**Análise de Requisitos**

**Projeto:** Sistema de Gestão de Tarefas para Funcionários de Supermercado (TCC)

**Tecnologias propostas:** HTML, CSS, JavaScript, React.js (web), Node.js (backend), React Native (mobile), SQL Server (banco de dados).

**1. Resumo do sistema**

Aplicação web e mobile para gerenciamento de tarefas operacionais em supermercados. Permite cadastrar funcionários, atribuir tarefas com evidência fotográfica, registrar check-ins com data/hora, controlar acesso por perfis (administrador, supervisor, funcionário), notificar atribuições e prazos, acompanhar métricas de produtividade e integrar controle de estoque com alertas de falta.

**2. Stakeholders**

* Dono / Gerente da rede (administrador)
* Supervisor de setor
* Funcionários de loja
* Administrador do sistema / TI
* Auditor/Orientador (TCC)

**3. Objetivos principais**

* Organizar e rastrear tarefas diárias por setor e funcionário.
* Garantir evidência de execução via upload de fotos e registro de check-in/check-out.
* Notificar automaticamente novas tarefas e prazos críticos.
* Gerar métricas e relatórios para supervisão por setor ou funcionário.
* Integrar consulta de estoque e alertas de falta de produto.

**4. Escopo**

**Inclusões:** cadastro de funcionários, níveis de acesso, CRUD de tarefas, atribuição, upload de fotos, check-in (data/hora), notificações, painel de tarefas (pendentes, em andamento, concluídas), registro de tempo, métricas, relatórios, consulta de estoque por código de barras/nome, alerta de falta.

**Exclusões (fora do TCC):** integração com leitores físicos de códigos de barras via hardware (pode-se simular/usar API de terceiros), integração contábil/ERP profunda, pagamentos.

**5. Requisitos Funcionais (RF)**

1. RF-01: Cadastro de funcionários — inserir nome, função, setor, e-mail/telefone (opcional), foto de perfil.
2. RF-02: Gestão de perfis — criar/editar papéis: Administrador, Supervisor, Funcionário; associar permissões.
3. RF-03: Autenticação — login por e-mail/usuário + senha; recuperação de senha.
4. RF-04: CRUD de tarefas — criar, ler, atualizar, excluir tarefas; cada tarefa tem: título, descrição, setor alvo, prioridade, prazo (data/hora), tempo estimado.
5. RF-05: Atribuição de tarefas — atribuir tarefas a um ou mais funcionários; opção de atribuição por setor (fila) ou individual.
6. RF-06: Notificações automáticas — enviar notificação (push no mobile e alert no web) quando uma tarefa for atribuída a um usuário ou a seu setor.
7. RF-07: Check-in/Marcar como iniciada — funcionário marca tarefa como iniciada; app registra data/hora de início; pode anexar foto como evidência.
8. RF-08: Upload de evidência — permitir upload de 1..N fotos por evento (início/conclusão); armazenamento seguro.
9. RF-09: Marcar como concluída — funcionário envia confirmação de conclusão; notificação enviada ao supervisor para validação (ou aprovação automática, conforme regra).
10. RF-10: Alertas de prazo — notificação automática para tarefas próximas do prazo (configurável: ex. 24h, 2h antes).
11. RF-11: Painel de tarefas — visão com filtros: setor, funcionário, status (pendente, em andamento, concluída), prioridade e data.
12. RF-12: Registro de tempo — sistema calcula tempo decorrido entre início e conclusão (ou check-out/manual) e armazena histórico.
13. RF-13: Métricas de produtividade — calcular KPIs: tempo médio por tarefa, tarefas concluídas por funcionário/turno, taxa de reabertura.
14. RF-14: Relatórios — gerar relatórios por período, setor ou funcionário; exportar CSV/PDF.
15. RF-15: Controle de estoque — pesquisa de produto por código de barras ou nome; exibir estoque atual.
16. RF-16: Tela de falta de produtos («Glândula») — listagem de produtos com estoque abaixo do limite; permitir transformar falta em tarefa (reposição).
17. RF-17: Painel específico de faltas — notificações e listagem de itens críticos por loja.
18. RF-18: Filtros avançados — filtros por setor, funcionário, status, prazo, prioridade para dashboards e relatórios.
19. RF-19: Multi-unidade — suporte para múltiplas lojas/unidades com separação de dados por unidade.
20. RF-20: Histórico e auditoria — manter logs de criação/edição/exclusão de tarefas, uploads e ações de usuário.

**6. Requisitos Não-Funcionais (RNF)**

**Usabilidade**

* RNF-U1: Interface intuitiva e responsiva para celulares e tablets (React + React Native); operações em 1–3 toques para ações comuns.
* RNF-U2: Design limpo adaptado para uso rápido em ambiente de trabalho.

**Segurança**

* RNF-S1: Autenticação por usuário/senha com hashing seguro (bcrypt/scrypt) e HTTPS obrigatório.
* RNF-S2: Controle de acesso baseado em papéis (RBAC) — validação no frontend e backend.
* RNF-S3: Armazenamento seguro de fotos (armazenamento em blob com acesso restrito; considerar criptografia em descanso ou políticas de acesso no SQL Server ou blobs externos).
* RNF-S4: Registro de logs de auditoria com retenção configurável.

**Performance**

* RNF-P1: Tempo de resposta para operações críticas (abrir painel, enviar foto, atualizar status) abaixo de 2s em condições normais de rede.
* RNF-P2: Upload de fotos com feedback de progresso; compressão/resizing no cliente antes do upload para reduzir latência e uso de dados.

**Disponibilidade / Offline**

* RNF-A1: Modo offline temporário no app mobile: permite marcar início/conclusão e armazenar fotos localmente; sincronização automática quando a conexão for restabelecida.
* RNF-A2: Conflitos de sincronização resolvidos por versão/último horário ou política definida (ex.: operações em ordem cronológica).

**Escalabilidade**

* RNF-ES1: Arquitetura escalável (Node.js stateless + banco SQL Server escalável / réplicas) para suportar múltiplas lojas e maior número de usuários.
* RNF-ES2: Possibilidade de mover storage de fotos para serviço de blobs (Azure Blob Storage / S3) se necessário.

**Conformidade & Privacidade**

* RNF-C1: Proteção de dados pessoais — armazenar apenas o necessário e permitir exclusão conforme regras da empresa/lei.

**7. Papéis e Permissões (matriz)**

* **Administrador:** tudo — gestão de unidades, usuários, relatórios, configurações de sistema.
* **Supervisor:** criar/atribuir tarefas, aprovar/validar conclusões, ver relatórios por setor, transformar faltas em tarefas.
* **Funcionário:** ver tarefas atribuídas, marcar início/conclusão, fazer upload de fotos, comentar.

**8. Casos de Uso / Histórias de Usuário (exemplos)**

* *Como funcionário, quero ver minhas tarefas do dia para saber o que fazer.*
* *Como supervisor, quero receber notificação quando uma tarefa é marcada como concluída para validar.*
* *Como administrador, quero gerar relatório de produtividade por setor no mês passado.*
* *Como responsável pela loja, quero uma tela com produtos em falta para criar tarefas de reposição rapidamente.*

**9. Modelo de Dados (tabelas principais — alto nível)**

* **Employee**(EmployeeID, Nome, Email, SenhaHash, Função, SetorID, RoleID, UnidadeID, ativo, criado\_em)
* **Role**(RoleID, nome, permissoes\_json)
* **Unit/Store**(UnitID, nome, endereco, timezone)
* **Task**(TaskID, titulo, descricao, setorID, prioridade, prazo, status, estimativa\_minutos, criado\_por, criado\_em)
* **TaskAssignment**(AssignmentID, TaskID, EmployeeID, atribuido\_em, atribuido\_por)
* **TaskEvent**(EventID, TaskID, tipo[inicio,conclusao,comentario], descricao, employeeID, timestamp)
* **TaskPhoto**(PhotoID, TaskID, EventID, url\_armazenamento, thumbnail\_url, uploaded\_by, uploaded\_at)
* **Notification**(NotificationID, tipo, target\_user, payload, lido, created\_at)
* **InventoryItem**(ItemID, codigo\_barras, nome, estoque\_atual, estoque\_minimo, unidadeID)
* **InventoryAlert**(AlertID, ItemID, UnidadeID, nivel, created\_at, resolved)
* **Metrics/Logs** conforme necessidade.

**10. API — Endpoints sugeridos (REST)**

* POST /api/auth/login — login
* POST /api/auth/refresh — refresh token
* GET /api/employees — listar funcionários (admin/supervisor)
* POST /api/employees — criar funcionário
* GET /api/tasks — listar/filtrar tarefas
* POST /api/tasks — criar tarefa
* PUT /api/tasks/:id — editar
* DELETE /api/tasks/:id — excluir (auditar)
* POST /api/tasks/:id/assign — atribuir
* POST /api/tasks/:id/start — marcar início (com foto)
* POST /api/tasks/:id/complete — marcar conclusão (com foto)
* POST /api/uploads — upload de foto (retorna URL)
* GET /api/inventory/search?q= — busca por código/nome
* GET /api/inventory/alerts — listar faltas
* GET /api/reports/productivity?unit=&from=&to= — relatórios

Autenticação com JWT e verificação de escopo/roles no middleware.

**11. UX / Telas principais**

* Tela de login (web + mobile)
* Dashboard geral (admin) — unidades, métricas rápidas
* Painel de tarefas (por unidade/setor) — abas: pendentes / em andamento / concluídas
* Tela de tarefa (detalhes) — descrição, fotos, histórico de eventos, atribuições, botão iniciar/concluir
* Tela de atribuição rápida (supervisor)
* Tela "Glândula" — produtos com falta e botão criar tarefa de reposição
* Painel de relatórios e filtros
* Configurações de sistema (papéis, unidades, limites de estoque)

**12. Regras de Negócio importantes**

* Uma tarefa pode ter múltiplas atribuições (ex.: equipe) — mas somente o funcionário que registrou início será considerado para tempo de execução, a menos que política determine soma/colaboração.
* Fotos anexadas são obrigatórias para tarefas críticas (configurável por categoria/priority).
* Supervisores podem rejeitar conclusão e reabrir tarefa, com justificativa.
* Alertas de estoque disparam quando estoque\_atual <= estoque\_minimo.

**13. Critérios de Aceitação (exemplos)**

* CA-01: Ao atribuir uma tarefa a um funcionário, este recebe notificação push no app em < 30s (em rede normal).
* CA-02: Funcionário consegue iniciar tarefa offline e app sincroniza evento e foto quando voltar online.
* CA-03: Relatório de produtividade exibe corretamente o tempo médio por tarefa no período selecionado.

**14. Considerações Técnicas / Implementação**

* **Frontend Web:** React.js + React Router + state management (Context/Redux conforme complexidade). UI responsiva (componentes acessíveis).
* **Mobile:** React Native; implementar armazenamento local (SQLite/AsyncStorage) para fila offline de eventos e fotos; compressão e enqueue de uploads.
* **Backend:** Node.js (Express ou Fastify); endpoints REST; autenticação JWT; middleware RBAC.
* **Banco:** SQL Server — modelagem relacional, índices para consultas por unidade/funcionário/status.
* **Armazenamento de fotos:** Inicialmente armazenamento no próprio servidor ou em um blob storage (Azure Blob ou S3) com U