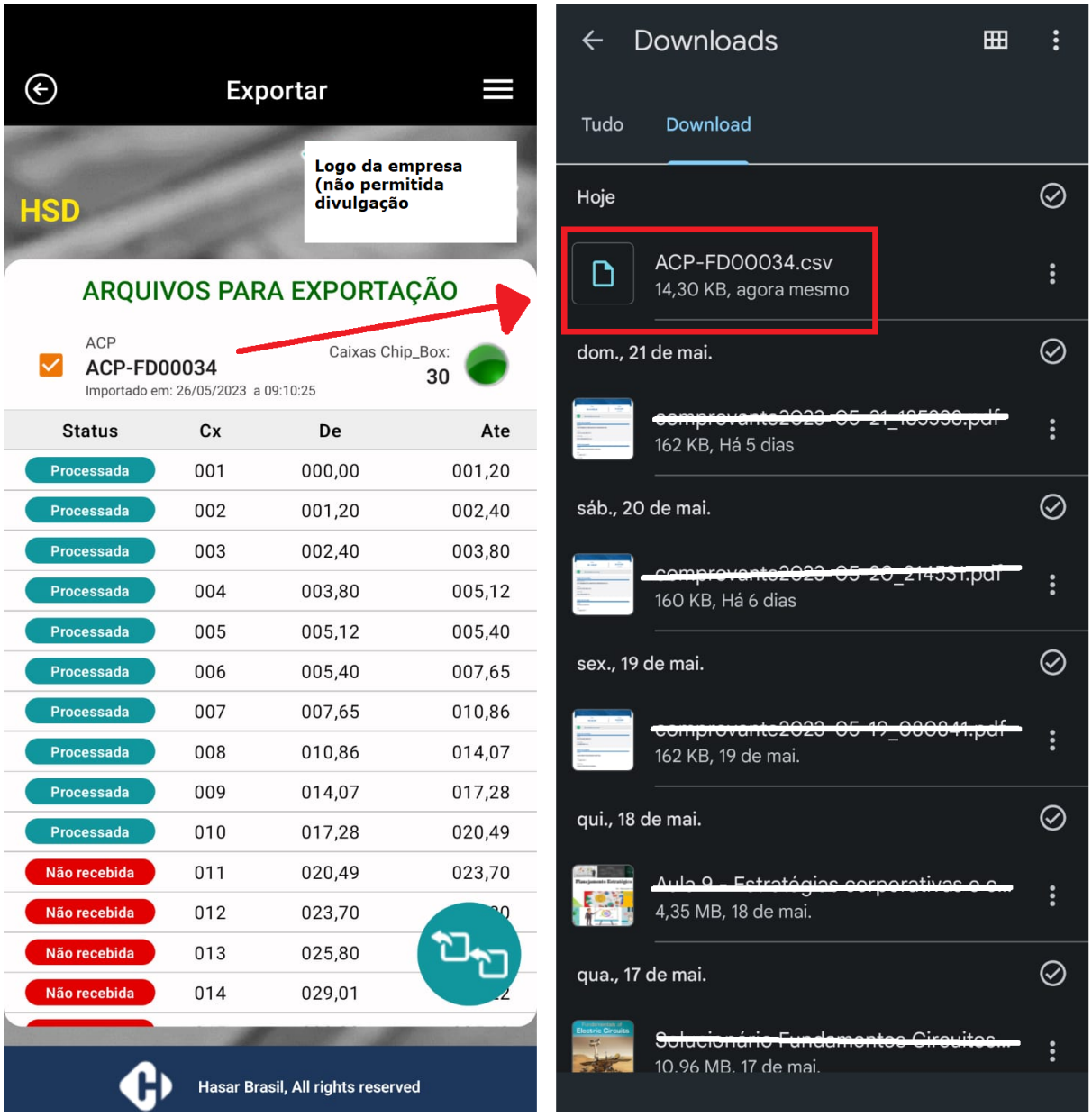
3.5.3 Exportação final das informações

Ao realizar todas as etapas de processamento, automaticamente o sistema altera a informação contida no furo “$finalizado$” de “$false$” para “$true$”, e ao realizar essa ação, a planilha dos dados processados pode ser finalmente exportada pelo administrador.

A planilha exportada é a saída final do sistema, pois posteriormente, a empresa contratante usará os dados fornecidos para controle e gestão de rendimento da operação,

bem como dos funcionários, visto que todas essas informações constam na planilha de dados. No menu, ao selecionar a opção de exportar, o administrador seleciona o furo cujos dados deseja exportar, e pressiona o botão no canto inferior direito na Figura 40. Após isso, o arquivo estará disponível na pasta "*downloads*" do dispositivo:

Figura 40 – Exportação das informações de processamento



Fonte: Autor, 2023

A lógica para geração da planilha acontece quando o usuário seleciona o *checkbox*, conforme a Figura 40. Ao pressionar o botão de exportar, é feita uma busca por todas as caixas que pertencem ao furo selecionado. Através de um *loop*, percorre-se por todas as caixas, lendo os dados de cada uma, e escrevendo uma *string* separada por pontos e

vírgula (;), que é o formato *Comma-separated values* (csv). Ao final da execução, salva o arquivo na pasta downloads. O arquivo final gerado tem o formato demonstrado na Figura 41:

Figura 41 – Planilha exportada

	A	B	C	D	E	F	G	H	I	J
1	Furo	Caixa	Epc	De	Ate	data_extraida	Conf_Ini	Conf_Fim	Conf_User	Marc_Ini
2	ACP-FD00034	1	EAAC020221037F0225503725	0	1,2	02/02/2023	26/05/2023 09:12:06	26/05/2023 09:22:41	hasarbrasil@teste.com	26/05/2023 09:28:50
3	ACP-FD00034	2	EAAC020221067F0225506743	1,2	2,4	03/02/2023	26/05/2023 09:23:38	26/05/2023 09:23:40	hasarbrasil@teste.com	26/05/2023 09:28:39
4	ACP-FD00034	3	EAAC020221047F022550472F	2,4	3,8	04/02/2023	26/05/2023 09:23:49	26/05/2023 09:23:59	hasarbrasil@teste.com	26/05/2023 09:28:29
5	ACP-FD00034	4	EAAC020221031F022550311F	3,8	5,12	04/02/2023	26/05/2023 09:24:08	26/05/2023 09:24:28	hasarbrasil@teste.com	26/05/2023 09:28:13
6	ACP-FD00034	5	EAAC020221079F022550794F	5,12	5,4	04/02/2023	26/05/2023 09:24:36	26/05/2023 09:24:49	hasarbrasil@teste.com	26/05/2023 09:28:03
7	ACP-FD00034	6	EAAC020221049F0225504931	5,4	7,65	04/02/2023	26/05/2023 09:25:39	26/05/2023 09:25:48	hasarbrasil@teste.com	26/05/2023 09:27:32
8	ACP-FD00034	7	EAAC020221056F0225505638	7,65	10,86	04/02/2023	26/05/2023 09:25:55	26/05/2023 09:26:03	hasarbrasil@teste.com	26/05/2023 09:27:23
9	ACP-FD00034	8	EAAC020221027F022550271B	10,86	14,07	04/02/2023	26/05/2023 09:26:11	26/05/2023 09:26:19	hasarbrasil@teste.com	26/05/2023 09:27:10
10	ACP-FD00034	9	EAAC020221093F022550935D	14,07	17,28	04/02/2023	26/05/2023 09:25:00	26/05/2023 09:25:13	hasarbrasil@teste.com	26/05/2023 09:27:55
11	ACP-FD00034	10	EAAC020221050F0225505032	17,28	20,49	04/02/2023	26/05/2023 09:25:22	26/05/2023 09:25:32	hasarbrasil@teste.com	26/05/2023 09:27:44

Fonte: Autor, 2023

Não é viável colocar imagens de toda a planilha preenchida, visto que ela possui 46 colunas. Contudo, pela Figura 41, é possível observar o formato que ela tem, mostrando todas as informações pertinentes a cada processo.

3.5.4 Banco de dados de desenvolvimento e produção

Para proporcionar flexibilidade no ambiente de desenvolvimento e produção, foram criados comandos distintos para executar o aplicativo em diferentes modos, o de *desenvolvimento* e o de *produção*.

Quando se inicializa o servidor da aplicação desenvolvida em *React Native* localmente, deve-se rodar um comando no terminal do computador. O arquivo de configuração do banco de dados *Firebase Firestore* é inserido dentro de uma pasta chamada *Android* dentro da pasta do projeto, e como a ideia é trabalhar com banco de dados diferentes, gerou-se então dois arquivos. Com isso, desenvolveu-se dois *scripts* personalizados que executam após diferentes comandos de terminal, onde um copia o arquivo do banco de desenvolvimento para a pasta *Android*, enquanto o outro copia o arquivo do banco de produção. Esses comandos permitem alternar entre as bases de dados sem comprometer sua integridade. Após essa troca de arquivos acontecer, o projeto é executado em modo de desenvolvimento (localmente).

Enquanto o comando para executar em modo de desenvolvimento permite testar e depurar o aplicativo com o banco de dados de teste, o comando para executar em modo de produção gera o arquivo de instalação (com extensão .apk) e permite a instalação do aplicativo com o banco de dados do cliente. Essa abordagem garante uma transição automatizada entre os ambientes de desenvolvimento e produção, mantendo a integridade dos dados e facilitando a manutenção do aplicativo. Ao executar o comando que conecta à base de produção, é exigido autenticação do usuário.

Como a conexão do *Firebase Firestore*, em aplicativos móveis, é feita através de um arquivo em formato JSON, não foi possível usar técnicas conhecidas de variáveis de ambiente. A solução proposta foi criar os dois bancos *Firebase*, obter os dois arquivos de conexão, e ao executar uma linha de comando específica, copiar o arquivo de conexão do banco para então compilar o aplicativo.

4 CONCLUSÃO

Neste trabalho, foi desenvolvido um aplicativo utilizando a tecnologia React Native com o objetivo de solucionar problemas no fluxo do processo de tratamento de dados de uma empresa extratora de amostras de solo. O aplicativo foi desenvolvido durante o estágio do autor na Hasar Brasil, buscando melhorar a eficiência e precisão do processo de análise de dados, além de automatizar tarefas manuais que aumentavam a possibilidade de erros humanos e falhas no preenchimento, contagem e organização do estoque.

O objetivo geral do aplicativo era permitir o tratamento de dados importantes com a maior precisão e cuidado possível, uma vez que eles representam o patrimônio mais valioso da empresa. Para alcançar esse objetivo geral, foram definidos objetivos específicos que abordaram diferentes aspectos do processo, como a inserção direta de dados provenientes de planilhas no banco de dados, o detalhamento e inspeção simplificados das amostras de solo, a associação de cada amostra a uma tag RFID única, a organização das amostras em diferentes níveis (amostras, caixas e paletes) e o controle de tempo e usuários responsáveis por cada processo realizado em cada amostra.

Através do desenvolvimento do aplicativo, foi possível atender todos os objetivos estabelecidos. A inserção direta de dados no banco de dados eliminou as divergências de valores e reduziu a possibilidade de erros no processo. A simplificação do detalhamento e inspeção das amostras tornou o processo mais eficaz e acessível, mesmo para usuários com poucos conhecimentos em tecnologia. A associação das amostras a tags RFID únicas permitiu uma busca prática e eficiente das amostras em outra solução implementada pela Hasar Brasil, denominada de inventário, buscando os itens que contém os “epcs” capturados pelo leitor RFID. A organização das amostras em diferentes níveis e o controle de tempo e usuários garantiram uma gestão mais eficiente do estoque e dos processos realizados em cada amostra.

Com a utilização do aplicativo desenvolvido, a empresa foi capaz de obter ganhos em termos de precisão, velocidade de execução e praticidade. A automação dos processos de análise de dados, aliada ao uso da tecnologia RFID, trouxe benefícios como redução de erros, aumento da eficiência, rastreamento preciso de produtos e melhoria da segurança. Além disso, a solução desenvolvida permitiu a exportação das informações pertinentes em formato de planilha para facilitar a leitura e análise dos dados.

Em conclusão, o desenvolvimento do aplicativo utilizando React Native atendeu aos objetivos propostos, proporcionando uma solução eficiente para a empresa extratora de amostras de solo. A automatização dos processos de tratamento de dados, aliada à tecnologia RFID, trouxe melhorias significativas em termos de produtividade, eficiência, precisão e segurança. A aplicação desenvolvida representa uma solução prática e eficaz para empresas que desejam melhorar seus processos de análise de dados e aumentar sua eficiência operacional.

REFERÊNCIAS

ALI, M.; KHAN, S. U.; ISLAM, S. RFID technology: A review of its applications, advantages, limitations and challenges. **Journal of King Saud University-Computer and Information Sciences**, v. 27, n. 4, p. 317–329, 2015.

BANKS, Alex; PORCELLO, Eve. **Learning React: Modern Patterns for Developing React Apps**. 2nd. [*S.l.*]: O'Reilly Media, 2021. P. 276. https://ebin.pub/learning-react-modern-patterns-for-developing-react-apps-2nbsped-1492051721-9781492051725.html. Acessado em 19º de março de 2023. ISBN 9781492051725.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Sistemas Distribuídos: Conceitos e Projeto**. 4th. Porto Alegre: Bookman, 2011.

CROCKFORD, Douglas. **JavaScript: The Good Parts**. [*S.l.*]: O'Reilly Media, Inc., 2008. https://www.abebooks.com/JavaScript-Good-Parts-Douglas-Crockford-OReilly/30109993546/bd. Acessado em 18º de março de 2023.

DUCKETT, Jon. **JavaScript and JQuery: Interactive Front-End Web Development**. [*S.l.*]: Wiley, 2014. https://doceru.com/doc/cx1ccn. Acessado em 17º de março de 2023.

FINKENZELLER, KLAUS. **RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication**. [*S.l.*]: John Wiley Sons, 2010.

FIREBASE. **Firebase**. [*S.l.*: *s.n.*], 2018. https://firebase.google.com/docs/. Acessado em 2023.

FLANAGAN, David. **JavaScript: The Definitive Guide**. [*S.l.*]: O'Reilly Media, Inc., 2011. https://pepa.holla.cz/wp-content/uploads/2016/08/JavaScript-The-Definitive-Guide-6th-Edition.pdf. Acessado em 1º de fevereiro de 2023.

FOWLER, Martin; SADALAGE, Pramod J. **NoSQL Essencial – Um Guia Conciso Para o Mundo Emergente da Persistência Poliglota**. [*S.l.*]: Novatec Editora, 2013. ISBN 978-85-7522-338-3.

GITHUB. **Octoverse 2022**. [*S.l.*: *s.n.*], 2022. Disponível em: https://octoverse.github.com/2022/top-programming-languages.

KORTH, Henry F.; SILBERSCHATZ, Abraham; SUDARSHAN, S. **Sistema de Banco de Dados**. 5th. São Paulo: Pearson, 2005.

LIKOVYI, Vladyslav. React Native Performance: Tips and Tricks. **Medium**, 2018. Acesso em 19 de março de 2023.

REACT DOCUMENTATION. **React - A JavaScript library for building user interfaces**. [*S.l.*: *s.n.*], 2023. https://reactjs.org/. Acesso em 12 de abril de 2023.

ROBBINS, Jennifer. **Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics**. 4th. Sebastopol, CA: O'Reilly Media, 2018.

SANTOS, Shalton Viana dos. **RFID: conceitos, implementação e desempenho com baixo custo computacional**. [*S.l.*]: Editora Dialética, 2022. P. 92. Acessado em 26 de maio de 2023.

SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGNE, Greg. **Fundamentos de Sistemas Operacionais**. 8th. São Paulo: LTC, 2013.

SRIPARASA, Sai Srinivas. **JavaScript and JSON Essentials**. [*S.l.*]: Packt Publishing Ltd, 2013.

TANENBAUM, A. S.; STEEN, M. V. **Sistemas Distribuídos: Princípios e Paradigmas**. 2. ed. São Paulo: Pearson, 2007.

UFRJ. **RFID: conceitos, aplicações e perspectivas**. [*S.l.*: *s.n.*], [s.d.] [S.l.: s.n.] Disponível em: https://www.gta.ufrj.br/grad/07_1/rfid/RFID_arquivos/Index.htm. Acesso em: 20 mar. 2023.

WANT, Roy; PERING, Trevor. The RFID Roadmap. **IEEE Pervasive Computing**, v. 5, n. 1, p. 26–33, 2006.

ZEBRA (ZEBRA TECHNOLOGIES CORPORATION). **Zebra**. [*S.l.*: *s.n.*]. [Online]. Available: https://www.zebra.com/. 20 mar. 2023.