UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

ÉVERTON PAULOUSKI

ROUTINE CHECKLIST: UM SISTEMA PARA ORGANIZAÇÃO E AUDITORIA DE ROTINAS EM SUPERMERCADOS

GUARAPUAVA

2024

ÉVERTON PAULOUSKI

ROUTINE CHECKLIST: UM SISTEMA PARA ORGANIZAÇÃO E AUDITORIA DE ROTINAS EM SUPERMERCADOS

Routine Checklist: A system for organization and audit of supermarket routines

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Dr. Roni Fabio Banaszewski

GUARAPUAVA

2024

ÉVERTON PAULOUSKI

ROUTINE CHECKLIST: UM SISTEMA PARA ORGANIZAÇÃO E AUDITORIA DE ROTINAS EM SUPERMERCADOS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 05/julho/2024

Prof. Roni Fabio Banaszewski
Doutor
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

---

Prof. Diego Marczal
Doutor
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

---

Prof. Renata Luiza Stange Carneiro Gomes
Doutora
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

GUARAPUAVA

2024

Esta dedicação é para Luciana, minha esposa, cujo apoio incansável foi fundamental para o sucesso deste projeto. Sua presença constante e amor genuíno foram a luz que guiou meu caminho, mesmo nos momentos mais desafiadores. Obrigado por estar ao meu lado e por todo o amor que compartilhamos.

AGRADECIMENTOS

Primeiramente, expresso minha gratidão a Deus por me guiar e fortalecer minha determinação ao longo desta jornada.

Quero estender meu sincero agradecimento ao meu orientador, o Professor Dr. Roni Fabio Banaszewski, por sua aceitação deste projeto e por sua orientação constante que foi fundamental para sua conclusão.

Aos estimados Professores do curso de Tecnologia em Sistemas para Internet da UTFPR, expresso minha profunda admiração pelo notável comprometimento e excepcional trabalho dedicado ao longo de minha jornada acadêmica. Generosamente, vocês compartilharam seu vasto conhecimento com entusiasmo e dedicação, desempenhando um papel fundamental na entrega de um ensino de excelência. Graças a essa dedicação, fui capacitado a atingir meu potencial máximo como profissional.

Por fim, expresso meu reconhecimento a todos os membros da comunidade acadêmica, incluindo funcionários e docentes, que contribuíram para fazer da UTFPR uma instituição de referência em ensino superior. Seu apoio foi fundamental para o meu desenvolvimento pessoal e profissional.

A todos, muito obrigado.

RESUMO

O século XXI tem sido marcado por uma crescente necessidade das empresas em aprimorar a gestão de seus processos organizacionais. A busca por eficiência operacional e produtividade tem levado à adoção de soluções de tecnologia da informação como parte integrante das operações cotidianas. Nesse contexto, o controle de processos e a obtenção de informações precisas se tornaram cruciais, visando a redução de perdas e a maximização de resultados. Uma abordagem eficaz para essa otimização é o uso de *checklists*, que verificam a conformidade com procedimentos padrão, identificando discrepâncias e permitindo correções. Anteriormente, essa atividade era manual, consumindo tempo e recursos. Com o advento de ferramentas de *checklist* automatizadas, impulsionadas pela tecnologia da informação, houve uma evolução significativa na organização de tarefas diárias e na auditoria de processos. Este projeto visa desenvolver uma solução de software chamada *Routine Checklist*, que irá aprimorar a gestão de processos por meio de um sistema de *checklist* que guiará as rotinas, auditará ações e proporcionará uma interface amigável. Espera-se que essa ferramenta reduza consideravelmente o tempo gasto em verificações e auditorias, aumentando a eficácia operacional. Este projeto representa uma contribuição valiosa para a otimização de processos em diversos setores e contextos no ambiente de supermercados.

Palavras-chave: checklist; rotina; processo; tarefa; gestão.

ABSTRACT

The 21st century has been marked by a growing need for companies to improve the management of their organizational processes. The pursuit of operational efficiency and productivity has led to the adoption of information technology solutions as an integral part of daily operations. In this context, process control and obtaining accurate information have become crucial, aiming for the reduction of losses and the maximization of results. An effective approach for this optimization is the use of checklists, which verify compliance with standard procedures, identifying discrepancies and allowing for corrections. Previously, this activity was manual, consuming time and resources. With the advent of automated checklist tools, driven by information technology, there has been significant evolution in organizing daily tasks and auditing processes. This project aims to develop a software solution called Routine Checklist, which will enhance process management through a checklist system that guides routines, audits actions, and provides a user-friendly interface. It is expected that this tool will considerably reduce the time spent on checks and audits, increasing operational efficiency. This project represents a valuable contribution to process optimization in various sectors and contexts within the supermarket environment.

Keywords: checklist; routine; proccess; task; manage.

LISTA DE FIGURAS

Figura 1 – Estrutura de ambiente web cliente X servidor 15

Figura 2 – Estrutura de aplicação utilizando infraestrutura MVC 19

Figura 3 – Quadro de Organização MoSCoW 21

Figura 4 – Fluxo de dados em um sistema gerenciado com SGBD. 23

Figura 5 – Tela dos *checklists* aplicados. 24

Figura 6 – Tela inicial FastField Checklist 25

Figura 7 – Tela inicial Task 26

Figura 8 – Relacionamentos muitos-para-muitos 32

Figura 9 – Unidades e localidades 33

Figura 10 – Usuário e grupos de acessos 34

Figura 11 – *Checklist* e perguntas associadas 35

Figura 12 – *Checklist* movimento e perguntas associadas 36

Figura 13 – Tela Cadastro de Setor - Web - Implementação fase 2 37

Figura 14 – Tela Cadastro de Classificação da aplicação web 37

Figura 15 – Tela cadastro de unidade da aplicação web 38

Figura 16 – Tela cadastro de grupo de usuários da aplicação web 39

Figura 17 – Tela cadastro de *Checklist* da aplicação web 40

Figura 18 – Tela Cadastro de Perguntas do *Checklist* da aplicação web 41

Figura 19 – Tela inicial da aplicação web 42

Figura 20 – Tela Inicial de tarefas pendentes do aplicativo móvel 43

Figura 21 – Tela de exibição das perguntas de um *checklist* do aplicativo móvel . . 44

Figura 22 – Tela para resposta de uma pergunta do aplicativo móvel 45

Figura 23 – Tela gerenciamento de unidades 47

Figura 24 – Tela edição de unidade 48

Figura 25 – Tela gerenciamento de *checklists* 49

Figura 26 – Tela de seleção de unidades para gerar tarefas manuais 50

Figura 27 – Tela de gerenciamento das tarefas de *checklist* - WEB 53

Figura 28 – Estatísticas e pontuações do relatório web de execução de uma tarefa . 54

Figura 29 – Perguntas e respostas do relatório web de execução de uma tarefa . . 55

Figura 30 – Tela inicial da aplicação web 56

Figura 31 – Tela de configuração para comunicação com a API **58**
Figura 32 – Autenticação do usuário **59**
Figura 33 – Apresentação das tarefas pendentes **60**
Figura 34 – Apresentação das perguntas de uma tarefa selecionada **61**
Figura 35 – Tela para resposta de uma pergunta de uma tarefa **62**
Figura 36 – Tela de ampliação de foto de uma pergunta **63**
Figura 37 – Modelo Lógico de Dados - MLD **70**

LISTA DE TABELAS

Tabela 1 – Levantamento dos requisitos não funcionais. **29**

Tabela 2 – Levantamento dos requisitos funcionais. **30**

LISTA DE QUADROS

Quadro 1 – Recursos Disponíveis **27**

LISTAGEM DE CÓDIGOS FONTE

Listagem 1 – Enum dos Tipos de processos do *Routine Checklist* **51**

Listagem 2 – Processamento Automatizado **52**

SUMÁRIO

1 INTRODUÇÃO 12

1.1 Objetivos 13

1.1.1 Objetivo geral 13

1.1.2 Objetivos específicos 13

2 REFERENCIAL TEÓRICO 15

2.1 Tecnologias do lado cliente (*front-end*) 16

2.2 Tecnologias do lado servidor (back-end) 18

2.3 Processo de desenvolvimento 20

2.4 Ferramentas de desenvolvimento 21

3 SISTEMAS CORRELATOS 24

3.1 Checklist Fácil 24

3.2 FastField Mobile Forms 25

3.3 Task 26

3.4 Estudo Comparativo 26

4 ANÁLISE E PROJETO DO SISTEMA 28

4.1 Levantamento de requisitos 28

4.2 Modelagem do banco de dados 31

4.3 Protótipos de Telas 36

5 DESENVOLVIMENTO DO SISTEMA 46

5.1 Desenvolvimento da aplicação web 46

5.2 Desenvolvimento da aplicação móvel 57

5.3 Considerações sobre o desenvolvimento 63

6 CONSIDERAÇÕES FINAIS 65

6.1 Trabalhos Futuros 65

REFERÊNCIAS 67

APÊNDICE A MODELO LÓGICO DE DADOS - MLD 70

APÊNDICE B RELATÓRIO EM PDF DE EXECUÇÃO DE UMA TAREFA DE CHECKLIST 72

1 INTRODUÇÃO

O início do século XXI tem testemunhado um aumento constante na demanda das empresas por aprimorar a gestão de seus processos organizacionais. Esse impulso em direção à otimização das atividades visa, fundamentalmente, aperfeiçoar a eficiência operacional, com o intuito de aumentar a produtividade e, assim, manter uma posição competitiva em um mercado cada vez mais acirrado. Como resposta a esse desafio, as empresas têm adotado uma série de recursos da tecnologia da informação como parte integrante de suas operações diárias. Essa integração tem como propósito automatizar tarefas e fornecer ferramentas que promovam melhorias nas atividades do cotidiano (PRADELLA; FURTADO, 2012).

Assim, com o objetivo de aprimorar o controle dos *processos* [1], as empresas buscam incessantemente soluções que possam oferecer controle sobre esses processos, e informações de alta precisão. Essa abordagem visa a redução de perdas e a maximização de resultados, transformando a busca contínua pela excelência em uma meta tangível e alcançável. Vislumbrando este objetivo, muitas empresas fazem uso de softwares de auditoria e controle que possibilitem a centralização de informações que ajudem a gestão do negócio (PAIN *et al.*, 2009).

Observa-se, portanto, que a gestão de processos desempenha um papel crucial no atual ambiente de negócios. Conforme destacado por Pain *et al.* (2009), a busca por aprimoramento nessa área é evidente. Esse aprimoramento é claramente evidenciado pelos crescentes investimentos em Tecnologia da Informação (TI) feitos pelas empresas. De acordo com a pesquisa anual da Fundação Getulio Vargas (2023), referência no estudo do uso de TI nas empresas, o investimento em TI no Brasil cresceu cerca de 6% ao ano nos últimos 35 anos, aumentando de 1,3% do faturamento líquido das empresas de médio e grande porte em 1988 para 9% em 2022/23. Além disso, a pesquisa aponta que ainda há necessidade de mais investimentos no setor para que o Brasil alcance níveis semelhantes aos de países mais desenvolvidos. Isso reflete a necessidade evidente de informatizar os processos de negócios, levando as empresas a investirem consideravelmente para otimizar suas atividades e manter sua competitividade no mercado.

Uma abordagem eficaz amplamente adotada pelas empresas na otimização de processos é o uso de *checklists*. De acordo com a definição da Softplan Planejamento e Sistemas S.A. (2023a), um *checklist* é uma ferramenta que verifica a conformidade com procedimentos padrão, identificando discrepâncias e possibilitando correções e melhorias nos processos auditados. Isso resulta em melhores resultados nas atividades, promovendo a inspeção, organização e padronização do processo de auditoria e verificação.

O *checklist* é uma ferramenta versátil e poderosa que encontra aplicação em diversos setores, incluindo indústrias, varejo, hospitais e outros. Pode ser elaborado em planilhas eletrô-

[1] *"Processo é uma agregação de atividades e comportamentos executados por humanos ou máquinas para alcançar um ou mais resultados".* (ABPMP, 2013)

nicas ou em papel, oferecendo mobilidade ao auditor durante a execução das verificações. Um exemplo notável de sua aplicação é no controle de qualidade, assegurando que produtos ou serviços atendam aos padrões exigidos pela empresa. Na área da saúde, é usado para agendar horários de administração de medicamentos aos pacientes, garantir a limpeza e em várias outras situações.

Apesar de sua utilidade, a gestão de informações em *checklists* manuais, seja em papel ou planilhas eletrônicas, pode ser desafiadora, resultando em desperdício de tempo, baixa confiabilidade dos dados coletados e possíveis perdas de informações relevantes, bem como dificuldades na leitura e interpretação dos dados. Para solucionar esse problema e atender à crescente demanda por soluções confiáveis e ágeis, surgiram ferramentas de *checklist* automatizadas baseadas em software. Essas ferramentas, impulsionadas pela tecnologia da informação, se mostraram mais eficazes na organização, execução e auditoria das tarefas diárias, permitindo a garantia na conformidade com as especificações da empresa. Como resultado desse avanço, surgiram aplicativos de *checklist* online e para dispositivos móveis, que são capazes de incluir imagens, texto, respostas objetivas e muitos outros recursos, de modo a comprovar sem questionamentos a aplicação correta da ferramenta e da execução do processo.

O presente projeto, é o desenvolvimento de uma solução de software com o propósito de evoluir a gestão de processos em supermercados. Para alcançar este objetivo, foi desenvolvido um sistema de *checklist* intitulado como: *Routine Checklist*, capaz de orientar a execução de rotinas, auditar as ações da operação do supermercado, buscando manter uma interface de fácil utilização e intuitiva. Espera-se que essa ferramenta reduza significativamente o tempo necessário para verificações e auditorias das atividades, aumentando a eficácia das operações. Acredita-se que este projeto será uma contribuição valiosa para otimizar processos em diversos contextos e departamentos do setor supermercadista, podendo eventualmente, ser estendida para outras áreas de atuação além do comércio varejista como por exemplo a indústria ou até mesmo na área de serviços.

1.1 Objetivos

1.1.1 Objetivo geral

Desenvolver um sistema de *checklist* dedicado a organização e auditoria de rotinas de um supermercado.

1.1.2 Objetivos específicos

- Organizar e padronizar as atividades do setor de operação de um supermercado;

- Auditar as atividades do setor de operação requerendo fotos para comprovar a execução correta dos processos;
- Avaliar a execução dos processos operacionais através de uma pontuação em cada questão executada;
- Exibir informações referente a execução geral das tarefas de *checklists* na página inicial da aplicação em forma de quadros;
- Gerar relatório em forma de gráfico na página inicial da aplicação, comparando os *checklists* executados, não executados e em execução entre as unidades de uma rede de supermercados;
- Agendar a geração automatizada de tarefas de *checklists* para execução.

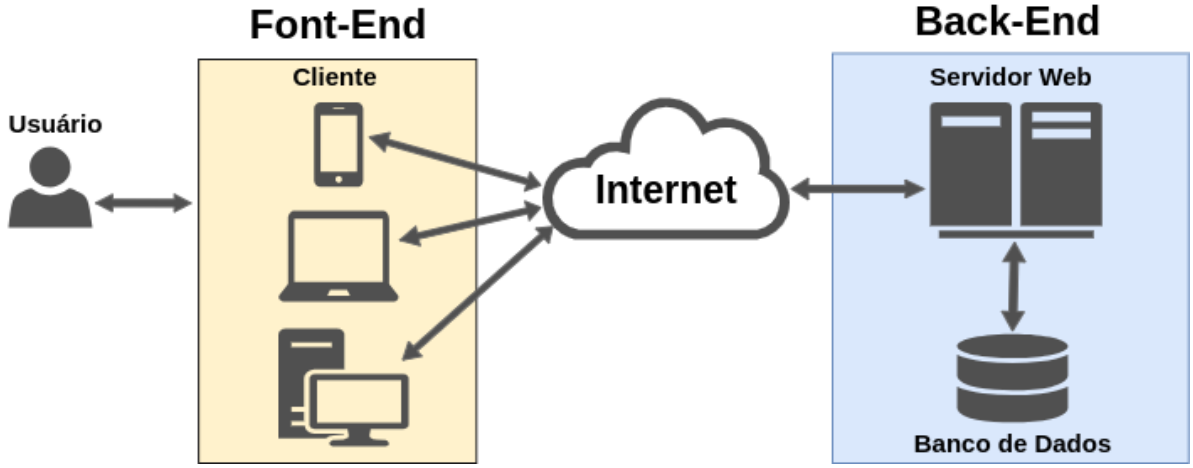
2 REFERENCIAL TEÓRICO

Neste capítulo, são delineadas as tecnologias e conceitos que orientam o projeto proposto. A intenção é oferecer um embasamento teórico que aprofunde a compreensão acerca da utilização de cada tecnologia utilizada, bem como seu papel no desenvolvimento da aplicação proposta.

O desenvolvimento de aplicações no contexto web pode ser categorizado em dois segmentos distintos: *front-end* e *back-end*. O *front-end* representa o aspecto voltado para o cliente (*client-side*), onde se concentra o desenvolvimento da interface visual do sistema, englobando elementos como telas, botões e todos os componentes que permitem a interação direta ou indireta do usuário. Por sua vez, o *back-end*, representa o lado do servidor (*server-side*), onde ocorre o processamento da aplicação, englobando o tratamento das informações manipuladas e a execução das regras de negócio específicas. Nesse ambiente, é possível gerenciar e processar dados, bem como executar ações como armazenar, modificar, excluir e recuperar informações.

A Figura 1 representa visualmente esses dois domínios, com o *front-end* sendo representado pelas máquinas acessadas pelo usuário/cliente e o *back-end* consistindo no servidor web e no banco de dados. As tecnologias empregadas no lado do cliente se comunicam com o servidor por meio de uma rede de computadores, podendo se tratar de uma rede privada corporativa ou da internet em geral.

Figura 1 – Estrutura de ambiente web cliente X servidor



Fonte: Autoria própria (2024).

Nas seções subsequentes, é analisado em maior profundidade as metodologias e tecnologias passíveis de aplicação para a estruturação do ambiente *front-end*, *back-end* e o acesso à base de dados.

2.1 Tecnologias do lado cliente (*front-end*)

De acordo com Eis e Ferreira (2012), “o desenvolvimento *client-side* é composto por três camadas principais: informação, formatação e comportamento”. A Linguagem de Marcação de Hipertexto, ou HTML (*HyperText Markup Language*), é uma linguagem de marcação amplamente utilizada no desenvolvimento da primeira camada, que é a camada de informação. Cada uma dessas camadas é responsável por manipular independentemente os elementos da página, permitindo a criação de interfaces mais elegantes e dinâmicas. No caso do HTML, sua função principal é definir a estrutura de um documento, ou seja, a página HTML em si.

O CSS, cuja sigla representa *Cascading Style Sheets*, é um mecanismo que permite estilizar fontes, cores, espaçamentos e elementos de um documento da web. Esta tecnologia assume um papel de extrema importância na segunda camada no desenvolvimento *client-side*: a formatação. Ela possibilita a estilização de elementos HTML, resultando em uma significativa melhoria na apresentação do conteúdo e na experiência do usuário (CONSÓRCIO WORLD WIDE WEB - W3C, 2023).

Para além da aprimoração estética dos documentos da web, o CSS concede um nível substancial de controle sobre o comportamento e a renderização de diversas formas de mídia, abrangendo áudio, vídeo e elementos visuais. Adicionalmente, possibilita a adaptação do conteúdo para distintos tipos de dispositivos e tamanhos de tela, incluindo monitores, dispositivos móveis, televisões e outros dispositivos. A habilidade de criar estilos específicos para cada contexto de exibição desempenha um papel fundamental na otimização da experiência do usuário na web (SILVA, 2011).

Com a crescente demanda por maior produtividade no desenvolvimento de aplicações web, diversos *frameworks*[1] CSS foram criados para atender a essa necessidade. Entre eles, destacam-se o Materialize (2023), baseado no Material Design desenvolvido pela Google, e o Bootstrap (2023), criado pelo Twitter. Especificamente, o *framework* Bootstrap, oferece uma ampla gama de funcionalidades que vão desde a estilização básica até configurações avançadas do HTML. Além disso, ele disponibiliza recursos prontos para uso, como a criação de diversos modelos de menus, formulários avançados, além de muitos outros recursos. Além, o Bootstrap é altamente adaptável a várias resoluções de tela e é compatível com os mais diversos navegadores disponíveis no mercado, incluindo Mozilla, Chrome, Opera e outros. Devido às vantagens apresentadas, o Bootstrap foi o *framework* escolhido para o desenvolvimento da camada de estilização do HTML do Routine Checklist (ZABOOT; MATOS, 2020).

Integrando a terceira camada do *front-end*, o JavaScript é uma linguagem de programação que está disponível no lado do cliente. Ela está implementada nos navegadores web modernos e desempenha um papel crucial na criação de eventos e conteúdos dinâmicos, incluindo animações de tela, imagens em movimento, controle multimídia, validação de formu-

[1] *Frameworks* são ferramentas que oferecem funcionalidades essenciais para acelerar o desenvolvimento de projetos de forma eficiente (ZABOOT; MATOS, 2020).

lários e uma variedade de outras funcionalidades. Sua aplicação é amplamente difundida no desenvolvimento de aplicações web e para dispositivos móveis, e, devido a sua versatilidade ganhou destaque. Segundo David (2013), o JavaScript tornou-se a “linguagem de programação mais onipresente da história”.

No cenário atual, o JavaScript transcendeu várias categorias de desenvolvimento, deixando de ser apenas uma ferramenta para validação de páginas web e evoluindo para se tornar uma linguagem de programação poderosa. Ela é capaz de impulsionar o desenvolvimento completo de aplicações, abrangendo tanto o front-end quanto o *back-end*. Nesse processo de expansão, surgiram diversos *frameworks* para orientar e otimizar o desenvolvimento, economizando tempo e tornando o uso dessa linguagem mais eficiente.

Dentre a vasta seleção de *frameworks* e bibliotecas em javascript disponíveis no mercado, o JQuery (2023) se destaca pela sua acentuada utilização em diversos tipos de projetos. Essa biblioteca tem como objetivo padronizar e agilizar o processo de desenvolvimento. Embora algumas ferramentas tenham começado a diminuir sua relevância, o jQuery ainda é amplamente utilizado no desenvolvimento de aplicações, sendo incorporado por muitos *frameworks* devido à sua capacidade de simplificar tarefas comuns e aprimorar a experiência de desenvolvimento. Como exemplo de sua utilização, o jQuery, oferece a capacidade de realizar chamadas AJAX[2] de forma assíncrona para o servidor de maneira simplificada. Isso proporciona recursos avançados para a manipulação de conteúdo, permitindo uma interatividade mais eficaz entre o cliente e o servidor (SILVA, 2013).

O *Framework* Ionic (2023) é uma poderosa ferramenta de interface do usuário amplamente utilizada para criar aplicativos móveis de alta qualidade. Ele se baseia nas tecnologias fundamentais do desenvolvimento web, como HTML, CSS e JavaScript. Isso permite o desenvolvimento de aplicativos e ferramentas usando as mesmas habilidades e recursos usados na criação de páginas da web. Uma das principais vantagens do Ionic é sua capacidade de ser multiplataforma, o que significa que você pode criar uma única aplicação que funcione em vários ambientes, incluindo navegadores da web, dispositivos móveis e *desktops*. O Ionic também é compatível com diversos *frameworks* JavaScript populares, como React, Vue e Angular.

É importante destacar que o Angular (2023), é outro framework JavaScript de destaque. Mantido pela Google, o Angular é amplamente utilizado na construção de páginas de aplicativos de página única, onde a interação entre os elementos da página é gerenciada de forma dinâmica pelo Angular. Isso permite que as alterações nos elementos sejam refletidas instantaneamente na visualização do cliente, sem a necessidade de recarregar a página. Dessa forma, o Angular possibilita a criação de aplicativos web altamente responsivos e eficientes.

É importante reconhecer que as três camadas fundamentais do *front-end* web - informação, formatação e comportamento - são representadas, respectivamente, pelo HTML, CSS e JavaScript. Essas tecnologias trabalham em conjunto de forma sinérgica para criar uma ex-

[2] Permite que aplicações trabalhem de modo assíncrono, processando qualquer requisição ao servidor em segundo plano, sem a necessidade de atualizar a página HTML

periência de usuário coesa. Embora cada uma delas seja processada de forma independente pelo navegador, elas estão intrinsecamente interligadas e exercem influência mútua. Portanto, qualquer inadequação na estrutura da página pode acarretar em problemas na apresentação do conteúdo ao usuário final, mesmo que navegadores modernos possam, em alguns casos, corrigir pequenas falhas estruturais no HTML.

Para completar a estrutura de desenvolvimento *front-end*, a utilização de *templates*, frequentemente integrados a *frameworks* como Bootstrap, jQuery, Angular e outros, desempenha um papel crucial na definição de layouts pré-definidos. Isso otimiza o processo de desenvolvimento, reduzindo o tempo necessário para criar interfaces e elementos visuais. Essa otimização, por sua vez, melhora a fluidez e a usabilidade do sistema, proporcionando uma experiência mais eficiente aos usuários.

No mercado, existem várias opções de *templates*, um exemplo notável sendo o Phoenix Template da ThemeWagon (2023), também disponibilizado pelo Bootstrap de forma proprietária, que oferece uma ampla gama de recursos prontos para uso. Esses recursos abrangem desde a padronização de formulários até a simplificação da implementação de gráficos e recursos avançados para manipulação da página web. Essa abordagem facilita, padroniza e otimiza o processo de desenvolvimento de interfaces e a apresentação de conteúdo, resultando em uma experiência de usuário aprimorada e maior uniformidade no desenvolvimento de software.

2.2 Tecnologias do lado servidor (back-end)

Uma linguagem de programação amplamente utilizada para o desenvolvimento de aplicações web no lado do servidor é o PHP. O acrônimo PHP representa "*Hypertext Preprocessor*" ou "Pré-processador de Hipertexto", como definido na documentação oficial da linguagem. Esta tecnologia é baseada em *scripts* interpretados, o que significa que a execução dos comandos e ações definidos no código ocorre em tempo real, independentemente do sistema operacional em uso. Em meio a uma ampla gama de tecnologias disponíveis, o PHP se destaca por sua facilidade de aprendizado e pela disponibilidade de uma documentação detalhada de suas funcionalidades. Isso favorece o desenvolvimento rápido e produtivo de aplicações usando esta linguagem. Além disso, assim como muitas outras linguagens de programação, o PHP dispõe de vários *frameworks* no mercado, incluindo o CakePHP, CodeIgniter, Symfony e Laravel (PHP GROUP, 2023).

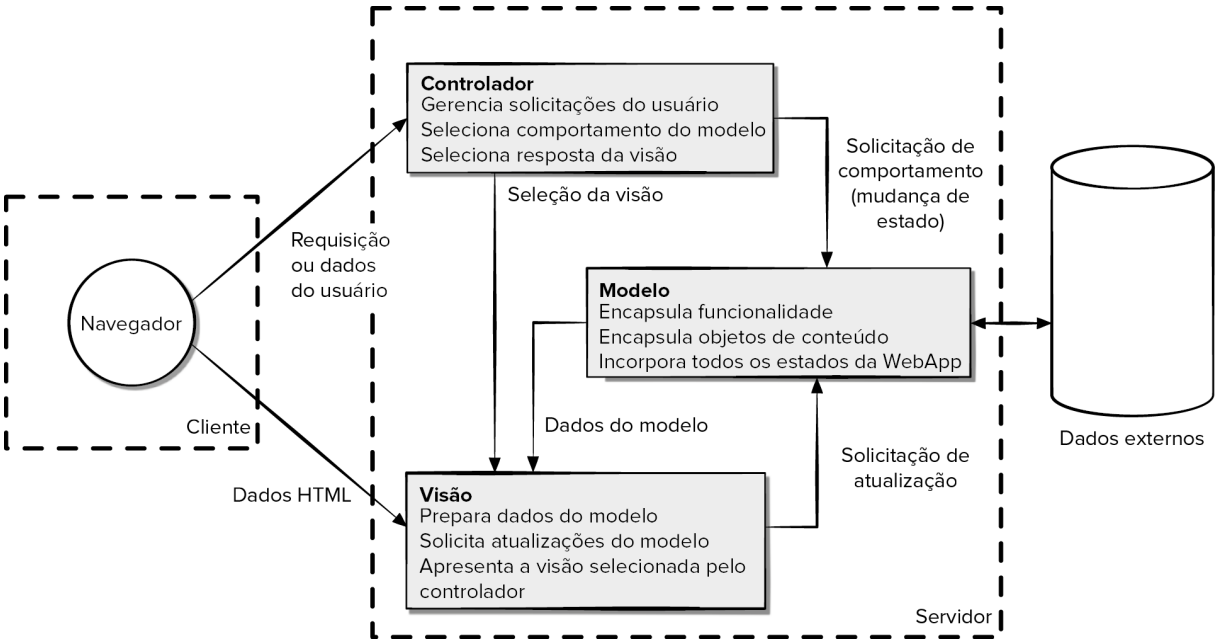
Para permitir o desenvolvimento mais eficiente e produtivo com o PHP, o *framework* Laravel se destaca como uma excelente escolha. Ele oferece uma estrutura bem definida que permite o desenvolvimento eficiente e altamente organizado de aplicações. O Laravel tem a capacidade de abstrair a complexidade do desenvolvimento, fornecendo recursos prontos para

uso, como o acesso ao banco de dados por meio de uma ORM[3](Mapeamento objeto-relacional) chamada *Eloquent*, manipulação de coleções de dados com o *Collect* e muitos outros recursos que agilizam e simplificam o desenvolvimento eficiente e rápido (LARAVEL, 2023a).

Além disso, o Laravel possui uma documentação completa e organizada que resulta em uma curva de aprendizado significativamente baixa. Isso, por sua vez, reduz o esforço necessário e permite que os desenvolvedores se concentrem no que realmente importa no projeto, como as regras de negócio, enquanto deixam a complexidade do acesso aos dados e as peculiaridades da linguagem PHP serem abstraídas pelo *framework* Laravel. Combinando o Laravel com o Phoenix Template, a velocidade de desenvolvimento se torna notável, devido à abstração da complexidade das funcionalidades de desenvolvimento e à criação das interfaces do sistema.

Em relação ao modelo de infraestrutura utilizado pelo Laravel, é adotada a arquitetura conhecida como MVC, sigla em inglês para *Model-View-Controller*, que estrutura as aplicações em três camadas distintas. A primeira camada é a *View* (Visão), onde a interface do sistema está presente, permitindo ao usuário interagir e trocar informações com a aplicação. A segunda camada é atribuída aos *Controllers* (Controladores), os quais têm a responsabilidade de gerenciar a comunicação e a troca de informações entre a visão e o modelo. Por fim, o *Model* (Modelo) representa as regras de negócio, bem como todo o processamento, lógica e acesso aos dados sensíveis do sistema. Este modelo de arquitetura é representado no esquema pela Figura 2 (PRESSMAN; MAXIM, 2021).

Figura 2 – Estrutura de aplicação utilizando infraestrutura MVC



Fonte: (PRESSMAN; MAXIM, 2021).

[3] ORM (*Object Relational Mapper*) consiste no mapeador objeto-relacional, que permite fazer uma relação dos objetos a nível de linguagem de programação, com os dados que os mesmos representam no banco de dados, de forma simples e agradável (LARAVEL, 2023b).

Para o desenvolvimento deste projeto, uma variedade de tecnologias discutido nas sessões 2.1 e 2.2 foi empregada. Na criação do aplicativo móvel, utilizou-se o Ionic Framework em conjunto com o Angular, visando garantir uma interface fluida e agradável para o usuário. Adicionalmente, foram empregadas marcações HTML e o *framework* Bootstrap para estilização, permitindo uma integração eficiente entre as tecnologias. A utilização do Javascript facilitou ainda mais a manipulação dos elementos de tela, proporcionando uma maior fluidez nas atividades apresentadas ao usuário.

No que tange ao desenvolvimento do *back-end*, optou-se pela utilização do PHP em conjunto com o Laravel, uma escolha que assegura robustez e agilidade. Além disso, foram integradas as tecnologias Phoenix Template, JQuery e Bootstrap na base de desenvolvimento, criando um ambiente de trabalho ágil e eficaz. Essa combinação de ferramentas possibilitou o desenvolvimento tanto da aplicação web quanto da API, que é essencial como interface de comunicação com o aplicativo móvel. A API desempenha um papel crucial ao garantir a sincronia e a troca eficiente de dados entre os sistemas web e o aplicativo móvel, assegurando um funcionamento harmonioso e integrado.

2.3 Processo de desenvolvimento

O método MoSCoW é empregado no contexto do planejamento e organização de projetos, processos e atividades. Ele permite uma gestão mais eficiente do tempo e dos recursos, pautada na consideração da importância de cada tarefa. Dessa forma, o método MoSCoW contribui para o aumento da produtividade e assegura que as tarefas de maior relevância sejam priorizadas e concluídas primeiro.

A palavra MoSCoW deriva-se do acrônimo em inglês, no qual cada categoria é representada pelas letras **MSCW**, sendo acrescida da letra “o” com o intuito de conferir uma pronúncia mais fluída. As tarefas são então classificadas em quatro categorias fundamentais:

1. ***M**ust Have* (Deve Ter): Representa as tarefas essenciais e indispensáveis para o projeto.

2. ***S**hould Have* (Deveria Ter): Engloba as tarefas importantes, mas não cruciais, que podem ser consideradas na evolução do projeto ou atividade.

3. ***C**ould Have* (Poderia Ter): Compreende as tarefas desejáveis, porém não prioritárias, podendo ser consideradas se houver recursos disponíveis.

4. ***W**on't Have* (Não Terá): Refere-se às tarefas que não serão incorporadas ao projeto ou atividade atual, mas podem ser consideradas em futuros desenvolvimentos.

Essas categorias desempenham um papel crucial na formação do quadro de tarefas, no qual cada atividade é meticulosamente organizada e categorizada com base em sua relevância para o projeto, conforme ilustrado no quadro representado na Figura 3 (SEBRAE, 2023).

Figura 3 – Quadro de Organização MoSCoW

Fonte: Autoria própria (2024).

2.4 Ferramentas de desenvolvimento

O Trello é uma aplicação online que oferece uma abordagem colaborativa para a gestão de projetos, notoriamente baseada no sistema Kanban[4]. Sua interface gráfica facilita a organização de projetos em listas, com o propósito de definir e visualizar o fluxo de trabalho, em linha com os princípios fundamentais do Kanban. Isso possibilita a definição clara do estado de desenvolvimento de cada funcionalidade, bem como a identificação dos membros da equipe responsáveis por sua implementação, com base na movimentação dos cartões entre as diferentes listas. A flexibilidade do Trello permite a criação de listas personalizadas conforme as necessidades específicas do projeto, proporcionando à equipe uma visão detalhada do fluxo de trabalho e do desempenho no desenvolvimento de cada atividade (ATLASSIAN CORPORATION, 2023).

No contexto do desenvolvimento de software, a organização das atividades é de extrema importância, mas não menos crucial é o controle das versões do código-fonte gerado. Esse controle é efetivamente alcançado por meio de ferramentas de versionamento, sendo o Git a escolha mais amplamente reconhecida e utilizada na atualidade. O Git representa uma tecnologia de sistemas de versão distribuída, notável por seu enfoque em viabilizar a colaboração eficiente em equipe durante o desenvolvimento.

[4] O Kanban é um método de organização que segmenta as tarefas em cartões, posicionando-os em colunas correspondentes às etapas do processo, com o objetivo de sinalizar e monitorar o progresso das atividades (PRESSMAN; MAXIM, 2021)

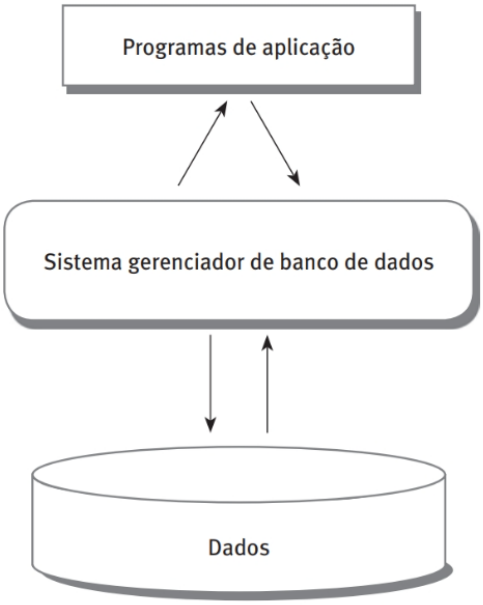
Para a implementação bem-sucedida desta tecnologia, existem diversos provedores de serviços à disposição. Destes, destacam-se notavelmente o Github, Bitbucket, Azure DevOps, e outros. Cada um deles oferece um conjunto de recursos e funcionalidades que podem se alinhar de maneira distinta às necessidades e preferências de diferentes equipes de desenvolvimento.(MONTEIRO *et al.*, 2021).

Após a cuidadosa análise e levantamento das atividades a serem realizadas, chegou o momento crucial de iniciar o processo de codificação das funcionalidades do sistema. Neste estágio, é de fundamental importância empregar uma ferramenta que otimize e agilize esse processo. As chamadas IDEs[5] desempenham um papel crucial nesse contexto. No mercado atual, encontramos uma ampla gama de ferramentas com essa finalidade, abrangendo tanto opções gratuitas quanto proprietárias. Dentre essas alternativas, o Visual Studio Code se destaca como uma escolha amplamente adotada no desenvolvimento de *software*. Esta ferramenta oferece suporte abrangente para diversas tecnologias, incluindo Java, Python, PHP, GIT, entre outras, que podem ser facilmente incorporadas por meio de extensões disponíveis para a plataforma. O Visual Studio Code possibilita, portanto, o desenvolvimento e a gestão do projeto por meio de uma interface centralizada, o que, por sua vez, contribui para tornar o processo de desenvolvimento mais produtivo e eficaz (MICROSOFT CORPORATION, 2023).

A manipulação de um sistema de informação é inerente à geração de informações e dados, os quais demandam armazenamento seguro e estruturado, assegurando sua pronta recuperação quando necessário. Dentro desse contexto, o banco de dados desempenha um papel de suma importância. Um banco de dados pode ser definido como um conjunto de dados organizado e inter-relacionado, delimitado por um domínio específico, com a finalidade de preservar informações para consultas futuras e garantir a sua integridade. Além de proporcionar uma estrutura ordenada para a retenção de dados, os sistemas de gerenciamento de banco de dados (SGBDs) conferem atributos fundamentais, tais como integridade dos dados, facilidade na administração de informações, robustez e segurança, juntamente com diversas outras funcionalidades, cujas especificidades variam conforme a escolha do SGBD. Desse modo, o fluxo de dados é gerenciado de maneira eficaz pelo SGBD, de modo que as informações percorrem uma trajetória conforme representado na Figura 4, que ilustra de maneira clara e concisa o ciclo de entrada, processamento, armazenamento e recuperação de dados em um sistema de informação.

[5] "A sigla IDE significa (*Integrated Development Environment*). Um ambiente de desenvolvimento integrado (IDE) é um software para criar aplicações que combina ferramentas comuns de desenvolvedor em uma única interface de usuário gráfica (GUI). Um IDE geralmente consiste em: Editor de código-fonte, Automação de compilação local e *Debugger*" (RED HAT, INC, 2023).

Figura 4 – Fluxo de dados em um sistema gerenciado com SGBD.



Fonte: (CARDOSO; CARDOSO, 2012).

No mercado atual, existem diversas opções de SGBDs disponíveis, incluindo nomes notáveis como Oracle, SQL Server, Postgres e MySQL. Todos os exemplos mencionados são representativos de bancos de dados relacionais, caracterizados pela sua estrutura tabular interconectada que forma uma rede abrangente de informações. O acesso a essas informações é viabilizado por meio da linguagem SQL (*Structured Query Language*), cujo nome traduzido para o português significa Linguagem de Consulta Estruturada, sendo amplamente utilizada em todos os sistemas de banco de dados relacionais (CARDOSO; CARDOSO, 2012).

Entre os SGBDs de código aberto mais populares, o PostgreSQL (2023) destaca-se como uma solução significativa no panorama do armazenamento de dados. Ele é capaz de lidar com cargas elevadas de dados, permitindo a escalabilidade do sistema. Ademais, destaca-se pela sua robustez e pela disponibilidade de ferramentas abrangentes para pesquisa e manipulação de dados, sendo por estes motivos o SGBD escolhido para a desenvolvimento deste projeto de *checklist*.

3 SISTEMAS CORRELATOS

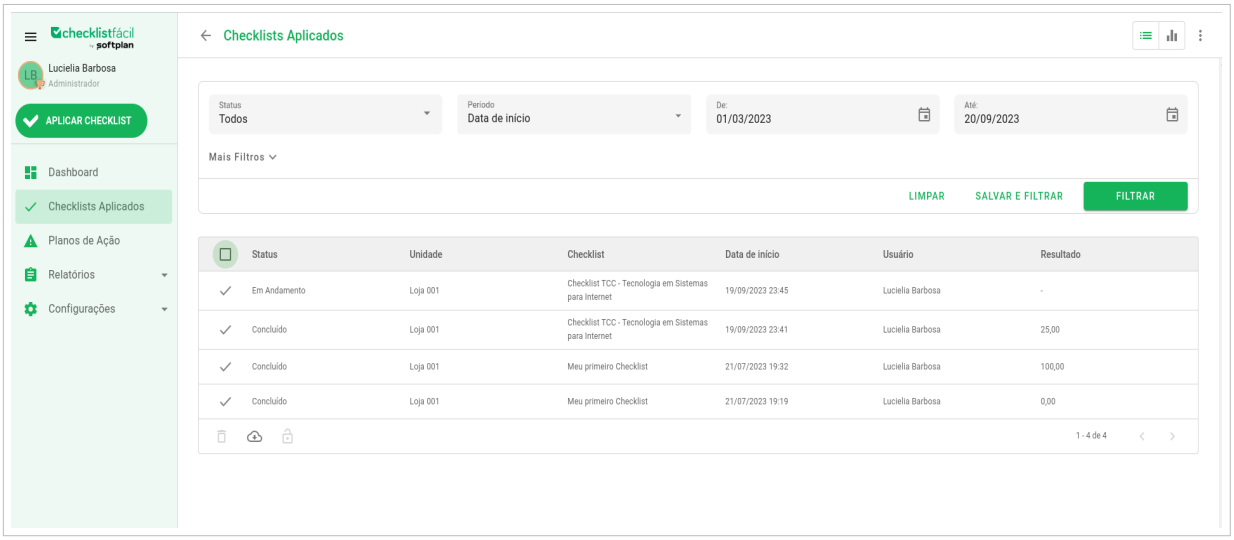
Da mesma forma que o sistema proposto, o qual tem a finalidade de auxiliar empresas no ramo de supermercados no controle da execução das rotinas internas do seu negócio, existem diversos outros sistemas disponíveis no mercado que implementam essa solução. Dentre estes, três com funcionalidades mais próximas ao sistema proposto serão descritos a seguir.

3.1 Checklist Fácil

O CheckList Fácil, mantido pela empresa Softplan, é uma ferramenta especializada na elaboração de *checklists*, oferecendo uma vasta gama de opções para a criação de questionários. Essas opções englobam a criação de perguntas avaliativas, textuais, listas de seleção, entre outras. Além disso, em relação às respostas, a aplicação proporciona funcionalidades adicionais, permitindo a inclusão de diversos elementos, que vão desde texto simples até a inserção de arquivos de multimídia, como imagens, áudio e vídeo nas respostas.

Este sistema de *checklist* é constituído por uma API[1] online fornecida pela Softplan, a empresa responsável pela aplicação, acompanhada por um aplicativo (APP) que simplifica a interação do usuário com o sistema. Além disso, ele oferece a capacidade de executar *checklists* offline, permitindo o registro das respostas no dispositivo móvel para posterior sincronização com o sistema online. Adicionalmente, é possível registrar as respostas por meio da plataforma web *online*, acessível através de um navegador de internet, eliminando assim a necessidade de utilizar o APP de *checklist*. Na plataforma web, é possível visualizar os *checklists* e o status de execução na tela de "*Checklists* Aplicados" conforme mostra a Figura 5.

Figura 5 – Tela dos *checklists* aplicados.



Fonte: Softplan Planejamento e Sistemas S.A. (2023b).

[1] Uma API é um conjunto de regras que permite que aplicativos e sistemas se comuniquem e compartilhem dados e funcionalidades de maneira organizada e padronizada.

3.2 FastField Mobile Forms

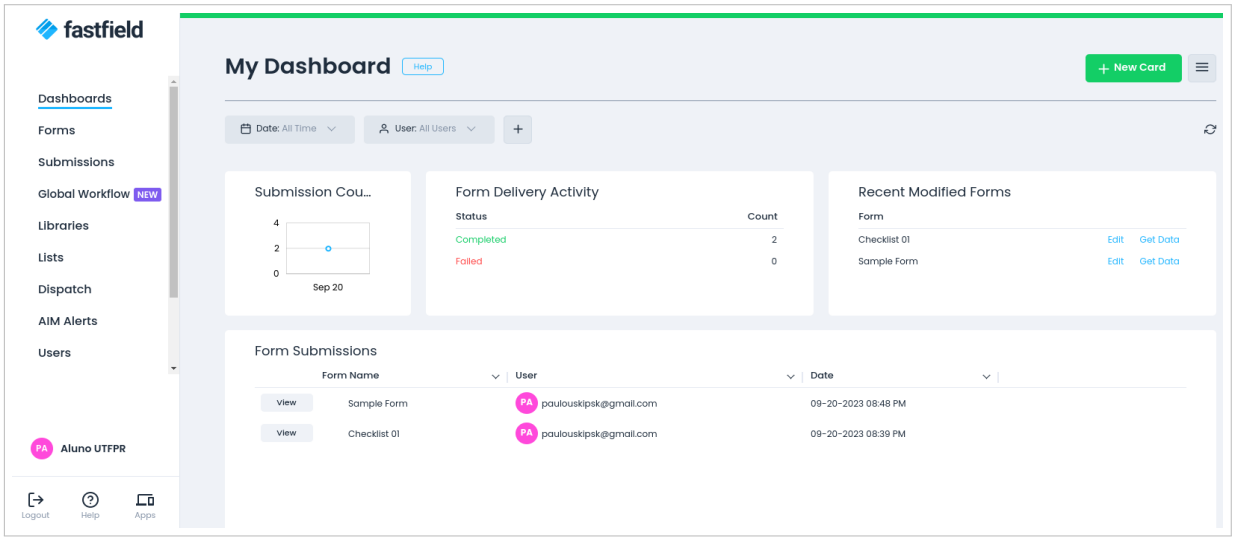
O sistema de *checklist* oferecido pela FastField se destaca, assim como o Checklist Fácil, devido à sua notável capacidade de personalização. Dentro de uma ampla gama de funcionalidades, merece destaque o recurso de cadastramento de perguntas online, que oferece uma variedade significativa de opções para a criação de listas de verificação, capazes de auxiliar empreendedores na gestão de processos de negócios. Além disso, o sistema também dispõe de recursos multimídia, como suporte para áudio, imagens e vídeo, juntamente com outras funcionalidades úteis, como listas de seleção e leitura de códigos de barras.

A interface do sistema se destaca pela sua notável facilidade de uso e intuição, o que o torna tão vantajoso quanto o Checklist Fácil. Essa interface intuitiva simplifica consideravelmente o processo de elaboração de formulários.

O FastField Mobile Forms, também permite a execução de *checklists* não apenas através de um navegador web, mas também por meio de um aplicativo móvel, tanto *online* quanto *offline*. Além disso, a interação com o usuário se destaca por sua simplicidade e objetividade, tornando a execução do *checklist* acessível e fácil, mesmo para usuários com diversos níveis de habilidade tecnológica.

Assim como no Checklist Fácil, é possível visualizar os status dos *checklists*, no entanto, essa informação é disponibilizada na tela inicial (*home*) da aplicação web, conforme demonstra a Figura 6.

Figura 6 – Tela inicial FastField Checklist



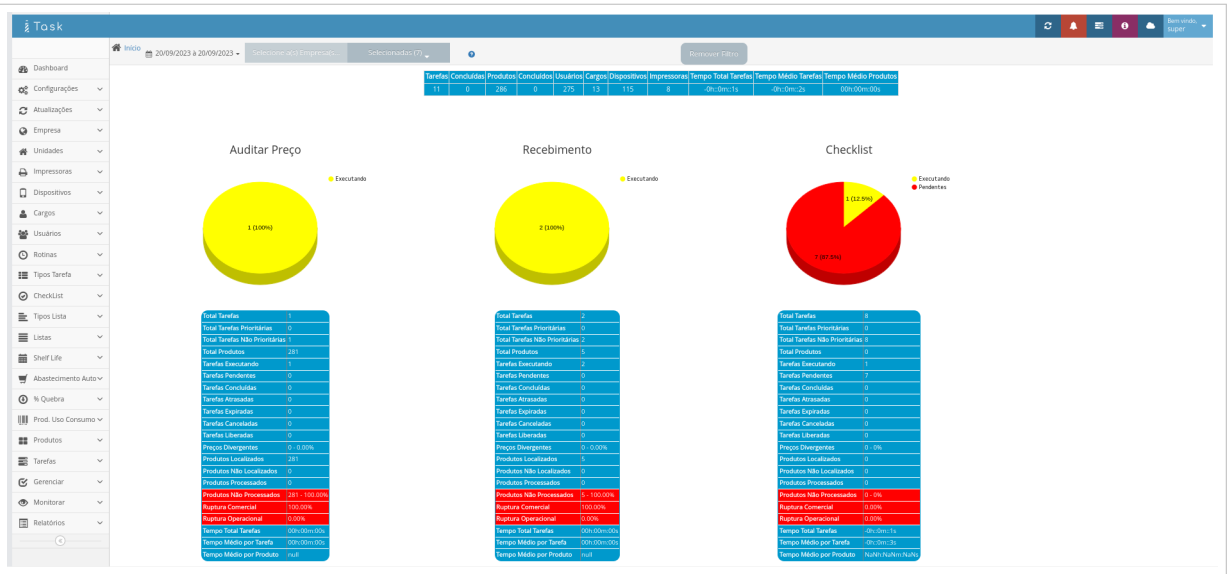
Fonte: FastField, Inc. (2023).

3.3 Task

A ferramenta Task é um eficaz gerenciador de tarefas desenvolvido pela empresa RP Info Sistemas, projetada para otimizar a organização das atividades diárias de uma empresa. Essa ferramenta é especialmente direcionada ao setor de varejo, mais precisamente a supermercados, atacadistas e centrais de compras, onde desempenha um papel fundamental na gestão operacional. O *checklist* é um dos módulos integrantes dessa ferramenta.

No Task, o *checklist* opera exclusivamente em modo *online*, não oferecendo a funcionalidade de execução *offline*. Além disso, sua execução é restrita ao aplicativo móvel dedicado, não sendo possível realizá-la por meio de um navegador da web. Apesar dessa limitação, o Task oferece uma série de recursos valiosos, incluindo a capacidade de criar Planos de Ação para a resolução de problemas, a facilidade de envio de e-mails para os setores responsáveis e a possibilidade de avaliação por meio de pontuação nas perguntas, semelhante ao Checklist Fácil. Embora não permita o envio de arquivos de mídia como respostas, o Task demonstra ser uma ferramenta robusta e eficiente para a gestão de tarefas e operações em ambientes de varejo, fornecendo soluções que atendem às necessidades específicas desse setor. Assim como no Fastfield, os status dos checklists e demais tarefas, são exibidas na tela inicial do sistema conforme exemplificado na Figura 7.

Figura 7 – Tela inicial Task



Fonte: RP Info Sistemas (2023).

3.4 Estudo Comparativo

Após estudo detalhado e simulação de uso tanto na plataforma web quanto nas aplicações mobile (APP) das três soluções apresentadas (Checklist Fácil, FastField Mobile Forms e Task), foi observado que todas oferecem uma variedade de recursos que facilitam a auditoria e

gestão de processos empresariais. Estes incluem a capacidade de inclusão de imagens, texto, planos de ação e a execução *offline*, entre outras funcionalidades importantes.

No entanto, ao comparar essas ferramentas, cada uma se destaca em pontos nos quais as outras podem não ser tão fortes. Esta disparidade pode ser explorada, permitindo a consolidação das funcionalidades relevantes encontradas nas três aplicações em um único projeto.

Assim, em conformidade com as particularidades identificadas nas aplicações similares ao projeto proposto, foi criado um quadro comparativo detalhado entre as tecnologias e os recursos disponíveis. Esse quadro possibilitou a definição das funcionalidades que o *Routine Checklist* incorporará, conforme evidenciado no Quadro 1.

Quadro 1 – Recursos Disponíveis

RECURSO	Ch.Fácil	Fastfield	Task	Routine
01 - Permite alterar a ordem das perguntas	X	X		X
02 - Permite atribuir pontuação para as perguntas	X		X	X
03 - Permite setorizar perguntas por Área/Setor	X		X	X
04 - Permite restringir horário de resposta por pergunta específica				X
05 - Permite inativar a pergunta em unidades específicas			X	X
06 - Permite limitar a quantidade de fotos nas perguntas			X	X
07 - Permite restringir a execução de um *checklist* baseado na permissão de grupo de acesso	X	X	X	X
08 - Permite configurar o tempo de expiração do *checklist* após a sua criação	X		X	X
09 - Permite agendar a criação de uma tarefa de *checklist* para execução através de uma configuração	X		X	X
10 - Permite criar plano de ação baseado nas respostas das perguntas	X		X	
11 - Permite acesso para a execução de *checklist* a partir de um navegador web	X	X		
12 - Acesso para a execução *offline* de *checklist* a partir de um aparelho mobile	X	X		
13 - Disponibilidade de resposta multimídia com áudio e vídeo	X	X		
15 - Possui responsividade na exibição em navegadores web	X	X		X
16 - Envio do relatório do *checklist* por e-mail cadastrado para cada *checklist* específico	X	X	X	

4 ANÁLISE E PROJETO DO SISTEMA

Neste capítulo, será apresentada a estratégia utilizada para orientar o desenvolvimento deste projeto de *checklist*. Para alcançar esse objetivo, foi empregado o método MoSCoW, citado na seção 2.3, para o gerenciamento e planejamento das atividades, visando aprimorar a eficiência em cada fase do projeto. Com essa abordagem, o capítulo destaca o planejamento organizando sistematicamente cada etapa do desenvolvimento. Dessa forma, é possível proporcionar uma visão clara do percurso percorrido e da utilização das tecnologias selecionadas, garantindo a qualidade e o sucesso no desenvolvimento do projeto.

O planejamento para desenvolvimento do projeto foi cuidadosamente estruturado e segmentado em cinco fases distintas, com o intuito de conduzir a análise e execução de maneira metódica e eficaz. Esse planejamento foi organizado e dividido da seguinte maneira:

- Levantamento de requisitos, prototipação de telas da aplicação web e do aplicativo móvel, prototipação da base dados e início do projeto Laravel com a integração do *Template Phoenix* e as ferramentas essenciais para o desenvolvimento da aplicação;
- Desenvolvimento de modelos, migrações para geração das tabelas de banco de dados e implantação das regras de negócios para execução das tarefas, bem como o desenvolvimento das telas de cadastros baseado nas prototipações executadas na fase1;
- Implementação dos processos de execução em segundo plano para geração e fechamento automatizado de tarefas;
- Desenvolvimento de relatórios e telas para gestão das tarefas executadas;
- Execução de testes finais, ajustes e otimização no processo de execução das aplicações web e mobile.

4.1 Levantamento de requisitos

Com a estrutura organizacional definida, deu-se início à primeira fase de desenvolvimento do projeto, cujo foco é abordado neste capítulo. O ponto de partida foi o levantamento das funcionalidades necessárias para o desenvolvimento da aplicação, processo conhecido como levantamento de requisitos. Para realizar essa etapa inicial, foi feita uma análise detalhada das informações coletadas durante o estudo abordado no Capítulo 3. Essa análise permitiu obter as informações necessárias para a síntese dos requisitos funcionais e não funcionais. Esses requisitos foram identificados a partir das características das ferramentas de *checklist* Task, FastField Mobile Forms e Checklist Fácil.

Juntamente com a elaboração dos requisitos funcionais, foram definidas as regras para o acesso ao sistema. Essas regras definem a configuração do cadastro dos usuários, que permitirá duas modalidades de acesso: Usuário administrador e Usuário *mobile*. Os administradores

desfrutam de permissões abrangentes, permitindo-lhes acesso completo aos cadastros do sistema, o que inclui o registro de *checklists* e das perguntas vinculadas, o gerenciamento de usuários e todas as funcionalidades disponíveis na aplicação web.

Em contraste, os usuários com a função de operador do aplicativo móvel, possibilita o acesso ao sistema para a execução dos *checklists*, permitindo que o usuário realize as tarefas específicas às quais está autorizado, diretamente a partir de um dispositivo móvel.

Com essa organização, o acesso ao sistema torna-se claramente bem definido. É importante ressaltar que um usuário com acesso à versão web (Administrador), também pode utilizar o acesso móvel, se necessário, desde que esteja a função habilitada no cadastro de usuário. Assim, o *Routine Checklist* oferece a flexibilidade de uso do aplicativo e para a plataforma web de acordo com as necessidades.

Com base nas funcionalidades encontradas, foi adotada uma abordagem subjetiva para determinar quais elementos eram essenciais, e quais tinham menor prioridade para a construção do quadro MoSCoW. Este quadro, é representado na Tabela 1, que descreve os requisitos não funcionais para o desenvolvimento da aplicação, enquanto os requisitos funcionais estão representados na Tabela 2.

Tabela 1 – Levantamento dos requisitos não funcionais.

ID	Descrição do requisito não funcional
RNF001	Permitir a execução do *checklist* em dispositivos mobile como smartphones e tablets.

Fonte: Autoria Própria (2024).

Tabela 2 – Levantamento dos requisitos funcionais.

ID	Prior.	Dom.	Descrição da Funcionalidade
RF001	*Must*	WEB	Permitir que o usuário faça a autenticação.
RF002	*Must*	WEB	Permitir que o usuário logado possa fazer *logout*.
RF003	*Must*	WEB	Possuir um usuário administrador do sistema com acesso total, para configurações e cadastros iniciais.
RF004	*Must*	WEB	Permitir o usuário administrador, fazer CRUD de classificação de *checklist*.
RF005	*Must*	WEB	Permitir o usuário administrador, fazer CRUD de Grupos de usuários.
RF006	*Must*	WEB	Permitir o usuário administrador, fazer CRUD de Unidade.
RF007	*Must*	WEB	Permitir o usuário administrador, fazer CRUD de Setores.
RF008	*Must*	WEB	Desenvolver a funcionalidade que gere automaticamente, baseado nas configurações dos *checklists*, as tarefas que devem ser executadas pelos usuários da operação.
RF009	*Must*	API	Criar end-point para autenticação do usuário via APP.
RF010	*Must*	API	Criar end-point para o usuário logado no APP no possa fazer logout no sistema.
RF011	*Must*	API	Criar end-point para consulta das tarefas do usuário logado pertinentes aos grupos de usuário a que ele pertence.
RF012	*Must*	APP	Permitir que o usuário faça a autenticação no sistema mobile.
RF013	*Must*	APP	Permitir que o usuário logado no sistema possa fazer logout no sistema mobile.
RF014	*Must*	APP	Permitir que um usuário possa visualizar as tarefas pertinentes aos grupos de usuário a que ele pertence.
RF015	*Must*	APP	Permitir que o usuário assuma uma tarefa e a execute.
RF016	*Must*	WEB	Permitir o usuário administrador, fazer CRUD de Usuários.
RF017	*Must*	APP	Permitir que um usuário libere uma tarefa assumida.
RF018	*Must*	WEB	Disponibilizar relatório, onde seja possível visualizar as respostas das perguntas de um *checklist* executado.
RF019	*Must*	WEB	Disponibilizar Relatório para visualizar o resultado dos *checklists* executados, com filtros que o usuário aplicar na consulta.
RF020	*Must*	WEB	Criar na *home page*, alguns gráficos que demonstrem o andamento dos *checklists* nas unidades.
RF021	*Must*	API	Criar end-point para resposta das perguntas do *checklist*.
RF022	*Must*	API	Criar end-point para Liberação de uma tarefa assumida.
RF023	*Must*	WEB	Desenvolver a rotina que feche automaticamente, baseado nas configurações dos *checklists*, as tarefas que já estão expiradas.
RF024	*Won't*	APP	Permitir gravar a geolocalização na resposta da pergunta, quando configurado na pergunta do *checklist*.
RF025	*Won't*	APP	Permitir a criação de plano de ação, baseado na resposta da pergunta, quando configurado na pergunta do *checklist*.

Fonte: Autoria Própria (2024).

Uma vez elaborado o quadro MoSCoW, foi possível organizar as tarefas a serem executadas através da ferramenta Trello, que permite a gestão do projeto através do Kanban, que são cartões que representam requisitos funcionais relacionados na Tabela 2. O fluxo de desenvolvimento foi distribuído em sete quadros:*Must Have*, *Should Have*, *Could Have* , *Won't Have*, *In progress*, *Tests/Review* e *Done*. Onde, a movimentação dos cartões acontece entre as três

Listas: *In progress*, *Tests/Review* e *Done*, indicando o status do processo em que se encontra cada atividade em execução.

Dessa forma, o fluxo do desenvolvimento teve um entendimento claro das funcionalidades a serem implementadas. À medida que cada funcionalidade foi iniciada, o respectivo cartão foi transferido da lista *Must Have* no quadro MoSCoW para a lista *In Progress*, indicando que a funcionalidade está, de fato, em processo de desenvolvimento. Quando a etapa de desenvolvimento estiver concluída e a fase de testes começar, o cartão será movido para o quadro *Tests/Review*. Após a finalização dos testes da funcionalidade, o cartão foi então movido para a lista *Done*, sinalizando o encerramento do desenvolvimento da respectiva funcionalidade.

A seleção das funcionalidades para o início do desenvolvimento da aplicação foi estruturada com base na lista apresentada na Tabela 2. A ordem de prioridade para a implementação de cada requisito funcional foi cuidadosamente determinada considerando sua importância e os pré-requisitos associados a cada um.

Após a registrado e organizado as tarefas a serem realizadas no sistema Trello, deu-se inicio à realização da modelagem do banco de dados, visando garantir a flexibilidade necessária para a incorporação de novas funcionalidades e a realização de manutenções futuras. Esse planejamento foi crucial para assegurar a evolução contínua do sistema. Além disso, o sistema deve oferecer a capacidade de gerar diversos relatórios gerenciais com informações essenciais para a gestão da empresa onde será implementado. Dada a importância dessa base de dados, optou-se pela utilização do Sistema de Gerenciamento de Banco de Dados (SGBD) PostgreSQL (2023), amplamente reconhecido como um dos mais robustos e respeitados no mercado.

O PostgreSQL destaca-se por sua capacidade de gerenciar grandes volumes de dados de forma eficiente e rápida. Adicionalmente, ele oferece integração com o *framework* Laravel, facilitando a interação entre o banco de dados e a aplicação web. Essa escolha estratégica pelo PostgreSQL fundamenta-se em sua sólida reputação no mercado e em sua capacidade de atender às necessidades do sistema, contribuindo para a robustez e eficiência da solução proposta.

4.2 Modelagem do banco de dados

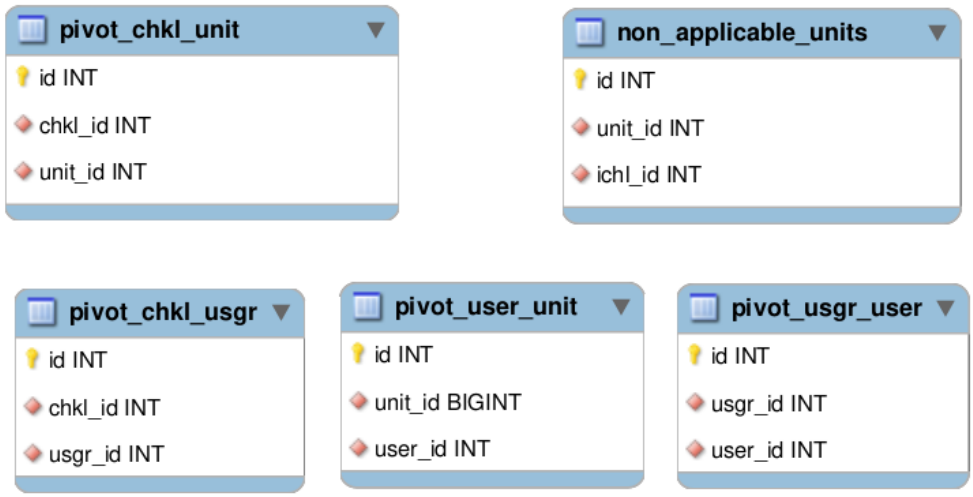
Com a definição dos requisitos funcionais e a escolha do sistema gerenciador de banco de dados, foi possível desenvolver um artefato fundamental para o sistema: o Modelo Lógico de Dados (MLD). Este artefato desempenha um papel importante ao fornecer uma representação visual detalhada da estrutura necessária do banco de dados. O MLD inclui os relacionamentos entre tabelas, seus atributos e os tipos de dados que acomodarão as informações geradas pelo sistema. Dessa forma, ele garante a integridade, a organização e a eficiência do armazenamento e da recuperação dos dados.

A seguir, os relacionamentos presentes no Modelo Lógico de Dados serão descritos individualmente, a fim de promover um entendimento mais claro das conexões entre as tabelas

do banco de dados. Contudo, o Modelo Lógico completo, incluindo todos os relacionamentos, pode ser visualizado na Figura 37, apresentada no Apêndice A.

Para dar início à apresentação das tabelas responsáveis por manter relacionamentos muitos-para-muitos, é fundamental destacar a relevância dessas tabelas no contexto do sistema. Elas desempenham um papel crucial e serão devidamente refletidas na representação visual das regras que governam os principais grupos de relacionamentos. É importante observar que essas tabelas podem ser identificadas pelo prefixo pivot_, com a exceção da tabela `non_applicable _units`. Na Figura 8, é apresentada essas tabelas de forma elucidativa.

Figura 8 – Relacionamentos muitos-para-muitos

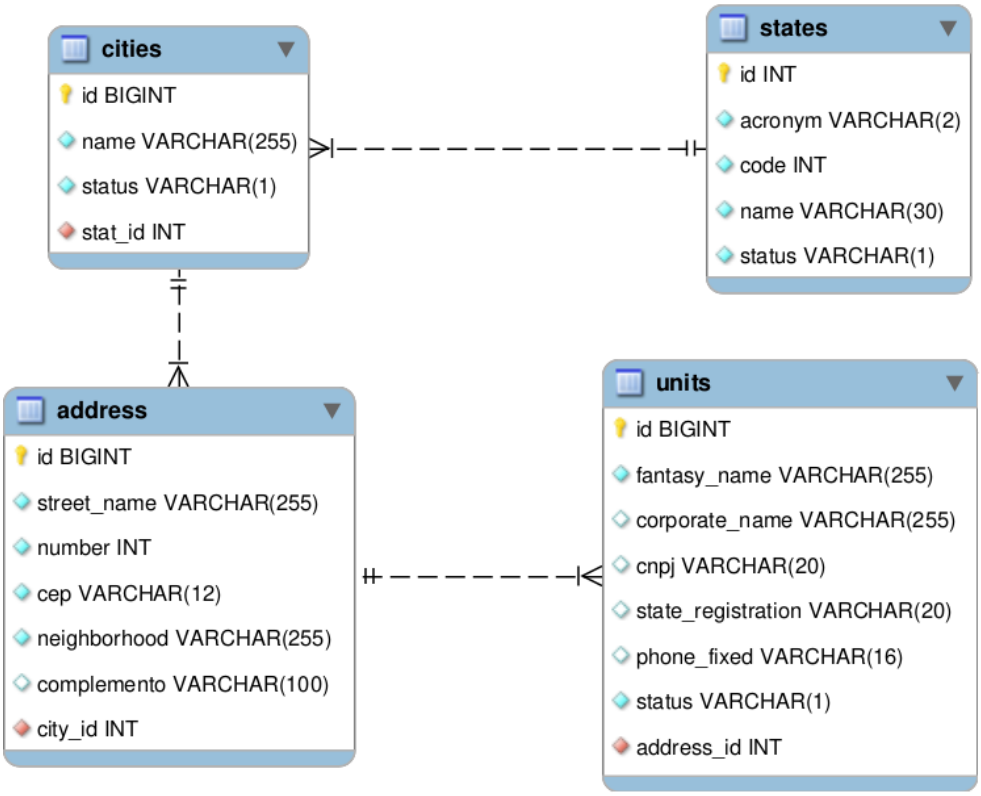


Fonte: Autoria própria (2024).

O relacionamento mais elementar no banco de dados, a tabela `units` que é uma representação das filiais de uma empresa. Este relacionamento fundamental serve como guia central para o sistema quanto aos procedimentos e particularidades de cada filial que utilizará o sistema de *checklist*. Portanto, a fim de utilizar essa ferramenta, é imperativo realizar o cadastro de, no mínimo, uma unidade, pois é por meio desse cadastro que as tarefas poderão ser geradas. Além disso, dentro deste contexto, encontram-se as tabelas `cities` e `states`, que têm a finalidade de representar os municípios e os estados brasileiros, incluindo o Distrito Federal. Essas tabelas foram alimentadas com informações provenientes do portal do Instituto Brasileiro de Geografia e Estatística (IBGE) (2023).

A Figura 9, apresenta visualmente os relacionamentos entre as tabelas `units`, `cities` e `states`.

Figura 9 – Unidades e localidades



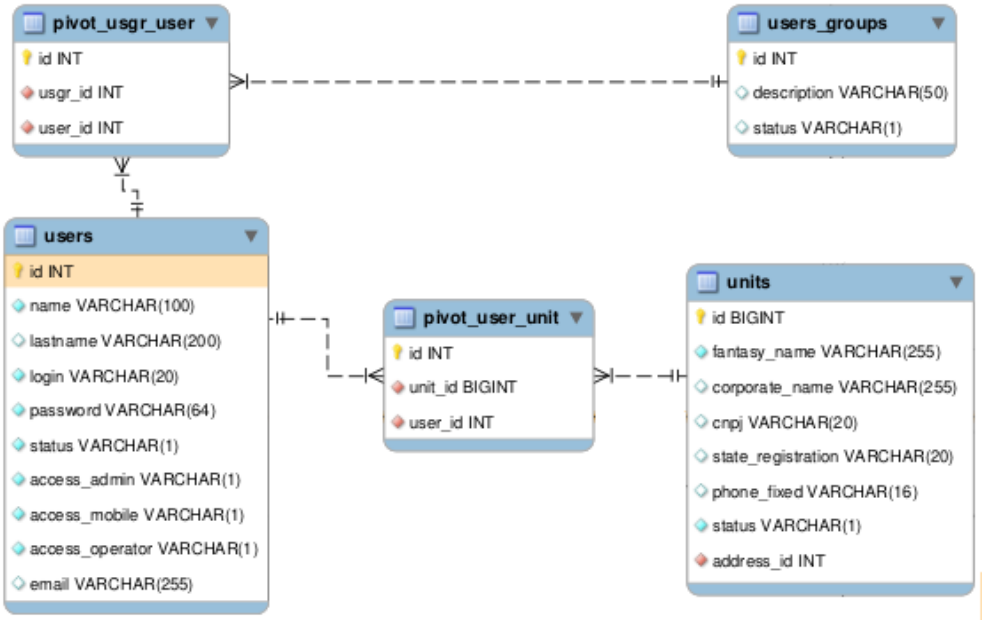
Fonte: Autoria própria (2024).

No que diz respeito ao acesso ao sistema, a tabela `users` desempenha um papel crucial. Ela não só permite o acesso ao sistema mediante o cadastro, mas também é responsável por rastrear as tarefas executadas pelos usuários. Além disso, ela proporciona diferentes níveis de acesso, personalizados para cada tipo de usuário, através da implementação de grupos de acesso representados na tabela `users_groups`.

Vale destacar que um usuário pode ser associado a múltiplos grupos de usuários, conferindo assim maior flexibilidade na gestão dos privilégios de acesso. Além disso, cada usuário pode ser vinculado a uma ou mais unidades, o que implica que o usuário somente terá visibilidade das tarefas relacionadas às unidades às quais ele está vinculado e de acordo com o grupo de usuários ao qual pertence. Isso assegura uma abordagem segmentada e controlada no acesso às informações e tarefas do sistema.

A Figura 10, apresenta visualmente os relacionamentos entre as tabelas que definem as regras de acesso ao usuário.

Figura 10 – Usuário e grupos de acessos



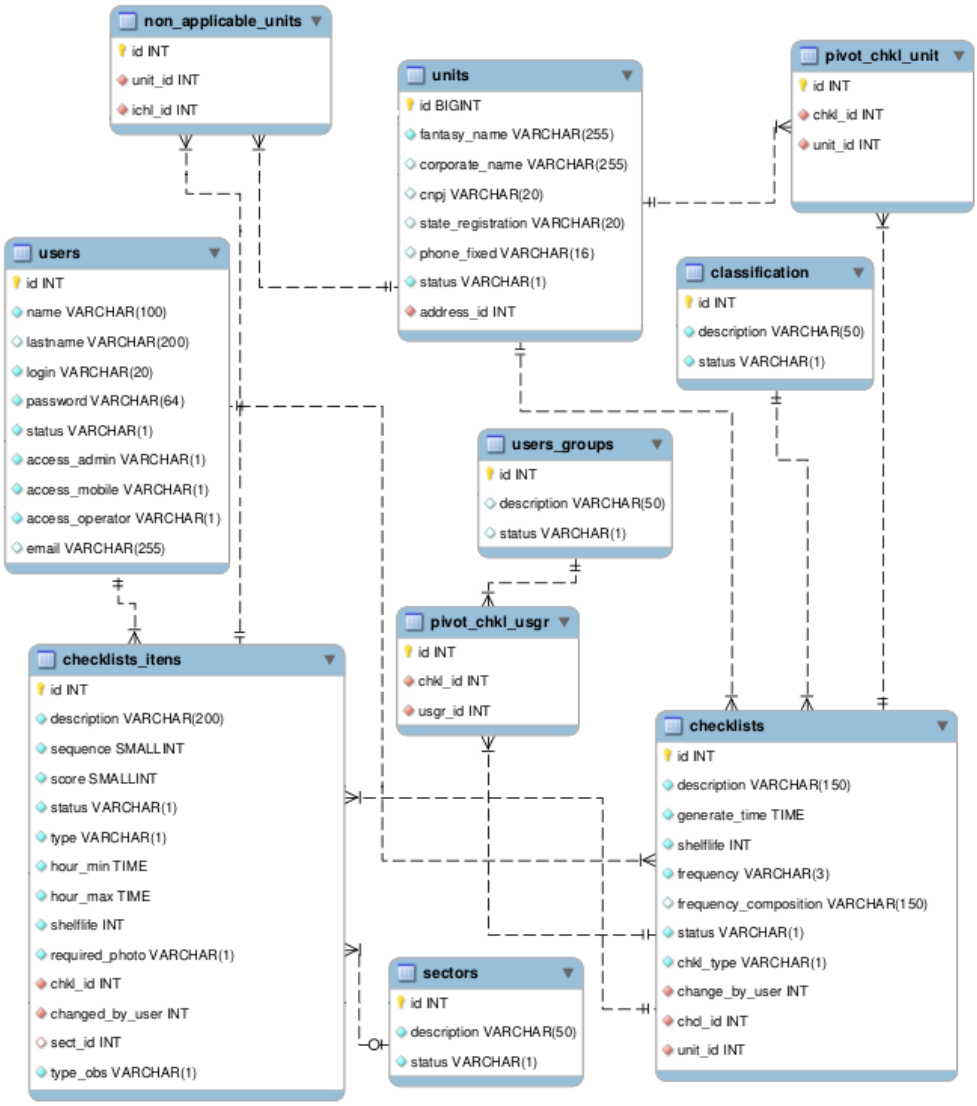
Fonte: Autoria própria (2024).

A tabela `checklists` assume a responsabilidade de armazenar os dados relativos aos registros e configurações dos *checklists*, sendo essenciais para o processamento e geração das tarefas. A tabela `checklist_itens`, por sua vez, contém as perguntas associadas aos *checklists*, juntamente com os dados e configurações pertinentes a cada uma delas. Apesar da representação envolver um grande número de relacionamentos, algumas tabelas se destacam por sua simplicidade notável. A tabela `sectors`, por exemplo, tem a função de especificar a qual setor uma pergunta está vinculada. Outra tabela de destaque é a `classification`, que permite a categorização dos *checklists* conforme a necessidade. Esses cadastros serão úteis, sobretudo para a gestão e a geração de relatórios.

Ademais, há o relacionamento de um `checklist` com a tabela `users_groups`, que possibilita o filtro na visualização e execução das tarefas, de acordo com as atribuições dos usuários estabelecidas para cada grupo. É relevante salientar a importância da tabela `non_applicable_units` no contexto das perguntas dos *checklists*. Esse relacionamento tem a capacidade de excluir uma ou mais unidades específicas para uma pergunta, quando não faz sentido incluí-la para aquela filial.

Na Figura 11, é possível observar os relacionamentos entre as tabelas que constituem o núcleo fundamental da aplicação.

Figura 11 – *Checklist* e perguntas associadas

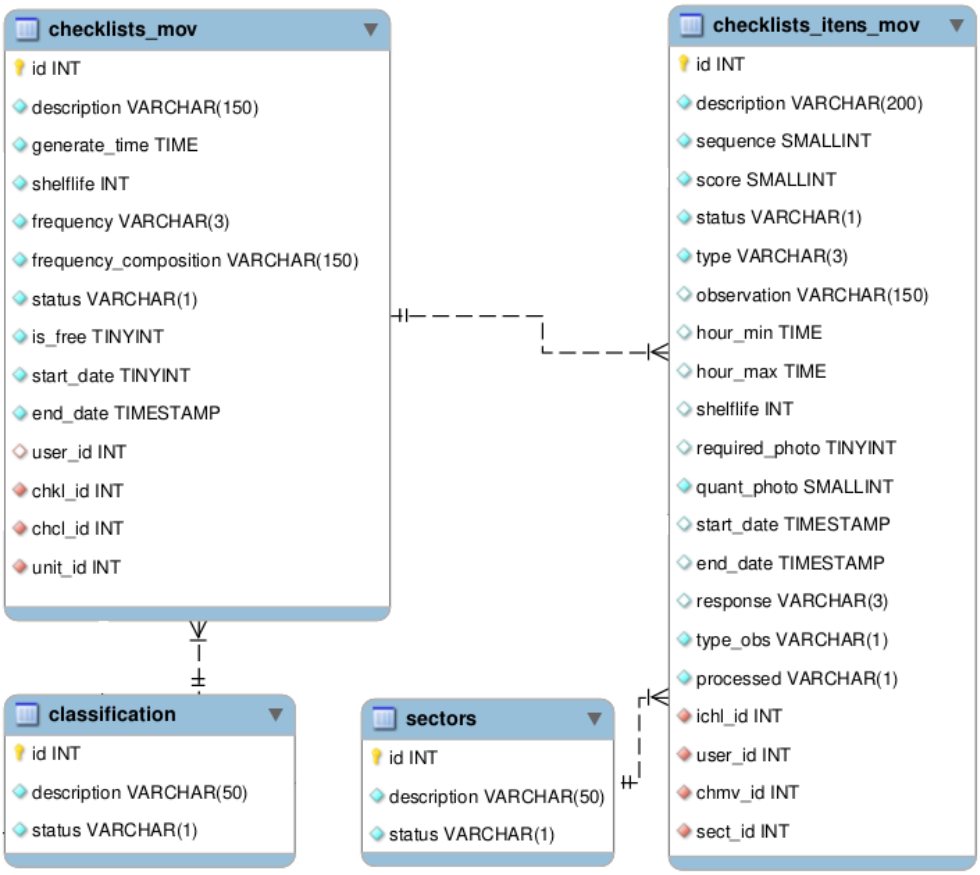


Fonte: Autoria própria (2024).

Por fim, são apresentadas as tabelas `checklists_mov` e `checklists_itens_mov`, cuja responsabilidade recai sobre o armazenamento das respostas geradas durante a execução das tarefas. A estrutura dessas tabelas guarda semelhanças com a estrutura da tabela de `checklists`, uma vez que as mesmas regras se aplicam a elas. A concepção de sua estrutura foi cuidadosamente elaborada para preservar as informações críticas dos *checklists* e de suas perguntas, que foram geradas no momento da criação da tarefa. Isso assegura que, mesmo que o *checklist* seja modificado, a integridade dos registros relativos ao movimento (execução da tarefa) permaneça intacta.

Na Figura 12, é possível visualizar de maneira simplificada os relacionamentos entre as tabelas que armazenam as tarefas em si e as respostas relacionadas à execução dessas tarefas.

Figura 12 – *Checklist* movimento e perguntas associadas



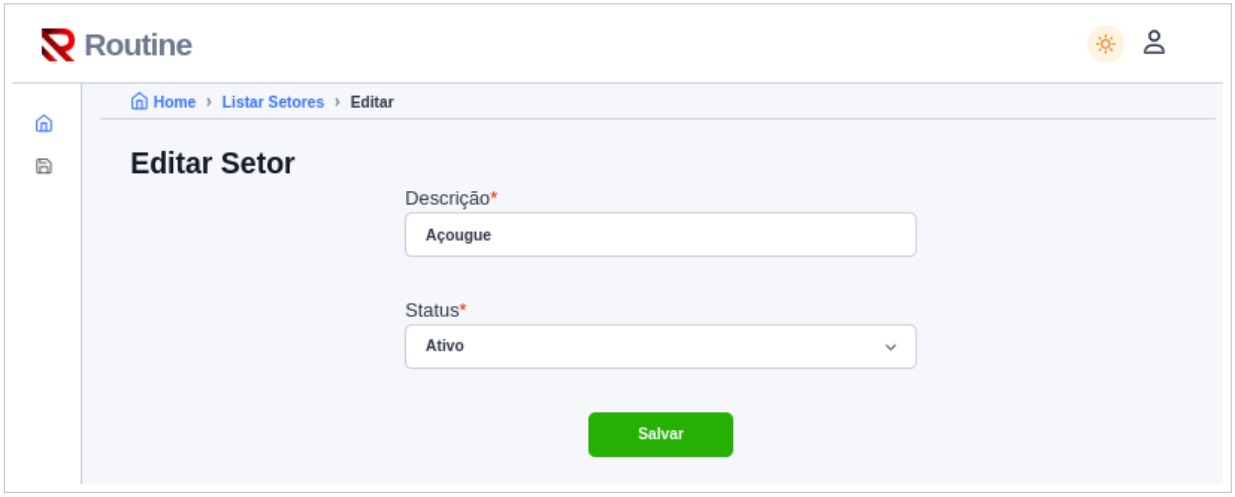
Fonte: Autoria própria (2024).

4.3 Protótipos de Telas

Após a conclusão do levantamento de requisitos e o estabelecimento da estrutura do banco de dados, o próximo estágio do projeto consistiu na implementação dos protótipos das telas. O início desse processo teve início nos cadastros das principais funcionalidades do sistema. A seguir, serão apresentados os protótipos das telas de cadastros mais elementares, a fim de agilizar a construção e a estruturação da aplicação.

Uma das primeiras telas desenvolvida no processo de prototipação foi a tela de cadastro de Setor, a qual tem como finalidade especificar o setor ou departamento relacionado às perguntas do *checklist*. A representação visual dessa tela pode ser conferida na Figura 13.

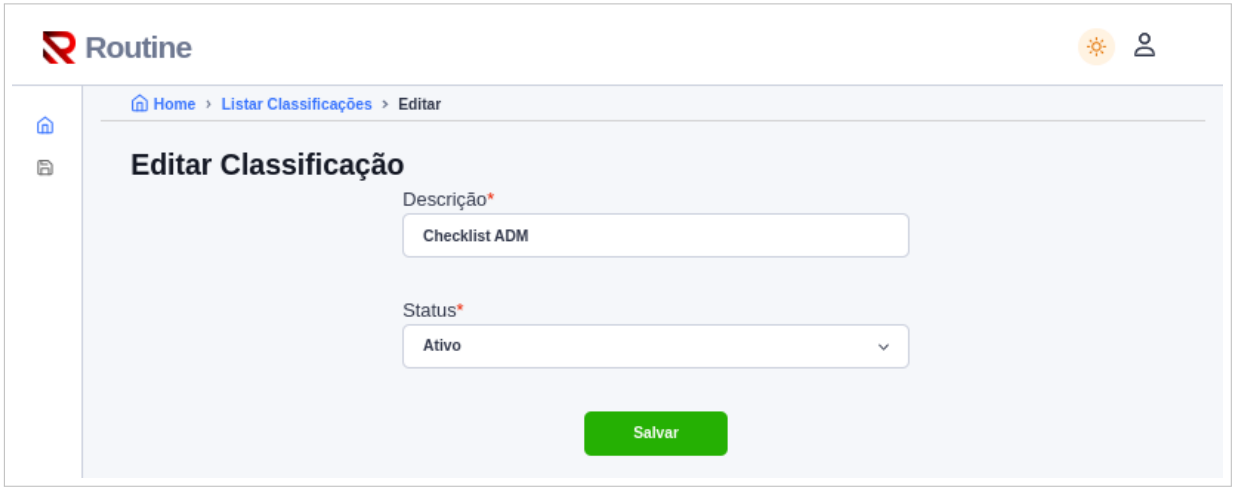
Figura 13 – Tela Cadastro de Setor - Web - Implementação fase 2



Fonte: Autoria própria (2024).

A principal função do cadastro de setor, é permitir a visualização da execução das perguntas respondidas de acordo com as particularidades cada setor ou departamento específico do supermercado. Na sequência, prossegue-se com o desenvolvimento da tela de Classificação. Esta tela desempenha um papel fundamental ao determinar a natureza do trabalho para a aplicação de um *checklist*. Uma visualização desta tela, pode ser observada por meio da Figura 14. Assim como o cadastro de Setor, a Classificação é um cadastro simples, no entanto, resulta em um importante recurso na criação de filtros para relatórios, e possibilita uma visualização mais abrangente dos processos executados pelo setor operacional da empresa.

Figura 14 – Tela Cadastro de Classificação da aplicação web

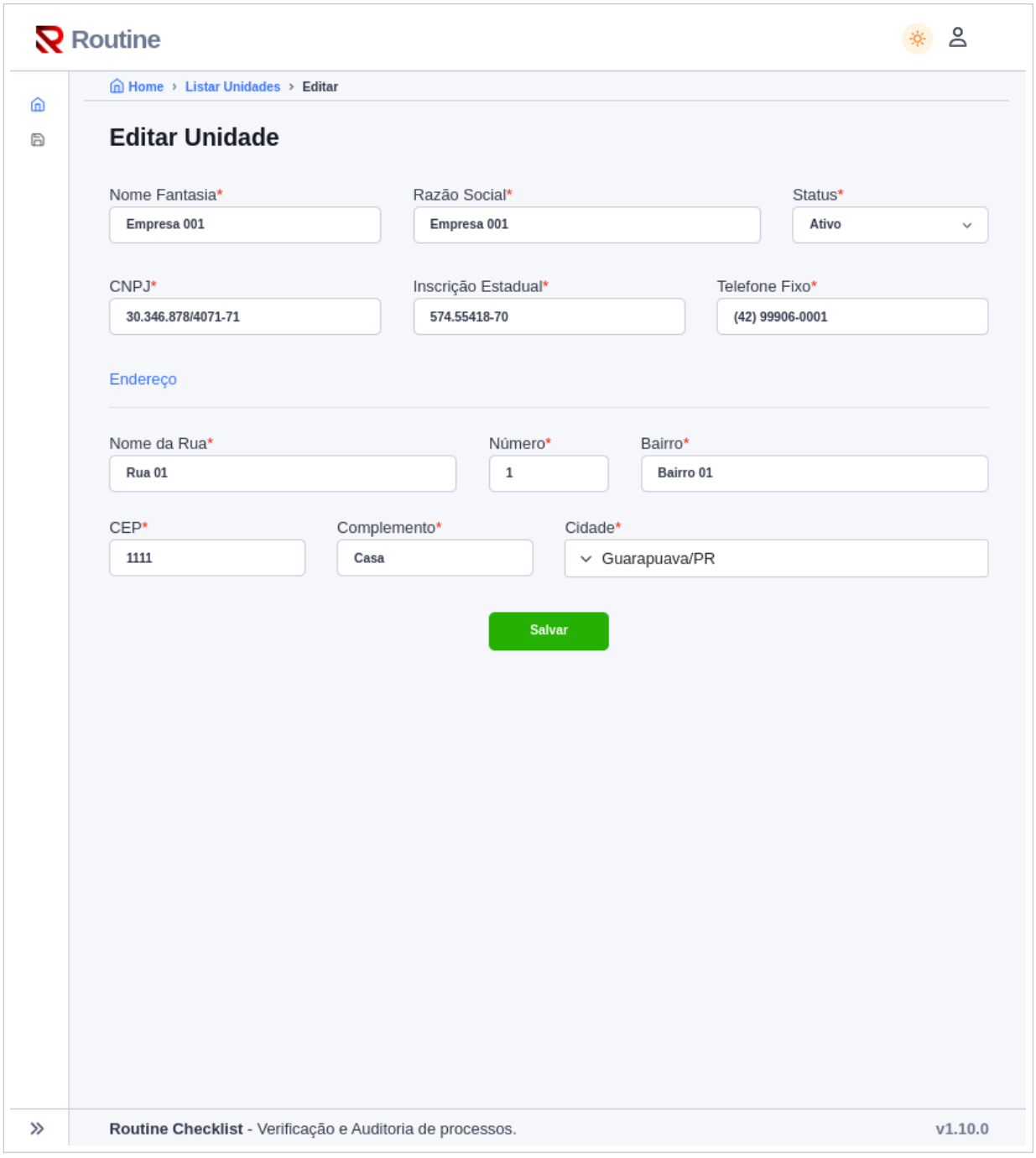


Fonte: Autoria própria (2024).

Por outro lado, o registro de unidades desempenha um papel de grande relevância no contexto da aplicação, uma vez que cada *checklist* deve estar vinculado, no mínimo, a uma unidade para que a tarefa correspondente seja gerada. Além disso, no âmbito deste cadastro,

o registro de endereços assume uma função crucial, devido a capacidade de gerar relatórios e disponibilizar informações para a análise dos resultados das unidades com base em sua localização geográfica. Dessa forma, posteriormente poderão ser implementados relatórios baseados em regiões onde as empresas estão localizadas. A tela de cadastro de unidades pode ser visualizada na Figura 15, que demostra as informações necessárias para o cadastro de uma unidade, que por sua vez, representa uma filial de uma rede de supermercados.

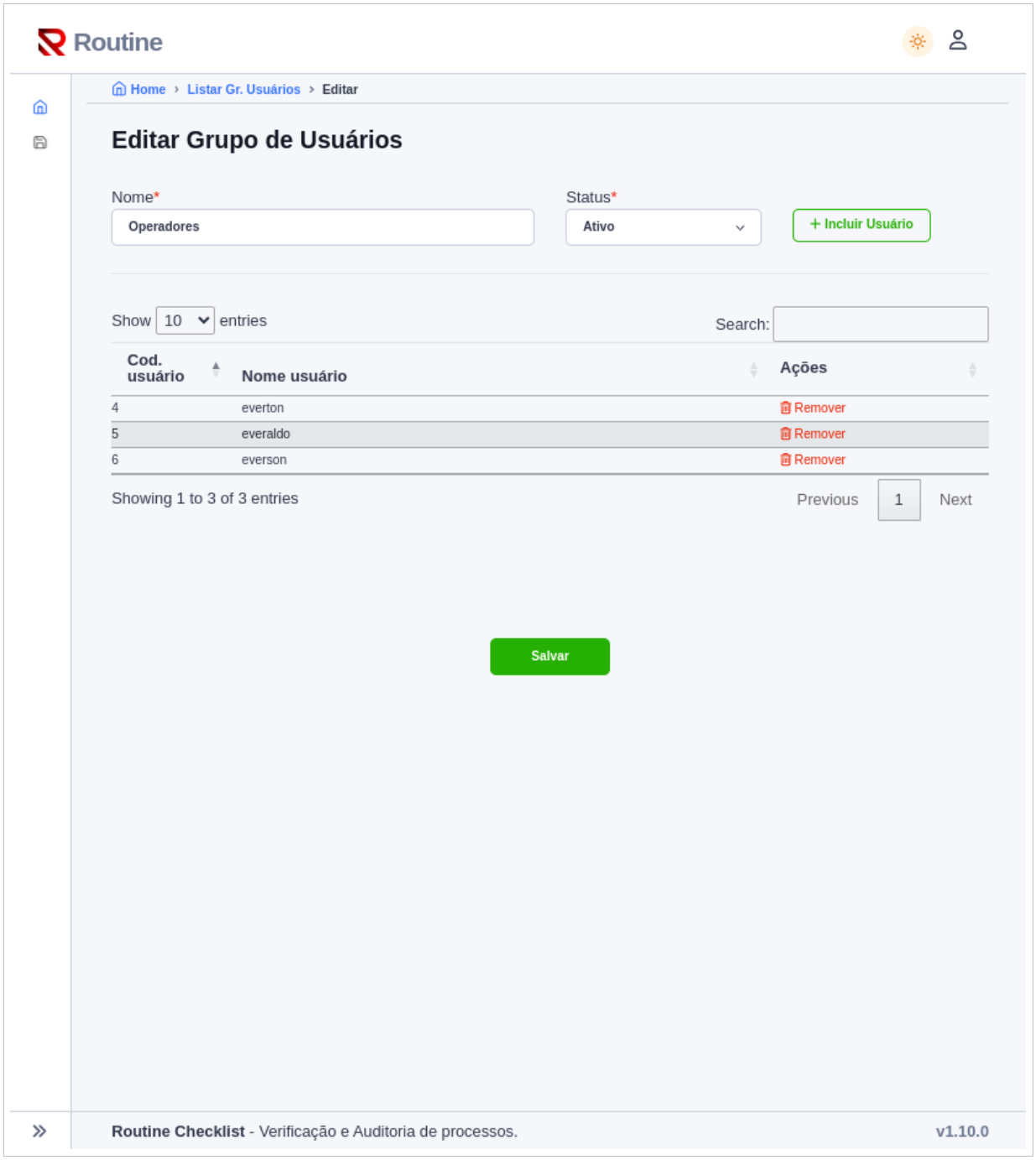
Figura 15 – Tela cadastro de unidade da aplicação web



Fonte: Autoria própria (2024).

Outra tela de extrema importância é a de cadastro de grupos de usuários. Essa funcionalidade possibilita agrupar usuários baseado em grupos, isso torna viável a restrição de acesso a *checklists* e funcionalidades específicas do sistema. Esta etapa de cadastro de grupos de usuários pode ser visualizada na Figura 16.

Figura 16 – Tela cadastro de grupo de usuários da aplicação web

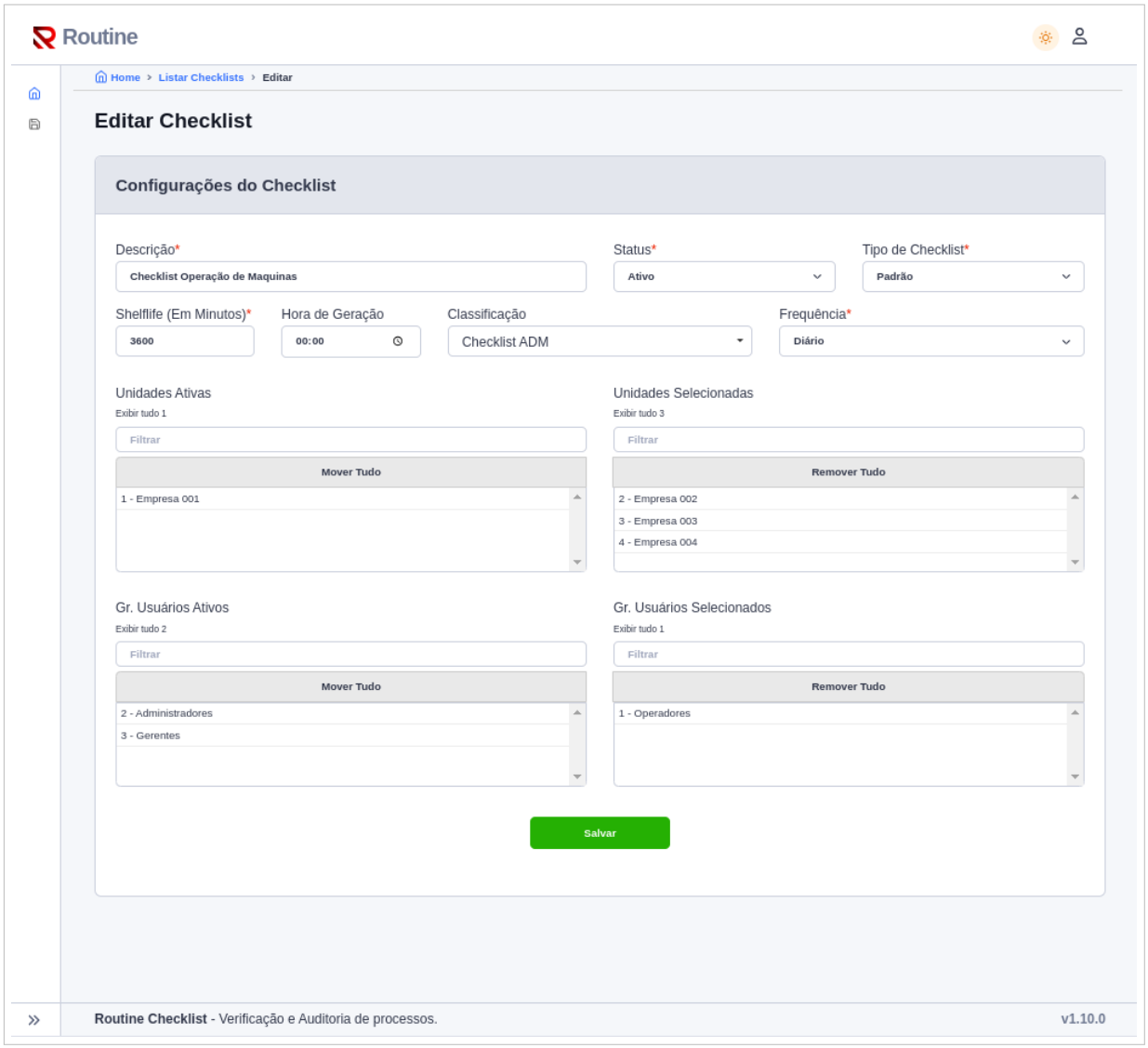


Fonte: Autoria própria (2024).

O ponto central e de máxima relevância reside no cadastro de *checklists*, uma vez que engloba as configurações e especificações essenciais relacionadas à criação e ao processa-

mento das tarefas. Todas as configurações desta tela são cruciais, e podem ser visualizadas na Figura 17, que exibe além dos campos de configurações importantes, os relacionamentos com os cadastros de Classificação, Unidades e Grupos de usuários.

Figura 17 – Tela cadastro de *Checklist* da aplicação web



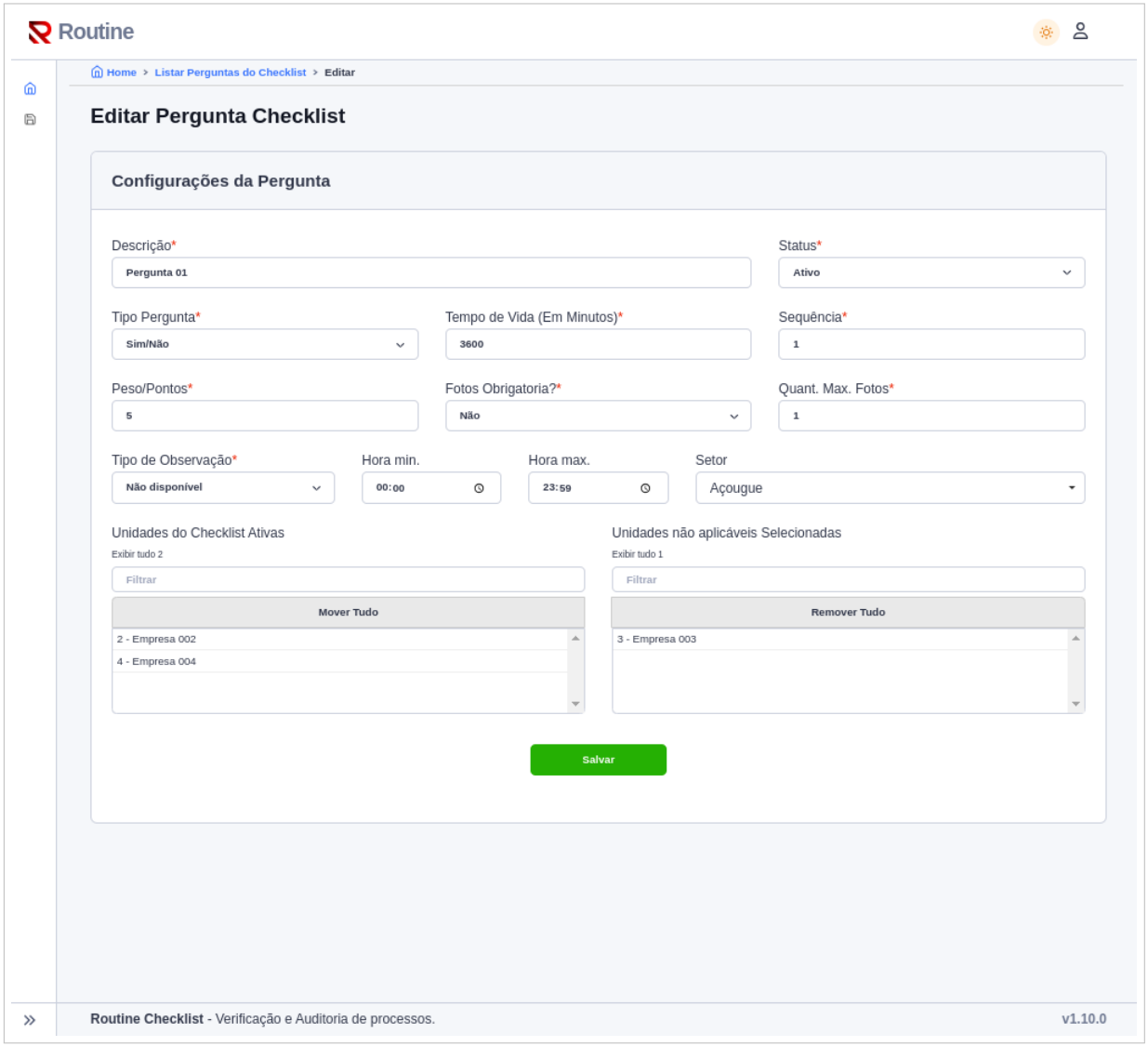
Fonte: Autoria própria (2024).

As perguntas, por sua vez, assumem um papel de extrema importância, complementando o cadastro dos *checklists*. Conforme a regra estabelecida, a ausência do cadastro de perguntas em um *checklist* impede a geração de tarefas para a execução. Portanto, a relevância deste cadastro é evidente. Além da própria pergunta, o processo de cadastro desse recurso também envolve configurações e especificações que influenciam diretamente a geração de tarefas.

A representação visual desta tela pode ser vista na Figura 18, onde são exibidos os campos necessários para o cadastro de uma pergunta e os relacionamentos com os cadastros de Setor e Unidade. Vale salientar que o relacionamento de unidade no cadastro de uma

pergunta é utilizado exclusivamente para não gerar uma pergunta específica em uma tarefa. Por definição, todas as perguntas são geradas para as unidades associadas a um *checklist*. No entanto, esse cadastro de exclusão permite maior flexibilidade ao cadastrar um *checklist*, permitindo que, para uma filial específica onde uma determinada pergunta não faça sentido, seja possível cadastrar um único *checklist* que atenda às individualidades de todas as lojas.

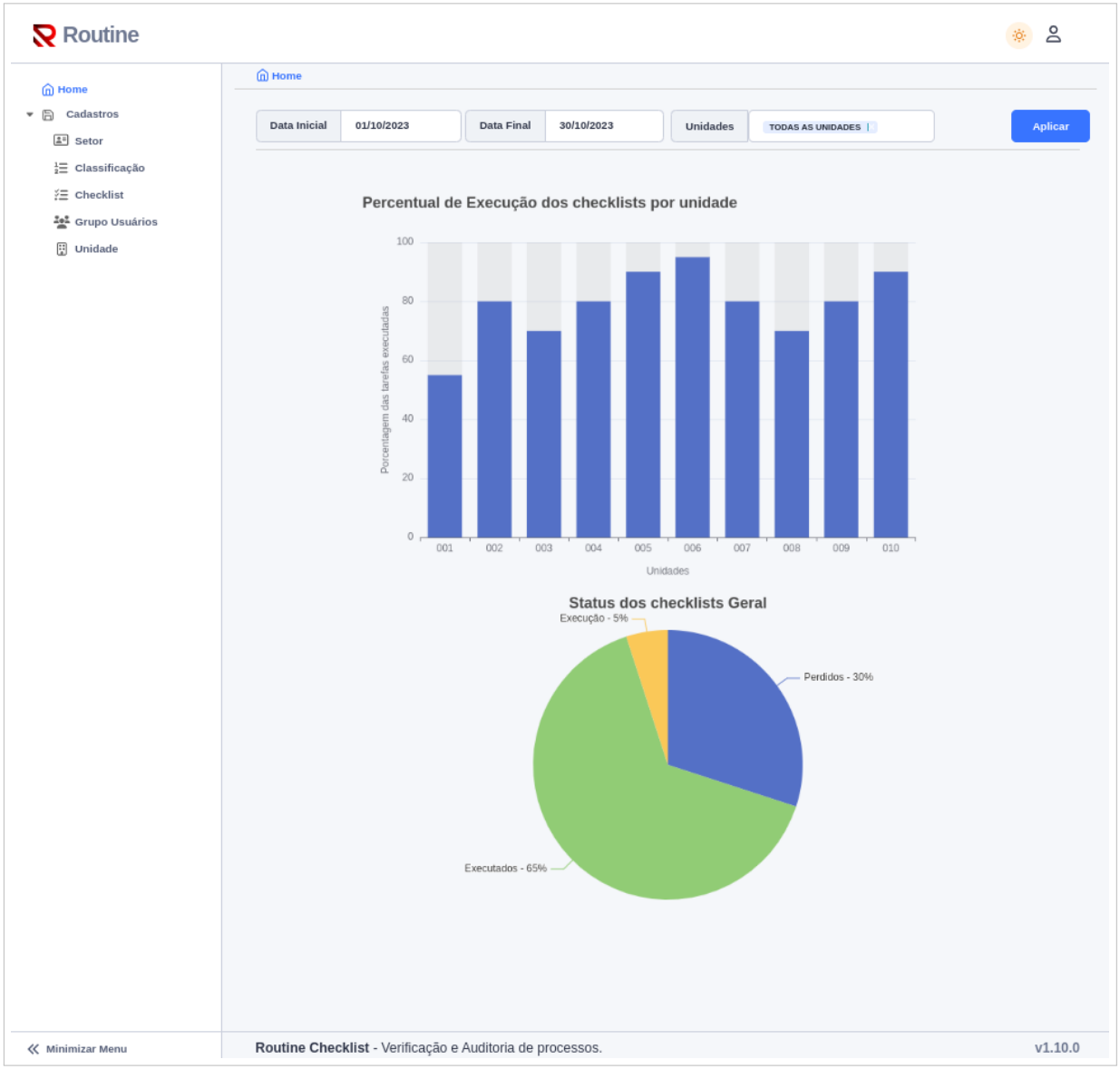
Figura 18 – Tela Cadastro de Perguntas do *Checklist* da aplicação web



Fonte: Autoria própria (2024).

Considerando os procedimentos de criação e encerramento de tarefas, a aplicação irá gerar informações acerca dos processos executados. Isso possibilitará a elaboração de relatórios essenciais para a gestão e aprimoramento dos processos. A Figura 19 ilustra uma visão prospectiva da aparência da tela inicial da aplicação web após o início da execução das tarefas de *checklists* pelos usuários.

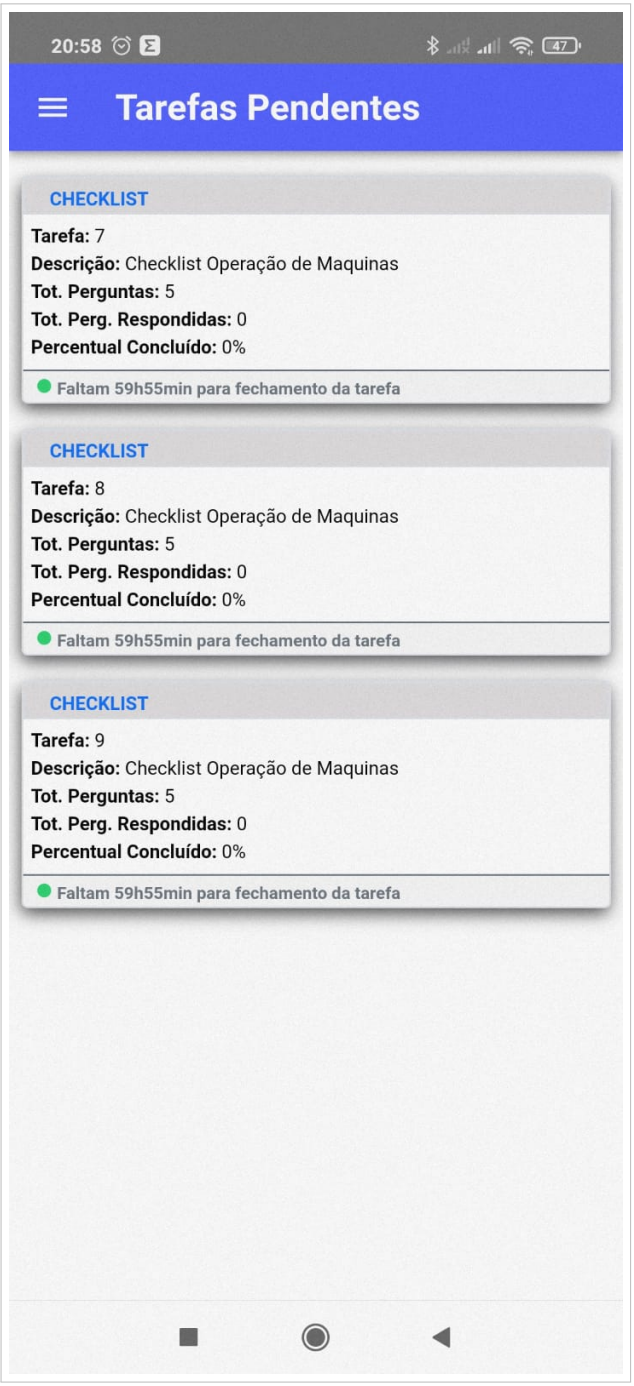
Figura 19 – Tela inicial da aplicação web



Fonte: Autoria própria (2024).

Seguindo com o desenvolvimento da fase de planejamento, visando possibilitar a execução e respostas das tarefas de *checklist*, foi imperativo o desenvolvimento de uma aplicação móvel que fosse capaz de manter comunicação com o sistema de retaguarda por meio de uma API. Esta aplicação, construída com Ionic Framework para a plataforma Android, teve a responsabilidade de receber tarefas exclusivamente em modo online, para a execução da tarefa de *checklist*. No que tange à representação visual e contextualização desta aplicação, destacamos, como ponto de partida, o protótipo da tela inicial do APP, que é responsável por apresentar as tarefas pendentes para o usuário, conforme ilustrado na Figura 20.

Figura 20 – Tela Inicial de tarefas pendentes do aplicativo móvel



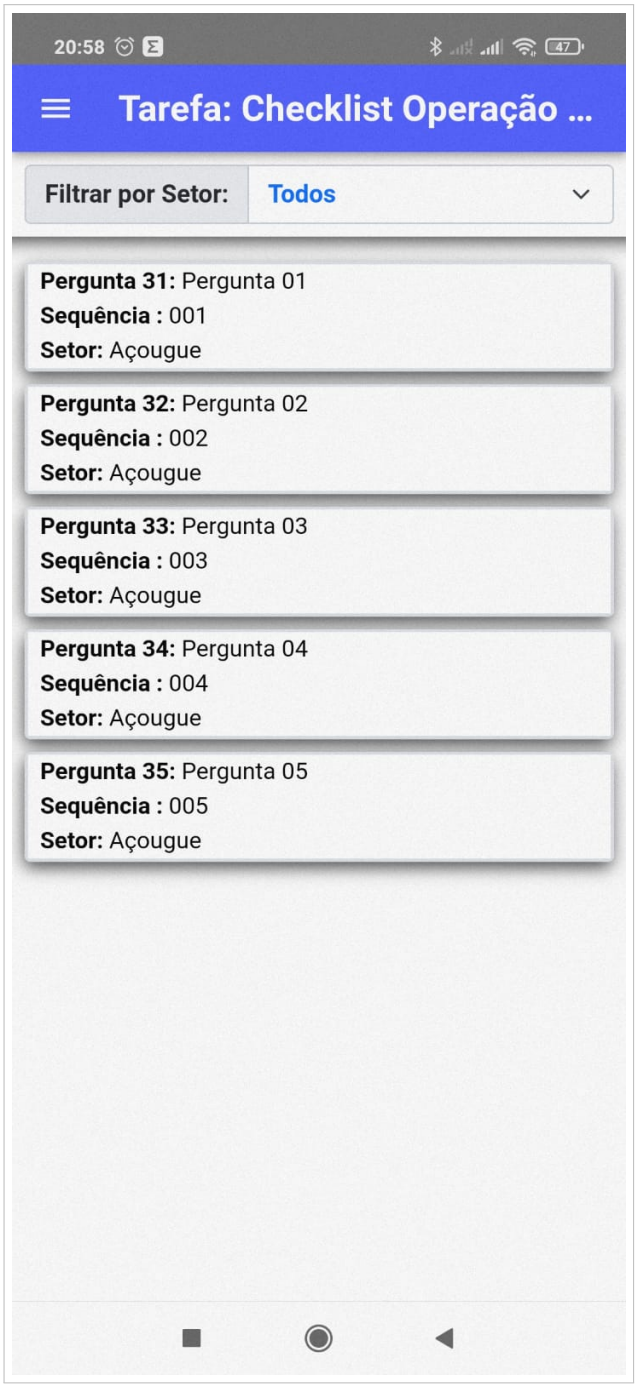
Fonte: Autoria própria (2024).

Após a seleção de um *checklist* para execução, é crucial que ele seja vinculado ao usuário que o selecionou. Dessa maneira, a tarefa torna-se indisponível para outros usuários até que seja devidamente liberada pelo usuário responsável e sem que haja a finalização dessa tarefa. Esse procedimento permite um controle preciso e uma gestão eficaz das tarefas, contribuindo para o funcionamento ordenado das rotinas operacionais.

Uma vez selecionado uma tarefa de *checklist* para a execução, é apresentada a tela de perguntas que compõe a tarefa, conforme ilustra a Figura 21. Nesta imagem é possível

perceber que existe um filtro na barra superior. Este filtro refere-se ao cadastro de Setor, onde o usuário pode filtrar as perguntas de um determinado setor e responde-las de forna ordenada, possibilitando maior agilidade na execução de uma tarefa.

Figura 21 – Tela de exibição das perguntas de um *checklist* do aplicativo móvel

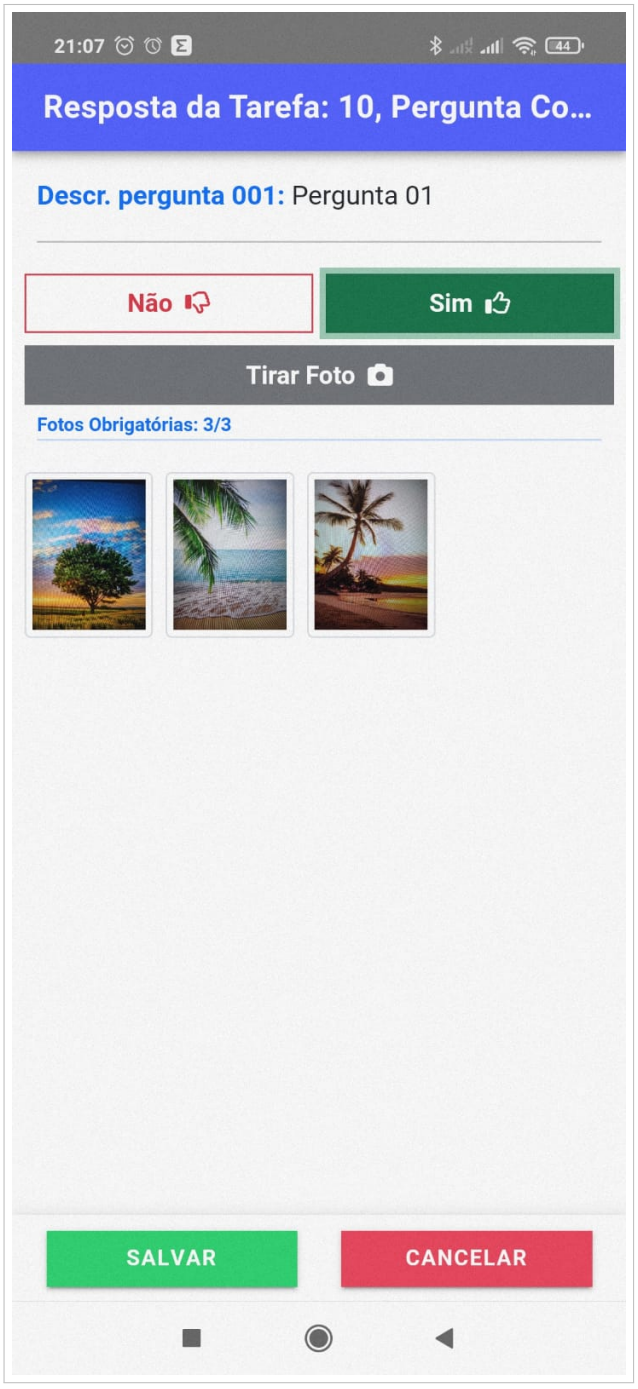


Fonte: Autoria própria (2024).

Além disso, logo após a seleção de uma pergunta pelo usuário, é fundamental que uma tela de resposta seja apresentada, exibindo os itens obrigatórios e opcionais. Esse processo é claramente exemplificado na Figura 22 que demonstra essa funcionalidade em ação. Na imagem apresentada, é possível visualizar uma pergunta objetiva que exige três fotos que devem ser

tiradas no momento da execução da resposta da tarefa. Esta imagem obrigatoriamente deve ser registrada através da câmera do aparelho. Através de controles internos o *Routine Checklist*, bloqueia que seja anexado uma imagem a partir da galeria do aparelho, garantindo dessa forma maior controle e garantia da execução do processo desejado.

Figura 22 – Tela para resposta de uma pergunta do aplicativo móvel



Fonte: Autoria própria (2024).

5 DESENVOLVIMENTO DO SISTEMA

Neste capítulo, são detalhadas as fases do desenvolvimento desde o início da codificação da aplicação web e também do aplicativo móvel. Cada fase será minuciosamente descrita, destacando as atividades, funcionalidades implementadas e as dificuldades encontradas na execução deste projeto. Esta abordagem permitirá uma compreensão aprofundada do progresso e das técnicas empregadas, evidenciando a evolução do desenvolvimento desde o projeto da infraestrutura, análise de requisitos e integração entre as tecnologias necessárias, que teve início na primeira fase de desenvolvimento, conforme apresentado com detalhes no Capítulo 4, até a finalização e entrega do projeto com a conclusão da quinta fase.

5.1 Desenvolvimento da aplicação web

No estágio atual de desenvolvimento, iniciou-se a segunda fase, que abordou a codificação da estrutura lógica e das regras de negócio. Este marco foi caracterizado pela inclusão dos modelos que representam as informações geradas pelos processos a serem armazenadas na base de dados, acompanhados pelas migrações correspondentes para cada modelo. As migrações, por sua vez, consistem nas definições de campos, tipagens e regras que asseguram a integridade e confiabilidade dos dados armazenados no banco de dados. Essas definições incluem a imposição de restrições nas tabelas do banco de dados, garantindo a consistência dos dados ao longo do sistema. Esta estrutura inicial foi concebida tendo como base o protótipo do modelo lógico de dados apresentado na seção 4.2.

Após a codificação dos modelos e das migrações iniciais, foi possível iniciar o desenvolvimento da estrutura inicial da aplicação. Primeiramente, implementou-se a autenticação na área gerencial web, seguida pela implementação da autenticação e geração de *tokens* pela API RESTful, responsável pela interface de comunicação entre o servidor e o aplicativo móvel. Neste estágio, foi criado um modelo estruturado de resposta para a API, visando padronizar a comunicação entre o servidor remoto e o aplicativo Android.

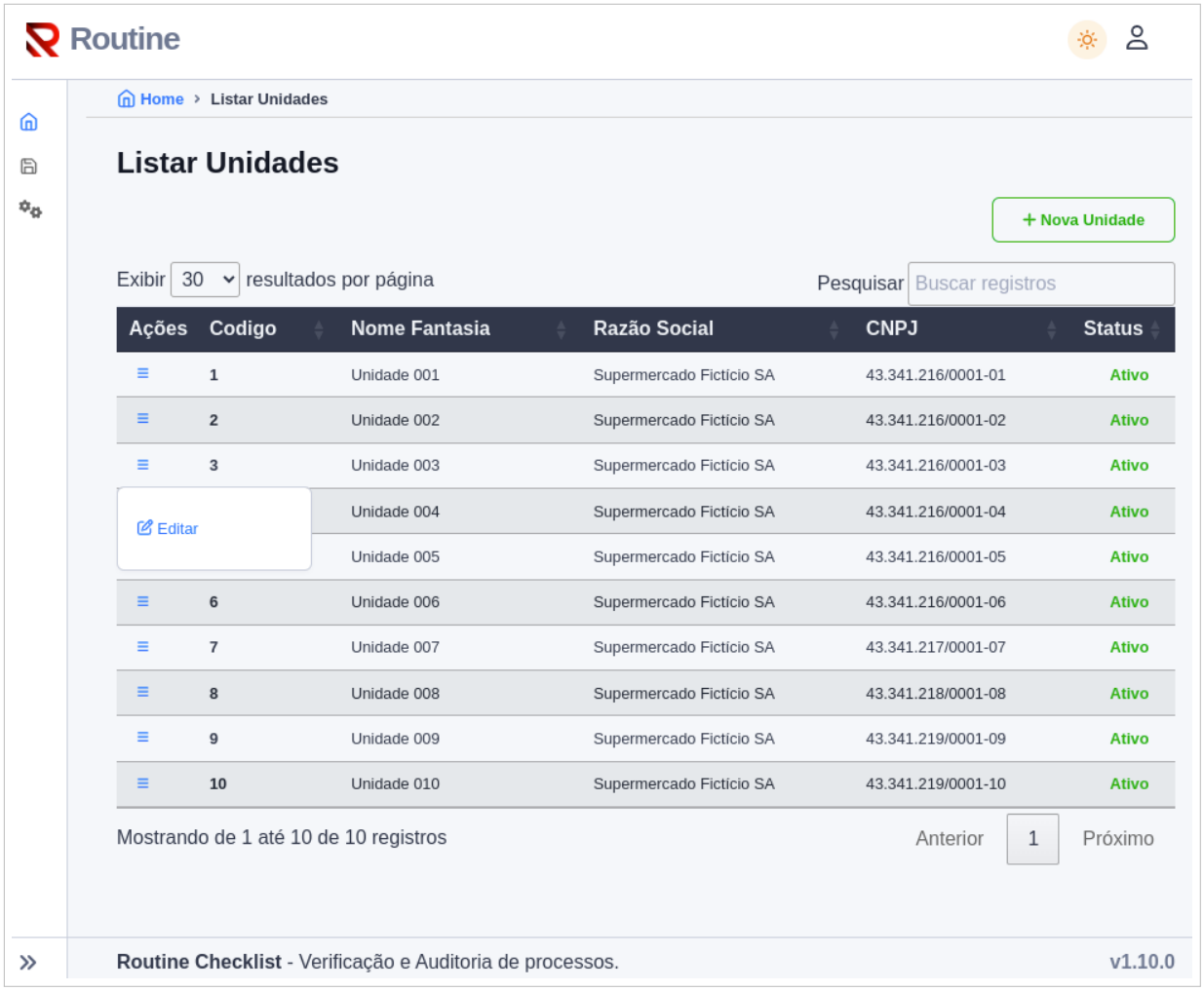
Do lado do aplicativo, concebeu-se um modelo de comunicação centralizado para processar as requisições e respostas da API. Essas medidas permitiram que, desde o início do trabalho de codificação, fossem estabelecidas diretrizes claras para a interação do aplicativo móvel com o servidor do *Routine Checklist*. Embora o desenvolvimento do aplicativo tenha ocorrido em paralelo ao da aplicação web, a abordagem do desenvolvimento do aplicativo será abordada com mais detalhes na seção 5.2.

Uma vez estabelecido o sistema de comunicação e segurança, iniciou-se a codificação das interfaces para interação com o usuário, que compreendem as telas e artefatos onde o usuário pode interagir de fato com a aplicação do *Routine Checklist*. As primeiras telas desenvolvidas foram as mais simples e básicas. Essa estratégia visou definir e agilizar o processo de desenvolvimento com o *template* adotado e definir os padrões de codificação que nortea-

ram o sistema como um todo. Assim, as primeiras telas desenvolvidas foram as interfaces para o cadastro de Setor, Classificação e Unidade. Estas telas não tiveram mudanças em relação aos protótipos desenvolvidos na fase de projeto, porém foram estabelecidas novas telas, com o intuito de definir um fluxo padrão para a inclusão, edição e listagem dos registros cadastrados.

Tomando as telas de cadastro de unidades como exemplo da estrutura base da aplicação web, o fluxo de gestão dos cadastros seguiu este processo: Tela de listagem dos registros com a possibilidade de acesso à tela de inclusão através do botão `Nova Unidade` no canto superior direito da tela. Através da primeira coluna da tabela de listagem dos registros cadastrados, é possível acessar as outras funções para a manipulação dos cadastros de acordo com as funcionalidades implementadas para cada tela. Um exemplo da tela de listagem segue na Figura 23, que apresenta de forma padronizada para todos os cadastros.

Figura 23 – Tela gerenciamento de unidades

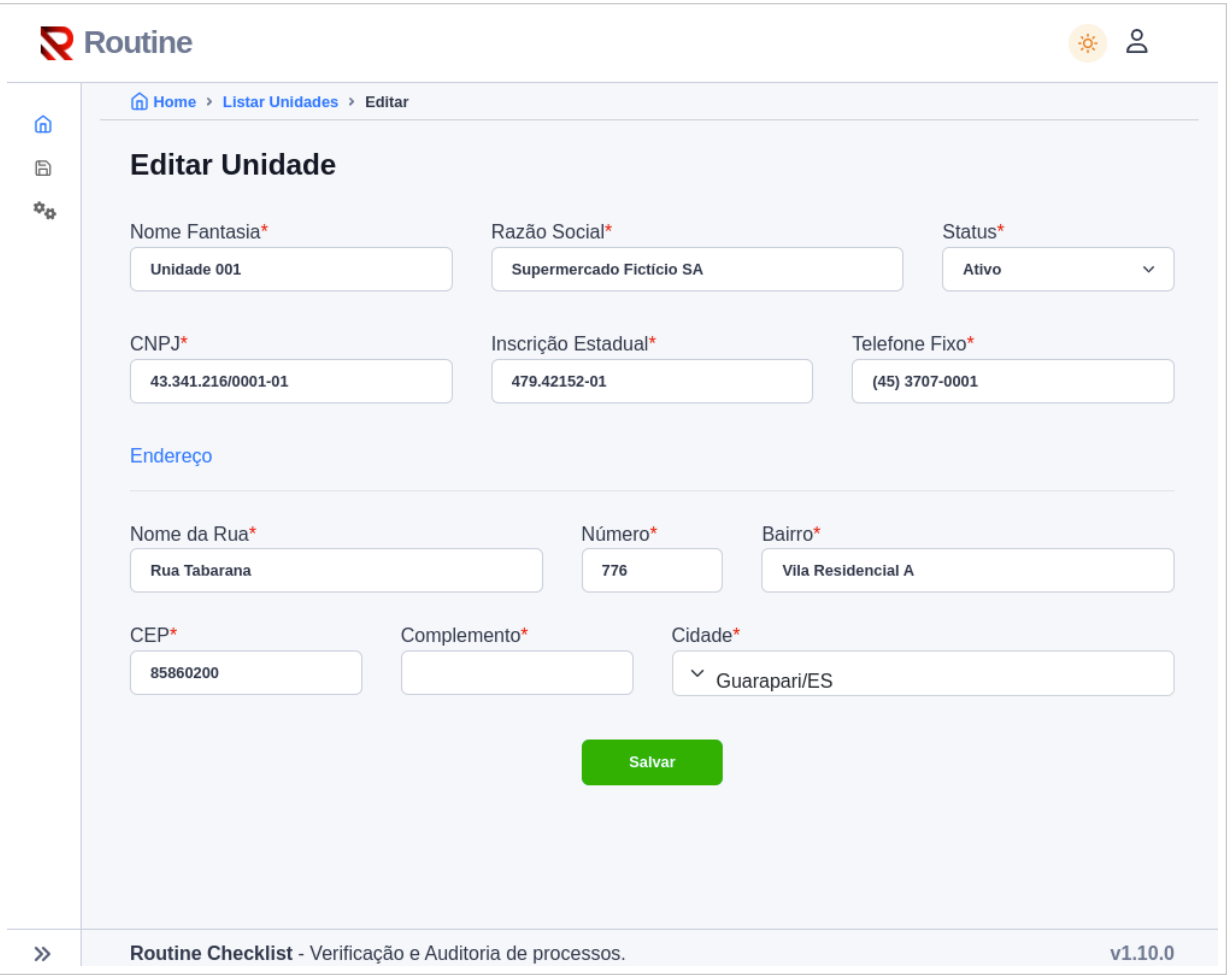


Fonte: Autoria própria (2024).

Por outro lado a tela de edição de unidade, representada na Figura 24, demonstra o resultado do desenvolvimento desta tela , e o padrão adotado para a criação dos cadastros do *Routine Checklist*. Vale destacar, que o relacionamento entre Unidade e Cidade presente no

cadastro de endereço de uma unidade, não foi concebido através de um cadastro convencional. Pois o cadastro das Unidades e Estados, foram obtidos através da sincronização junto ao cadastro fornecido pela API do IBGE. Desta forma, temos no *Routine Checklist*, o cadastro de todas as localidades e estados brasileiros, que pode ser sincronizado sempre que necessário.

Figura 24 – Tela edição de unidade



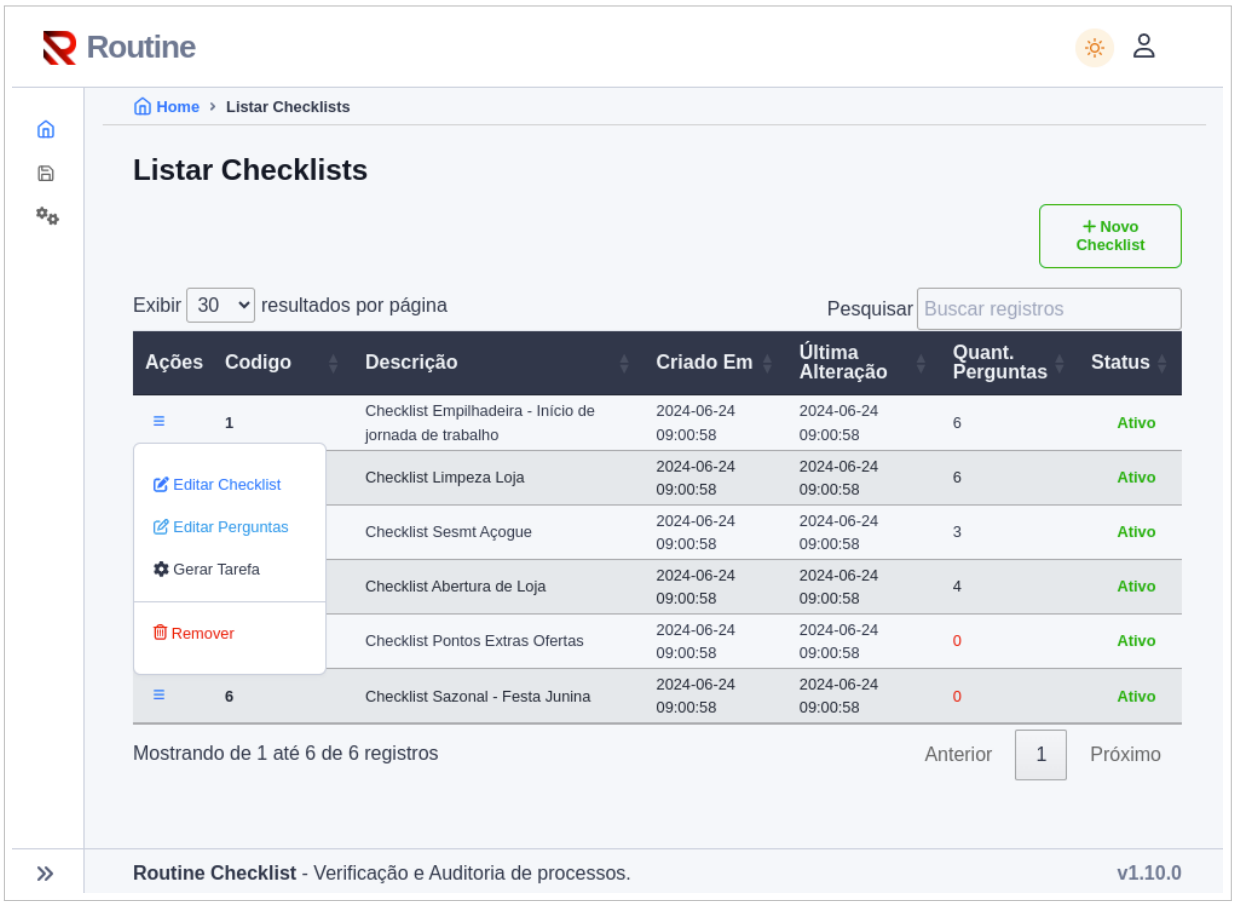
Fonte: Autoria própria (2024).

Outro cadastro fundamental de destaque foi a criação efetiva da tela gerenciamento de *checklist*, representada na Figura 25. Esta tela desempenha um importante papel na gestão de *checklists* e criação de tarefas, atuando como uma importante engrenagem nesse processo. A tabela apresenta na primeira coluna, um menu de ações que permite executar vários comportamentos em um *checklist* específico. Além disso, pode-se observar que a lista de opções inclui o cadastro de perguntas associadas ao *checklist*, de modo que o acesso às perguntas só pode ser feito através do próprio *checklist*.

Adicionalmente, a tela conta com a opção de deleção de um *checklist*. No entanto, para manter a integridade do banco de dados, caso o *checklist* já esteja associado a uma tarefa, a deleção do registro ocorre por meio do recurso *soft delete*, uma funcionalidade Laravel que permite marcar um registro como excluído sem removê-lo de fato da base de dados.

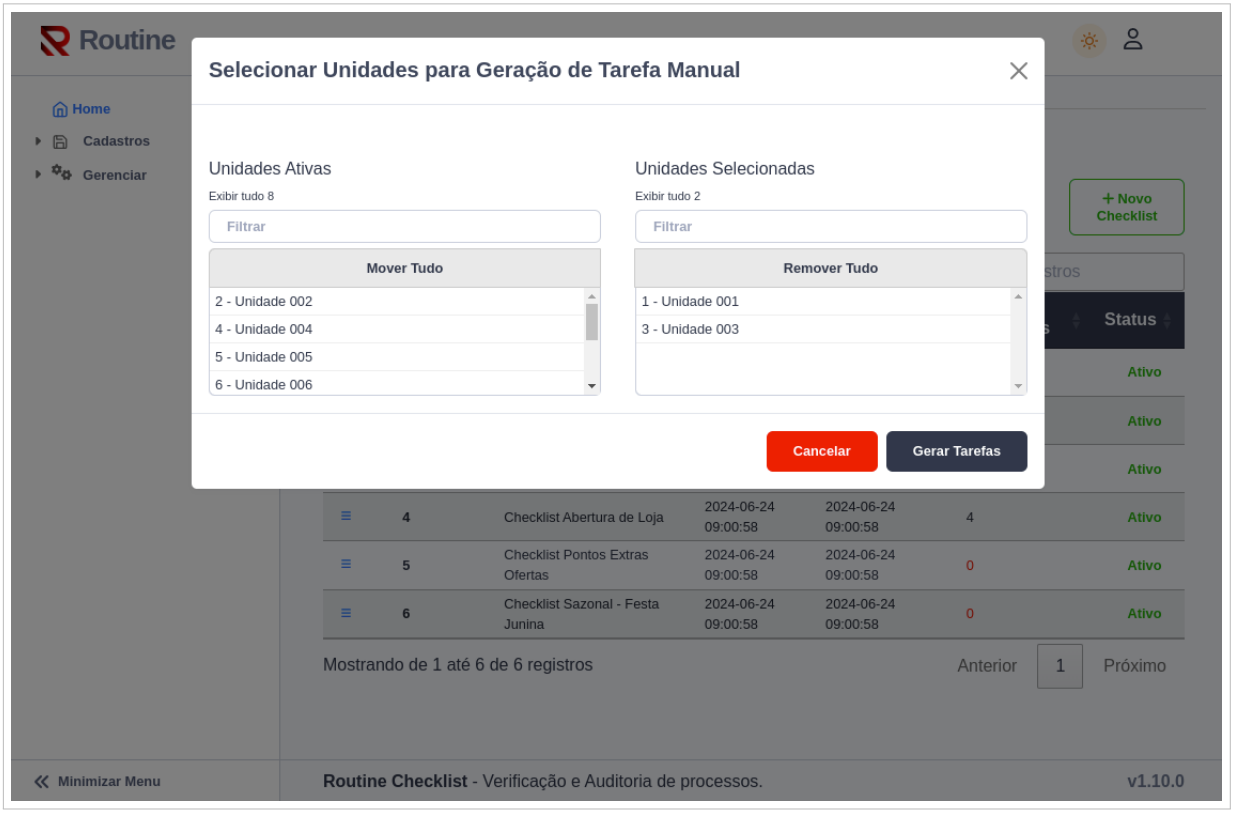
Por fim, foi incorporado na opção `Gerar Tarefa`, a capacidade de gerar manualmente as tarefas de *checklist*. Para este recurso, foi adicionalmente incluído a possibilidade de seleção das unidades desejadas para este procedimento como pode ser observado na Figura 26. Essa funcionalidade acrescentou um nível mais refinado de gerenciamento de tarefas ao sistema, proporcionando ao gestor ou auditor dos processos, maior flexibilidade para aplicar um *checklist* conforme necessário, sem precisar modificar as configurações previamente programadas ou desconfigurar um planejamento existente.

Figura 25 – Tela gerenciamento de *checklists*



Fonte: Autoria própria (2024).

Figura 26 – Tela de seleção de unidades para gerar tarefas manuais



Fonte: Autoria própria (2024).

A tela de edição ou inclusão de um *checklist*, manteve-se fiel ao protótipo desenvolvido no Capítulo 4. No entanto, em relação à funcionalidade, devido às restrições de tempo, não foi possível implementar a criação de *checklists* de acordo com o tipo, inicialmente idealizado. Através dessa opção, desejava-se incorporar outras modalidades de tarefas, como *checklist* surpresa, que poderiam ser gerados aleatoriamente, com a finalidade de auditar um setor sem aviso prévio para identificar inadequações e situações que não seriam observadas em um *checklist* convencional. No entanto, essa funcionalidade foi suprimida do projeto ficando para o desenvolvimento de trabalhos futuros.

Após a implementação da estrutura do sistema e dos cadastros essenciais para a operação, seguiu-se para o desenvolvimento da camada de processamento que traria dinamismo ao *Routine Checklist*. Essa etapa, caracterizou a terceira fase do desenvolvimento e envolveu a criação do processamento em segundo plano do sistema.

Inicialmente, concentrou-se na elaboração da estrutura responsável por gerar automaticamente as tarefas de *checklists* com base nos registros ativos disponíveis no banco de dados, e nas configurações definidas para cada registro. Com isso, conseguimos automatizar a geração das tarefas de acordo com as necessidades do cliente, sem a necessidade de intervenção humana. Após o processamento de criação, as tarefas são armazenadas no banco de dados e aguardam o momento programado para serem apresentadas ao usuário responsável pela execução.

O motor de execução automatizado foi concebido com base no conceito de reflexão, que consiste em um método que executa determinado procedimento sendo chamado dinamicamente pela aplicação. Para colocar em prática esse procedimento, foi utilizado como base, uma estrutura de *Enum*, que lista os tipos de processos que o *Routine Checklist* é capaz de processar. Essa abordagem tornou a execução dos procedimentos automatizados altamente flexível. Embora a implementação possa parecer complexa em um primeiro momento, ela oferece grande versatilidade para a inclusão de novas funcionalidades e rotinas de processamento, bastando adicionar novos itens ao *Enum* e implementar os métodos correspondentes para processar as funcionalidades adicionais. Como resultado, o sistema torna-se capaz de incluir e processar automaticamente, as novas rotinas na execução em segundo plano.

Uma demonstração da lógica aplicada na execução automatizada é apresentada a seguir. No código da Listagem 1, é exibida a estrutura do *Enum* que define os tipos de processos disponíveis. Já no código da Listagem 2, é exibida a classe responsável por iniciar processamento automatizado de cada rotina.

Esses trechos de código exemplificam como a flexibilidade e a versatilidade foram incorporadas ao sistema, permitindo a inclusão de novos tipos de processos e a execução automatizada de rotinas adicionais com facilidade e eficiência.

Listagem 1 – Enum dos Tipos de processos do *Routine Checklist*

```
1  <?php
2  namespace App\Enums;
3  use ArchTech\Enums\Options;
4
5  enum Routine: string {
6      use Options;
7      case TASK_CREATE = 'C';
8      case FINISH_EXPIRED_TASKS = 'F';
9
10     public function description(): string {
11         return static::getDescription($this->value);
12     }
13
14     public static function getDescription(int $value): string {
15         return match ($value) {
16             'C' => 'Criar Tarefas Checklist',
17             'F' => 'Finalizar Tarefas de Checklists Expiradas',
18         };
19     }
20 }
```

Fonte: Autoria própria (2024).

Listagem 2 – Processamento Automatizado

```php
<?php

namespace App\Console\Commands;

use App\Enums\Routine;
use App\Services\ChecklistMovService;
use App\Services\ChecklistService;
use Carbon\Carbon;
use Illuminate\Console\Command;
use Illuminate\Support\Facades\App;
use Illuminate\Support\Facades\Log;

class RunRoutineCommand extends Command {

  protected $signature='run:routines';
  protected $description='Executa o processamento de tarefas automatizado';

  public function handle() {
        $now = Carbon::now()->second(0)->millisecond(0);
        foreach (Routine::options() as $routineKey => $routineValue) {
            $runRoutine = '';
            try {
                $runRoutine="execute".$this->convertToCommand($routineKey);
                $this->$runRoutine($now);
            } catch (\Throwable $th) {
                Log::error('Erro ao Executar rotina: '.$th->getMessage());
            }
        }
  }

  private function convertToCommand(string $str){
        $command = str_replace("_", "", ucwords(strtolower($str), " /_"));
        return $command;
  }

  private function executeTaskCreate(Carbon $time){
        $checklistService = App::make(ChecklistService::class);
        $checklistService->processGenerateTasksChecklists();
  }

  private function executeFinishExpiredTasks(Carbon $time){
        $checklistMovService = App::make(ChecklistMovService::class);
        $checklistMovService->finishExpiredTasks($time);
  }
}
```

Fonte: Autoria própria (2024).

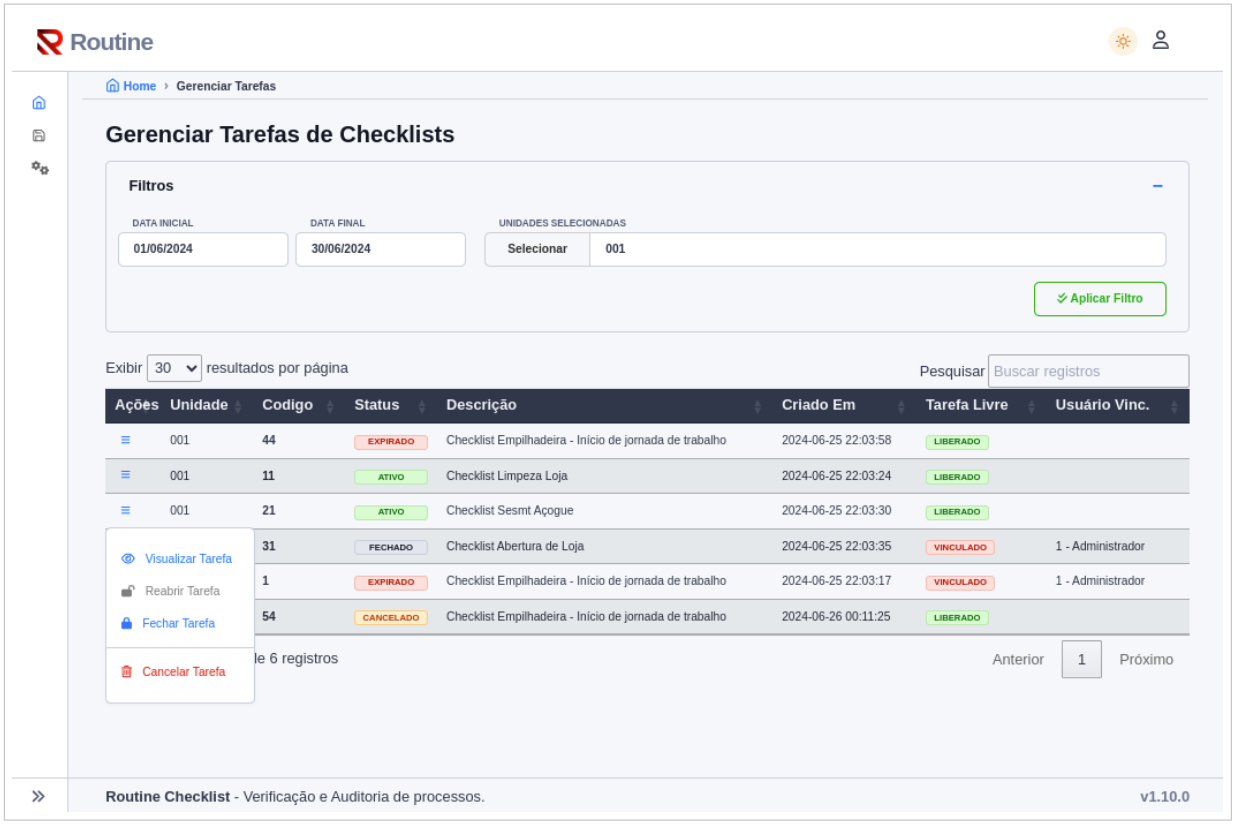
Neste estágio do projeto, obteve-se a implementação de um sistema plenamente funcional. Este sistema não apenas é capaz de gerar tarefas, mas também oferece ao usuário a

capacidade de executá-las de maneira eficiente. Além disso, integrou-se o processamento em segundo plano capaz de automatizar a criação e encerramento de tarefas expiradas, de acordo com as configurações especificadas para cada *checklist*.

Com isto, concentrou-se esforços na elaboração de relatórios permitindo aos usuários ou auditores dos processos, acompanhar e analisar os trabalhos realizados. A partir da tela de gerenciamento de tarefas, representada na Figura 27, os usuários administradores podem gerenciar as tarefas de acordo com com o status de cada tarefa. Para esclarecer melhor as funcionalidades disponíveis nesta tela, apresentamos a seguir as variações dos procedimentos disponíveis, que podem ser executados nas tarefas de acordo com o status:

- O fechamento de uma tarefa pode ser executado quando o status encontra-se ativo;
- Para reabrir uma tarefa, ela deve estar com status Expirado ou Cancelado;
- O cancelamento de uma tarefa, aplica-se aos status de Expirado ou Ativo;
- A visualização do relatório de uma tarefa, pode ser acessado independente do status em que se encontra.

Figura 27 – Tela de gerenciamento das tarefas de *checklist* - WEB



Fonte: Autoria própria (2024).

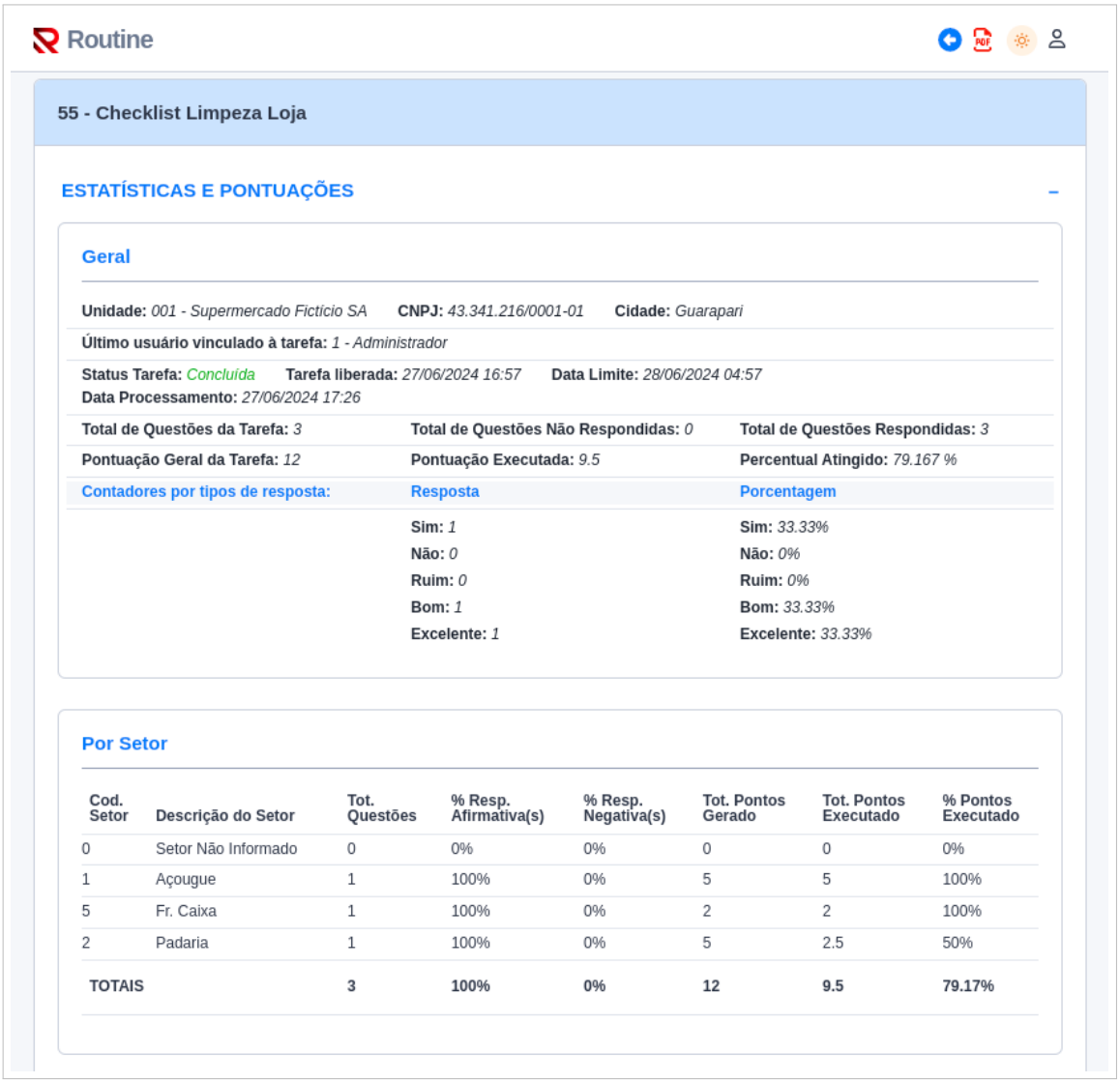
A funcionalidade que permite visualizar no menu de ações permite acessar o relatório web, que fornece informações detalhadas sobre a execução das tarefas e a visualização das

respostas fornecidas. Devido à extensão do relatório no navegador, a representação foi dividida em duas partes.

A Figura 28 apresenta a visualização de um resumo geral referente à tarefa executada. Esse resumo permite análises baseadas na execução das perguntas em um contexto geral, além de um resumo dividido por setor. Essa divisão possibilita uma análise mais refinada, considerando as peculiaridades de cada setor e permitindo a comparação das pontuações de acordo com o cadastro das perguntas do *checklist*.

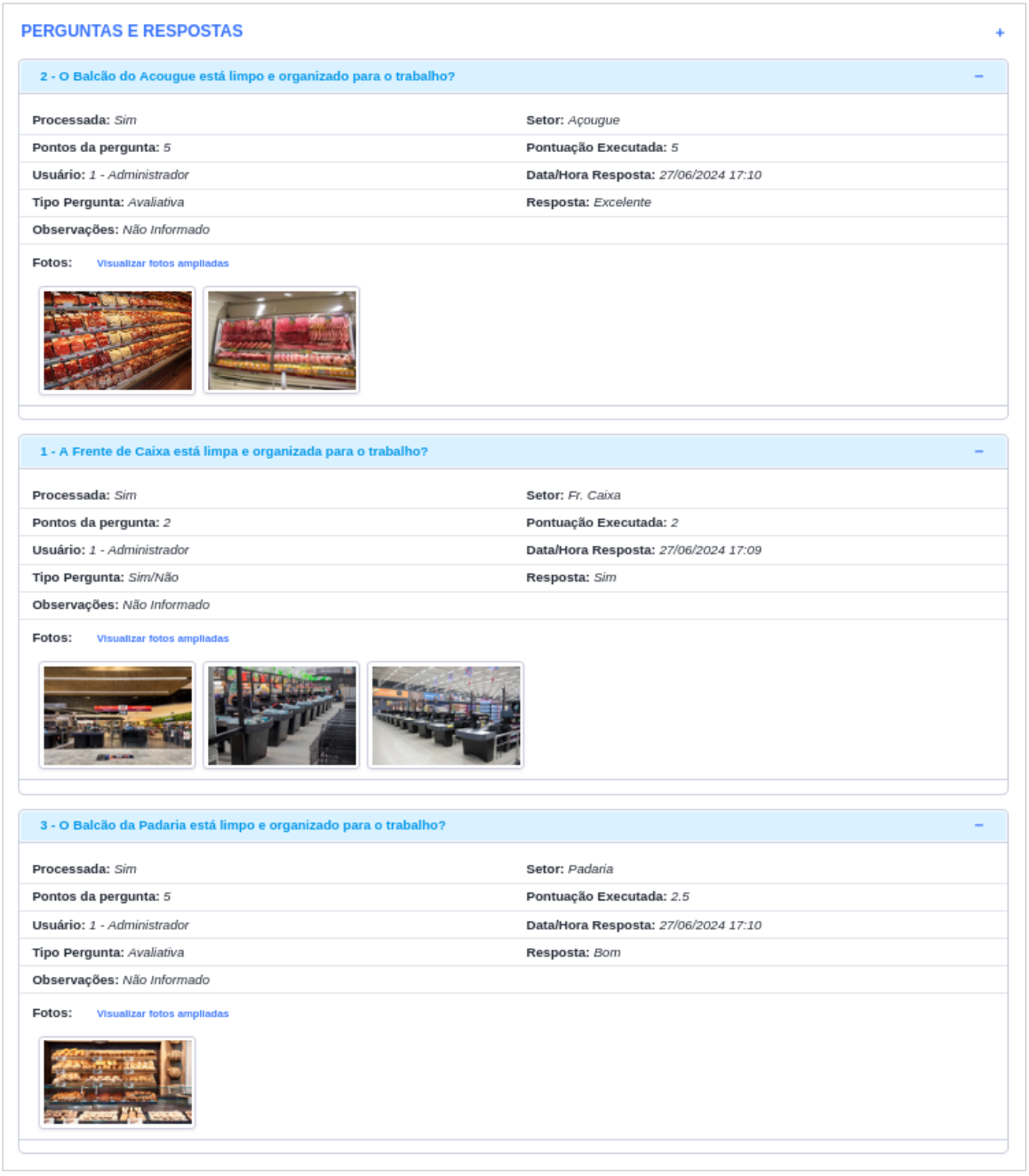
Na segunda parte do relatório, representada pela Figura 29, são apresentadas as perguntas da tarefa e as respostas fornecidas pela operação. Isso permite uma abordagem mais específica, onde o auditor pode verificar a resposta, o usuário que executou a pergunta, fotos registradas e demais campos preenchidos durante a execução da tarefa. Adicionalmente, nesta tela, o usuário pode optar por baixar o relatório em formato PDF (uma amostra do relatório gerado pode ser visualizada no Apêndice B).

Figura 28 – Estatísticas e pontuações do relatório web de execução de uma tarefa



Fonte: Autoria própria (2024).

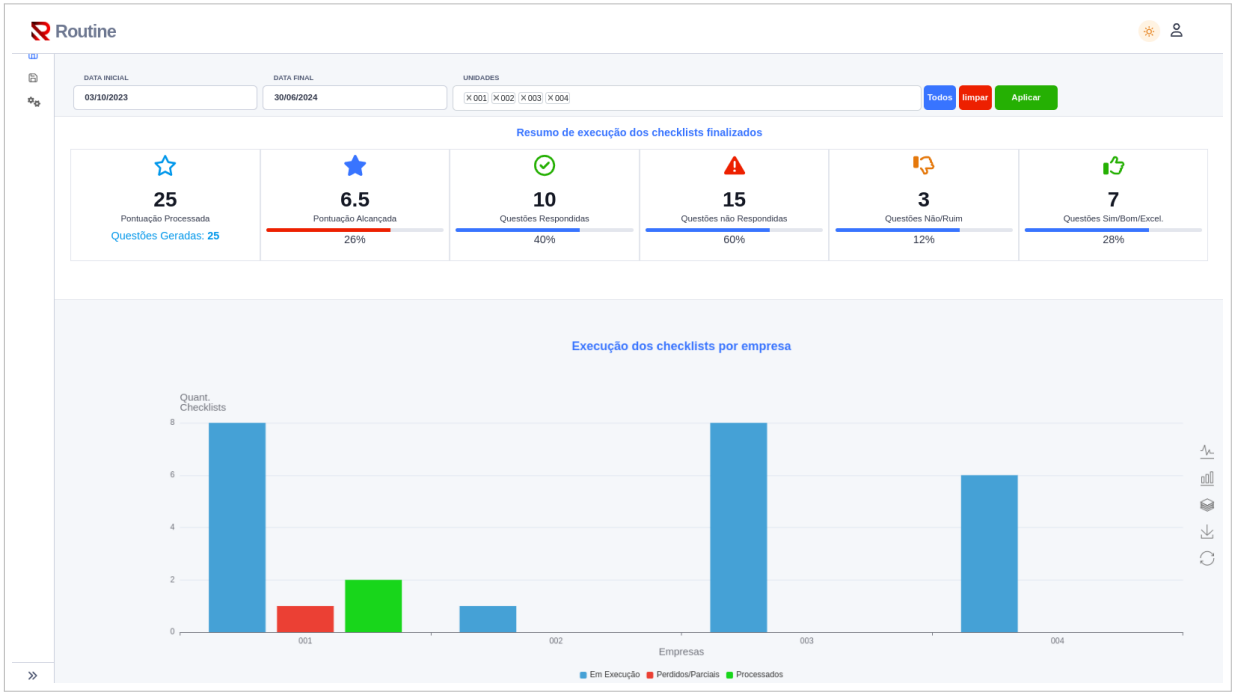
Figura 29 – Perguntas e respostas do relatório web de execução de uma tarefa



Fonte: Autoria própria (2024).

Também foram realizados ajustes na tela inicial da aplicação web, com base no protótipo apresentado na Figura 19 do Capítulo 4. Nesta tela, foram desenvolvidos os gráficos para a comparação da execução das tarefas. Esses gráficos permitem que o auditor compare os status das tarefas entre as unidades, proporcionando uma visão ampla da execução dos processos entre as diversas unidades da empresa. A nova tela inicial, com as melhorias implementadas, pode ser visualizada na Figura 30.

Figura 30 – Tela inicial da aplicação web



Fonte: Autoria própria (2024).

Na fase final do desenvolvimento da aplicação, foram conduzidos diversos testes de usabilidade pelo autor, utilizando o aplicativo android para a execução das tarefas. O objetivo desses testes foi detectar possíveis falhas e inconsistências que poderiam surgir ao submeter as respostas a um servidor remoto. Dentre os testes manuais aplicados, foram analisados:

- Tempo de resposta de uma pergunta objetiva simples e também com envio de fotos: Esses testes visavam medir a eficiência do sistema em processar e armazenar diferentes tipos de respostas;
- Tempo de carregamento das tarefas disponíveis para o usuário: Verificou-se a velocidade com que as tarefas eram apresentadas ao usuário após a solicitação;
- Tempo de carregamento das perguntas de um *checklist*: Avaliou-se a rapidez com que as perguntas eram carregadas na interface do usuário.

Os resultados desses testes denotaram que o tempo de processamento e carregamento das tarefas, foi similar quando comparado com as ferramentas citadas no Capítulo 3.

Além disso, foram realizados testes de criação de tarefas no modo manual através da tela de gerenciamento de *checklists*, e dos processos automatizados com rotinas que executam em segundo plano, bem como criação e fechamentos automatizados das tarefas. Esses testes foram fundamentais para garantir que todas as funcionalidades essenciais da aplicação estivessem operando corretamente e sem falhas.

Após a execução dos testes, foram realizados os ajustes finais necessários para garantir que a aplicação estivesse pronta para a entrega, marcando assim o encerramento do desenvolvimento da aplicação web. Esses ajustes incluíram:

- Revisão de código: Uma revisão de código minuciosa foi realizada para identificar e corrigir problemas de qualidade, como código duplicado, complexidade excessiva e possíveis erros;
- Otimização de performance: A performance da aplicação foi analisada e otimizada para garantir que ela seja eficiente e responda de forma adequada às solicitações dos usuários;
- Verificação de segurança: Uma análise de segurança foi conduzida para identificar vulnerabilidades e garantir que a aplicação estivesse protegida contra possíveis ataques.

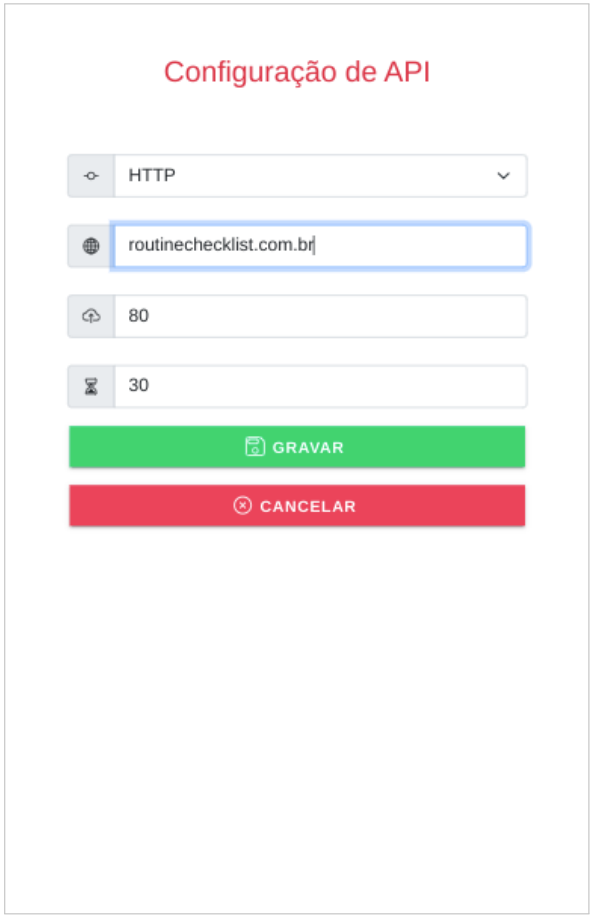
5.2 Desenvolvimento da aplicação móvel

O desenvolvimento do aplicativo móvel, foi executado em paralelo ao da aplicação web e API, conforme citado no autoseção 5.1. Por conta disso, a implementação do aplicativo móvel corresponde à segunda fase de desenvolvimento, onde foram implementadas as telas e principais funcionalidades das aplicações web, API e também para o aplicativo móvel (APP).

O início do desenvolvimento do aplicativo, foi marcado pela definição da tela de configuração inicial para a comunicação com a API, e o procedimento de autenticação para o aplicativo.

Primeiramente, a tela de configuração foi incluída junto a tela de login, onde ao relacionar o logotipo do *Routine Checklist*, à chamada de uma tela de configuração, é apresentada ao clicar, redirecionando o usuário para a inclusão das configurações iniciais do APP, conforme ilustra a Figura 31. Nesta tela, devem ser informado o endereço do servidor e protocolo de comunicação que pode ser HTTP ou HTTPS, a porta para comunicação com o servidor, que por definição, será a porta padrão 80, e o tempo de resposta que o aplicativo deve esperar até que seja considerado uma requisição perdida.

Figura 31 – Tela de configuração para comunicação com a API

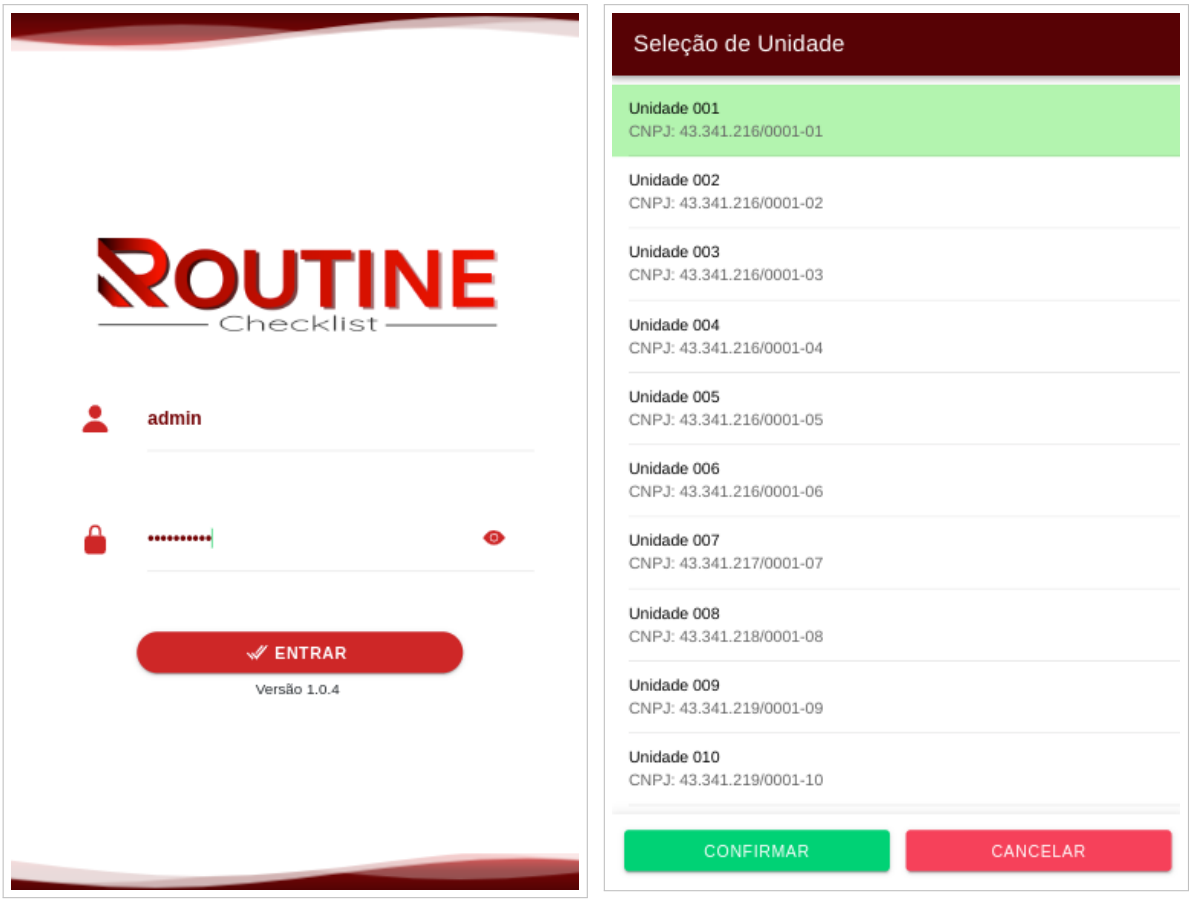


Fonte: Autoria própria (2024).

Logo após a definição da comunicação com a API, foi desenvolvido a funcionalidade através de duas etapas para realização da autenticação do usuário no aplicativo móvel. Primeiramente o usuário deve informar o login e senha conforme mostra a tela de login na Figura 32. Assim, o aplicativo solicita ao servidor através de uma requisição, enviando o usuário e senha, as unidades que o usuário tem acesso para efetuar o login. Uma vez recebido as unidades disponíveis, a seleção da unidade desejada deve ser executada conforme exibida na tela de seleção de unidades na Figura 32, para então a autenticação ser efetivada e o token de acesso ser gerado.

Figura 32 – Autenticação do usuário

(a) Tela de login **(b) Tela de seleção de unidade**



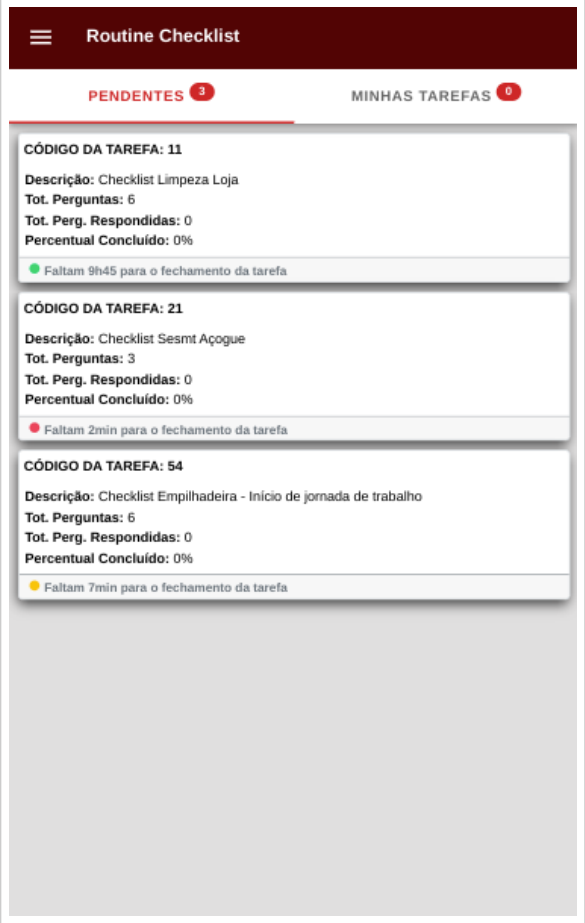
Fonte: Autoria própria (2024).

Após a autenticação realizada, o aplicativo carrega as tarefas pendentes para disponibilizando-as para a execução. No entanto, durante a implementação do APP, identificou-se a necessidade de inclusão de uma nova aba na tela inicial denominada `Minhas Tarefas`. Este novo recurso, destina-se a separar as tarefas as quais o usuário está vinculado, sinalizando que são de sua responsabilidade exclusiva e que já foram iniciadas. Isso permite ao usuário, organizar-se e dar prioridade às tarefas que já estão em andamento. As tarefas pendentes que ainda estão livres para serem vinculadas a um usuário, manteve-se sendo exibidas na aba `Pendentes`, conforme apresentado na tela inicial de exibição de tarefas da Figura 33.

Ainda na tela inicial, pode-se notar, no rodapé de cada tarefa, um importante indicador de tempo que mostra quanto falta para a expiração da tarefa. Esse recurso auxilia o usuário a priorizar as tarefas mais urgentes, que estão mais próximas do prazo final. Os indicadores utilizam cores para sinalizar a proximidade da expiração: o verde indica que a tarefa não está próxima de expirar e o tempo restante é igual ou superior a uma hora; o amarelo indica que faltam menos de 60 minutos para a expiração; e o vermelho, por sua vez, indica que a tarefa está a menos de 30 minutos de expirar.

Adicionalmente, destaca-se, além da alteração da disposição dos elementos na tela inicial, a padronização da cor do aplicativo para vermelho. Essa escolha permitiu a uniformização das cores nas aplicações web e móvel, criando uma identidade visual consistente para a ferramenta.

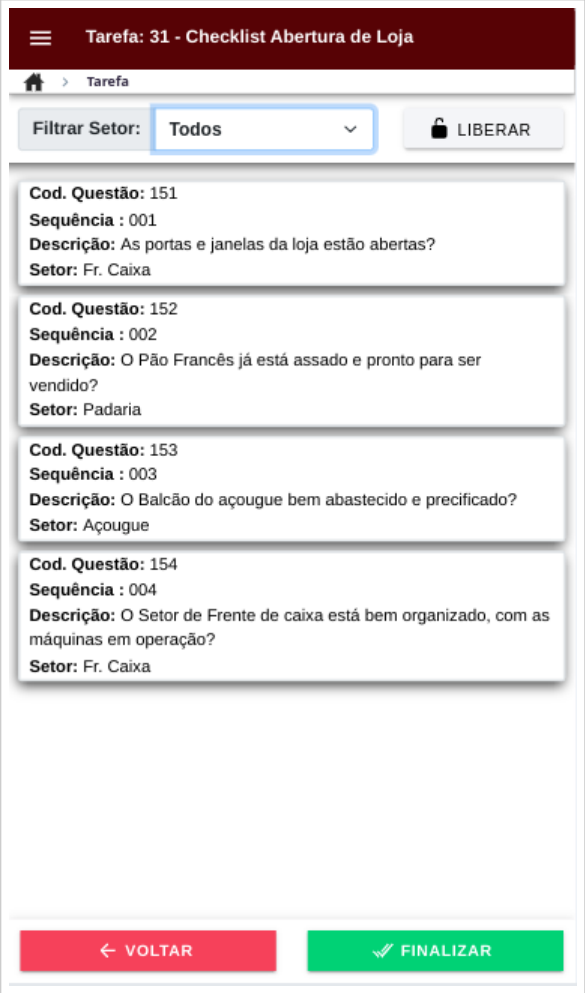
Figura 33 – Apresentação das tarefas pendentes



Fonte: Autoria própria (2024).

Uma vez que uma tarefa é selecionada e a atribuição ao usuário é confirmada, o aplicativo exibe a tela de perguntas da tarefa para seleção e execução, conforme a Figura 34. Esta tela permite, se necessário, filtrar a visualização das perguntas de acordo com o setor a elas atribuídas. Além disso, é possível nesta tela, desvincular uma tarefa de um usuário permitindo que outra pessoa execute a tarefa. Esse procedimento pode ser realizado através do botão `Liberar`. Essa funcionalidade oferece flexibilidade na execução de uma rotina, prevendo possíveis situações que podem ocorrer na operação de um supermercado. No entanto, mesmo que a tarefa seja liberada e outro usuário assuma a execução, cada resposta é associada ao usuário que a forneceu, mantendo assim a rastreabilidade do processo e garantindo a integridade das informações geradas.

Figura 34 – Apresentação das perguntas de uma tarefa selecionada



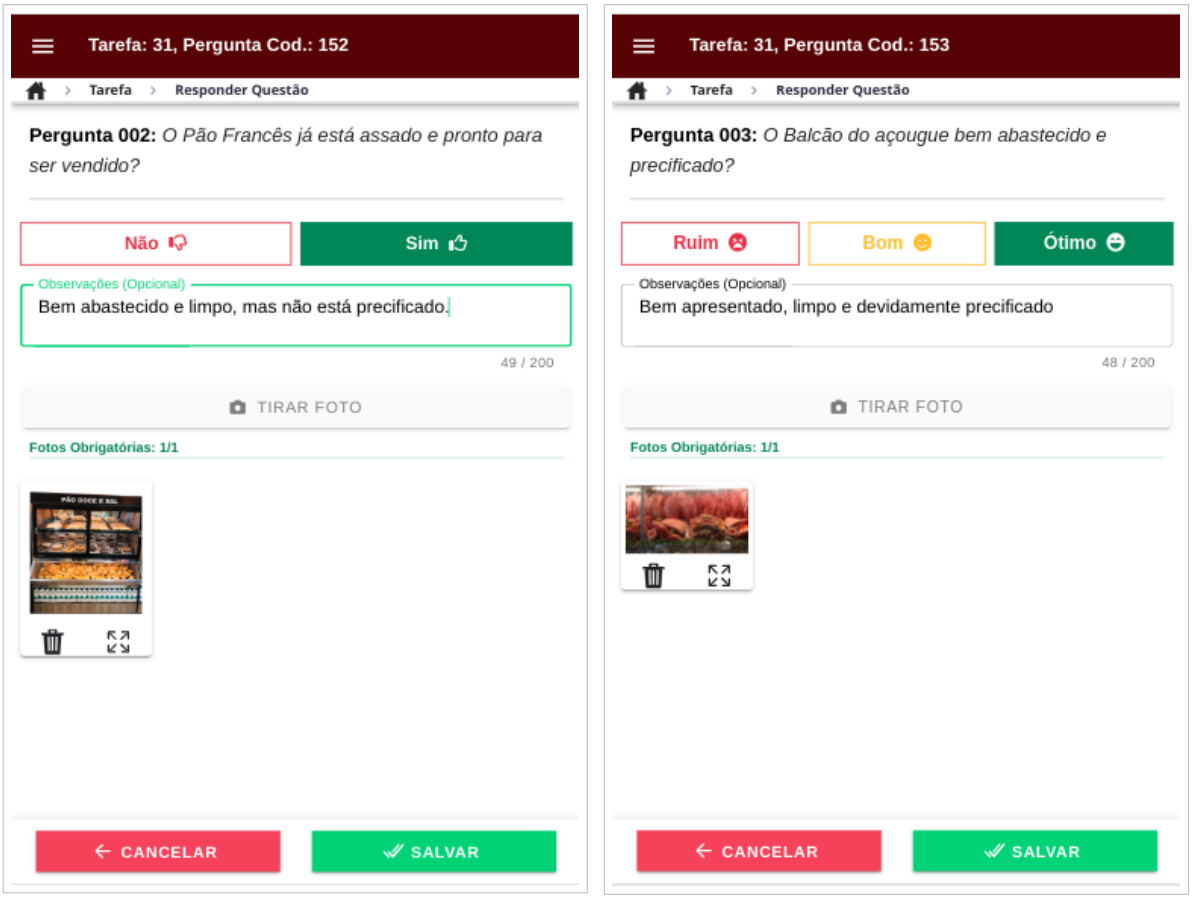
Fonte: Autoria própria (2024).

Ao responder uma pergunta, é possível optar por respostas objetivas, como sim ou não, ou por respostas avaliativas, com as opções ruim, bom ou ótimo, conforme demonstrado na Figura 35. Essa flexibilidade permite diversos níveis de avaliação do processo, adaptando-se às diferentes situações que podem ocorrer na operação de um supermercado. Além disso, através da configuração das perguntas, é possível ativar determinados comportamentos no aplicativo, como exigir observações, torná-las opcionais ou simplesmente não oferecer essa opção na resposta. Também é possível restringir a quantidade de fotos e torná-las obrigatórias. Esses recursos tornam a auditoria de processos mais eficiente, garantindo a comprovação da execução de um determinado processo por meio de perguntas e imagens.

Ao submeter uma resposta, ela é avaliada pelo servidor e se estiver de acordo com as configurações exigidas no cadastro da questão, ela é gravada na base de dados e deixa de ser exibida para o usuário, impedindo qualquer alteração posterior e garantindo a integridade das informações fornecidas.

Figura 35 – Tela para resposta de uma pergunta de uma tarefa

(a) Resposta objetiva **(b) Resposta Avaliativa**



Fonte: Autoria própria (2024).

A tela de respostas, oferece ainda, a capacidade de remover as imagens indesejadas ou mesmo ampliá-las conforme demonstra a Figura 36. Esse processo permite o usuário avaliar a imagem com mais detalhes e refazer o processo caso julgue necessário, para que a resposta seja executada da melhor forma.

Figura 36 – Tela de ampliação de foto de uma pergunta

Fonte: Autoria própria (2024).

5.3 Considerações sobre o desenvolvimento

O desenvolvimento deste projeto abrangeu três frentes principais de trabalho: o aplicativo móvel (APP), a API para comunicação com o APP e a aplicação web para cadastros e gestão de tarefas. Cada uma dessas frentes desempenhou um papel importante na entrega do produto final, sendo necessário um esforço coordenado para garantir a integração e funcionalidade de todos os componentes.

O desenvolvimento do APP focou na manipulação direta dos objetos Json recebidos do servidor, abordando uma utilização mas restrita da orientação a objetos neste ponto do desenvolvimento do APP. Esta abordagem foi adotada para acelerar o desenvolvimento do APP, além de permitir maior engajamento nas regras de negócio e no desenvolvimento da aplicação web e da API. A decisão de tratar os dados de forma direta possibilitou maior velocidade na implementação, porém, sem comprometer a qualidade final do aplicativo.

A API foi desenvolvida para fornecer um canal de comunicação robusto e seguro com o APP, enquanto o modulo web foi projetada para gerenciar cadastros e tarefas. No entanto, devido ao volume de trabalho e ao prazo estipulado, também foram adotadas algumas medidas para garantir a entrega do projeto dentro do cronograma. Entre as principais estratégias,

destaca-se a abordagem final do desenvolvimento, que foi particularmente desafiadora devido à restrição de tempo. O objetivo era realizar testes mais abrangentes que pudessem prever uma maior variedade de situações de erro. No entanto, optou-se por realizar testes básicos que garantissem o funcionamento mínimo viável dos métodos mais importantes do sistema. Apesar das limitações, esses testes foram suficientes para assegurar a estabilidade do sistema.

Durante o desenvolvimento, optou-se por um modelo de acesso mais simples do que o inicialmente planejado. O acesso foi restringido ao uso do aplicativo, onde o usuário deve se autenticar para utilizar as funcionalidades. Este acesso é controlado de acordo com as permissões definidas no cadastro de usuário, determinando se o usuário pode ou não acessar o aplicativo.

Da mesma forma, na plataforma web, o acesso é exclusivo para o administrador. O administrador também pode acessar o aplicativo, desde que tenha a permissão necessária concedida no cadastro.Por outro lado, a implementação do perfil de usuário operador, que estava prevista na fase inicial do projeto, foi adiada para uma implementação futura.

Além disso, foram incorporados ao projeto conceitos de segurança e implantação que não estavam previstos inicialmente. A aplicação web foi instalada e disponibilizada em um servidor EC2 da Amazon. Também foi adquirido o domínio ***routinechecklist.com.br*** para a publicação do *Routine Checklist* na internet. Além disso, visando para garantir a segurança dos dados trafegados entre o aplicativo e a API, foi implementado a certificação SSL, fornecido pela Let's Encrypt[1], buscando através dessas ações, fornecer ao presente projeto segurança para troca de informações em um ambiente publicado na internet.

Apesar das limitações enfrentadas, o autor considera o desenvolvimento deste projeto uma experiência única e extremamente positiva. Este trabalho permitiu a aquisição de novos conhecimentos, especialmente em relação aos processos de automação de tarefas no desenvolvimento de software e à implementação de regras de negócio mais complexas, além da publicação do sistema em um servidor bem estruturado e compatível com uma aplicação em produção na internet.

Adicionalmente, o projeto revelou uma vasta quantidade de detalhes e regras necessárias para o funcionamento correto dos processos, proporcionando um aprendizado significativo que vai além dos conceitos abordados em sala de aula. Esta experiência tem sido de grande valor para o crescimento pessoal e profissional do autor, permitindo a aplicação prática dos conhecimentos adquiridos na universidade e a exploração de novas técnicas e abordagens no desenvolvimento de software.

[1] Let's Encrypt é uma autoridade certificadora (CA) que oferece certificados SSL/TLS gratuitos, facilitando a implementação de segurança em sites. A principal missão da Let's Encrypt é tornar a web mais segura e acessível, incentivando o uso universal de HTTPS (LET'S ENCRYPT, 2024).

6 CONSIDERAÇÕES FINAIS

A motivação que originou a proposta deste projeto reside na escassez de ferramentas de *checklist* capazes de atender de forma eficaz as necessidades das instituições que buscam aprimorar seus processos de negócios e operações internas. Considerando a indiscutível utilidade de uma ferramenta de *checklist* automatizada, apta a fornecer informações precisas para a administração de uma empresa, podemos afirmar que este projeto tem o potencial de aprimorar a execução das tarefas, identificar deficiências nos processos e indicar como essas falhas podem ser sanadas.

É igualmente relevante destacar que um sistema de auditoria equipado com recursos de mídia, como a capacidade de anexar fotos para comprovar a execução das atividades, constitui um diferencial significativo na análise e gestão de procedimentos nas empresas que dependem de rigorosas diretrizes operacionais. Portanto, desenvolver um software que otimize esses processos é o objetivo primordial. Ademais, o desenvolvimento das funcionalidades internas do aplicativo é pautado pela premissa de proporcionar simplicidade e automação na manipulação do software, sem comprometer a integridade e organização das informações geradas pelos processos. Isso significa que oferecemos flexibilidade tanto para os usuários que são proficientes em tecnologia quanto para aqueles que têm menos familiaridade com dispositivos de informática.

Para alcançar esse objetivo, o projeto foi desenvolvido em conformidade com as melhores práticas recomendadas para o desenvolvimento de software, utilizando os conceitos e padrões disponíveis na documentação do *framework* Laravel e das ferramentas incorporadas ao projeto. Ao concluir este trabalho, obteve-se um software funcional capaz de oferecer recursos de controle de processos, juntamente com relatórios que facilitam a tomada de decisões e simplificam a identificação de deficiências e melhorias nos processos da empresa.

6.1 Trabalhos Futuros

O presente projeto oferece um sólido ponto de partida para futuras expansões e aprimoramentos das suas funcionalidades, alavancando o progresso tecnológico das ferramentas empregadas. Com o intuito de tornar o sistema ainda mais abrangente e eficaz, diversas melhorias podem ser implementadas.

Uma das adições em potencial é o desenvolvimento de uma funcionalidade de *checklist* acessível por meio de um navegador web. Isso permitiria aos usuários executar tarefas de *checklist* de maneira mais flexível e acessível. Além disso, a capacidade de anexar recursos multimídia, como áudio e vídeos curtos, às respostas de cada pergunta, agregaria um valor significativo à ferramenta. Também seria benéfico incorporar a geolocalização nas perguntas, o que pode ser vital para determinados tipos de *checklists*.

Outra funcionalidade de destaque que poderia ser desenvolvida é a implementação de planos de ação associados às respostas das perguntas. Isso impulsionaria a melhoria dos processos, permitindo que o sistema emita alertas aos usuários e crie tarefas com base nas respostas. Esse recurso ajudaria a garantir que situações críticas não sejam negligenciadas e que os problemas sejam resolvidos ou adequadamente justificados dentro de um período definido, sem serem esquecidos.

Além disso, visando uma expansão ainda mais abrangente da ferramenta, poderia ser desenvolvido um novo modelo de *checklist*. Neste modelo, os usuários, por meio do aplicativo móvel, poderiam iniciar uma tarefa de *checklist* sem perguntas pré-definidas, criando-as durante a execução da atividade. Isso seria particularmente útil para auditorias aleatórias em setores específicos da empresa ou para situações em que um *checklist* elaborado e monitorado não seja necessário. Essa abordagem “aleatória” de *checklist* também poderia ser aplicada para fazer anotações durante reuniões ou outras situações similares.

Ainda, poderia ser implementada a opção de acesso ao sistema por um usuário operador, conforme previsto na fase inicial do projeto. No entanto, devido à falta de tempo, essa funcionalidade foi adiada para uma implementação futura. Esse modelo de usuário teria acesso apenas a relatórios e análises da operação, com acesso restrito às demais funcionalidades do sistema, como a criação ou alteração de rotinas.

Essas propostas de desenvolvimento futuro demonstram o potencial de aprimoramento significativo do sistema, tornando-o mais versátil e adaptável às necessidades variadas dos usuários e das empresas.

REFERÊNCIAS

ABPMP. **BPM CBOK: Guia para o gerenciamento de processos de negócio corpo comum de conhecimento v3.0**. 1. ed. [*S.l.*]: Association of Business Process Management Professionals, 2013.

ANGULAR. **Docs**. [*S.l.*], 2023. Disponível em: https://angular.io/docs. Acesso em: 09 de outubro de 2023.

ATLASSIAN CORPORATION. **O que é o Trello?** [*S.l.*], 2023. Disponível em: https://trello.com/pt-BR/tour. Acesso em: 10 de outubro de 2023.

BOOTSTRAP. **Home**. [*S.l.*], 2023. Disponível em: https://getbootstrap.com/. Acesso em: 08 de outubro de 2023.

CARDOSO, G.; CARDOSO, V. **Sistemas de Banco de Dados: Uma abordagem introdutoória e aplicada**. 1. ed. São Paulo: Saraiva, 2012.

CONSÓRCIO WORLD WIDE WEB - W3C. **O Quê é CSS?** [*S.l.*], 2023. Disponível em: https://www.w3.org/Style/CSS. Acesso em: 08 de outubro de 2023.

DAVID, F. **JavaScript: o guia definitivo**. 6. ed. Porto Alegre: Bookman Companhia Editora Ltda, 2013.

EIS, D.; FERREIRA, E. **HTML5 e CSS3: Com farinha e pimenta**. 1. ed. São Paulo: Tableless, 2012.

FASTFIELD, INC. **Dashboard**. [*S.l.*], 2023. Disponível em: https://portal.fastfieldforms.com/. Acesso em: 20 de setembro de 2023.

FUNDAçÃO GETULIO VARGAS. **Uso de TI no Brasil: Gastos e investimentos em TI**. [*S.l.*], 2023. Disponível em: https://portal.fgv.br/noticias/uso-ti-brasil-pais-tem-mais-dois-dispositivos-digitais-habitante-revela-pesquisa. Acesso em: 24 de outubro de 2023.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATíSTICA (IBGE). **API de localidades: Distritos**. [*S.l.*], 2023. Disponível em: https://servicodados.ibge.gov.br/api/v1/localidades/distritos. Acesso em: 17 de outubro de 2023.

IONIC. **Docs**. [*S.l.*], 2023. Disponível em: https://ionic.io/docs. Acesso em: 09 de outubro de 2023.

JQUERY. **Home**. [*S.l.*], 2023. Disponível em: https://jquery.com. Acesso em: 08 de outubro de 2023.

LARAVEL. **Conheça o Laravel**. [*S.l.*], 2023. Disponível em: https://laravel.com/docs/10.x. Acesso em: 10 de outubro de 2023.

LARAVEL. **Eloquente: primeiros passos**. [*S.l.*], 2023. Disponível em: https://laravel.com/docs/10.x/eloquent. Acesso em: 10 de outubro de 2023.

LET'S ENCRYPT. **Sobre**. [*S.l.*], 2024. Disponível em: https://letsencrypt.org/about/. Acesso em: 12 de junho de 2024.

MATERIALIZE. **Home**. [*S.l.*], 2023. Disponível em: https://materializecss.com/. Acesso em: 08 de outubro de 2023.

MICROSOFT CORPORATION. **Começando**. [*S.l.*], 2023. Disponível em: https://code.visualstudio.com/docs. Acesso em: 11 de outubro de 2023.

MONTEIRO, E. R. *et al.* **DevOps**. 1. ed. Porto Alegre: Sagah, 2021.

PAIN, R. *et al.* **Gestão de processos: Pensar, agir e aprender**. 1. ed. Porto Alegre: Editora Bookman, 2009.

PHP GROUP. **Documentação Oficial PHP**. [*S.l.*], 2023. Disponível em: https://www.php.net/manual/pt_BR/introduction.php. Acesso em: 10 de outubro de 2023.

POSTGRESQL. **Sobre**. [*S.l.*], 2023. Disponível em: https://www.postgresql.org/about. Acesso em: 11 de outubro de 2023.

PRADELLA, S.; FURTADO, L. M. K. joão C. **Gestão de processos: Da teoria a prática**. 1. ed. Porto Alegre: Editora Atlas S.A, 2012.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software: Uma abordagem profissional**. 9. ed. Porto Alegre: AMGH Editora Ltda, 2021.

RED HAT, INC. **O que é IDE (Ambiente de desenvolvimento integrado)?** [*S.l.*], 2023. Disponível em: https://www.redhat.com/pt-br/topics/middleware/what-is-ide. Acesso em: 11 de outubro de 2023.

RP INFO SISTEMAS. **Dashboard**. [*S.l.*], 2023. Disponível em: https://www.rpinfo.com.br/produto/task/79. Acesso em: 20 de setembro de 2023.

SEBRAE. **O método MoSCoW para definição de prioridades**. [*S.l.*], 2023. Disponível em: https://sebrae.com.br/Sebrae/Portal%20Sebrae/Arquivos/ebook_sebrae_metodologia_moscow.pdf. Acesso em: 03 de outubro de 2023.

SILVA, M. S. **CSS: Desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3**. 1. ed. São Paulo: Novatec, 2011.

SILVA, M. S. **JQuery - A Biblioteca do Programador JavaScript**. 3. ed. Porto Alegre: Bookman Companhia Editora Ltda, 2013.

SOFTPLAN PLANEJAMENTO E SISTEMAS S.A. **Checklist**. [*S.l.*], 2023. Disponível em: https://blog-pt.checklistfacil.com/checklist/. Acesso em: 05 de outubro de 2023.

SOFTPLAN PLANEJAMENTO E SISTEMAS S.A. **checklists Aplicados**. [*S.l.*], 2023. Disponível em: https://app.checklistfacil.com.br/. Acesso em: 19 de setembro de 2023.

THEMEWAGON. **Phoenix – Painel de administração e modelo de WebApp**. [*S.l.*], 2023. Disponível em: https://themes.getbootstrap.com/product/phoenix-admin-dashboard-webapp-template. Acesso em: 08 de outubro de 2023.

ZABOOT, D.; MATOS, E. **Aplicativos com Bootstrap e Angular - Como desenvolver APPS responsivos**. 1. ed. São Paulo: Érica, 2020.

APÊNDICE A – Modelo Lógico de Dados - MLD

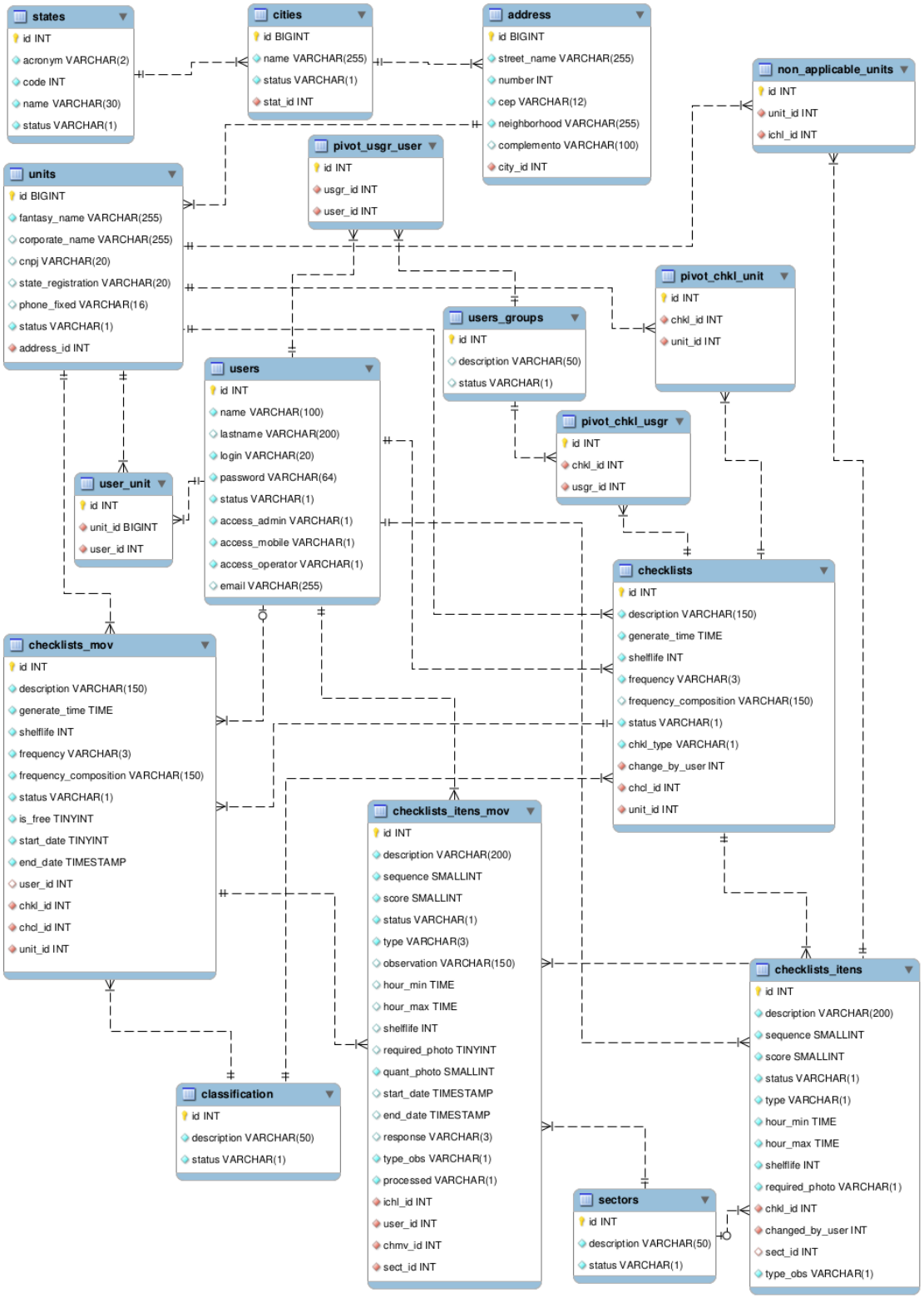


Figura 37 – Modelo Lógico de Dados - MLD

Fonte: Autoria Própria (2023).

APÊNDICE B – Relatório em PDF de execução de uma tarefa de checklist

ROUTINE Checklist

Razão Social: *Supermercado Fictício SA*
CNPJ: *43.341.216/0001-01* **Telefone:** *(45) 3707-0001*
Endereço: *Rua Tabarana, n° 776 - Vila Residencial A - Guarapari/ES*
Dt. Emissão: *27/06/24 18:27:29* **Usuário:** *1*

RELATÓRIO DE EXECUÇÃO DE TAREFA

Estatísticas e Pontuações

GERAL

Tarefa: *55 - Checklist Limpeza Loja*
Unidade: *001 - Supermercado Fictício SA* **CNPJ:** *43.341.216/0001-01* **Cidade:** *Guarapari*
Último usuário vinculado à tarefa: *1 - Administrador* **Status Tarefa:** *Concluída*
DT Início: *27/06/2024 16:57* **DT Limite:** *28/06/2024 04:57* **Processada em:** *27/06/2024 17:26*
Tot Questões da Tarefa: *3* **Tot. Não Respondidas:** *0* **Tot. Respondidas:** *3*
Pontuação geral da Tarefa: *12* **Pontuação Executada:** *9.5* **Percentual Atingido:** *79.167%*

Contadores por tipos de resposta	Resposta	Porcentagem
	Sim: *1*	**Sim:** *33.33%*
	Não: *0*	**Não:** *0%*
	Ruim: *0*	**Ruim:** *0%*
	Bom: *1*	**Bom:** *33.33%*
	Excelente: *1*	**Excelente:** *33.33%*

POR SETOR

Descrição do Setor	Tot. Questões	% Resp. Afirmativa(s)	% Resp. Negativa(s)	Tot. Pontos Gerado	Tot. Pontos Executado	% Pontos Executado
Setor Não Informado	0	0%	0%	0	0	0%
Açougue	1	100%	0%	5	5	100%
Fr. Caixa	1	100%	0%	2	2	100%
Padaria	1	100%	0%	5	2.5	50%
TOTAIS	**3**	**100%**	**0%**	**12**	**9.5**	**79.17%**

ROUTINE Checklist

Razão Social: *Supermercado Fictício SA*
CNPJ: *43.341.216/0001-01* **Telefone:** *(45) 3707-0001*
Endereço: *Rua Tabarana, n° 776 - Vila Residencial A - Guarapari/ES*
Dt. Emissão: *27/06/24 18:27:29* **Usuário:** *1*

Perguntas X Respostas

258 - O BALCÃO DO ACOUGUE ESTÁ LIMPO E ORGANIZADO PARA O TRABALHO?

Processada: *Sim*	**Setor:** *Açougue*
Pontos da pergunta: *5*	**Pontuação Executada:** *5*
Usuário: *1 - Administrador*	**Data/Hora Resposta:** *27/06/2024 17:10*
Tipo Pergunta: *Avaliativa*	**Resposta:** *Excelente*
Observações: *Não Informado*	

Fotos:


257 - A FRENTE DE CAIXA ESTÁ LIMPA E ORGANIZADA PARA O TRABALHO?

Processada: *Sim*	**Setor:** *Fr. Caixa*
Pontos da pergunta: *2*	**Pontuação Executada:** *2*
Usuário: *1 - Administrador*	**Data/Hora Resposta:** *27/06/2024 17:09*
Tipo Pergunta: *Sim/Não*	**Resposta:** *Sim*
Observações: *Não Informado*	

Fotos:



259 - O BALCÃO DA PADARIA ESTÁ LIMPO E ORGANIZADO PARA O TRABALHO?

Processada: *Sim*	**Setor:** *Padaria*
Pontos da pergunta: *5*	**Pontuação Executada:** *2.5*
Usuário: *1 - Administrador*	**Data/Hora Resposta:** *27/06/2024 17:10*
Tipo Pergunta: *Avaliativa*	**Resposta:** *Bom*
Observações: *Não Informado*	

Fotos: