UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COINT - TECNOLOGIA EM SISTEMAS PARA INTERNET
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

EVERTON PAULOUSKI

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

PROJETO DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO

GUARAPUAVA
2023

EVERTON PAULOUSKI

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

Projeto de Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 1, do Curso Superior de Tecnologia em Sistemas para Internet – TSI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Guarapuava, como requisito parcial para obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Dr. Roni Fabio Banaszewski
Universidade Tecnológica Federal do Paraná

GUARAPUAVA
2023

RESUMO

PAULOUSKI, Everton. Routine Checklist: Sistema para Organização e Auditoria de Rotinas em Supermercados. 2023. 44 f. Projeto de Trabalho de Conclusão de Curso de Graduação – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Guarapuava, 2023.

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: gestão; controle; processo; tarefa; checklist.

ABSTRACT

PAULOUSKI, Everton. Routine Checklist: System for Organization and Audit of Supermarket Routines. 2023. 44 f. Projeto de Trabalho de Conclusão de Curso de Graduação – Curso de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná. Guarapuava, 2023.

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: management; control; process; task; checklist.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AJAX	Javascript Assíncrono e XML (do ingês *Asynchronous Javascript* and XML)
CSS	Folhas de Estilo em Cascata (do inglês *Cascading Style Sheets*)
MLD	Modelo Lógico de Dados
HTML	Linguagem para Marcação de Hipertexto (do inglês *HyperText Markup Language*)
MVC	Modelo-Visão-Controle (do inglês *Model-View-Controller*)
ORM	Mapeamento Objeto Relacional (do inglês *Object Relational Mapper*)
PHP	Processador de Hipertexto (do inglês *Hypertext Preprocessor*)
SGBD	Sistema Gerenciador de Bando de Dados
SQL	Linguagem de Consulta Padrão (do inglês *Standard Query Language*)
TCC	Trabalho de Conclusão de curso
TSI	Tecnologia em Sistemas para Internet
UTFPR	Universidade Tecnológica Federal do Paraná
W3C	Consórcio *World Wide Web*
WEB	Rede Mundial de Computadores - Internet
API	Interface de Programação de Aplicação (do inglês *Application Programming Interface*)

SUMÁRIO

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

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

eletrônicas 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.

Este projeto tem como objetivo desenvolver uma solução de software para melhorar a gestão de processos em supermercados. Propõe-se a criação de um sistema de *checklist* que se chamará: "Routine Checklist", e orientará a execução de rotinas, auditará as ações e fornecerá uma interface de fácil utilização. 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.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Desenvolver um sistema de *checklist* dedicado a organizar, padronizar e auditar Rotinas de um supermercado.

1.1.2 Objetivos Específicos

- Organizar e padronizar as atividades de um supermercado;
- Auditar requerendo fotos para comprovar a execução correta dos processos;
- Avaliar a execução dos processos através de uma pontuação em cada questão executada;
- Gerar relatório na página inicial da aplicação, comparando a pontuação geral dos *checklists* executados entre unidades de uma rede de supermercados;
- Gerar relatório na página inicial da aplicação, comparando os *checklists* executados e não executados entre as unidades de uma rede de supermercados;
- Agendar a geração automatizada de tarefas de *checklists* para execução.

2 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.

2.0.1 Checklist Fácil

O CheckList Fácil 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 1.

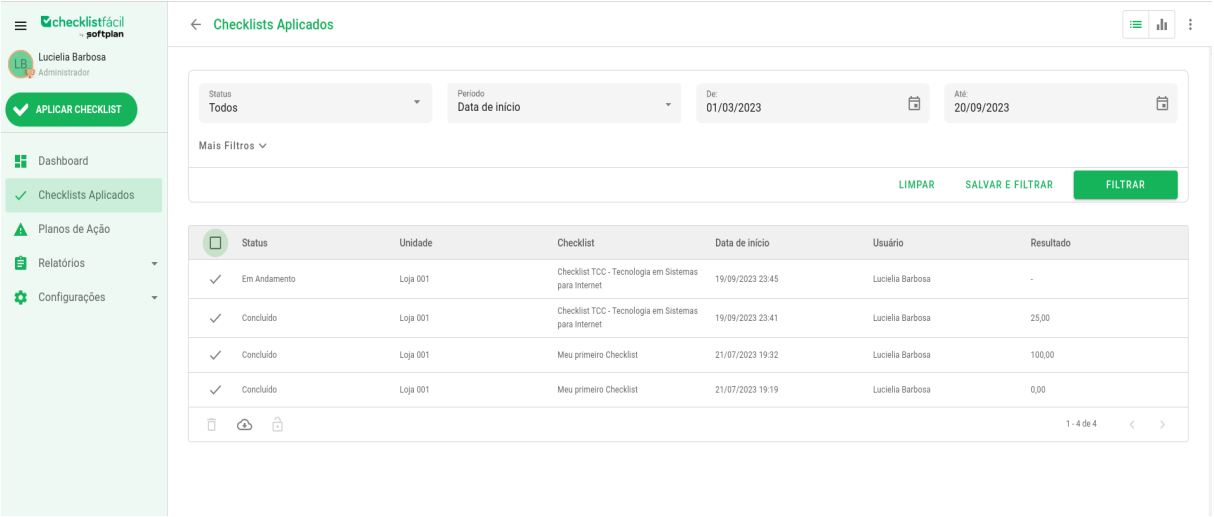

Figura 1 – Tela dos *checklists* aplicados
Fonte: Softplan Planejamento e Sistemas S.A. (2023b)

[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.

2.0.2 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 2.

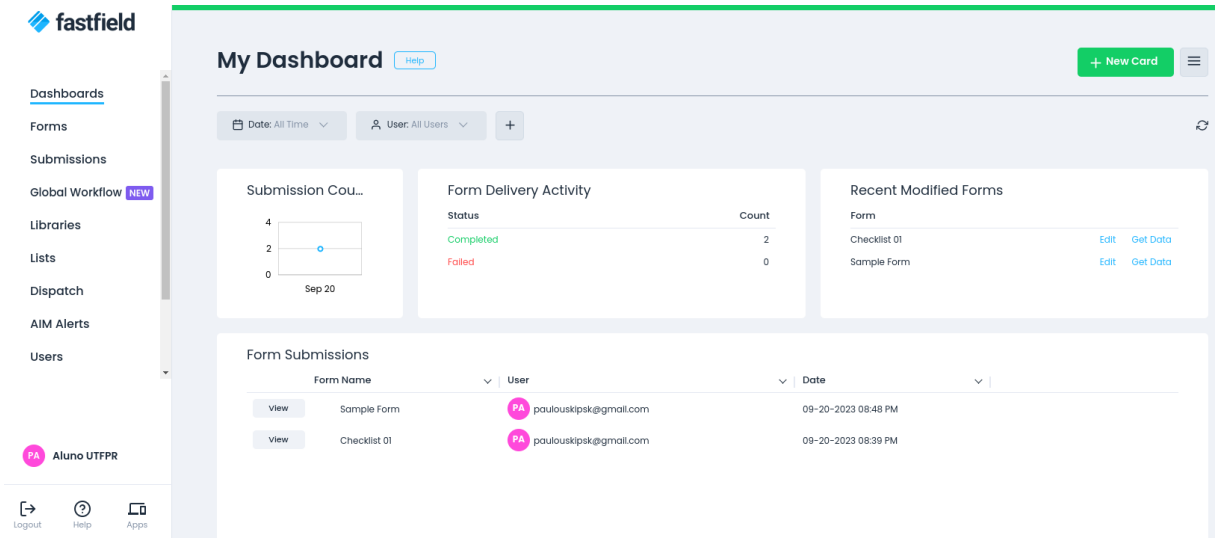

Figura 2 – Tela *Dashboard* FastField
Fonte: FastField, Inc. (2023)

2.0.3 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 supermer-

cados, 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 3.

Figura 3 – Tela Dashboard Task
Fonte: RP Info Sistemas (2023)

2.1 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	

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão delineadas as tecnologias e conceitos que orientarão 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 4 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.

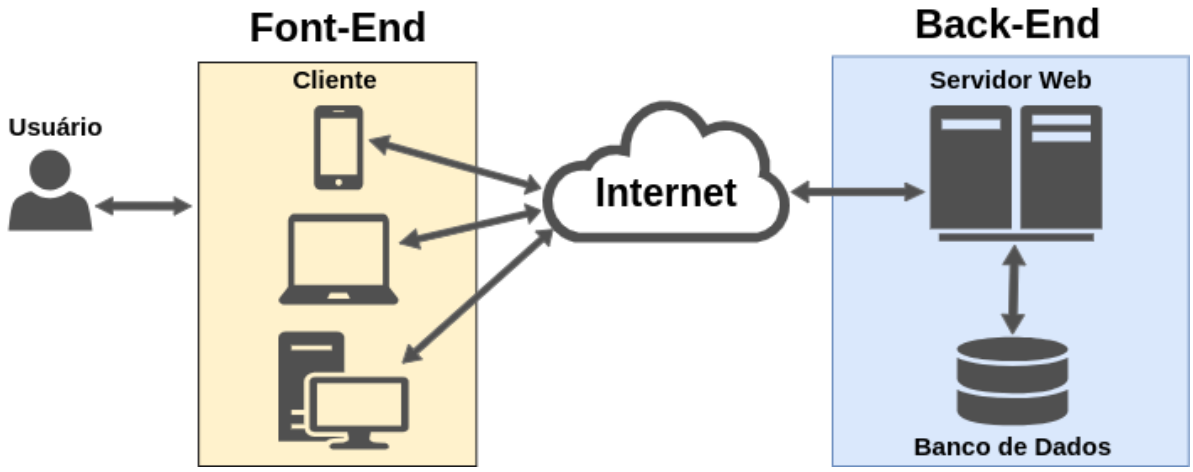

Figura 4 – Estrutura de Ambiente Web - Cliente X Servidor
Fonte: Autoria Própria (2023)

Nas seções subsequentes, analisaremos 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.

3.1 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 definido pelo Consórcio World Wide Web - W3C (2023a)[1], como "*um mecanismo simples para adicionar estilo (por exemplo, fontes, cores, espaçamento) a documentos 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.

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*[2] 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 (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 for-

[1] "*O Consórcio World Wide Web (W3C) é um consórcio internacional no qual organizações filiadas, uma equipe em tempo integral e o público trabalham juntos para desenvolver padrões para a web*". (CONSóRCIO WORLD WIDE WEB - W3C, 2023b)

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

mulários e uma variedade de outras funcionalidades. Sua aplicação é amplamente difundida no desenvolvimento de aplicações web e para dispositivos móveis, e, conforme observado por David (2013), o JavaScript devido a sua versatilidade, ganhou destaque como 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[3] 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 torna 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,

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

CSS e JavaScript. Essas tecnologias trabalham em conjunto de forma sinérgica para criar uma experiência 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.

3.2 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**[4] chamada *Eloquent*,

[4]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

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 5 (PRESSMAN; MAXIM, 2021).

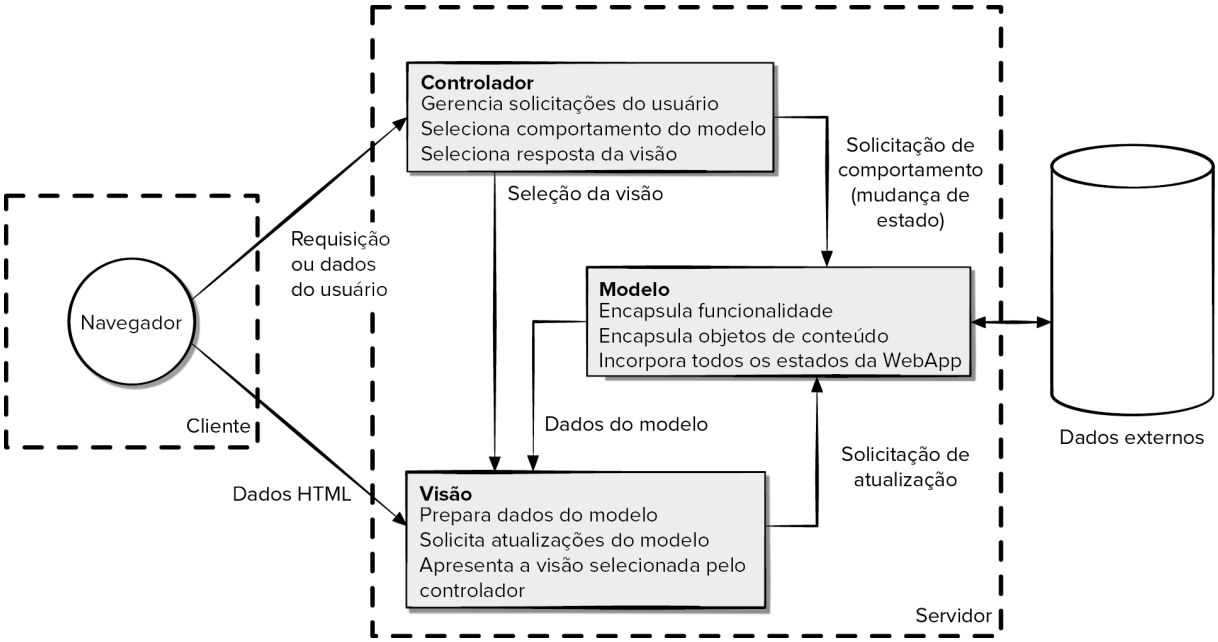

Figura 5 – Estrutura de aplicação utilizando infraestrutura MVC.
Fonte: (PRESSMAN; MAXIM, 2021)

dados, de forma simples e agradável (LARAVEL, 2023b).

3.3 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 6 (SEBRAE, 2023).

Figura 6 – Quadro de Organização MoSCoW
Fonte: Autoria Própria (2023)

3.4 Ferramentas de desenvolvimento

O Trello é uma aplicação online que oferece uma abordagem colaborativa para a gestão de projetos, notoriamente baseada no sistema Kanban[5]. 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.

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, chegamos ao 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[6] 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

[5] 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)

[6] "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).

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 7, que ilustra de maneira clara e concisa o ciclo de entrada, processamento, armazenamento e recuperação de dados em um sistema de informação.

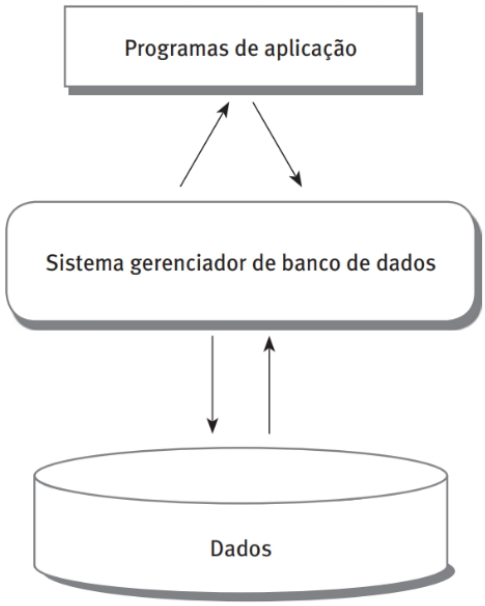

Figura 7 – Fluxo de dados em um sistema gerenciado com SGBD.
Fonte: (CARDOSO; CARDOSO, 2012)

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.

4 ANÁLISE E PROJETO DO SISTEMA

Neste capítulo, será explanado sobre estratégia que guia o desenvolvimento do projeto de *checklist* proposto. Para atingir esse objetivo, será empregado o método MoSCoW no gerenciamento e planejamento das atividades, visando aprimorar a eficiência em cada fase do projeto. Por meio dessa abordagem, destacamos o planejamento do projeto e analisamos a execução de fases específicas, proporcionando uma visão aprofundada das tecnologias selecionadas para a realização do projeto de software proposto. Este capítulo proporciona uma análise do trajeto trilhado para garantir a qualidade e o êxito no desenvolvimento desse projeto.

4.1 Planejamento do Desenvolvimento

O projeto proposto foi cuidadosamente planejado e segmentado em cinco fases distintas, com o intuito de conduzir a análise e o desenvolvimento da aplicação de maneira metódica e eficaz. A primeira fase representa o ponto de partida, na qual concentra os esforços no levantamento das funcionalidades necessárias para a construção do sistema, processo conhecido como "Levantamento de requisitos". Nesse estágio inicial, dedica-se uma atenção especial à definição da interface visual, destacando a integração do template Phoenix ao Laravel Framework. Além disso, priorizamos a integração de frameworks essenciais, como jQuery, Bootstrap e outros elementos fundamentais, estabelecendo a base sólida para o desenvolvimento do sistema web.

Ainda na primeira fase do projeto, haverá foco no desenvolvimento da base do sistema móvel (APP) utilizando o *framework* Ionic, com ênfase na plataforma Android. Nesta fase, priorizaremos a implementação das funcionalidades essenciais para a comunicação com a API do Laravel, responsável por conter as regras de negócio do sistema de retaguarda. Além disso, dedicaremos esforços à definição do layout e ao mapeamento do fluxo operacional para execução dos *checklists* no aplicativo.

A segunda fase foca na definição do sistema web em relação aos cadastros cruciais para a administração do sistema, juntamente com a elaboração das regras de negócio. Durante essa etapa, serão implementadas as funções de autenticação, cadastro dos *checklists*, perguntas associadas a esses *checklists*, e aos registros que desempenham um papel fundamental na geração de relatórios e na classificação dos processos, como a classificação de *checklists* e áreas, por exemplo. Em relação ao APP, nesta fase serão desenvolvidas as primeiras telas para execução de um *checklist*, as quais se destacam as funcionalidades de buscar os *checklists* pendentes, selecionar uma tarefa para execução e abrir a tela responsável por permitir responder uma pergunta.

A terceira fase concentra-se no desenvolvimento da área funcional do sistema, responsável por gerar tarefas, encerrar e processar tarefas expiradas, além de executar regras em segundo plano, sem a necessidade de intervenção humana por parte dos operadores do sistema.

A quarta fase, define a implementação dos principais relatórios e gráficos que serão disponibilizados no painel da página inicial do sistema de gerenciamento web. Além disso, efetuamos implementações adicionais para os ajustes finais da aplicação.

A quinta e última fase concentrou-se na realização de testes para verificar as funcionalidades do sistema, visando executar os processos na prática e identificar falhas na comunicação, além de possíveis erros que ainda possam ocorrer. O objetivo primordial foi assegurar a integridade na execução do software. Embora o desenvolvimento de testes automatizados seja uma prática adotada em todas as fases de desenvolvimento, é fundamental realizar testes manuais para garantir a resolução de possíveis problemas na comunicação, especialmente em relação ao ambiente no qual os sistemas irão interagir.

Com a conclusão bem-sucedida da quinta fase, espera-se obter uma aplicação capaz de oferecer qualidade e integridade nas informações geradas por esse sistema de gerenciamento e auditoria de processos, marcando assim o término do desenvolvimento do sistema proposto.

4.2 Levantamento de requisitos

Para iniciar a primeira fase do projeto, o ponto de partida envolveu o levantamento dos requisitos do sistema. Para realizar esta etapa inicial, foi realizado a análise detalhada das informações coletadas durante o estudo abordado no capítulo referente aos sistemas correlatos (Capítulo 2). Essa análise permitiu obter os dados essenciais para a síntese dos requisitos funcionais e não funcionais. Estes requisitos foram identificados a partir das características das ferramentas de *checklist* mencionadas: Task, Checklist Fácil e FastField Mobile Forms.

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

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

Tabela 1 – Levantamento dos requisitos não funcionais.
Fonte: Autoria Própria (2023)

ID	Prioridade	[Domínio] 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*.

Tabela 2 – Levantamento dos requisitos funcionais.

Fonte: Autoria Própria (2023)

4.3 Movimentação dos cartões do Kanban no quadro MoSCoW

A ferramenta Trello permite a gestão do projeto através do Kanban, que são os cartões que representam requisitos funcionais relacionados na Tabela 2. O fluxo de desenvolvimento será distribuído em sete quadros:*Must Have*, *Should Have*, *Could Have* , *Won't Have*, *In progress*, *Tests/Review* e *Done*. Onde, a movimentação dos cartões acontecerá 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, a equipe de desenvolvimento terá um entendimento claro das funcionalidades a serem implementadas. À medida que cada funcionalidade é iniciada, o respectivo cartão é 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 será então movido para a lista *Done*, sinalizando o encerramento do desenvolvimento da respectiva funcionalidade.

4.4 Seleção das funcionalidades e ordenação das prioridades

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.

No entanto, antes de iniciar a implementação dos requisitos funcionais, será necessário estabelecer os alicerces do sistema, bem como integrar as bibliotecas necessárias e o *template* que definirá o *layout* da aplicação. Nessa fase inicial, as integrações entre tecnologias desempenham um papel central no projeto, impulsionando a concretização da estrutura central do *software*, tanto na versão web quanto no aplicativo móvel (APP).

Assim, com a base do sistema construído, será possível dar início à implementação das funcionalidades de autenticação (RF001) e dos demais cadastros essenciais para a gestão do sistema, incluindo os principais CRUDs que orientarão o cadastro e as regras básicas de execução dos *checklists*. Isso permitirá obter uma compreensão mais profunda do funcionamento e do fluxo de trabalho que o sistema deveria seguir.

Portanto, após a conclusão dessas implementações, o objetivo é não apenas ter os principais cadastros do sistema, como *checklists* e perguntas com suas regras de execução, mas também um sistema funcional capaz de gerar tarefas de *checklist* sem a necessidade de intervenção dos usuários. Isso abrirá caminho para a implementação das funcionalidades necessárias, que complementarão as atividades de gestão e auditoria da ferramenta, como o desenvolvimento de relatórios e ajustes necessários para o pleno funcionamento do sistema.

Além disso, considerando a abordagem de implementação ágil, os testes das principais funcionalidades seriam realizados após a definição do fluxo e das regras de negócio do sistema.

Isso ocorreria após as primeiras execuções dos *checklists*, o que permitiria ajustar as regras para aprimorar a execução da ferramenta. Esses ajustes visam melhorar o controle, a integridade e a segurança das informações geradas pelos usuários que executam os *checklists*, ao mesmo tempo que proporcionam maior fluidez no fluxo de trabalho e a geração de relatórios necessários para análise e gestão dos processos auditados.

4.5 Definição de acesso aos sistemas web e mobile

O acesso ao sistema é configurado durante o cadastro dos usuários, oferecendo três modalidades de acesso: "Administrador", "Operador" e "*Mobile*". Os administradores desfrutam de permissões abrangentes, permitindo-lhes acesso completo aos cadastros do sistema, o que inclui o registro de *checklists*, perguntas associadas a *checklists*, o gerenciamento de usuários e outras funcionalidades disponíveis no menu "Cadastros".

Em contraste, os usuários com a função de "Operador" possuem um nível mais restrito de acesso, limitando-se aos relatórios gerados a partir da execução dos *checklists*, de acordo com as unidades às quais possuem acesso autorizado. A permissão "*Mobile*" 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. Importante ressaltar que um usuário com acesso à versão web também pode utilizar o acesso móvel, se necessário, oferecendo flexibilidade de uso das plataformas web e mobile de acordo com as necessidades.

4.6 Banco de dados

O sistema proposto requer uma estrutura de banco de dados que permita a flexibilidade necessária para a incorporação de novas funcionalidades e a realização de manutenções futuras, garantindo, assim, a evolução contínua. Além disso, é essencial que o sistema possa gerar diversos relatórios gerenciais contendo informações cruciais para a gestão da empresa à qual o sistema irá operar. Devido à importância dessa base de dados, optou-se pelo Sistema de Gerenciamento de Banco de Dados (SGBD) PostgreSQL (2023), amplamente reconhecido como um dos sistemas mais robustos e respeitados disponíveis no mercado.

O PostgreSQL é conhecido por sua capacidade de gerenciar grandes volumes de dados com eficiência e rapidez. Além disso, ele oferece integração com o *framework* Laravel, o que facilita a interação entre o banco de dados e a aplicação web. Essa escolha estratégica do PostgreSQL, baseia-se em sua sólida reputação no mercado e sua capacidade de atender às necessidades do sistema, contribuindo para a robustez e eficiência da solução proposta.

4.6.1 Modelo Lógico de Dados

Com base na definição dos requisitos funcionais, foi possível desenvolver um artefato fundamental para o sistema: o Modelo Lógico de Dados (MLD). Este artefato desempenha um papel crucial ao proporcionar uma representação visual detalhada da estrutura do banco de dados, incluindo os relacionamentos entre tabelas, seus atributos e os tipos de dados que acomodarão as informações geradas pelo sistema.

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 24, 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.

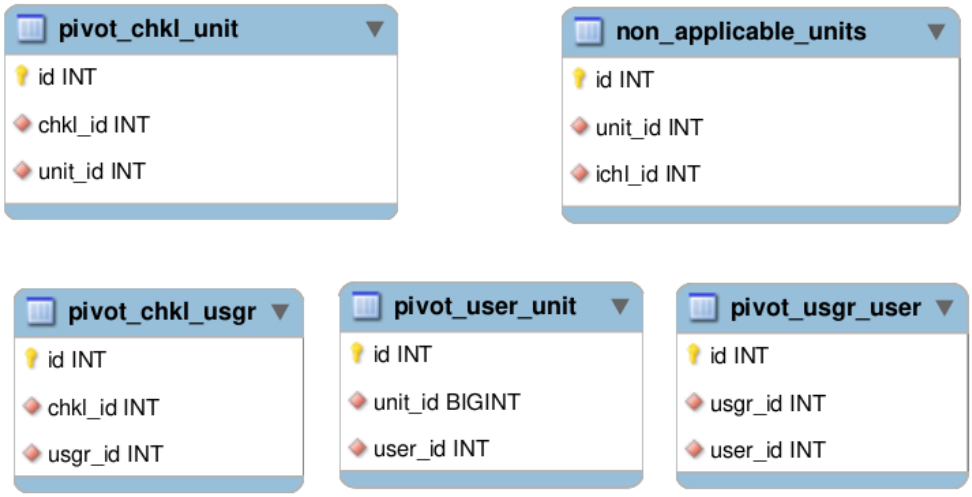

Figura 8 – MLD - *Pivotables*
Fonte: Autoria Própria (2023)

O relacionamento mais elementar no banco de dados, a tabela *units* que é uma representação das filiais de uma empresa. Este relacionamento fundamental servirá 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*.

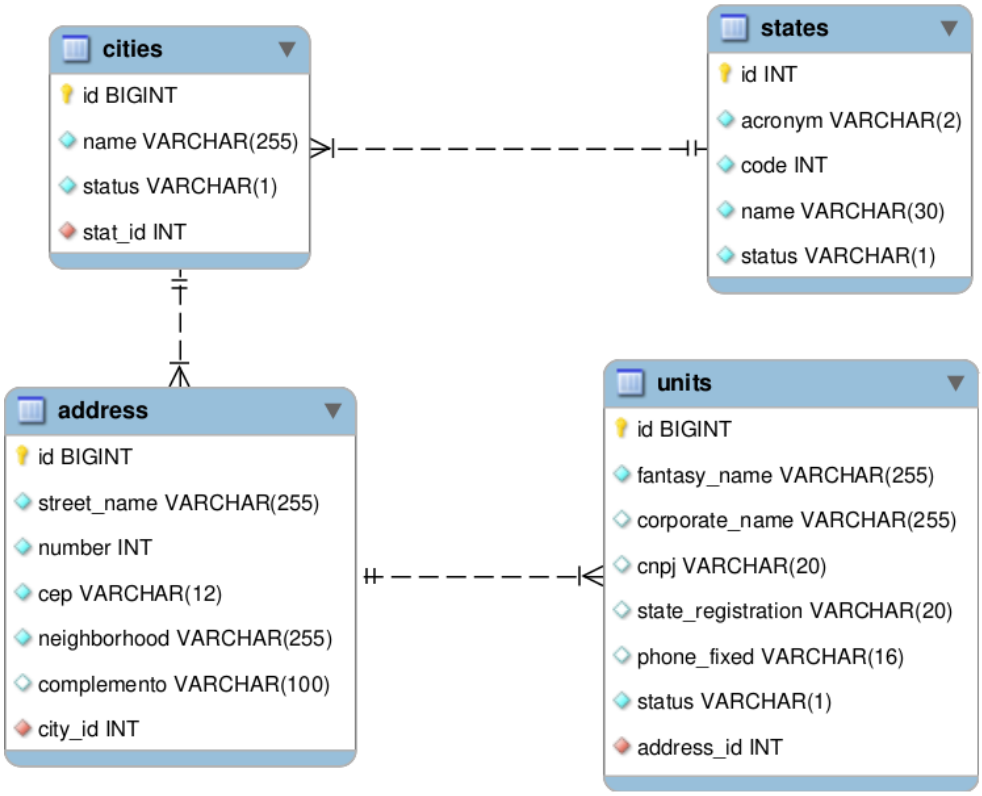

Figura 9 – MLD - Unidades e Localidades
Fonte: Autoria Própria (2023)

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.

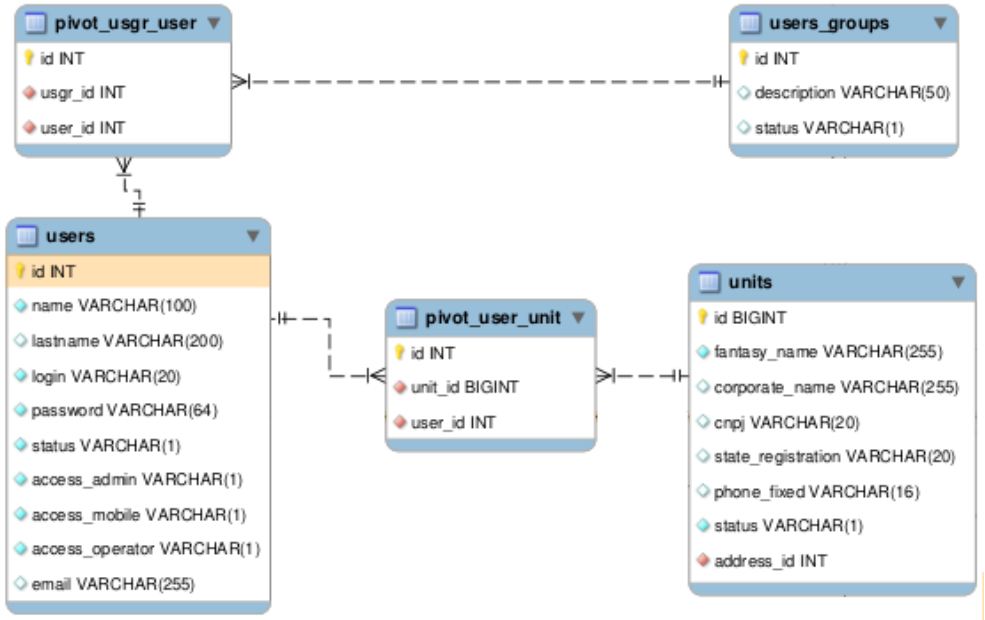

Figura 10 – MLD - Usuário e Grupos de acessos
Fonte: Autoria Própria (2023)

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 do *checklist* com grupos de usuários, 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.

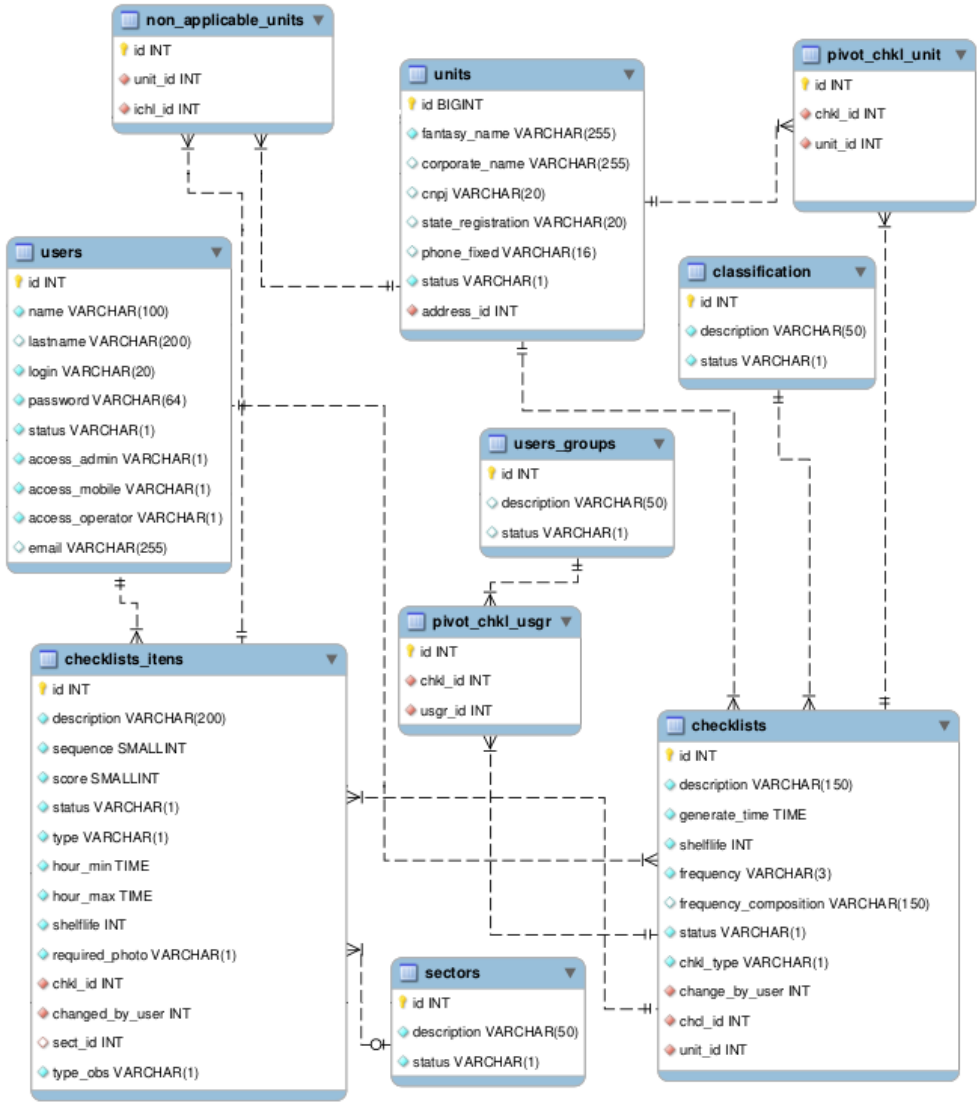

Figura 11 – MLD - *Checklist* e Perguntas associadas
Fonte: Autoria Própria (2023)

Por fim, encerramos com 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 dos *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.

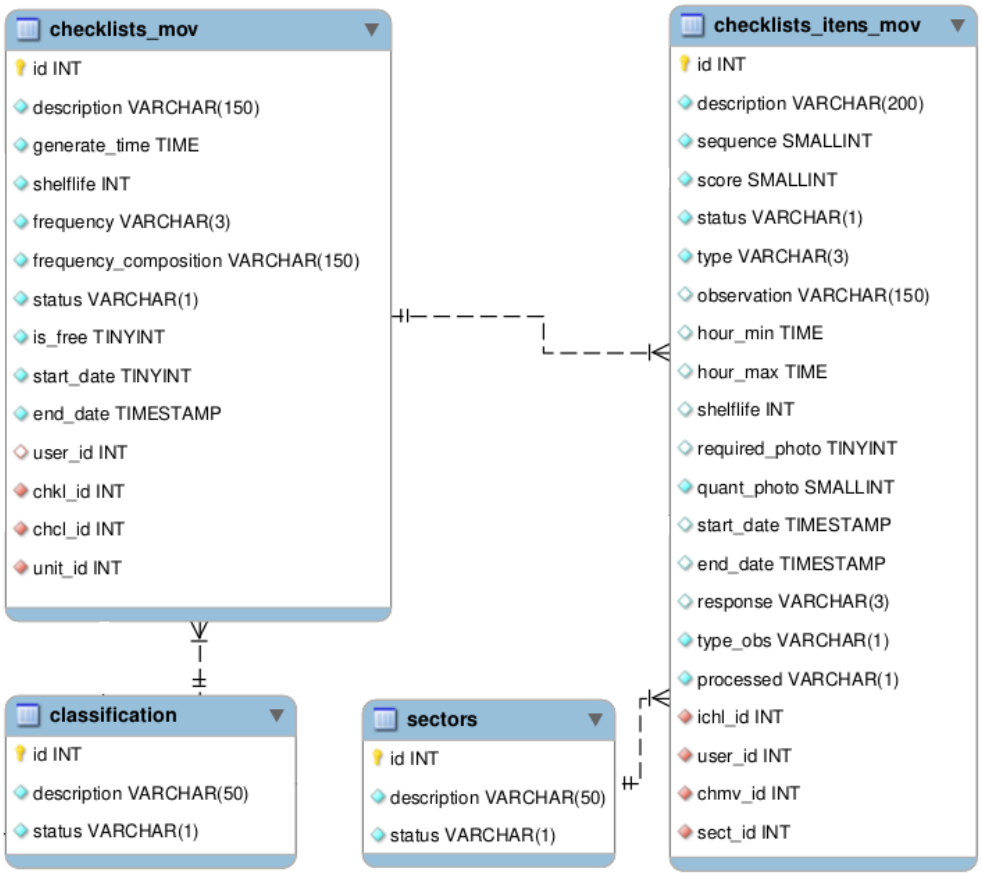

Figura 12 – MLD - *Checklist* Movimento e Perguntas associadas
Fonte: Autoria Própria (2023)

4.7 *Mockup* da Primeira Fase - Estrutura

Após a conclusão do levantamento de requisitos e o estabelecimento da estrutura do banco de dados, o próximo estágio do projeto consiste na implementação das telas do sistema. Nesta seção, serão abordados aspectos e telas fundamentais necessárias para o aprimoramento das funcionalidades do sistema, permitindo a criação dos cadastros e funcionalidades importantes descritas na fase 2, da Seção 4.1.

O início do processo de desenvolvimento na fase 1, implica na instalação de ferramentas essenciais, como o Template Phoenix, Bootstrap, JQuery e outras, que são fundamentais para o desenvolvimento ágil e eficaz. Após a conclusão deste trabalho, a expectativa é obter a base necessária para a construção da aplicação. O resultado esperado nesse estágio é a tela inicial *"home"* do sistema, conforme ilustrado na Figura 13.

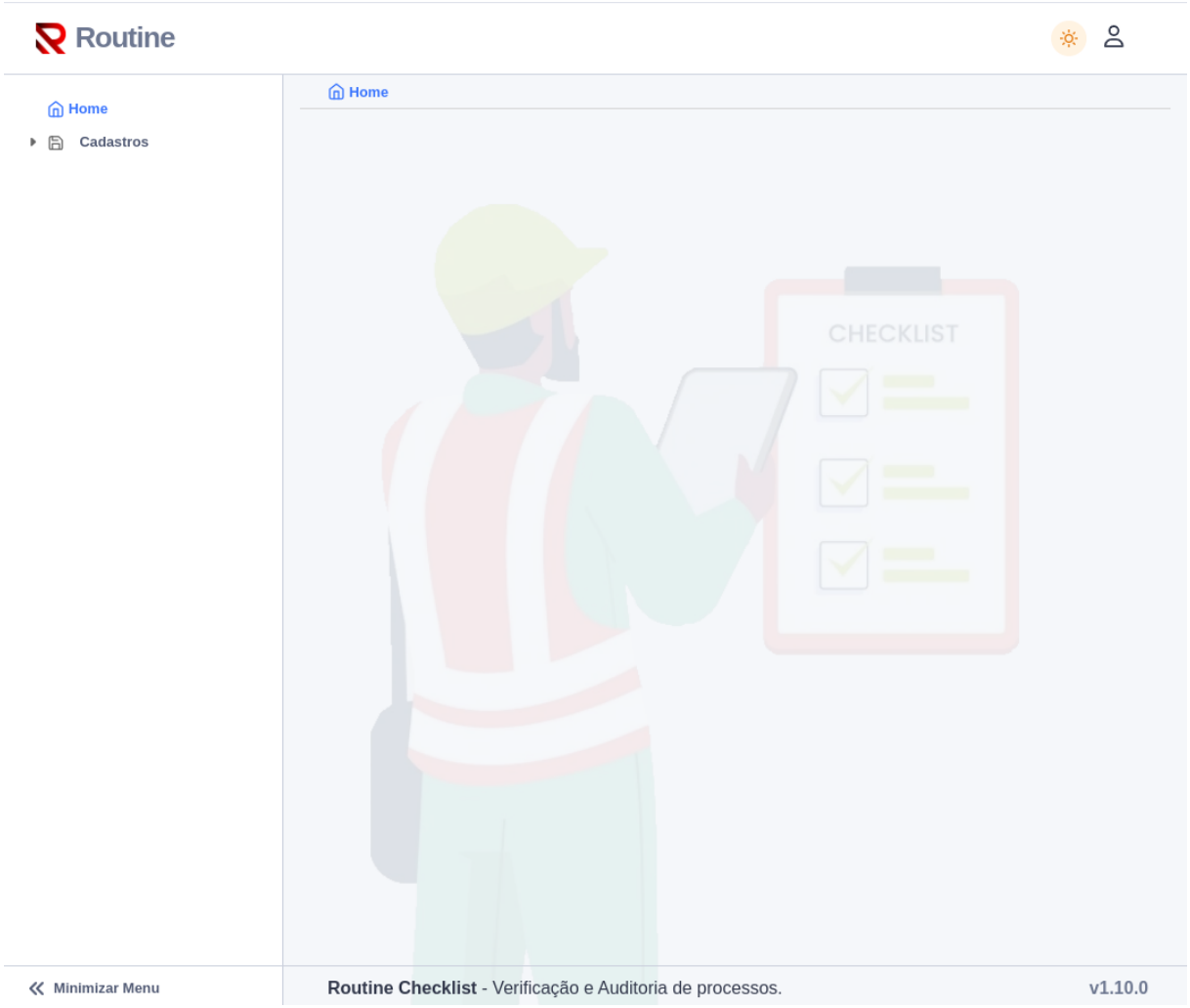

Figura 13 – Tela Home - Web - Implementação fase 1
Fonte: Autoria Própria (2023)

4.7.1 *Mockups* da Segunda Fase - Cadastros

Nesta fase do projeto, o foco recairá no desenvolvimento dos cadastros essenciais que permitirão o funcionamento adequado do sistema. A seguir, detalharemos as principais funcionalidades do sistema e apresentaremos os *mockups* das respectivas telas.

O início desta etapa contemplará o desenvolvimento dos cadastros mais elementares, a fim de agilizar a construção e a estruturação da aplicação. Uma das primeiras telas a ser desenvolvida será a 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 14.

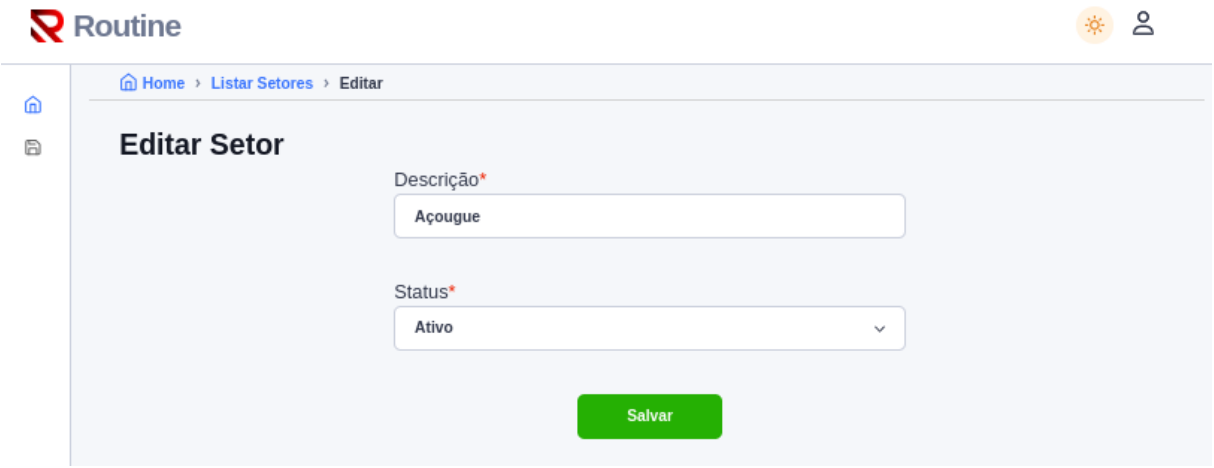

Figura 14 – Tela Cadastro de Setor - Web - Implementação fase 2
Fonte: Autoria Própria (2023)

Na sequência, prosseguiremos 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*. Para uma visualização mais clara dessa funcionalidade, consulte a Figura 15.

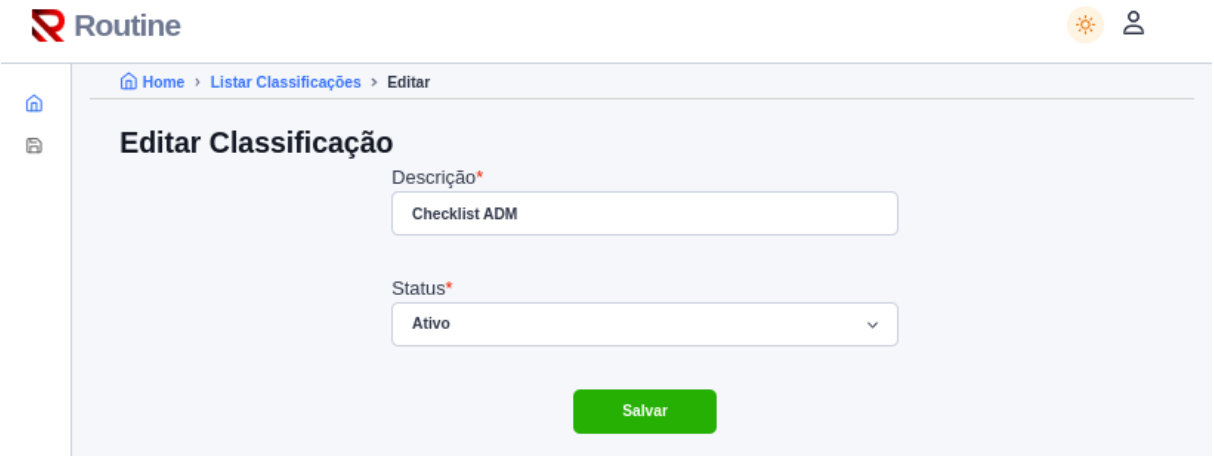

Figura 15 – Tela Cadastro de Classificação - Web - Implementação fase 2
Fonte: Autoria Própria (2023)

O registro de unidades desempenha um papel de grande relevância no contexto do sistema, 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. Isso ocorre devido à sua capacidade de gerar relatórios e disponibilizar informações para a análise dos resultados das unidades com base em sua localização geográfica. A tela de cadastro de unidades pode ser visualizada na Figura 16.

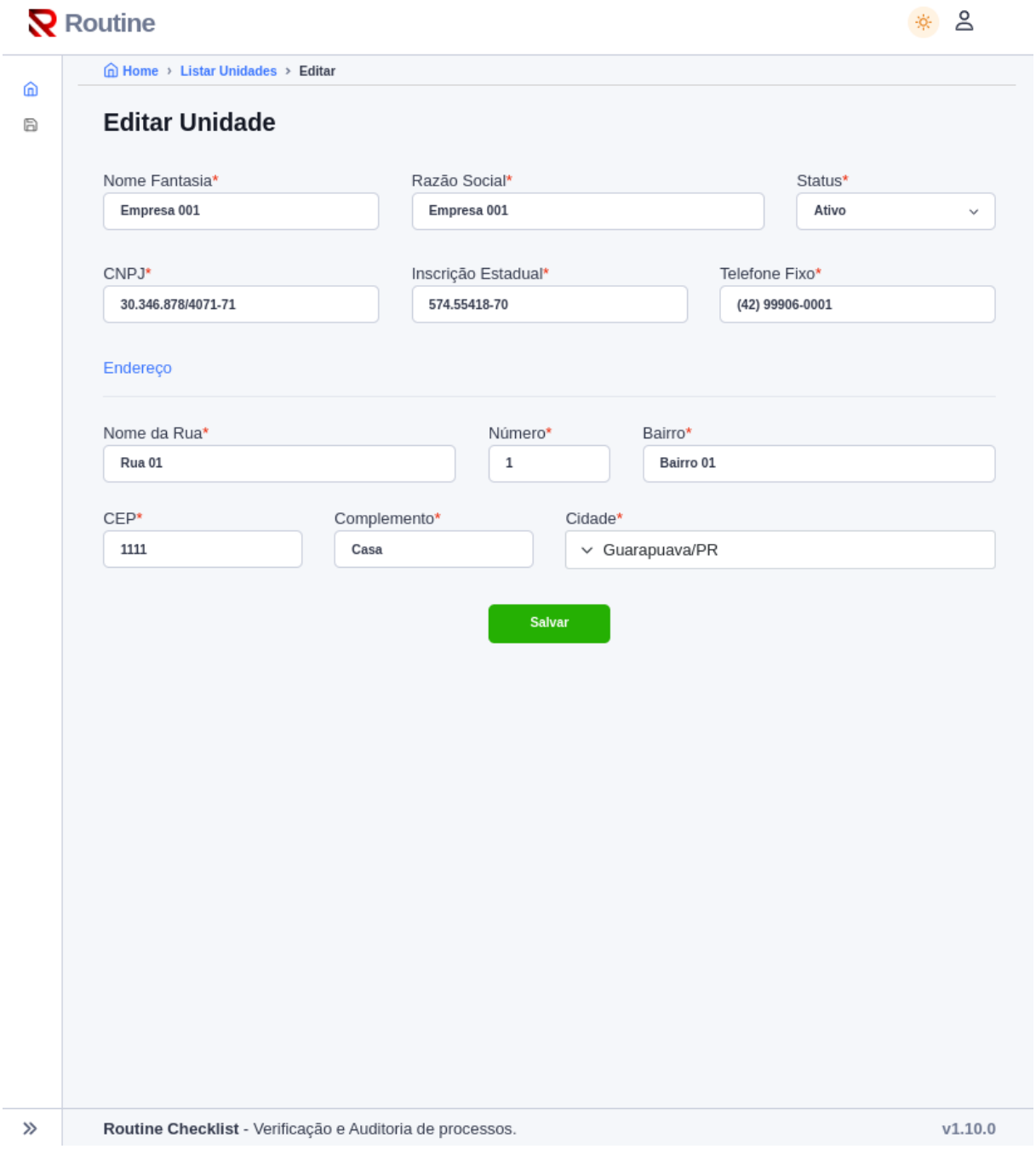

Figura 16 – Tela Cadastro de Unidade - Web - Implementação fase 2
Fonte: Autoria Própria (2023)

Outra tela de extrema importância é a de cadastro de grupos de usuários. Essa funcionalidade possibilita agrupar usuários, e com base nesses grupos, torna-se 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 17.

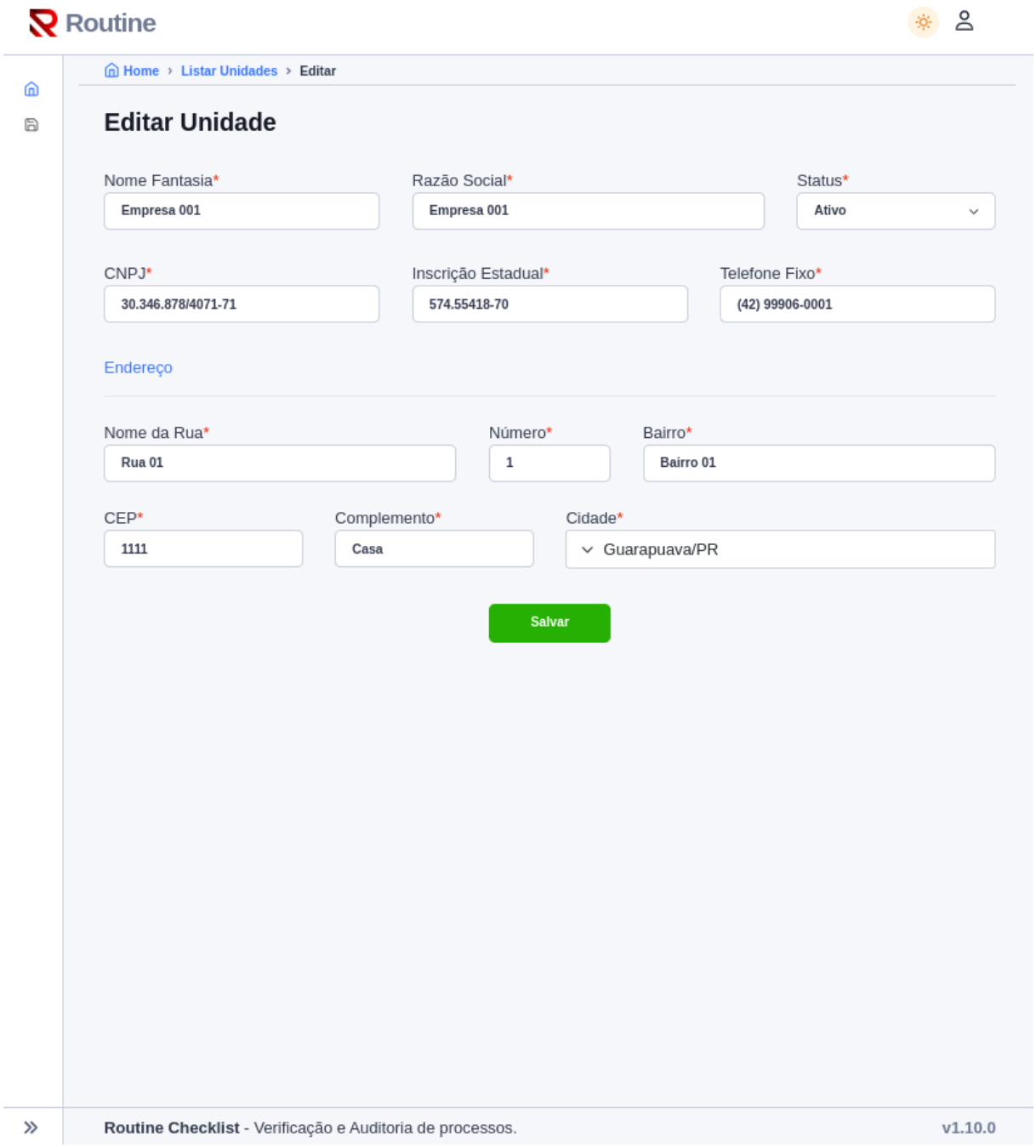

Figura 17 – Tela Cadastro de Grupo de usuários - Web - Implementação fase 2
Fonte: Autoria Própria (2023)

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 processamento das tarefas de *checklist*. Todas essas configurações cruciais são claramente apresentadas na Figura 18, destacando a importância desse componente no sistema em desenvolvimento.

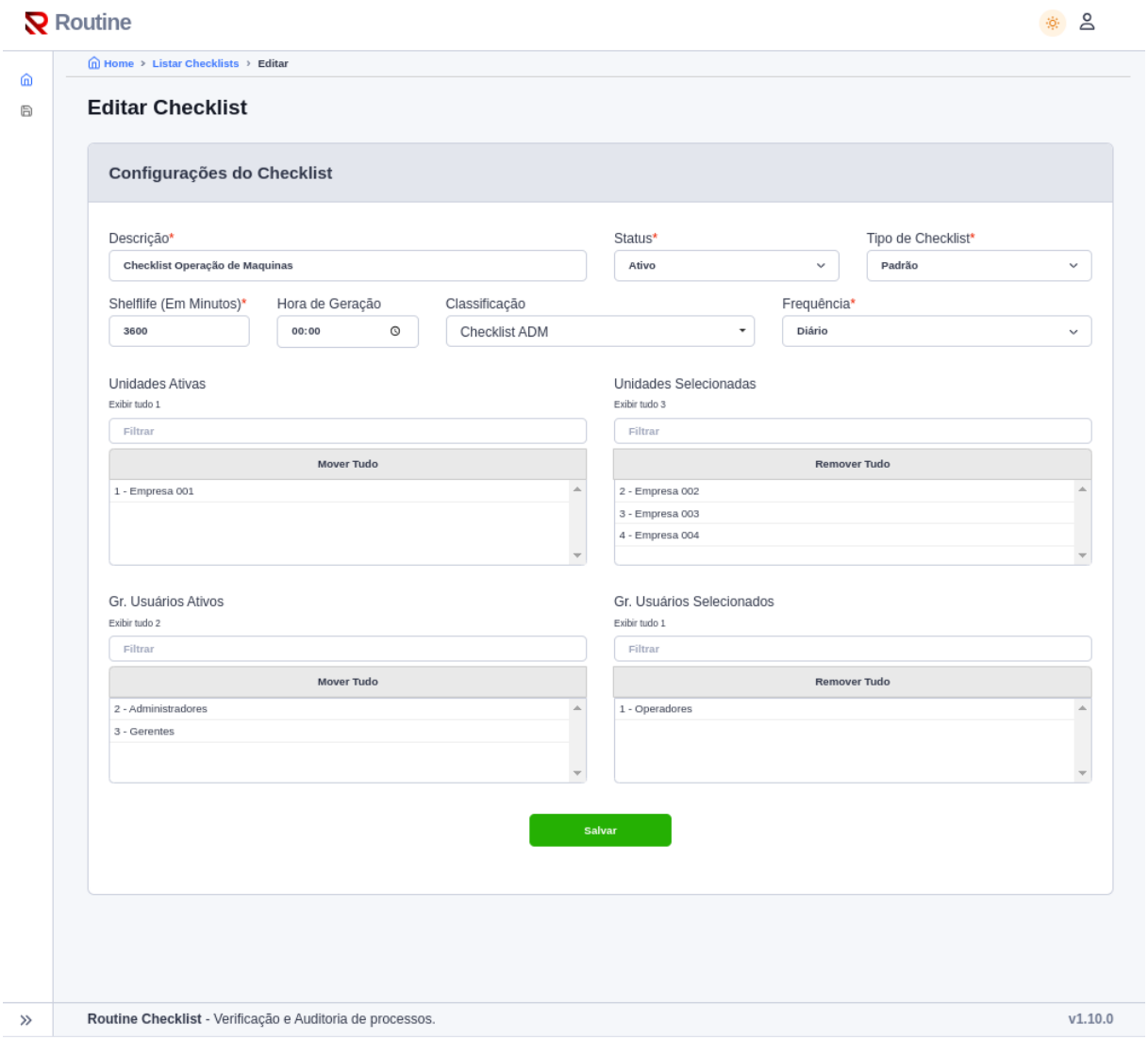

Figura 18 – Tela Cadastro de *Checklist* - Web - Implementação fase 2
Fonte: Autoria Própria (2023)

No contexto do cadastro do *checklist*, as perguntas assumem um papel de extrema importância. Conforme estabelecido pela regra, a ausência de perguntas registradas em um *checklist* impede a geração de tarefas. Portanto, é evidente a relevância deste cadastro. Além da própria pergunta, o processo de cadastramento desse recurso também envolve configurações e especificações que influenciam diretamente a geração de tarefas, em semelhança ao *checklist* em si. A representação visual desta tela encontra-se na Figura 19.

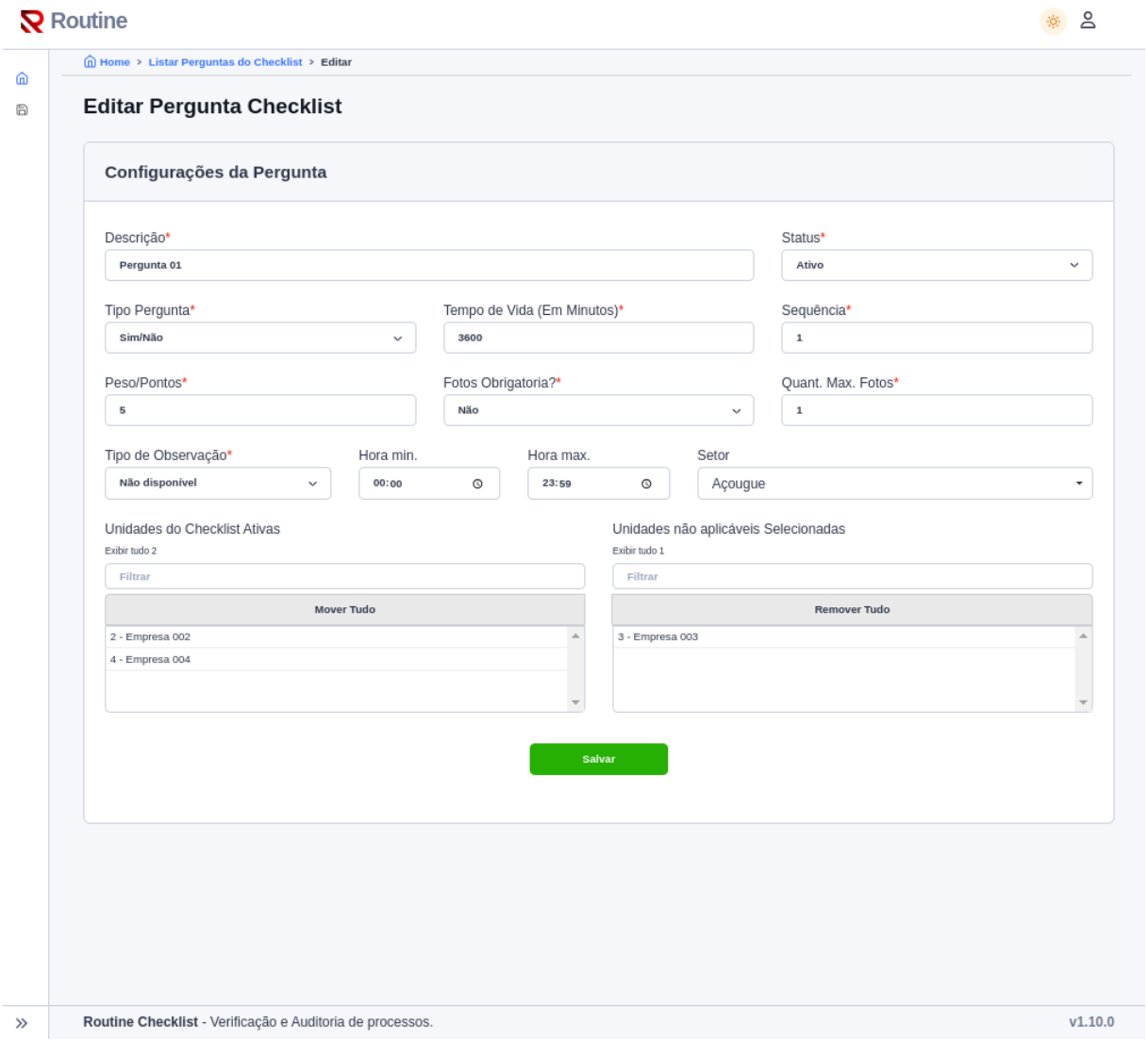

Figura 19 – Tela Cadastro de Perguntas do *Checklist* - Web - Implementação fase 2
Fonte: Autoria Própria (2023)

4.7.2 *Mockups* da Segunda Fase - Desenvolvimento do APP

Para possibilitar a execução e resposta dos *checklists*, é imperativo o desenvolvimento de uma aplicação mobile que mantenha comunicação com o sistema de retaguarda por meio de uma API. Esta aplicação, construída com *Ionic* para a plataforma Android, tem a responsabilidade de receber tarefas em modo online, para a execução do *checklist*. No que tange à representação visual e contextualização desta aplicação, destacamos, como ponto de partida, a tela inicial do APP, que é encarregada de carregar as tarefas pendentes para o usuário, conforme ilustrado na Figura 20.

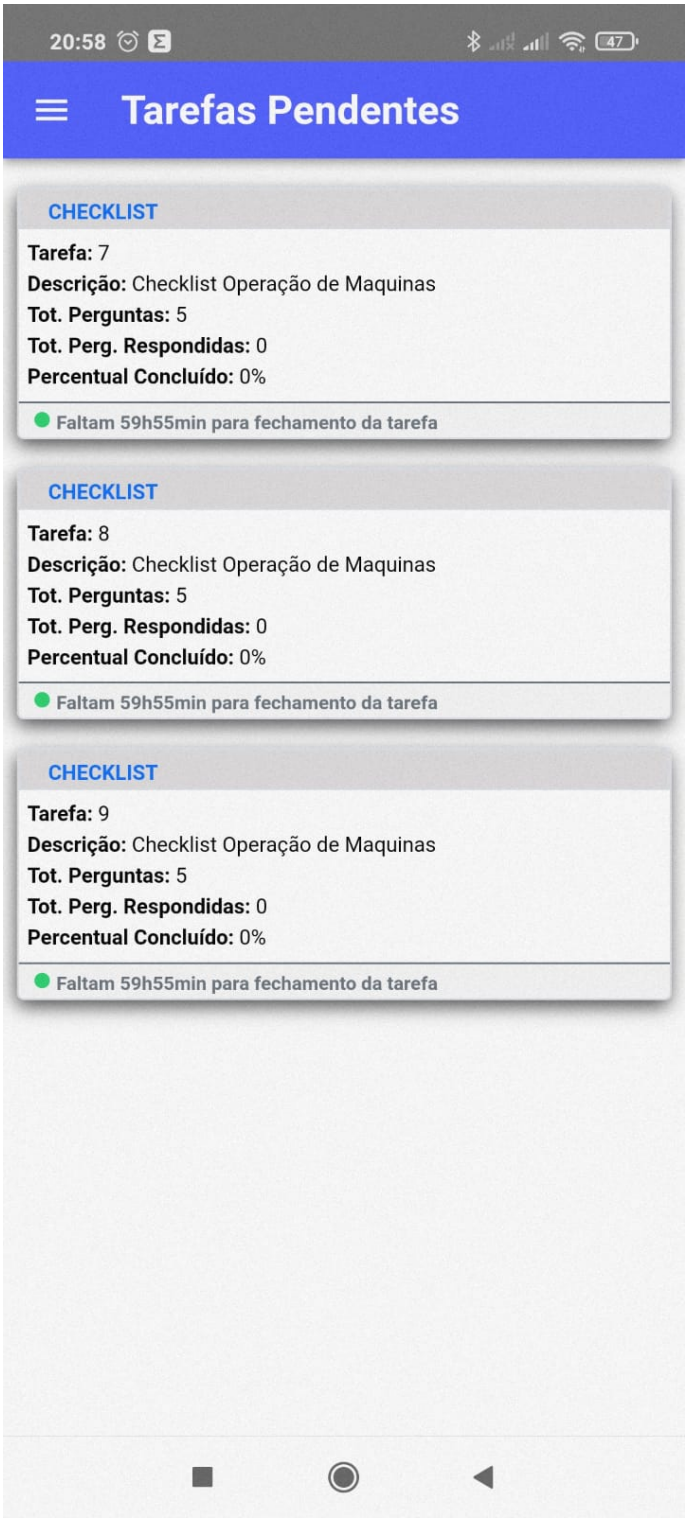

Figura 20 – Tela Inicial (*Home*) - APP - Implementação fase 2
Fonte: Autoria Própria (2023)

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 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. A Figura 21 exibe a lista de perguntas pendentes de resposta em um *checklist*, ilustrando essa tela.

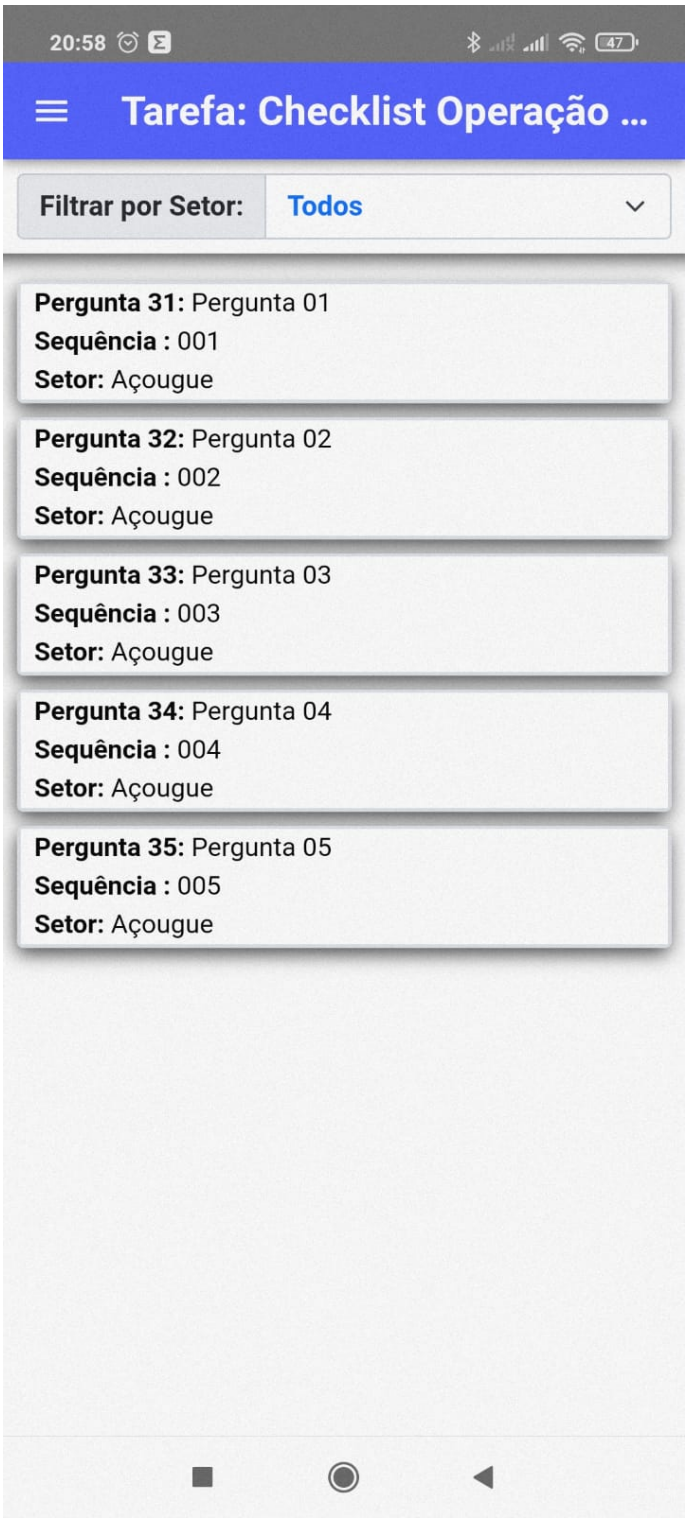

Figura 21 – Tela perguntas de um *checklist* - APP - Implementação fase 2
Fonte: Autoria Própria (2023)

Além disso, logo após a seleção de uma pergunta pelo usuário para registrar uma resposta, é fundamental que uma tela de opções de resposta seja apresentada, exibindo tanto os

itens obrigatórios quanto os opcionais. Esse processo é claramente exemplificado na Figura 22 que demonstra essa funcionalidade em ação.

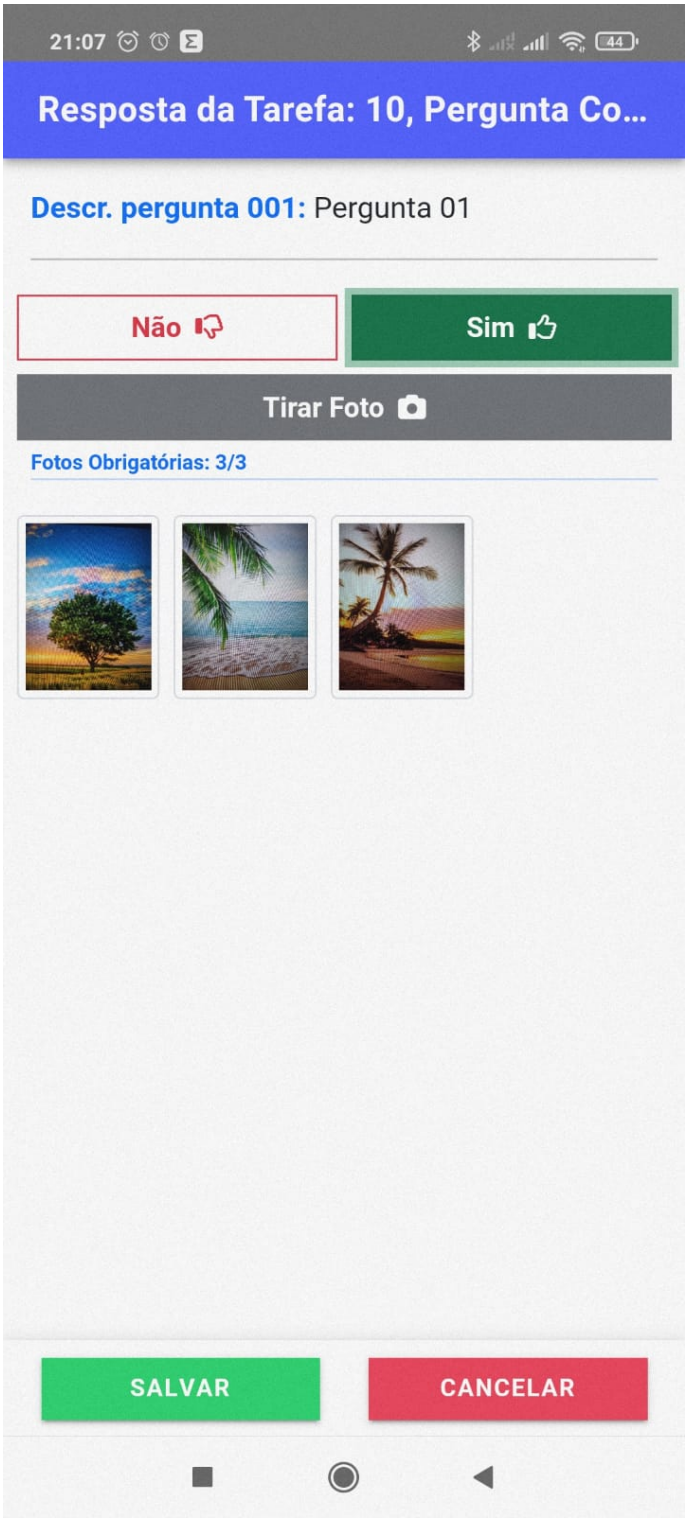

Figura 22 – Tela para Resposta de uma pergunta - APP - Implementação fase 2
Fonte: Autoria Própria (2023)

4.7.3 *Mockup* da Quarta Fase - Relatórios

Considerando a terceira fase, esta se concentrará no desenvolvimento dos processos em segundo plano da aplicação, tais como as regras de negócio e os procedimentos de criação e encerramento de tarefas. Nessa etapa, não haverá a criação de novas telas para o sistema.

Por outro lado, a quarta fase do desenvolvimento estará voltada para a elaboração de relatórios essenciais para a gestão e aprimoramento dos processos. A Figura 23 ilustra uma visão prospectiva da aparência da tela "*Home*" após o início da execução das tarefas de *checklists* pelos usuários.

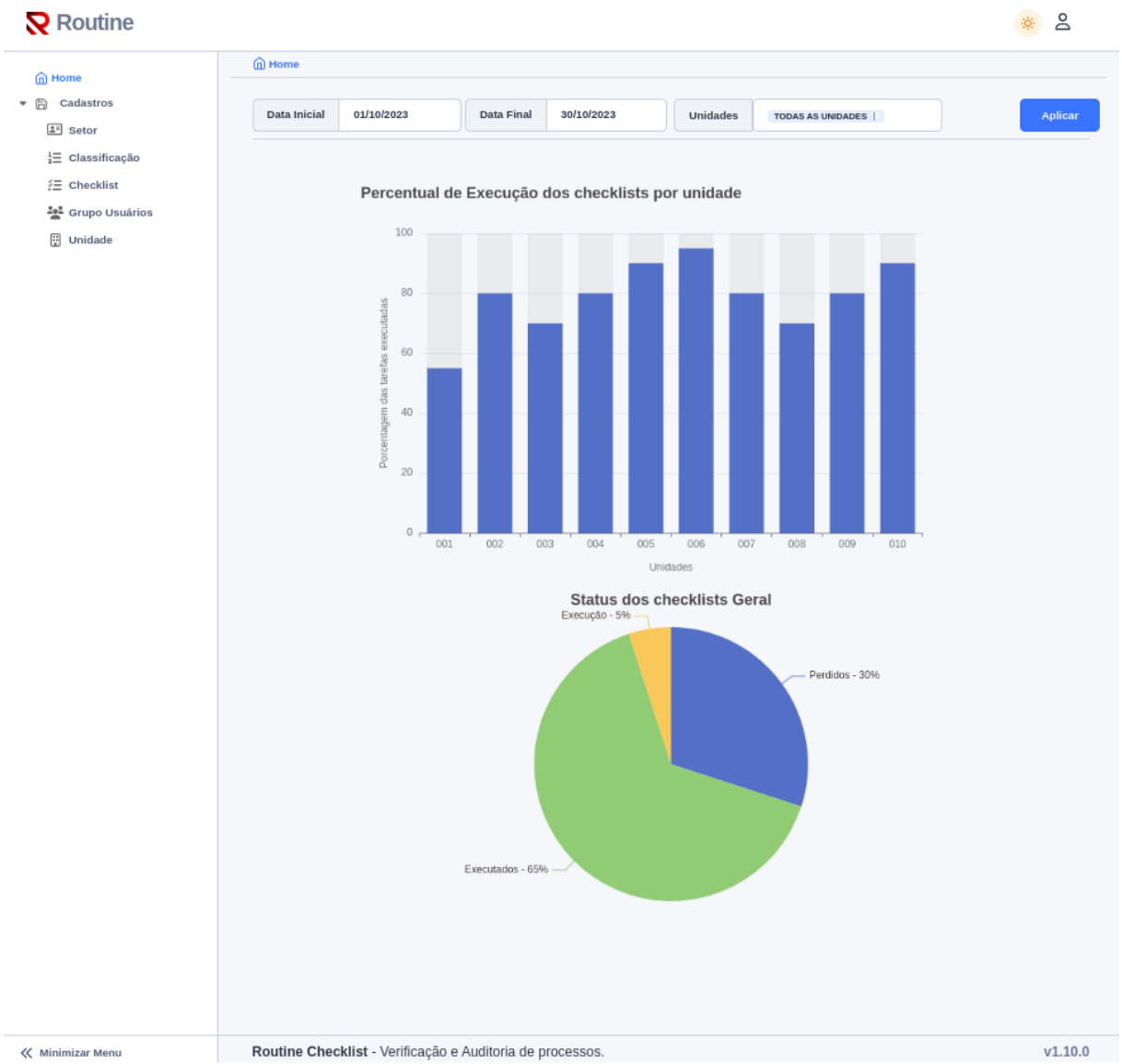

Figura 23 – Tela Home - WEB - Implementação fase 4
Fonte: Autoria Própria (2023)

5 PLANEJAMENTO DO TRABALHO

O planejamento do trabalho proposto que será desenvolvido ao longo do período letivo, está descrito no cronograma da Quadro 2. Neste cronograma constam todas as atividades com seus respectivos prazos para o cumprimento.

Quadro 2 – Cronograma de Atividades

Atividades	**Ago**	**Set**	**Out**	**Nov**	**Dez**
1.Elaboração do projeto de TCC					
2.Revisão bibliográfica					
3.Defesa do projeto de TCC					
4.Revisão dos apontamentos da banca					
5.Desenvolvimento do sistema					
6.Teste do programa resultante					
7.Redação da monografia do TCC					
8.Defesa e correções da monografia do TCC					

5.1 Recursos necessários

Para a execução e desenvolvimento do trabalho proposto, será necessário a disponibilidades de alguns recursos que são eles:

- Acesso a Internet;
- Acesso a acervo bibliográfico;
- Computador com configurações mínimas para desenvolvimento de *software* em ambiente WEB;
- Servidor na internet para disponibilizar aplicação para produção;
- Compra de Domínio para disponibilizar a aplicação na internet;
- *Smartphone* para testes, execução e desenvolvimento do APP de *checklist*.

5.2 Horário de trabalho

O horário destinado para realização das atividades do TCC, bem como o horário destinado para a reunião semanal/quinzenal com o orientador estão descritos no cronograma do Quadro 3. Este horário é definido com orientador levando em consideração a complexidade do trabalho a ser desenvolvido.

Quadro 3 – Horário de Trabalho.

Horário	**Seg**	**Ter**	**Qua**	**Qui**	**Sex**	**Sab**
18h50 - 19h40						
19h40 - 20h30			Orientaçao			TCC
20h30 - 21h20	TCC	TCC	TCC	TCC		TCC
21h30 - 22h15	TCC	TCC	TCC	TCC		TCC

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 está sendo desenvolvido em conformidade com as melhores práticas recomendadas para o desenvolvimento de software, conforme definido pela W3C, com base na documentação do *framework* Laravel e das ferramentas incorporadas no projeto. Ao concluir o cronograma deste trabalho, esperamos disponibilizar 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.

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. Citado na página 1.

ANGULAR. **Docs**. [S.l.], 2023. Disponível em: <https://angular.io/docs>. Acesso em: 09 de outubro de 2023. Citado na página 9.

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. Citado na página 13.

BOOTSTRAP. **Home**. [S.l.], 2023. Disponível em: <https://getbootstrap.com/>. Acesso em: 08 de outubro de 2023. Citado na página 8.

CARDOSO, G.; CARDOSO, V. **Sistemas de Banco de Dados: Uma abordagem introdutoria e aplicada**. 1. ed. São Paulo: Saraiva, 2012. Citado na página 14.

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. Citado na página 8.

CONSóRCIO WORLD WIDE WEB - W3C. **Sobre o W3C**. [S.l.], 2023. Disponível em: <https://www.w3c.br/Sobre>. Acesso em: 08 de outubro de 2023. Citado na página 8.

DAVID, F. **JavaScript: o guia definitivo**. 6. ed. Porto Alegre: Bookman Companhia Editora Ltda, 2013. Citado na página 9.

EIS, D.; FERREIRA, E. **HTML5 e CSS3: Com farinha e pimenta**. 1. ed. São Paulo: Tableless, 2012. Citado na página 8.

FASTFIELD, INC. **Dashboard**. [S.l.], 2023. Disponível em: <https://portal.fastfieldforms.com/>. Acesso em: 20 de setembro de 2023. Citado na página 4.

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. Citado na página 1.

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. Citado na página 22.

IONIC. **Docs**. [S.l.], 2023. Disponível em: <https://ionic.io/docs>. Acesso em: 09 de outubro de 2023. Citado na página 9.

JQUERY. **Home**. [S.l.], 2023. Disponível em: <https://jquery.com>. Acesso em: 08 de outubro de 2023. Citado na página 9.

LARAVEL. **Conheça o Laravel**. [S.l.], 2023. Disponível em: <https://laravel.com/docs/10.x>. Acesso em: 10 de outubro de 2023. Citado na página 11.

LARAVEL. **Eloquente: primeiros passos**. [S.l.], 2023. Disponível em: <https://laravel.com/docs/10.x/eloquent>. Acesso em: 10 de outubro de 2023. Citado na página 11.

MATERIALIZE. **Home**. [S.l.], 2023. Disponível em: <https://materializecss.com/>. Acesso em: 08 de outubro de 2023. Citado na página 8.

MICROSOFT CORPORATION. **Começando**. [S.l.], 2023. Disponível em: <https://code.visualstudio.com/docs>. Acesso em: 11 de outubro de 2023. Citado na página 14.

MONTEIRO, E. R. et al. **DevOps**. 1. ed. Porto Alegre: Sagah, 2021. Citado na página 13.

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

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. Citado na página 10.

POSTGRESQL. **Sobre**. [S.l.], 2023. Disponível em: <https://www.postgresql.org/about>. Acesso em: 11 de outubro de 2023. Citado 2 vezes nas páginas 15 e 20.

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

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software: Uma abordagem profissional**. 9. ed. Porto Alegre: AMGH Editora Ltda, 2021. Citado 2 vezes nas páginas 11 e 13.

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. Citado na página 13.

RP INFO SISTEMAS. **Dashboard**. [S.l.], 2023. Disponível em: <https://www.rpinfo.com.br/produto/task/79>. Acesso em: 20 de setembro de 2023. Citado na página 5.

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. Citado na página 12.

SILVA, M. S. **CSS: Desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3**. 1. ed. São Paulo: Novatec, 2011. Citado na página 8.

SILVA, M. S. **JQuery - A Biblioteca do Programador JavaScript**. 3. ed. Porto Alegre: Bookman Companhia Editora Ltda, 2013. Citado na página 9.

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. Citado na página 1.

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. Citado na página 3.

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. Citado na página 10.

ZABOOT, D.; MATOS, E. **Aplicativos com Bootstrap e Angular - Como desenvolver APPS responsivos**. 1. ed. São Paulo: Érica, 2020. Citado na página 8.

Apêndices

APÊNDICE A – Modelo Lógico de Dados - MLD

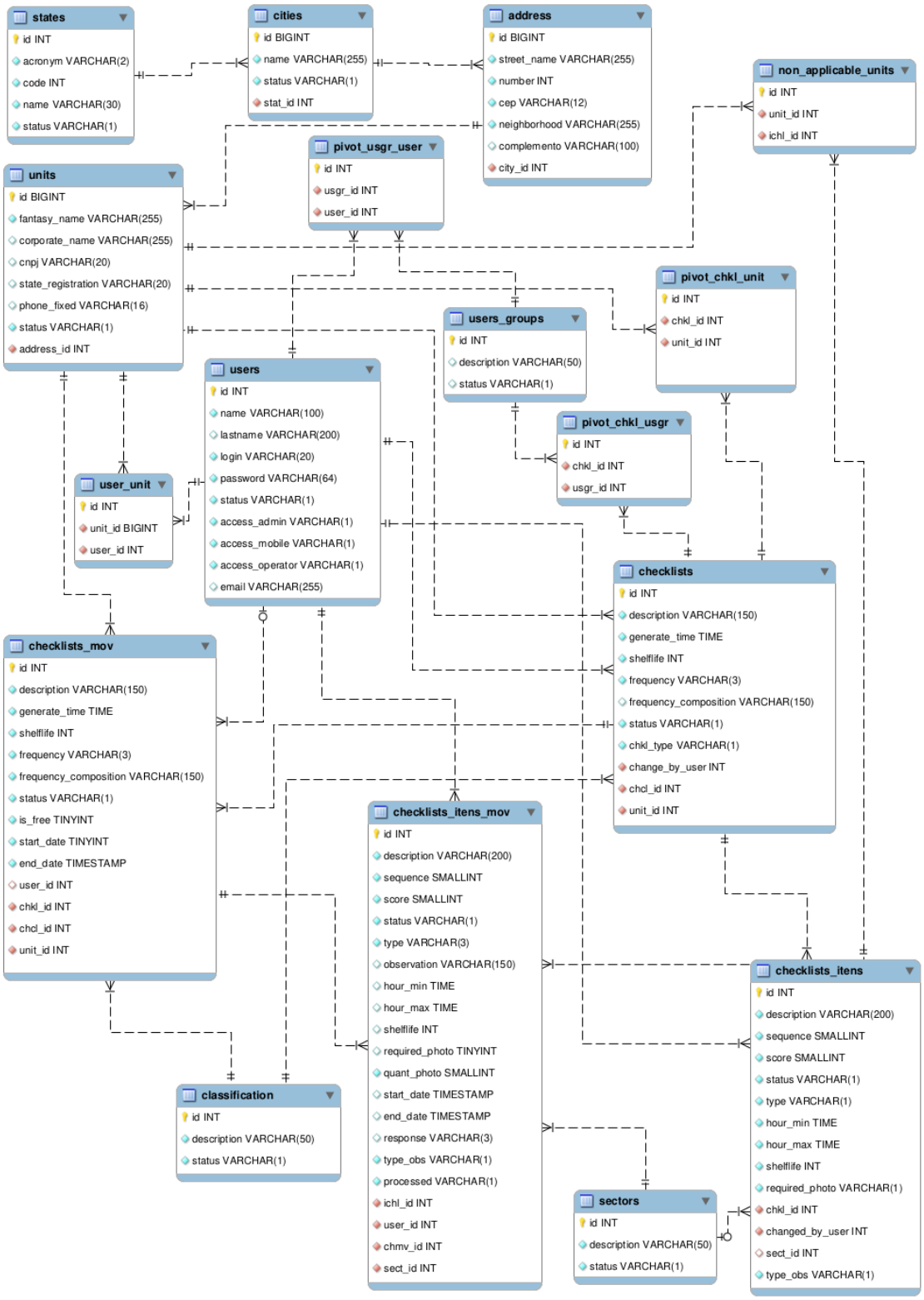

Figura 24 – Modelo Lógico de Dados - MLD
Fonte: Autoria Própria (2023)